Université Sorbonne Paris Nord

École doctorale Galilée

Laboratoire d'Informatique de Paris Nord

Span-Based Structured Prediction for Information Graph Extraction

Présentée par Urchade Zaratiana

Thèse de doctorat d' INFORMATIQUE

Soutenue publiquement le JJ/MM/AAAA devant le jury composé de :

Thierry CHARNOIS, Prof. HDR	Université Sorbonne Paris Nord	Directeur
Nadi TOMEH, MCF	Université Sorbonne Paris Nord	Encadrant

Abstract

Title: Span-based Structure Prediction for Information Graph Extraction.

Keywords: *Named Entity Recognition, Relation Extraction, Graph Algorithm.*

Abstract: This thesis proposes new deep learning architectures, training and decoding algorithm for Information extraction. Our contribution focus in Named Entity Recognition and Joint Entity and Relation Extraction. All of our contribution is centered around the graph structure of this task, we leverage for designing our model architecture but also for our efficient training and decoding algorithm. For Named Entity Recognition, we firstly tackle the overlap span problem in span-based NER, where the output entity should not have overlapping. For that, we propose an architecture called GNNer that integrates that enrich span representation with overlapping graph using graph neural networks to reduced overlapping. We then propose global span selection decoding, which is an exact decoding to span-based NER, by treating span decoding as maximum independent set problem. From that, we propose the Filtered Semi-Markov CRF, a more computationally efficient and better alternative to the classical CRF Semi-Markov CRF models.

For joint entity and relation extraction (JERE), we propose diverse set of new architectures. Our first contribution treat the task as an auto-regressive graph generation. Unlike previous works that either produce augmented text or structured textual output, our proposed model ATG generated textual spans using pointer mechanism, ensuring the inference to produce valid information extraction graph using constrained decoding. From this basis, we further proposes new architecture that get rid of the auto-regressive generation for more efficient inference.

Contents

List of Figures

List of Tables

Part I

Introduction and background

Chapter 1

Introduction and Overview

1.1 Information Extraction

In the rapidly evolving landscape of natural language processing (NLP), the extraction of structured information from unstructured data stands as a cornerstone, enabling a myriad of applications from knowledge base construction to information retrieval and question answering. The field of information extraction has existed for long time, starting from **Named Entity Recognition** (NER) where the goal is to identify and classify named entities such person, location and organization, and then **Relation Extraction** (RE) which consist in prediction the relationship between the entities, such as "person - Work for - organisation". While many other information task exist, such as event extraction and coreference resolution, in this thesis we primarily focus on NER and RE. The reason for this focus is twofold: firstly, I am pursuing my PhD in the industry, and NER and RE have more practical applications, such as data anonymization and knowledge graph construction. Secondly, other tasks are not as relevant in this industrial context.

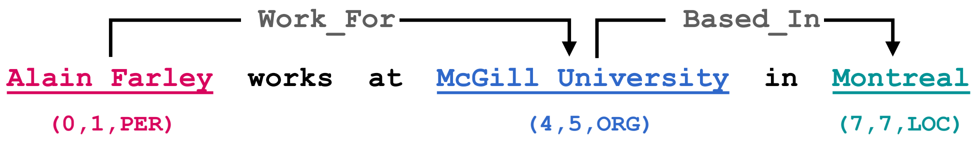


Figure 1.1: Named entity and relation extraction.

Traditionally, early models for these task relied on rule-based approaches, where experts manually designed rules [Appelt et al., 1993, Cunningham et al., 2002]. However, such models often lacked robustness and struggled with handling complex language patterns. To address these limitations, models based on statistical machine learning emerged, utilizing handcrafted features. Yet, these models still faced challenges in handling the diversity and nuances of natural language. The advent of deep learning revolutionized entity extraction by enabling automatic feature learning from raw text. Specialized architectures based on Convolutional Neural Networks (CNNs) [LeCun et al., 1998], Recurrent Neural Networks (RNNs) [Elman, 1990, Hochreiter and Schmidhuber, 1997], and more recently, transform-

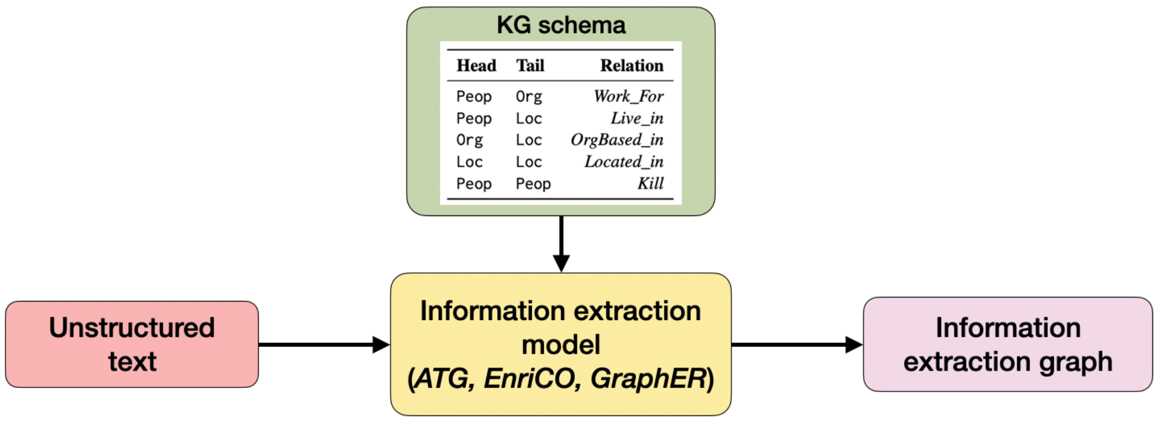


Figure 1.2: Research objective: having a single, lightweight model that can generate entity and relation from unstructured text, conditioned on a knowledge graph schema.

ers [Vaswani et al., 2017c], have been employed to automatically learn patterns from text data. However, despite their advancements, these models often exhibit inflexibility and are typically designed to extract entities and relations from a closed set of predefined types. Recent breakthroughs in Large Language Models (LLMs) have further pushed the boundaries of information extraction. By leveraging natural language instructions, these models can now extract novel entity and relation types with competitive performance, opening new avenues for information extraction tasks.

1.2 Research objective

The objective of information extraction is to transform raw, unstructured data into a structured, queryable database. This transformation is crucial for enabling easy access to information and facilitating the analysis of large datasets. During my PhD studies, I have focused on two main goals:

1. **Enhancing Reliability of Information Systems**: My primary goal is to develop efficient and scalable algorithms that improve the reliability of information extraction systems. Reliability, in this context, means enforcing a precise structural format and incorporating domain-specific knowledge. To this end, we conducted extensive research into developing a decoding algorithm for named entity recognition, which led to the creation of a global span selection algorithm. Additionally, I proposed efficient decoding techniques that ensure the output of relation extraction systems adheres to domain-specific knowledge, making the results more coherent. The ultimate aim is to make these systems as reliable and accurate as possible, addressing one of the significant challenges with large language models: their susceptibility to generating erroneous or fabricated information, i.e., hallucination.

2. **Developing strong models for resource-limited settings**: My second goal is to develop high-performance models that are easy to deploy in environments with

limited resources. Although large language models (LLMs) exhibit strong performance, their extensive size often restricts their accessibility for individuals with limited GPU resources. Furthermore, these models typically suffer from slow inference speeds due to their autoregressive, token-by-token generation process. In my thesis, I demonstrated that a carefully designed, lightweight named entity recognition model, GLiNER, can surpass larger models in zero-shot settings. My long-term vision is to expand this approach to more complex information extraction tasks. Specifically, I aim to develop lightweight models capable of efficiently identifying entities and their relationships, guided by a knowledge graph schema. I have already laid a solid foundation for this by proposing three innovative joint information extraction models: ATG, GraphER, and EnriCO. Looking forward, I plan to further scale these models to enhance their effectiveness.

1.3 Contributions

1.3.1 Span-Based Named Entity Recognition Training and Decoding

Traditionally, named entity recognition is performed by treating the task as sequence tagging using predefined schema such as the BIO tagging. Recently, span-base approached which enumerate all spans in the text and perform classification, has become popular, as it allow for richer representation. However, the independent span classification process make the output entity spans not respect some predefined constrained such as two entity should not overlap for flat NER. One of the focus of this thesis was to explore how to improve span-based NER models, especially how to train them and how to produce well-formed output

Implicit constraint To address the overlapping span issue, we propose GNNer, a novel architecture that leverages Graph Neural Networks (GNNs) [Kipf et al., 2018] to encode span overlap constraints implicitly within the model. By constructing a graph that captures the relationships between spans—including overlap information—and applying GNN layers, GNNer can learn to discriminate between competing spans, thereby reducing overlap without sacrificing performance. This is achieved by making the representation of each span directly influenced by its overlapping counterparts, encouraging the model to favor non-overlapping predictions during inference

Exact inference We then introduce an exact inference algorithm for span-based NER, aiming to select the set of non-overlapping spans that cumulatively maximize classification scores. This algorithm uses a dynamic programming approach to identify the optimal configuration of entities that adhere to the non-overlap constraints inherent in flat NER tasks by computing the maximum independent weight set (MWIS) in the interval graphs formed by the spans.

Global-aware training To further refine our approach, we integrate a global-aware training regimen that focuses on optimizing span selection during the model's training phase. In this scenario, each span's classification is optimized using a cross-entropy loss. Additionally, the selection of the final set of spans is optimized using a globally normalized training objective. By treating span selection as a dynamic programming problem, we can efficiently calculate the maximum weight independent set (MWIS) in interval graphs, which corresponds to the optimal selection of non-overlapping spans. This method not only enhances the model's ability to understand and respect span boundaries but also significantly improves performance by considering the global interaction between spans, as opposed to traditional local span classification objectives. Our training process is thereby aligned more closely with the actual inference conditions, where the selection of spans is influenced by the entire text structure rather than isolated predictions.

1.3.2 Generalist and Lightweight model for Named Entity Recognition

Large language models have gained a lot of popularity and have been applied to many tasks, and information extraction is not an exception. An advantage of LLMs is that they are highly flexible and can address a task by natural language instruction. For instance, they have been applied to the task of named entity recognition and show promising results in zero-shot scenarios across many domains and types of entities. However, their size makes them hard to deploy in low-resource setups. Moreover, they can pose some privacy problems as powerful LLMs are accessible only via external APIs. To address this problem, we propose GLiNER, a NER model that has strong generalization but uses a small bidirectional transformer encoder.

1.3.3 Joint Entity an relation extraction

The second part of my doctoral study focuses on designing a novel and scalable span-based architecture for joint entity and relation extraction. We first tackle the task as graph generation and then as constrained graph prediction.

Auto-regressive Text-to-Graph model Our first contribution to joint information extraction (IE) was to treat the task as graph generation. We were motivated by the increasing popularity of auto-regressive language models like GPT-3, which, despite their success, struggle to generate coherent structures for information extraction. In this work, we propose ATG, a model that generates spans using a pointer mechanism rather than generating textual outputs. Generating spans instead of tokens offers several benefits, including richer representation, span-level attention, the ability for spans to interact with the input sequence through cross-attention, and shorter sequences compared to generating textual tokens. Moreover, the ATG framework allows our model to be easily controlled to produce valid output graphs, addressing the hallucination problems often seen in language model-based approaches.

Enriched Representation for IE While ATG produced state-of-the-art performance, its decoding process is relatively slow due to the auto-regressive sequential nature. For this model, which we term EnRiCO, we retain many characteristics of ATG, such as span-level attention. However, EnRiCO introduces additional advantages, including span-pair interaction (i.e., relation level) and non-sequential prediction. Furthermore, EnRiCO integrates domain-specific constraints for decoding, implemented using ASP language, allowing for more coherent predictions.

IE as graph structure learning Our final contribution for joint entity and relation extraction is GraphER, which aims at exploiting the graph nature of the task. This formulation allows for structure-informed decisions for entity and relation prediction, in contrast to previous models that have untied predictions for these tasks. We found that using the transformer architecture for learning node and edge representation is highly effective compared to using specialized graph neural networks [Kipf et al., 2018, Veličković et al., 2018]. Specifically, we note that GNNs are not robust when the graph structure is noisy (missing nodes/edges).

1.4 Overview of the Structure

Chapter 2 provides a comprehensive review of the literature on information extraction, focusing particularly on named entity recognition and relation extraction. This review includes concise summaries of prominent methods developed over the years, detailing their architecture, objective functions, and inference algorithms.

Chapters 3, 4, and 5 discuss our work on span-based named entity recognition, presenting our findings and methodologies.

Chapter 6 introduces GLiNER, our novel model for named entity recognition. GLiNER utilizes a lightweight bidirectional transformer encoder to extract any entity, showcasing flexibility and efficiency in modern NER tasks.

Chapter 7 is dedicated to our research on generative information extraction, exploring innovative approaches and their implications.

In Chapters 8 and 9, I describe EnriCo and GraphER, two efficient models for relation extraction that demonstrate significant advancements in the field.

1.5 List of publications

1.5.1 Conference publications

1. Urchade Zaratiana, Nadi Tomeh, Pierre Holat, Thierry Charnois, GLiNER: Generalist Model for Named Entity Recognition using Bidirectional Transformer, *NAACL 2024.*

2. Urchade Zaratiana, Nadi Tomeh, Pierre Holat, Thierry Charnois, An Autoregressive Text-to-Graph Framework for Joint Entity and Relation Extraction, *AAAI 2024*.

3. Urchade Zaratiana, Nadi Tomeh, Niama El Khbir, Pierre Holat, Thierry Charnois, Filtered Semi-Markov CRF, *Findings of EMNLP 2023*.

1.5.2 Preprints

1. Urchade Zaratiana, Nadi Tomeh, Niama El Khbir, Pierre Holat, Thierry Charnois, GraphER: A Structure-aware Text-to-Graph Model for Entity and Relation Extraction , *Arxiv 2024*.

2. Urchade Zaratiana, Nadi Tomeh, Yann dauxais, Pierre Holat, Thierry Charnois, EnriCo: Enriched Representation and Globally Constrained Inference for Entity and Relation Extraction , *Arxiv 2024*.

1.5.3 Workshop publications

1. Urchade Zaratiana, Nadi Tomeh, Pierre Holat, Thierry Charnois, GNNer: Reducing Overlapping in Span-based NER Using Graph Neural Networks, *ACL Student Research Workshop 2022*.

2. Urchade Zaratiana, Nadi Tomeh, Pierre Holat, Thierry Charnois, Named Entity Recognition as Structured Span Prediction , *UM-IoS 2022*.

3. Urchade Zaratiana, Nadi Tomeh, Niama El Khbir, Pierre Holat, Thierry Charnois, Global Span Selection for Named Entity Recognition , *UM-IoS 2022*.

4. Niama El Khbir, Urchade Zaratiana, Nadi Tomeh, Thierry Charnois, Cross-Dialectal Named Entity Recognition in Arabic , *ArabicNLP 2023*.

5. Niama El Khbir, Urchade Zaratiana, Nadi Tomeh, Thierry Charnois, LIPN at Wojood-NER shared task: A Span-Based Approach for Flat and Nested Arabic Named Entity Recognition , *ArabicNLP 2023*.

Chapter 2

Background

2.1 Introduction

Information extraction (IE) is the process of automatically extracting structural information from unstructured or semi-structured data [Okurowski, 1993, Gaizauskas and Wilks, 1998]. This field plays a pivotal role in transforming raw data into meaningful information that can be analyzed and utilized across various applications. Its application inludes enhancing search engine [Gaizauskas and Wilks, 1998], powering recommendation systems, the automated construction of knowledge bases [Melnyk et al., 2022], question answering [Jijkoun et al., 2004, Ding et al., 2008, Mollá et al., 2006, Ng et al., 2001] and opinion mining [Popescu et al., 2005, Popescu and Etzioni, 2005]. At its core, Information Extraction (IE) encompasses several key tasks that enable the parsing and structuring of data from textual content. These tasks include entity recognition, relation extraction, event extraction, entity linking and coreference resolution [van Deemter and Kibble, 1999], each playing a crucial role in understanding and organizing information. We provide a brief description of these task in the following.

Entity Recognition This essential task in natural language processing involves the detection and classification of entities within textual data, which are typically names of people, organizations, locations, dates, monetary values, percentages, and other specific types of information [Chinchor, 1998, Zhou and Su, 2002].

Relation Extraction Following the identification of entities, relation extraction plays a pivotal role in interpreting the semantic associations between entities [Zelenko et al., 2002b, Kambhatla, 2004]. This process seeks to discern and categorize the relationships or interactions between identified entities based on their context within the text. For example, in the sentence "Alice lives in Paris," relation extraction would not only identify "Alice" and "Paris" as entities but also classify the "lives in" relationship that connects them.

Event Extraction This complex task involves recognizing and deciphering events that are described in text. An event can be understood as a significant occurrence involving multiple entities and expressed by one or more actions. For instance, an "election" event might encompass entities such as candidates, dates, and locations. Event extraction aims to not only identify these components and their roles but also to classify the type of event and its structural properties.

Entity Linking Entity linking connects identified entities in text to their corresponding entries in a knowledge base, like Wikipedia, by disambiguating them based on context [Bunescu and Paşca, 2006, Cucerzan, 2007]. For example, it determines whether "London" refers to the city or a person, enriching the text with structured knowledge.

Coreference Resolution Coreference resolution [van Deemter and Kibble, 1999, Soon et al., 2001] is critical for achieving text coherence and understanding narrative flow. This task aims to identify all expressions in a text that refer to the same real-world entity, thus linking pronouns and other referring expressions (like "the president" or "he") to the appropriate entities they denote.

Focus of This Thesis: This thesis specifically concentrates on advancing methodologies for entity (NER) and relation extraction (RE), as the primary objective is to build a knowledge graph for a specific domain, and NER and RE are the most important tasks for that.

2.2 Named Entity Recognition

Named Entity Recognition (NER) is a critical task in the field of natural language processing that involves identifying noun phrases in unstructured text that represent entities. Typically, these entities span one to several tokens and include named entities, which are specifically recognized and categorized during various notable competitions. Originally focused on conventional categories like persons, locations, and organizations [Carreras and Màrquez, 2004], the scope of entities recognized under NER has broadened significantly. Modern applications of NER now include entities such as disease names and protein names [Collier et al., 2004, Tsuruoka and Tsujii, 2003, Niu and Hirst, 2004]. NER methodologies have evolved significantly, transitioning from rule-based approaches [Petasis et al., 2001] to neural and generative models. In particular, the recent adoption of Large Language Models marks a notable advancement in this field, a topic we will delve into further in the following section.

2.2.1 Rule-based Methods

Rule-based methods for Named Entity Recognition (NER) rely on lexicon resources and domain-specific knowledge to identify entities within text [Appelt et al., 1993, Cunningham

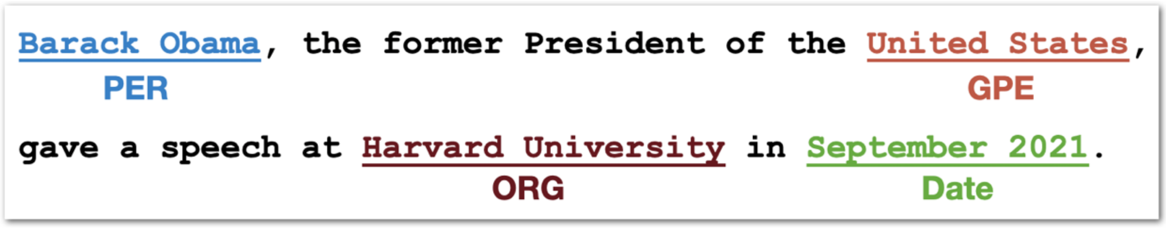

Barack Obama, the former President of the United States,
PER
GPE
gave a speech at Harvard University in September 2021.
ORG
Date

Figure 2.1: Task of Named Entity Recognition (NER).

et al., 2002]. These approaches operate by employing a collection of rules, which are either hand-coded by domain experts [Maloney and Niv, 1998, Tsuruoka and Tsujii, 2003]. Rule-based systems are particularly effective when the task is controlled and well-behaved, such as identifying structured entities like phone numbers, zip codes, or email addresses. One of the key advantages of rule-based systems is their speed and simplicity. Since they operate based on predefined rules, the processing time is often faster compared to more complex machine learning models [Reiss et al., 2008]. However, rule-based methods also have their limitations. They may struggle with ambiguous or irregular entities that do not conform to predefined patterns, making them less suitable for tasks where the context is highly variable or the entities are less structured. Additionally, maintaining and updating rule sets can be labor-intensive, requiring continuous effort to adapt to changes in language usage or domain-specific terminology. Despite these limitations, rule-based NER systems remain a valuable tool, particularly in scenarios where precision and speed are paramount, and where the entities of interest follow clear and consistent patterns.

2.2.2 Statistical Models

Statistical methods for Named Entity Recognition (NER) leverage statistical inference to predict entity labels. These methods are broadly divided into token-level and span-level models, each employing distinct approaches and features for entity recognition. We discuss the methodologies and specific models used within these categories.

a) Token-Level Models

In token-level models, the text is treated as a sequence of tokens (words or sub-words), and the task involves assigning an entity label to each token. This approach primarily uses the BIO tagging scheme [Lafferty et al., 2001] to mark the beginning (B), inside (I), and outside (O) of entities. This granularity allows the model to capture the boundaries of entities at the token level.

Feature Engineering: Effective feature engineering is crucial for the performance of token-level models. Features typically include token case, part-of-speech tags, chunking information, context words, and lexical features such as prefixes and suffixes. These features help the model to identify patterns and attributes indicative of named entities. Here are some examples used in prior works [Chieu and Ng, 2002, Lafferty et al., 2001, Zhou and Su, 2002]:

- **Token Case:** The capitalization of a token often provides significant clues, especially in languages like English where proper nouns (usually named entities) are capitalized.
- **Part-of-Speech Tags:** Identifying whether a token is a noun, verb, adjective, etc., can help in distinguishing named entities, particularly nouns and proper nouns.
- **Chunking Information:** This involves grouping tokens into "chunks" of phrases, which can be useful to determine if a token belongs to a noun phrase, often a characteristic of named entities.
- **Context Words:** The tokens immediately before and after a given token can provide context that hints at whether the token is part of a named entity. For example, 'President' before a name indicates a likely personal name entity.
- **Lexical Features:** These include prefixes (e.g., "un-", "pre-", "post-") and suffixes (e.g., "-tion", "-ing", "-ly") that help in understanding the role and meaning of tokens, though they are more indicative of common nouns or verbs than named entities.
- **Orthographic Features:** Patterns in the way tokens are written, such as the use of all-caps (e.g., USA), use of digits (e.g., 3M), or mixed alphanumeric tokens (e.g., iPhone5), can signal named entities.
- **Syntactic Dependencies:** The grammatical dependencies between tokens can indicate entity boundaries. For instance, the dependency relation might suggest that a token is an attribute or modifier of a potential named entity.
- **Semantic Features:** Information from semantic resources like WordNet or domain-specific ontologies can help in associating a token with potential entity types based on its meaning.

In the following is a list of the representative models from the initial stages through to the adoption of statistical NER techniques:

- **Maximum Entropy Model (MaxEnt):** The Maximum Entropy model have also been popular in named entity recognition [Chieu and Ng, 2002, 2003, Bender et al., 2003, Tsai et al., 2005]. It estimates the probability distribution of the possible labels for a token, based on the constraint that the expected value of the feature functions should match the empirical average:

  $$P(l \mid O) = \frac{1}{Z(O)} \exp\left(\sum_j \lambda_j f_j(l, O)\right)$$

  where l is the label for a token, O represents the observed features for the token, f_j are feature functions, λ_j are the parameters to be learned, and $Z(O)$ is a normalization factor ensuring that the probabilities sum to one.

- **Hidden Markov Model (HMM):** HMMs model the sequence of tokens as a Markov process with hidden states, which correspond to entity labels [Zhou and Su, 2002, Zhao, 2004, Liu et al., 2005]. These models calculate the probabilities of label sequences given the observed sequence of tokens, using known statistics about entity transitions and emissions. The probability of a label sequence given a sequence of tokens is given by:

  $$P(L \mid O) = \prod_{i=1}^{n} P(l_i \mid l_{i-1}) P(o_i \mid l_i)$$

  where L is the sequence of labels, O is the sequence of observations (tokens), and $P(l_i \mid l_{i-1})$ and $P(o_i \mid l_i)$ are the transition and emission probabilities, respectively.

- **Conditional Random Fields (CRFs):** CRFs [Lafferty et al., 2001] are particularly popular for NER [Altun et al., 2003, Collier et al., 2004, Peng and McCallum, 2004] as they model the conditional probability of a label sequence given a token sequence, directly optimizing the tag sequence globally. This avoids the label bias problem inherent in models like HMMs and is flexible in incorporating a wide range of input features. CRFs calculate the conditional probability of a sequence of labels given a sequence of tokens, without making independence assumptions:

  $$P(L \mid O) = \frac{1}{Z(O)} \exp\left(\sum_{i,j} \lambda_j f_j(l_{i-1}, l_i, O, i) \right)$$

  where $Z(O)$ is the normalization factor, f_j are feature functions, and λ_j are the learned weights.

b) Span-Level Models

Span-level models consider features defined over segments of text that comprise multiple tokens, which together form an entire entity [Sarawagi and Cohen, 2005]. These models are adept at capturing information about the entire entity, rather than individual tokens within it. Here are example of features used when dealing with span-level NER [Sarawagi and Cohen, 2005, Sarawagi, 2008]:

- **Similarity to an Entity in the Database:** This feature measures how closely a token span matches known entity names in a structured database, aiding in the recognition of named entities based on historical data.

- **Length of the Entity:** Often, the length of a token span (in terms of the number of tokens or characters) can be indicative of whether it is likely to be a named entity.

- **Part-of-Speech Tags:** The grammatical category of each word within a token span can provide significant cues about its likelihood of being a named entity. For example, proper nouns and capitalised nouns are more likely to be named entities.

- **Capitalization Pattern:** Observing the capitalization in token spans is crucial; named entities often start with capital letters in certain contexts (e.g., at the beginning of sentences or in titles).

- **Contextual Keywords:** The presence of certain keywords or phrases immediately before or after a token span can suggest its entity type. For example, words like "Mr.", "Company", "Inc." suggest personal names or organizations respectively.

- **Entity Consistency Across a Document:** Entities that appear multiple times in a document can be cross-verified for consistency in recognition, leveraging co-reference and context usage throughout the text.

- **Morphological Features:** This includes prefixes, suffixes, and other derivational forms of words that might indicate an entity. For example, suffixes like "-corp" or "-inc" can identify corporations.

- **Dependency Parse Structure:** Analyzing the grammatical relationships between words in a sentence, like who is doing what to whom, can help identify entities based on their roles within those structures.

Traditionnal model for span-based NER include the popular Semi-Markov Conditional Random Fields (Semi-CRFs) [Sarawagi and Cohen, 2005] which extend the CRF framework by allowing each label to be associated with segments of varying lengths, rather than single tokens. This approach is particularly useful for span-level NER as it directly models the entity spans. The probability model for a Semi-CRF is given by:

$$P(S \mid O) = \frac{1}{Z(O)} \exp\left(\sum_{t=1}^{T}\sum_{k=1}^{K} \lambda_k g_k(s_t, o_{t:t+s_t.length-1})\right)$$

where S is the sequence of segments, s_t represents the segment at position t, $o_{t:t+s_t.length-1}$ are the tokens in the segment, g_k are segment-level feature functions, and λ_k are the weights.

c) Training and Inference

Training and inference are two crucial phases in the lifecycle of any statistical model used for Named Entity Recognition (NER). These phases are designed to ensure that the models not only learn from the training data but also generalize well to new, unseen data during inference.

Training Training a statistical NER model involves optimizing the model parameters to best fit the training data. The goal is to minimize the difference between the predicted entity labels and the actual labels in the training set, a process often referred to as error minimization. Depending on the model type (e.g., HMM, CRF), various techniques are used to estimate the parameters. For example, CRFs typically use iterative algorithms like the L-BFGS [Lafferty et al., 2001] or Stochastic Gradient Descent [van der Maaten et al., 2011] to find the best parameter values that maximize.

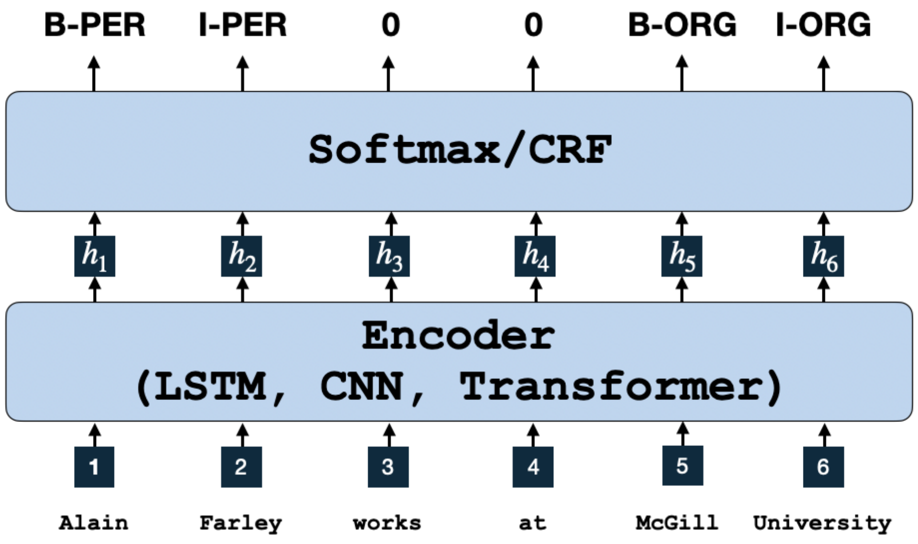


Figure 2.2: Named entity recognition with sequence labelling

Inference Inference, or prediction, involves applying the trained NER model to new, unseen texts to detect and classify named entities. The efficiency and accuracy of inference are pivotal for the practical deployment of NER systems. For models like HMMs and CRFs, a decoding process is used to find the most likely label sequence for a given input sequence. Techniques such as the Viterbi algorithm are typically employed.

2.2.3 Neural Representation Learning for NER

Named Entity Recognition (NER) has undergone a significant transformation with the advent of deep learning, shifting from traditional methods that rely on manual feature engineering to automated, end-to-end feature learning methodologies. These methodologies enable models to learn rich, hierarchical representations of text data, optimized through advanced optimization techniques like gradient descent and backpropagation.

a) Sequence labelling approaches Early approaches in the field of NER employed various neural network architectures [Wang and Manning, 2013, Gupta et al., 2018], which have evolved to process text data through several computational steps:

1. **Tokenization**: The raw text is segmented into a sequence of discrete units (tokens), typically words or subwords, represented as $x_1, x_2, \ldots, x_n$.
2. **Word Embedding**: Each token x_i is mapped to a dense vector e_i via an embedding matrix E, forming the basic feature representation for subsequent layers:
$$e_i = E[x_i]$$
3. **Contextual Representation**: The embeddings are then processed through a generic deep learning architecture f, which captures contextual dependencies and nuances

in the text:

$$h_0, h_1, \ldots, h_N = f(e_0, e_1, \ldots, e_N; \theta)$$

Here, f could be any neural architecture like LSTM or CNN, designed to model both the sequential and spatial relationships among the embeddings. More recently, this has evolved into more sofisticated representation based on pretrained transformer encoders such as BERT.

4. **Output Layer**: The contextual representations h_i output from f are used to compute label scores for each token. Depending on the modeling choice, various approaches can be applied:

   - **Probabilistic Softmax**: The softmax function is used to convert the raw scores into a probability distribution over the label set:

     $$p(y_i|h_i) = \text{softmax}(Wh_i + b)$$

     where W and b are the parameters of the output layer.

   - **Max Margin**: This approach focuses on maximizing the margin between the correct label and other possible labels, effectively enhancing classification robustness:

     $$\max(0, 1 - s_{y_i} + \max_{y \neq y_i} s_y)$$

     where s_y are the scores for each label computed from h_i.

   - **Conditional Random Field (CRF)**: A CRF layer models the dependencies between labels in a sequence, enhancing the prediction accuracy by taking into account the neighboring label information:

     $$p(\mathbf{y}|\mathbf{h}) = \frac{\exp(\sum_{i=1}^{n} \Psi(y_{i-1}, y_i, h_i))}{\sum_{\mathbf{y}'} \exp(\sum_{i=1}^{n} \Psi(y'_{i-1}, y'_i, h_i))}$$

     where Ψ is the potential function defining the transition scores from one label to another, learned during training.

To train neural models effectively for Named Entity Recognition (NER), the choice of an appropriate loss function is crucial, as it directly impacts the model's ability to learn accurate representations and make correct predictions. Different output layers and modeling choices require specific loss functions tailored to their characteristics:

b) Span-Based Approaches Span-based models in NER provide a sophisticated method for identifying entities by focusing on continuous spans of text rather than individual tokens [Luan et al., 2019a, Fu et al., 2021]. These models follow the same initial steps as token-based models (tokenization, embedding, and contextual representation) and then introduce additional processes to handle spans:

1. **Tokenization, Embedding, and Contextual Representation**: As previously described, the text is tokenized, each token is embedded into a vector space, and contextual representations $h_0, h_1, \ldots, h_N$ are computed using a neural architecture f.

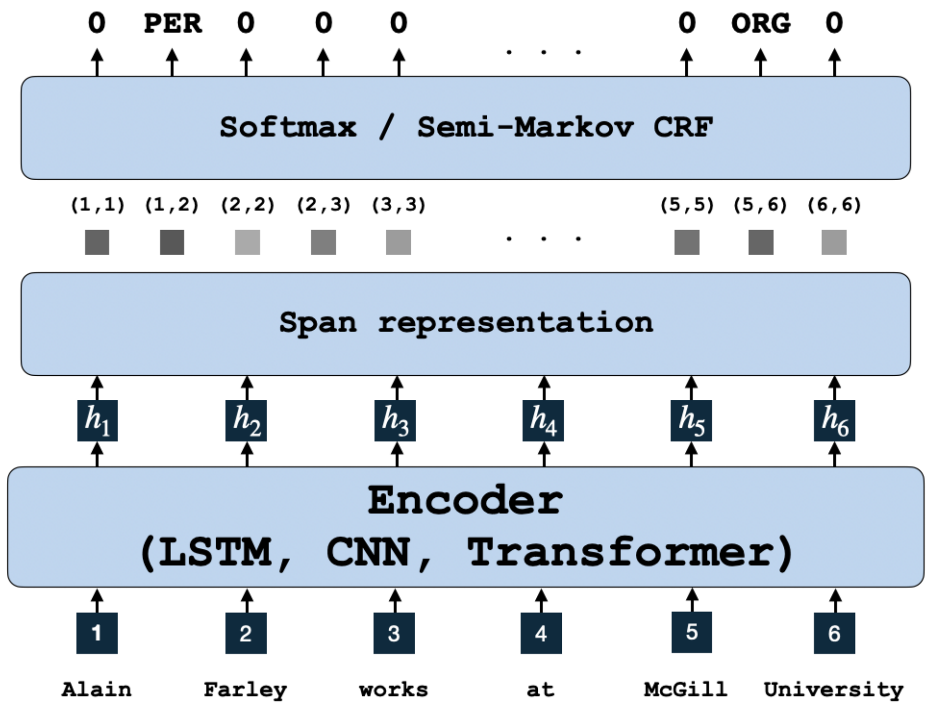


Figure 2.3: Architecture for span-based Named Entity Recognition

2. **Span Representation**: Each possible span in the text, defined by a start index i and an end index j, is represented by a function of the embeddings or contextual representations of the tokens within the span. A common approach is to use a combination of the boundary token representations and a transformation of the entire span:

$$r_{i,j} = g(h_i, \dots, h_j)$$

where g is a neural function that could be any function such as sum, mean, convolution, concatenation or a neural network layer itself designed to capture the properties of the span.

3. **Span Scoring**: Once spans are represented, each span (i, j) is scored for each possible entity type k. This is typically done using a fully connected layer with parameters specifically trained to score spans:

$$s^k_{i,j} = W_k r_{i,j} + b_k$$

where W_k and b_k are parameters specific to entity type k, enabling the model to differentiate between different types of entities.

4. **Output Layer**: The final decision for each span is made either through a softmax layer, which normalizes the scores across all entity types, or using a Semi-CRF layer, which considers both the scores and the transitions between entity labels in adjacent spans:

$$p(y^k_{i,j}|r_{i,j}) = \text{softmax}(s^k_{i,j})$$

or, for a Semi-CRF [Sarawagi and Cohen, 2005],

$$p(\mathbf{y}|\mathbf{r}) = \frac{\exp(\sum_{(i,j)} \Psi(y_{i-1,j-1}, y_{i,j}, r_{i,j}))}{\sum_{\mathbf{y}'} \exp(\sum_{(i,j)} \Psi(y'_{i-1,j-1}, y'_{i,j}, r_{i,j}))}$$

where Ψ includes the transition scores from one entity type to another and is trained to capture label dependencies.

Furthermore, unlike sequence labeling approaches, unconstrained span-based NER models inherently face the challenge of overlapping entity predictions. This can lead to contradictory or redundant entity identifications within the same text span. While Semi-CRF address this issue using the segmental virtbi decoding, the local span-based NER needs further processing to address the issue of overlapping spans. To address this issue, the literature commonly adopts a two-step decoding process, which enhances the precision of entity recognition while ensuring that entity spans do not overlap. Initially, a set of candidate entities is created where the maximum label is not null. That is, for each span (i, j), the entity type k with the highest score among non-null labels is determined:

$$\mathcal{K} = \{\arg\max_{k \neq \text{null}} (W_k r_{i,j} + b_k)\}$$

Then, each span is assigned a score based on the maximum score among the selected candidate entities:

$$s_{i,j} = \max_{k \in \mathcal{K}} (W_k r_{i,j} + b_k)$$

This process ensures that only non-null entity types are considered for scoring spans. To ensure that the predicted spans do not overlap, a greedy algorithm is applied. This algorithm iteratively selects spans based on their scores, prioritizing those with the highest scores first. For each selected span, any overlapping spans are removed from consideration. This is done using the following iterative process:

Algorithm 1: Greedy Span Selection Algorithm

```
Data: Predicted span $\mathcal{K}$
Result: Selected non-overlapping spans
Sort all spans $(i, j) \in \mathcal{K}$ by their score in descending order;
Initialize an empty set $S$ to store selected spans;
foreach span $(i, j)$ do
    if span $(i, j)$ does not overlap with any selected span $(i', j') \in S$ then
        Add span $(i, j)$ to set $S$;
    end
end
```

This greedy selection process ensures that the final set of spans are non-overlapping, maximizing the total score of the selected spans under the constraint of mutual exclusivity.

b) Biaffine Model

Originally developed for parsing models [Dozat and Manning, 2017, Attardi et al., 2021, Candito, 2022, Floquet et al., 2023], biaffine layers have also become a popular choice for named entity recognition [Yu et al., 2020c, Zhu and Li, 2022]. These layers efficiently compute span scores without explicitly constructing span representations, which can be memory-intensive. Instead, biaffine layers utilize a biaffine operation to directly compute the scores for each span. In biaffine models, the span score $s^k_{i,j}$ for each entity type k is calculated using a biaffine operation as follows:

$$s^k_{i,j} = (h_i^T W_k h_j) + (U_k^T h_i) + (V_k^T h_j) + b_k$$

where:

- h_i and h_j are the contextual representations of the start and end tokens of the span (i, j), respectively. These representations are typically derived from a deep neural network, such as a Transformer or LSTM, which processes the entire input sequence to capture the contextual nuances of each token.
- W_k, U_k, and V_k are matrices of learnable parameters that are specific to each entity type k. These matrices are crucial for the model as they enable it to learn distinct scoring patterns for different types of entities:
  - W_k captures the interaction between the start and end tokens of a span, effectively determining how these tokens combine to form a potential entity.
  - U_k and V_k separately weigh the importance of the start and end tokens, respectively, providing a mechanism to assess the relevance of each token independently within the context of a particular entity type.
- b_k is a learned bias term for entity type k, which adjusts the score based on the each entity type.

This operation directly computes the score for each entity type k for the span (i, j), considering the interactions between the start and end tokens as well as their individual representations.

2.2.4 Generative Models for Named Entity Recognition

Recent advancements have highlighted the significance of generative models, such as ChatGPT, which have gained considerable popularity in various computational linguistics tasks, including Named Entity Recognition (NER). These models typically employ mechanisms like encoder-decoder frameworks, text generation with pointers, and set generation methods.

Encoder-Decoder Models In the context of NER, encoder-decoder models conceptualize the task as a translation problem where input text is transformed into a structured format that linearly represents entities. The operational mechanics of these models are as follows:

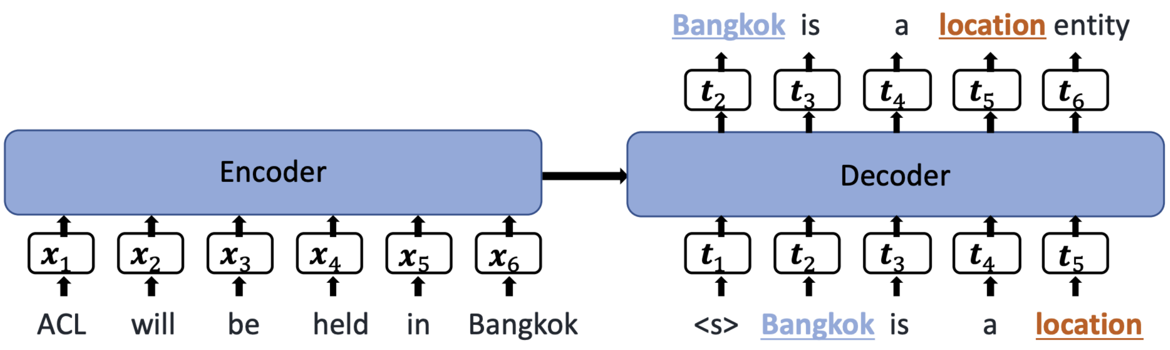

Bangkok is a location entity
t_2 t_3 t_4 t_5 t_6
Encoder
Decoder
x_1 x_2 x_3 x_4 x_5 x_6
ACL will be held in Bangkok
t_1 t_2 t_3 t_4 t_5
<s> Bangkok is a location

Figure 2.4: Named Entity Recognition with an Encoder-Decoder architecture. Figure taken from [Cui et al., 2021b]

- Input token sequence $x_1, x_2, \ldots, x_N$ is encoded into a latent space using an encoder function enc, which generates contextual embeddings $h_1, h_2, \ldots, h_N$:

$$\mathbf{h} = \text{enc}(x_1, x_2, \ldots, x_N)$$

- The contextual embeddings are then processed by a decoder function dec to sequentially produce an output sequence $t_1, t_2, \ldots, t_M$, which represents entities in a linearized format:

$$t_i = \text{dec}(h_1, h_2, \ldots, h_N; t_{1\ldots i-1}), \quad \text{for } i = 1 \text{ to } M$$

These encoder-decoder models are typically trained using the teacher forcing technique, where the correct output is always fed back into the model during training, regardless of the model's predictions. During inference, the models might employ various strategies such as greedy decoding, beam search, or sampling methods (e.g., ancestral sampling, top-k sampling, or nucleus sampling). Three notable implementations of the encoder-decoder framework in NER include:

- **Seq2seq NER:** This method, introduced by [Straková et al., 2019], pioneered the autoregressive approach for Named Entity Recognition (NER). It utilizes an encoder-decoder architecture based on Long Short-Term Memory (LSTM) networks. The encoder generates a contextualized representation of the input sequence, while the decoder is specifically trained to sequentially output entity BILOU tags. This model leverages the sequential nature of text to efficiently predict the structured tags that denote entity boundaries and types in a coherent manner.

- **BartNER:** This approach, as discussed in [Yan et al., 2021a], inputs a sentence and generates a sequence of entity pointer indices on the target side. This model is effective in addressing flat, continuous, and discontinuous NER tasks in a unified manner by leveraging a pretrained BART model.

- **Template-BART:** In contrast to BartNER, which produces pointer indices, Template-BART, as detailed in [Cui et al., 2021b], is trained to translate input text into a predefined natural language template. This method benefits from the text generation capabilities learned during the pretraining of the BART model.

```
Please identify Organization, Person, Location, and
Miscellaneous Entity from the given text, output using the
format as:

Entity: Organization: None | Person: None | Location:
Word1, Word2 | Miscellaneous: Word3

Text: {text}

Entity:
```

Figure 2.5: Prompting a large language model for Named Entity Recognition.

Decoder-Only Approaches Recent trends in Natural Language Processing (NLP) have seen a shift towards decoder-only models, particularly fueled by the widespread adoption of large pre-trained language models like GPT-4. These models for Named Entity Recognition (NER) typically utilize a prompting paradigm, where the task is reformulated as a natural language prompt. This approach enables zero-shot learning, where the model makes predictions based solely on its pre-trained knowledge without any task-specific training. However, encoder-only models can also be adapted to a fine-tuning paradigm using labeled NER datasets. In fine-tuning, the model is further trained on a task-specific dataset, enhancing its ability to precisely identify and classify named entities according to the dataset's annotations. Here are some representative models:

- **PromptNER:** This prompting-based NER model [Ashok and Lipton, 2023b] achieves state-of-the-art results in Few-Shot and Cross-Domain NER. The approach is built around four key components: a backbone Large Language Model (LLM), a modular definition of entity types documented for clarity, a set of example entities from the target domain, and a specific output format. This format is essential for instructing the model on how to present the extracted entities, implemented through the structure of the few-shot examples. To adapt to new domains, only the entity definitions and the examples need to be adjusted. Performance is enhanced by incorporating a chain of thought prompting.

- **GPT-NER:** This method by Wang et al. [2023a] introduces a prompting-based approach using GPT for NER tasks. The proposed prompt consists of two main components: a Task Description and Few-shot Demonstrations. To facilitate these demonstrations, a datastore of sentence-entity pairs is constructed. Entities are retrieved using a K-Nearest Neighbors (KNN) search, leveraging the datastore to provide relevant context and examples for the model during inference.

- **UniNER:** This approach by Zhou et al. [2024] involves fine-tuning an open-source large language model, such as LLaMa, using synthetic data generated by GPT-3.5. Results indicate strong performance on out-of-domain NER datasets, even surpassing the original capabilities of the teacher model, GPT-3.5, highlighting the effectiveness of synthetic training enhancements.

- **GoLLie:** Similar to UniNER, this model [Sainz et al., 2024] involves fine-tuning another large language model, LLaMa, but with a focus on detailed annotation guidelines for information extraction. When evaluated against standard benchmarks, GoL-Lie shows robust results, outperforming UniNER and demonstrating the value of guideline adherence in training.

2.2.5 Datasets

This section outlines the datasets employed in our study, detailing their source, size, entity types, and domain-specific relevance. Each dataset has been chosen for its particular characteristics, which are suited to testing the versatility and accuracy of NER systems across varied contexts and languages.

- **ACE05** Walker et al. [2006b] — Comprises 7,299 training, 971 development, and 1,060 test documents, with 7 entity types. It spans across several domains including news and conversational telephone speech, facilitating comprehensive entity recognition tasks involving persons, locations, and organizations.

- **AnatEM** Pyysalo and Ananiadou [2014] — Consists of 5,861 training, 2,118 development, and 3,830 test documents. This biomedical dataset is specifically annotated for anatomical entities, crucial for detailed biomedical research and applications.

- **bc2gm** Smith et al. [2008] — A gene mention dataset with 12,500 training, 2,500 development, and 5,000 test documents. It targets gene and protein entities, supporting genetic research through biomedical text mining.

- **bc4chemd** Krallinger et al. [2015] — Focuses on chemical compounds and drugs with 30,682 training, 30,639 development, and 26,364 test documents, serving as a fundamental resource for chemical entity recognition in biomedical texts.

- **bc5cdr** Li et al. [2016] — Includes 4,560 training, 4,581 development, and 4,797 test documents, annotated for chemical and disease entities. This dataset aids in linking chemical compounds to diseases, enhancing drug discovery and epidemiological studies.

- **Broad Tweet Corpus** Derczynski et al. [2016] — Contains 5,334 training, 2,001 development, and 2,000 test tweets, annotated for common entities like persons, locations, and organizations. It addresses the challenges of informal language in social media.

- **conll 03** Tjong Kim Sang and De Meulder [2003] — A well-known benchmark dataset in NER with 14,041 training, 3,250 development, and 3,453 test documents. It covers news text in English and German, focusing on entities such as persons, organizations, and locations.

- **GENIA** Kim et al. [2003] — Comprises 15,023 training, 1,669 development, and 1,854 test documents. This dataset is specific to the domain of molecular biology, annotated for entities like genes, proteins, and cells.

- **Ontonotes** Weischedel et al. [2013] — A large and diverse dataset with 59,924 training, 8,528 development, and 8,262 test documents. It includes a wide range of entities across different layers of linguistic annotation, used in news, broadcast, and web texts.
- **WikiANN** (PAN-X) Pan et al. [2017] — A multilingual dataset derived from Wikipedia, comprising 20,000 training, 10,000 development, and 10,000 test documents. It is designed for cross-lingual name tagging and includes common entities like person, location, and organization across more than 40 languages.
- **FindVehicle** Guan et al. [2023] — Designed for vehicle-related entity recognition with 21,565 training, 20,777 development, and 20,777 test documents. It features 21 entity types, making it ideal for applications in transportation and mobility.
- **CrossNER** Liu et al. [2020b] — A collection of domain-specific datasets with varying sizes, focusing on entities related to domains like AI, literature, music, politics, and science. Each subset is tailored to extract domain-specific knowledge effectively.
- **TDM** is a NER dataset that was recently published and it was designed for extracting Tasks, Datasets, and Metrics entities from Natural Language Processing papers.

These datasets collectively provide a robust framework for evaluating the effectiveness of NER systems across a spectrum of domains, ensuring a comprehensive assessment of performance and scalability.

2.2.6 Evaluation Metrics for Named Entity Recognition

To assess the performance of our Named Entity Recognition (NER) models, we employ span-level precision, recall, and F1 score. These metrics are crucial for evaluating how well the model identifies and classifies entities, considering both the boundaries (span) and types of entities. Here, we detail each of these metrics and their importance in NER tasks.

Precision Precision, also known as the *positive predictive value*, quantifies the accuracy of a model in identifying relevant entities. It is defined as the ratio of the number of correctly predicted positive entities (true positives) to the total number of entities predicted by the model (including both correct and incorrect predictions). The formula for precision is expressed as follows:

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}$$

where:

- TP (True Positives): This metric represents the number of entities correctly identified by the model as entities. Specifically, a true positive is recorded each time the model accurately predicts the start, end, and type of an entity that matches exactly with the corresponding entity in the ground truth data.

- FP (False Positives): This metric counts the instances where the model incorrectly identifies a segment of text as an entity when, in fact, it does not correspond to any entity in the ground truth annotations. False positives can occur due to the model misclassifying the text or recognizing an entity's boundaries incorrectly.

This metric ensures that the entities predicted by the model are not only correct but also precise, emphasizing both the accuracy of entity boundaries and their classification. Precision thereby measures the model's ability to minimize the error of falsely labeling non-entity text spans as entities.

Recall Recall, also known as *sensitivity* or *true positive rate*, measures the model's ability to identify all relevant instances within a dataset. This metric is particularly important in scenarios where missing a positive instance (such as failing to detect a disease in medical diagnostics) has serious consequences. The formula for recall is:

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}$$

where:

- FN (False Negatives): This metric refers to the occurrences where the model fails to recognize an actual entity present in the ground truth data. False negatives can result from the model's inability to detect the entity at all or its failure to recognize the entity within the correct boundaries or category. Reducing false negatives is essential for improving the model's recall, ensuring that it does not miss any entities.

Recall thus measures the completeness with which the model can capture all relevant entities.

F1 Score The F1 score is the harmonic mean of precision and recall. It is a balanced metric that considers both precision and recall, providing a single score to measure the overall effectiveness of the model. This metric is particularly useful when you want to find a balance between precision and recall, and it is calculated as follows:

$$\text{F1 Score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

The F1 score is especially valuable in NER, as it ensures that both the precision (correct identification and classification of entities) and recall (comprehensive coverage of all entities) are maintained at high levels. This balance is critical because an NER model could have high precision but very low recall (or vice versa), leading to an incomplete or inaccurate understanding of the text's content.

2.3 Joint Entity and Relation Extraction

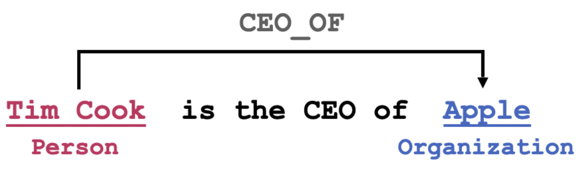


Figure 2.6: The task of joint IE is to indentify entities and their types along with the relation between pairs of entities.

Extracting entities and their relationships from text is a crucial and longstanding task in the fields of Natural Language Processing (NLP). The primary objective of this task is the identification of named entities and their pairwise relationships directly from unstructured text. This process is pivotal for task such as constructing knowledge bases, which play a foundational role in the semantic web. More recently, augmenting large language models with joint entity and relation extraction (JERE) can mitigate issues related to factual inaccuracies in generated text, commonly called hallucination. By grounding these models in knowledge bases that accurately reflect real-world facts and relationships, they can produce responses that are not only relevant but also factually correct, such as generating reliable text about specific companies and their executives, thereby maintaining fidelity to actual data. The methodologies for performing JERE have evolved significantly, ranging from pipeline-based approaches to recent end-to-end models and generative models, which we will review in the following sections.

2.3.1 Pipeline models for entity and relation extraction

Traditionally, JERE was tackled through a two-step pipeline where entities are first identified and then, in a separate process, relations between these entities are extracted. For example, initial models might identify entities like "Apple Inc." and "Tim Cook" and subsequently determine the relation "CEO of" between them.

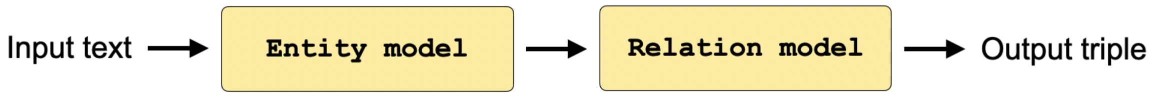


Figure 2.7: Pipeline model for entity and relation extraction.

The pipeline model, as illustrated in Figure 2.7, comprises two main blocks: an entity model and a relation model. The entity model, identifies entities within the input text sequence X. This model predicts a set of entities $E = \{e_1, e_2, \ldots, e_k\}$ given the input token sequence X:

$$E = \textbf{ENT}(X)$$

The layer **ENT** can be any model we described in the previous section, such sequence labelling based on CRFs or Span-based approaches. Furthermore, a relation model operates on the predicted entities and is tasked with classifying all pairs of predicted entities.

Formally, the relation model predicts relations between all entity pairs. This involves the following steps:

1. **Compute relation features**: For each pair of predicted entities $\boldsymbol{e}_n$ and $\boldsymbol{e}_m$, a relation representation $\boldsymbol{r}_{n,m}$ is computed, capturing the contextual information between the entities:
$$\boldsymbol{r}_{n,m} = \boldsymbol{F}(\boldsymbol{e}_n, \boldsymbol{e}_m, X)$$

2. **Compute relation Score**: The relation score between the entity pair e_n and e_m is computed using a softmax layer, which maps the relation representation to a probability distribution over relation types:
$$p(r|n, m) = \text{softmax}(W_r \boldsymbol{r}_{n,m} + \boldsymbol{b}_r)$$
where W_r and b_r are learnable parameters specific to relation classification. The parameters is learned by minimizing the cross-entropy loss during training.

3. **Decoding**: In this step, the model predicts the actual relation between entity pairs. A common approach is to select the relation type with the highest probability score for each entity pair:
$$\hat{r} = \arg\max_r p(r|n, m)$$
where $\hat{r}_{n,m}$ is the predicted relation type between entities e_n and e_m. This method independently predicts the highest scoring relation type for every pair of entities based on the computed relation scores.

Most relation classification methods vary in the approach used to compute relation features, which can be based on hand-crafted features or learned features obtained from neural networks. Here are some notable models for the relation feature space $\boldsymbol{F}$:

- Traditional approaches to relation classification [Roth and Yih, 2004] heavily rely on handcrafted features, encompassing various aspects at both the entity and relation levels. Entity-level features include capitalization patterns, suffix analysis (e.g., "-ing", "-ment") and entity length. On the other hand, relation-level features capture properties such as predefined patterns (e.g., entities separated by a comma), word context and others.

- Among of the earliest neural models proposed by [Zeng et al., 2014] computes relation features by integrating sentence-level features learned through 1D convolutional layers with lexical-level features. These lexical features include the words of the entity pairs themselves, the types of the entity pairs, and word sequences between the entities. Other convolution-based representation have been proposed subsequently, including Nguyen and Grishman [2015], dos Santos et al. [2015], Wang et al. [2016].

- Most recent relation models use pretrained transformer based representations, such as in PURE [Zhong and Chen, 2021a] and PL-markers [Ye et al., 2022a], which have obtained state of the art performance on relation classification benchmarks.

Drawback of pipeline models The pipeline model segregates entity recognition and relation extraction into distinct phases, enabling focused modeling for each task. However, the sequential nature of the pipeline may limit its ability to capture complex dependencies between entities and relations, leading to suboptimal performance in certain scenarios. To circumvent error propagation pipeline model, [Roth and Yih, 2004] proposed to perform joint inference of the two task by proposing a linear program formulation.

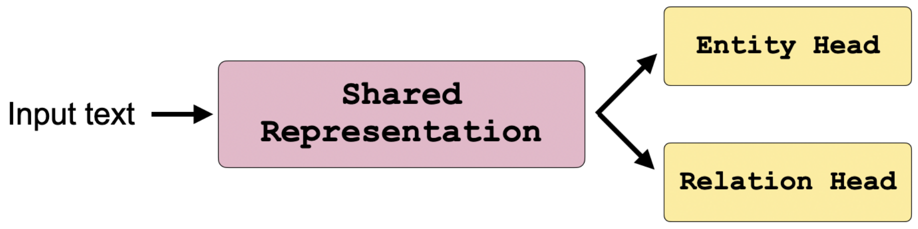


Figure 2.8: Jointly trained entity and relation model with shared representation.

2.3.2 End-to-end (or Joint) Model for Entity and Relation Extraction

The advent of deep learning has facilitated the emergence of jointly trained models for entity and relation extraction. These models typically adopt an architecture similar to the one depicted in Figure 2.8, where both entity and relation tasks are optimized simultaneously during model training through multitask learning:

1. **Token Representation**: The input text $x_1, \ldots, x_N$ is transformed into token representations $h_1, \ldots, h_N$. This is often achieved using pre-trained word embeddings or contextualized word representations obtained from models like BERT.

2. **Entity Classification**: Entity classification is performed using either sequence labeling or span-based classification. In sequence labeling, each token is assigned an entity label, while in span-based classification, contiguous spans of tokens are labeled with entity types. We provide a comprehensive way to perform entity classification in section 2.2.3, which can also be used in this context.

3. **Relation Classification**: Relation classification involves representing all pairs of entities and predicting the relation between them. This can use the same relation model as describe in section 2.3.1.

4. **Multitask Loss Function**: The model is trained using a multitask loss function that jointly optimizes both entity and relation classification tasks. This loss function typically combines the losses from both tasks, ensuring that the model learns to extract entities and relations effectively:

   $$\mathcal{L}_{\text{multi}} = \lambda_1 \mathcal{L}_{\text{entity}} + \lambda_2 \mathcal{L}_{\text{relation}}$$

   where λ_1 and λ_2 are hyperparameters controlling the relative importance of each task, and $\mathcal{L}_{\text{entity}}$ and $\mathcal{L}_{\text{relation}}$ are the respective loss functions for entity and relation classification.

This end-to-end model optimizes entity and relation extraction within a unified framework, facilitating joint optimization and reducing error propagation. In this multi-task setup, losses for entity and relation classification are both utilized to optimize the token representation layer, in contrast to the pipeline model.

Representative Models Recent years have seen numerous proposals for models capable of joint prediction of entities and relations within a text. These models principally differ in their architectural frameworks and the methodologies employed to compute the scores of entities and relations. Typically, these models adopt either a span-based approach or a table-filling strategy.

- **Span-based Models:** This paradigm utilizes span-based models for joint information extraction, as evidenced in various studies [Dixit and Al-Onaizan, 2019, Ji et al., 2020, Lin et al., 2020, Eberts and Ulges, 2021]. Notably, models like DyGIE [Luan et al., 2018] and DyGIE++ [Wadden et al., 2019b] exemplify this approach by incorporating graph propagation techniques to compute contextualized representations of spans.
- **Table Filling Models:** Alternatively, some models approach the tasks of entity recognition and relation classification as a table-filling problem, where the model predicts a matrix or table where rows and columns correspond to possible entities, and the cells represent potential relationships between them. Early implementations leveraged RNN-based frameworks [Gupta et al., 2016], while more recent developments harness the advanced capabilities of pretrained transformers [Ren et al., 2021a, Wang and Lu, 2020, Wang et al., 2021, Ma et al., 2022, Yan et al., 2023].

2.3.3 Generative Information Extraction

Generative models have significantly influenced a variety of tasks in natural language processing (NLP), extending their impact to the domain of joint relation extraction. Among the prominent approaches are models that fine-tune pre-trained encoder-decoder architectures like T5 [Raffel et al., 2019], as well as those that employ either prompting or fine-tuning strategies with encoder-only large language models such as GPT-3 [Ouyang et al., 2022] or LLaMA [Touvron et al., 2023].

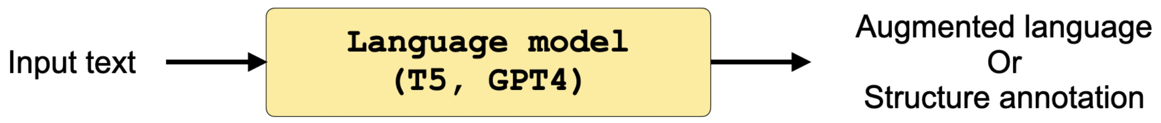
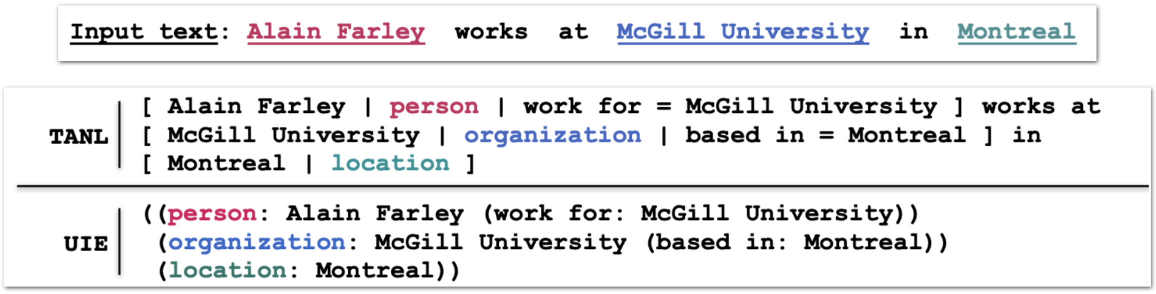


Figure 2.9: Autoregressive models for Information Extraction. An example of an input-output pair is illustrated in Figure 2.10.

Fine-tuning Approaches The principal approach to generative modeling is to fine-tune existing pre-trained encoder-decoder architectures such as T5 [Raffel et al., 2019]. Numerous models have recently been proposed, with the main distinction among them being the representation of the output, i.e., how to linearize the output entities and relationships from the text. Some models approach this by treating the task as a translation from natural language input to a so-called augmented language, as exemplified by TANL [Paolini et al., 2021]. The augmented language is a transformation of the original input text that embeds label information directly within the text. Conversely, some approaches produce a bespoke, easily-parsed structured format to directly output all relevant labels and mentions without replicating the entire input sequence, as seen in models like UIE. An illustration of the two types of representation is illustrated in figure 2.10.



```
Input text: Alain Farley works at McGill University in Montreal

TANL | [ Alain Farley | person | work for = McGill University ] works at
     | [ McGill University | organization | based in = Montreal ] in
     | [ Montreal | location ]

UIE  | ((person: Alain Farley (work for: McGill University))
     |  (organization: McGill University (based in: Montreal))
     |  (location: Montreal))
```

Figure 2.10: Paradigms of structured outputs in generative information extraction. TANL generated augmented languages, while UIE directly produce the output.

Prompting Another way for entity and relation extraction is to directly prompt a language model such as ChatGPT to produce the output. For instance, GPT-RE [Wan et al., 2023] combine specialized instruction and demonstration retrieval to perform the task of relation extraction, and obtain strong performance. Similarly, [Wadhwa et al., 2023] and [Li et al., 2023a] have both shown that when using chain of thought prompting, Large language model such as ChatGPT can obtain performance that are one par with strong supervised approaches.

2.3.4 Dataset Descriptions

In this research, we utilize three prominent datasets: ACE 05, CoNLL 2004, and SciERC. Each dataset is tailored for specific types of entity and relation extraction tasks, which are fundamental to our study. Below, we provide an extensive description of each dataset, detailing their contents and their relevance to our models.

ACE 05 The Automatic Content Extraction (ACE) program datasets, particularly ACE 05, are designed for tasks that involve the recognition and classification of entities and relations from English texts. ACE 05 is comprehensive, covering a wide range of entity and relation types that are crucial for understanding complex semantic structures in text. The dataset categorizes entities into types such as Persons (PER), Facilities (FAC), Locations (LOC), Geopolitical Entities (GPE), Organizations (ORG), Weapons (WEA), and Vehicles (VEH). It also identifies various relation types among these entities, including ART (Artifact relationships), PHYS (Physical relationships), GEN-AFF (General affiliations), ORG-AFF

(Organizational affiliations), PER-SOC (Personal social relationships), and PART-WHOLE (Part-whole relationships). This broad spectrum of annotations makes ACE 05 invaluable for improving and evaluating the accuracy of entity recognition and relation extraction systems.

CoNLL 2004 The Conference on Natural Language Learning (CoNLL) in 2004 introduced a dataset that focuses on named entity recognition and relation extraction, but with a slightly narrower scope than ACE 05. The CoNLL 2004 dataset includes annotations for entities such as People, Organizations, and Locations, and specifies relations like Work_For (a person working for an organization), Live_in (a person living in a location), OrgBased_in (an organization based in a location), Located_in (one location within another), and Kill (one person causing the death of another). The explicit focus on linking entities through specific interactions provides a robust framework for testing and enhancing models that predict relations based on context and entity interactions, making it an essential resource for studies in information extraction.

SciERC Developed specifically for scientific text, the SciERC dataset is tailored to identify and relate entities that commonly appear in scholarly articles. This includes entity types such as Task, Method, Metric, Material, Other-ScientificTerm, and Generic. Each entity is involved in various relationships such as Used-for (a method or material used for a task), Feature-of (describing features of an entity), Part-of (component relationships), Evaluate-for (evaluation scenarios), Compare (comparative relations), and Conjunction (entities linked conjunctively). The specialized nature of SciERC makes it an ideal dataset for advancing models that operate within academic and scientific domains, particularly those that automate the extraction and summarization of research findings.

2.3.5 Evaluation Metrics for Joint Entity and Relation Extraction

In the evaluation of joint entity and relation extraction models, precise metrics that accurately reflect the system's performance are essential. Three primary metrics are employed following standard practice in the literature:

1. **Entity F1 Score:** This metric is fundamental for evaluating the accuracy of entity recognition within our models. The Entity F1 Score calculates the harmonic mean of Precision and Recall, focusing specifically on the correct identification of entity boundaries and their types. The F1 score is calculated as:

$$F1 = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

   we refer the reader to section 2.2.6 for detailled and formal description of the Precision, Recall and F1 metric.

2. **Relation Strict F1 (Correct Entity and Type):** This metric assesses the accuracy of relation extraction when both the boundaries and types of entities are correctly

identified. It evaluates the predicted relations, requiring that the type of the relation and the involved entities (with correct boundaries and types) are accurately identified. The calculation of this F1 score follows the same formula as above, applied to relation predictions.

3. **Relation Relaxed F1 (Correct Entity Boundary):** This metric focuses on the correct identification of the entity boundaries involved in a relation, while the entity types are not considered. The relation type must still be correctly predicted. The formula for calculating this F1 score remains consistent with the standard F1 formula, emphasizing the role of entity boundary accuracy in relation prediction.

Each metric provides a unique perspective on the performance of our models, enabling a comprehensive analysis of their capabilities in both entity recognition and relation extraction.

2.4 Conclusion

This section has provided a comprehensive overview of the evolution of information extraction techniques over the past two decades, from rule-based models to the sophisticated paradigms of generative AI. Despite extensive research, the field remains fraught with challenges. Large language models (LLMs), which have demonstrated strong performance, are often resource-intensive and costly. Moreover, zero-shot approaches, though innovative, continue to lag behind supervised methods in effectiveness, underscoring the complexity of the task. The gaps identified through this historical overview set the stage for the interventions proposed in the following sections of this thesis. These include refining the integration of task-specific structures into both model architecture and decoding processes to ensure coherent outputs, and reintroducing domain-specific knowledge into model decoding—a practice that has been largely overlooked in recent years. Furthermore, addressing the controllability of generative AI models is crucial, as their potent capabilities come with significant management challenges. The subsequent sections will detail efforts to bridge these gaps, aiming to enhance both the efficacy and efficiency of information extraction systems.

Dataset	Type	Description
ACE 05	**Entity Types**	
	PER	Individual people or groups of people
	FAC	Buildings, airports, highways, bridges, etc.
	LOC	Geographical areas and landmasses
	GPE	Countries, cities, states, etc.
	ORG	Companies, governments, institutions, etc.
	WEA	Tools or instruments designed to inflict harm
	VEH	Means of transportation
	Relation Types	
	ART	Artifact relationship, indicating ownership or use
	PHYS	Physical proximity or location relationships
	GEN-AFF	General affiliations or relationships
	ORG-AFF	Affiliation with an organization
	PER-SOC	Personal relationships between people
	PART-WHOLE	One entity is a part of another entity
CoNLL 2004	**Entity Types**	
	Peop	Individuals or groups of people
	Org	Companies, institutions, governments, etc.
	Loc	Geographical entities like cities, countries, etc.
	Relation Types	
	Work_For	A person works for an organization
	Live_in	A person lives in a location
	OrgBased_in	An organization is based in a location
	Located_in	One location is situated within another
	Kill	One person is responsible for the death of another
SciERC	**Entity Types**	
	Task	Specific task or problem in a scientific context
	Method	Approach or technique used in research
	Metric	Standard of measurement in experiments
	Material	Tools, datasets, or materials used in research
	Other-ScientificTerm	Other relevant scientific terms
	Generic	General entities relevant in context
	Relation Types	
	Used-for	A method/material used for a task
	Feature-of	An entity is a feature of another
	Part-of	One entity is a component of another
	Evaluate-for	Evaluating a method/material/task
	Compare	Comparison between two or more entities
	Conjunction	Entities related in a conjunctive manner

Table 2.1: Combined Descriptions of Entity and Relation Types in ACE 05, CoNLL 2004, and SciERC datasets.

Part II

Span-based models for Named Entity Recognition

Chapter 3

GNNer: Reducing Overlapping in Span-based NER Using Graph Neural Networks

3.1 Context

One of the primary objectives of this thesis is to develop neural architecture capable of producing structured outputs that adhere to predefined forms and constraints, particularly for information extraction tasks.

The focus of this chapter is to explore innovative ways to incorporate structural constraints into a deep learning model without explicitly defining them.

To illustrate this concept, we examine the case of span-based Named Entity Recognition (NER), which approaches NER as a span classification problem, differing from the traditional sequence labeling approach of token classification. While span-based methods have demonstrated strong performance, they often generate overlapping spans, which is undesirable in flat NER tasks. This issue arises due to the independent classification of each span.

In this study, we aim to investigate whether the non-overlapping constraint can be effectively enforced by providing the model with information about span overlaps. To achieve this, we employ a graph neural network to update the representations of spans based on their overlap graph. The intention is to equip each span with knowledge about its overlapping counterparts. Our experimental results provide evidence that this implicit constraint can successfully reduce the occurrence of overlapping spans in the output, thus substantiating our initial hypothesis.

The content of this chapter is drawn from the first publication of my PhD study, which was presented at the ACL Student Research Workshop in 2022.

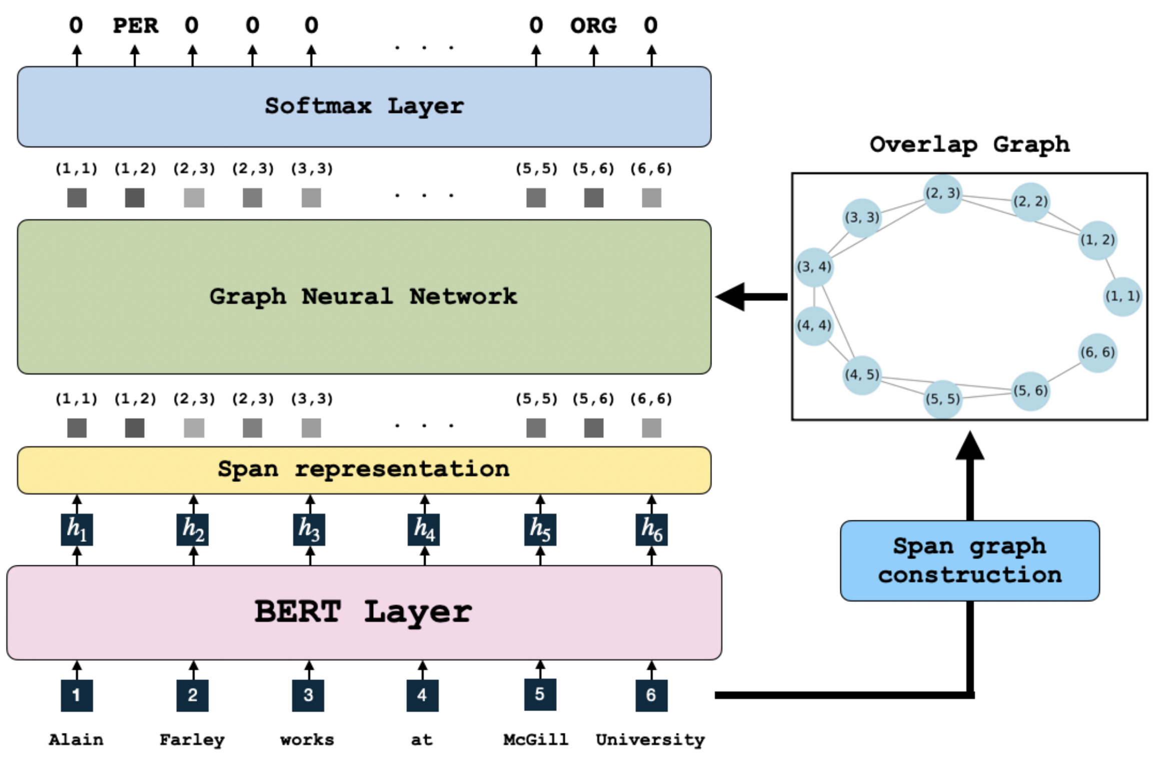


Figure 3.1: The architecture of GNNer

3.2 Introduction

Named Entity Recognition (NER) is an information extraction task that aims to identify named entities such as locations, organizations and person names from textual data. Frequently, NER is designed as a sequence labelling task where each word is classified into its respective label using an annotation scheme such as BIO [Huang et al., 2015, Lample et al., 2016b]. Such schemes are used to encode segment information on the token level. Recently, span-based NER has gained a lot of popularity by handling segments, instead of individual words, as the basic units for labelling Luan et al. [2018], Wadden et al. [2019a]. Specifically, span-based NER enumerates every segment in a text and classifies them by their entity label, whereby non-entity segments are classified into an allocated `null` label. While this method has shown good empirical results, it often assigns entity labels to overlapping spans, which is not desirable, especially for flat NER tasks.

Therefore, to ensure that entities do not overlap, a constraint must be explicitly applied during decoding through, for example, Semi-Markov CRFs [Sarawagi and Cohen, 2005, Sato et al., 2017]. Recent work by Fu et al. [2021] and Li et al. [2021b] address overlapping entities using heuristic decoding: conflict between overlapping spans is resolved by retaining the span with the highest prediction probability, dropping the others. This approach has proven effective, however, the no-overlap constraint is not imposed during learning, which is sub-optimal. In this work, we consider that the no-overlap constraint could be optimized directly by injecting inductive biases into the model.

In this regard, we propose a new approach to reduce overlapping in span-based NERs without affecting the efficiency of heuristic-based decoding. The idea is to make the representation of each span directly influenced by other spans overlapping with it. Specifically, we encode overlapping information as a graph and feed it into the span representation using an equivariant graph neural network layer. In this way, we bias the model towards predictions that implicitly respect the constraints without explicitly modeling them. Our results demonstrate that injecting this graph during model training significantly reduces the number of overlaps compared to our baseline model while achieving better performance. We propose, in this chapter, two variants of our model, GNNer-Conv based on the graph convolution network [Kipf and Welling, 2017] and GNNer-AT based on the graph attention network [Velickovic et al., 2018]. We observe that GNNer-AT is best at preventing span overlaps at the cost of a low recall, while GNNer-Conv provides a better trade-off between the number of violated constraints and metric performance (precision, recall and F-score).

3.3 Model

Given an input sequence, our task involves enumerating and classifying every span. The architecture of our model, summarized in Figure 8.2, includes the following components: token representation layer, span representation layer, GNN layer and span classification layer. Our model is similar to the vanilla span-based NER models [Lee et al., 2017, Luan et al., 2019b], to which we add the GNN layer.

3.3.1 Word Representation

The primary component of our architecture is the word representation layer. The purpose of this layer is to return a set of embedding vectors $\{\boldsymbol{h}^0, \boldsymbol{h}^1, \dots, \boldsymbol{h}^L\}$ from a sequence of tokens $\{w^0, w^1, \dots, w^L\}$. For this part, we employ pre-trained Transformer models such as BERT [Devlin et al., 2019]. However, since pre-trained Transformer models produce sub-word instead of word representations, we retain for each word its first sub-word representation. This choice works well in practice for token classification tasks [Devlin et al., 2019, Beltagy et al., 2019b].

3.3.2 Span Representation

After representing words with their contextualized embeddings, we enumerate all the spans of the sentence up to a maximum span width, which we set to 6 in all our experiments, following prior works [Sarawagi and Cohen, 2005, Xia et al., 2019]. Next, we compute the representation of a span as the concatenation of word embeddings of its left and right extremities, along with a learned embedding of the span width. Specifically, a span (i, j) of width k is represented by the vector $s_{ij} = \boldsymbol{h}^i \otimes \boldsymbol{h}^j \otimes \boldsymbol{z}_k$ where $\boldsymbol{h}^i$ and $\boldsymbol{h}^j$ are respectively

	Architecture	Precision	Recall	F1	Num. Ov.
Conll 2003	Baseline	89.83±0.48	90.31±0.26	90.06±0.15	83±27
	GNNer-CONV	90.12±0.32	89.88±0.36	**90.16±0.52**	52±1
	GNNer-AT	89.54±0.84	79.32±0.04	84.12±0.37	24±11
SciERC	Baseline	66.69±0.49	69.89±0.45	68.25±0.33	87±4
	GNNer-CONV	66.89±1.59	70.34±0.50	**68.57±0.96**	35±3
	GNNer-AT	63.21±0.51	58.06±0.86	60.53±0.69	13±2
NCBI	Baseline	85.30±0.45	89.59±0.74	87.39±0.13	43±12
	GNNer-CONV	85.98±0.45	88.93±0.45	**87.43±0.45**	16±5
	GNNer-AT	84.78±0.18	79.41±0.61	81.98±0.38	10±4

Table 3.1: The results of the experiments on the test datasets. We report the micro-averaged precision, recall and F1-score as well as Num. OV., the total number of overlapping spans on all the test set (without normalization). The numbers are the result of averaging across 3 different/independent runs using different random seeds.

the representation of the words at indexes i and j, and $\boldsymbol{z}_k$ corresponds to the embedding vector for spans of width k; the $\otimes$ symbol denotes the concatenation operation.

3.3.3 Graph construction

Given two spans s_1 and s_2, our graph as represented by the adjacency matrix $\boldsymbol{A}$ is defined as follows:

$$\boldsymbol{A}[s_1, s_2] = \begin{cases} 1, & \text{if } s_1 = s_2 \\ 0, & \text{if } |s_1 \cap s_2| = 0 \\ -1, & \text{otherwise} \end{cases} \tag{3.1}$$

In the adjacency matrix, the edge weight 1 corresponds to self-connection, 0 to non-overlapping nodes, and -1 to overlapping spans. The choice of -1 for the overlap case is supposed to bias the model to learn dissimilar representations for overlapping spans. However, we believe that there may be a better choice to achieve this objective, which would require more in-depth investigation. The addition of the span graph information to the model before the classification layer gives each span information about the spans connected to it and thus allows them to make predictions in a collaborative way, i.e. to make their predictions according to the predictions of their neighbours in the graph.

3.3.4 Span refinement with GNN

After the initial BERT-based representations of all spans are obtained, we refine them using a GNN layer exploiting the previously constructed graph. We propose two versions of the

GNN layer: GNN`er`-CONV, based on graph convolution; and GNN`er`-AT based on attention mechanisms. By exploiting the graph information, we expect the model to implicitly learn that two overlapping spans should not be predicted as a named entity at the same time by learning dissimilar representations for them.

GNNer-CONV

The first variant of our model uses a GCN [Kipf and Welling, 2017] layer, but since GCN is not well suited in the presence of negative edges [Derr et al., 2018], we run two independent 1-layer GCNs over the span representations S: a first GCN, GCN_+ using only positive edges E^+ and another GCN GCN_- using only negative edges E^- for which we concatenate the two representations to get the final span representation:

$$\begin{aligned} \boldsymbol{S}^+ &= GCN_+(\boldsymbol{S}, E^+) \\ \boldsymbol{S}^- &= GCN_-(\boldsymbol{S}, E^-) \\ \boldsymbol{S}^{final} &= \boldsymbol{S}^+ \otimes \boldsymbol{S}^- \end{aligned} \tag{3.2}$$

Note that running a 1-layer GCN on the positive edges is equivalent to a linear layer since the positive edges are self-connections.

GNNer-AT

The second variant of our method uses a graph attention network [Velickovic et al., 2018] but instead of using additive attention, we employ a dot product attention which is much faster and more space-efficient in practice, according to Vaswani et al. [2017a]. More specifically, we project the span representation into keys $\boldsymbol{K}$, queries $\boldsymbol{Q}$, and values $\boldsymbol{V}$ using a two-layer feed-forward network, and compute the attention score as the dot product of the queries and all keys. We further include the scaling factor $\frac{1}{\sqrt{d_{model}}}$ following [Vaswani et al., 2017a] to prevent saturation. We then multiply this attention score by the weighted adjacency matrix. We compute the final span representation as follows:

$$\boldsymbol{S}^{final} = \left(\frac{\boldsymbol{Q}\boldsymbol{K}^T}{\sqrt{d_{model}}} \odot \boldsymbol{A}\right)\boldsymbol{V} \tag{3.3}$$

In the above equation, $\odot$ denotes element-wise multiplication or Hadamard product which is used to mask the attention for null edges. One downside to this approach is that the self-attention mechanism has a quadratic complexity in the number of spans.

3.3.5 Span classification

Lastly, the final representation of the spans is passed to a linear layer with softmax activation to predict the span labels. Remember that for non-entity spans, we allocate a `null` label.

$$Y = \text{softmax}(\boldsymbol{S}^{final}\boldsymbol{W}^{(f)}) \tag{3.4}$$

Here, $\boldsymbol{W}^{(f)}$ is a weight matrix that project the span representations into the label space and the softmax activation function is applied to the label dimension.

3.4 Experiments

3.4.1 Experimental Setup

Datasets We evaluate our approach on three benchmark datasets: Conll-2003 [Tjong Kim Sang and De Meulder, 2003], SciERC NER [Luan et al., 2018] and NCBI [Doğan et al., 2014]. Conll-2003 is a general domain NER dataset that extracts person, organization and location entity mentions from text. SciERC is a dataset for scientific information extraction that consists of article abstracts extracted from Artificial Intelligence related articles. NCBI is a NER dataset that is designed to identify disease mentions in biomedical texts. For all the datasets, we employed the standard train, test and validation splits.

	Domain	Train	Dev	Test
Conll 2003	News	14,987	3,466	3,684
NCBI	Bio	5432	923	940
SciERC	CS	350	50	50

Table 3.2: The statistics of the datasets

Evaluation We evaluate our models on the test splits of the corresponding datasets. Our evaluation is based on the exact match between true and gold entities by discarding non-entity spans. We report the micro-averaged precision, recall and F1. In addition, we also measure the ability of each model to avoid entity overlaps during classification by reporting the number of entity overlaps (Num. Ov.) across all the test set, where a lower number is better.

Implementation details For all our experiments, we used either pre-trained BERT [Devlin et al., 2019] or SciBERT [Beltagy et al., 2019b] as the word encoder depending on the dataset used i.e. BERT for conll-2003, and SciBERT for SciERC and NCBI. We employed a span width embedding of 128 dimensions, and down-projected the span representation (768 * 2 + 128) into 128 units before the GNN layer, using a linear layer. We used only one layer for all GNN variants, which resulted in the best performance on the dev set. In fact, we noticed in our preliminary experiments that adding more layers resulted in decreased performance and slower convergence during training. For all experiments, we set our learning rate to 1e-5 and used Adam [Kingma and Ba, 2017] as our optimizer. We ran all our models

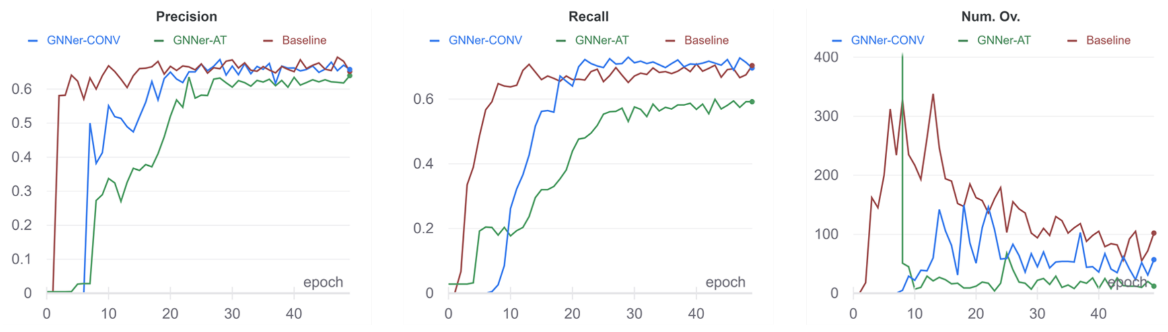


Figure 3.2: Evolution of precision, recall and number of overlaps (Num. Ov.) on the SciERC validation set.

for up to 50 epochs and kept the checkpoint with the best validation performance for testing. All our models are implemented in the PyTorch [Paszke et al., 2019b] and we used the heavily tested GCN layer provided by PyTorch Geometric library [Fey and Lenssen, 2019].

Baseline We used the same architecture without the GNN layer as our baseline. For fair comparisons, we increased the size of the baseline layers to obtain a comparable number of parameters to our proposed models.

3.4.2 Results

Table 3.1 summarizes the results of our experiments by reporting the performance measures (micro-averaged Precision, Recall and F1-score) and the Num. Ov. on the test set. The numbers are the result of averaging across 3 independent runs using different random seeds.

Main results The results in Table 3.1 show that GNNer-AT outperforms all other approaches in reducing the number of overlaps. On average, it produces four times fewer overlaps than the baseline model and twice as few as the GNNer-CONV model. However, it has a lower recall (-11 absolute points compared to the baseline on conll-2003) while maintaining a similar precision score. The low recall may be due to the restrictive span representation caused by the negative edges in our span graph, which could limit the model's ability to predict entities.

Additionally, GNNer-CONV achieves competitive results while keeping the number of overlaps low compared to the baseline model. This makes it the best balance between overlap reduction and metric performance.

Learning curves Figure 3.2 shows the evolution of precision, recall, and Num. Ov. during model training. The plot is shown for training on the SciERC dataset, we obtained similar curves on Conll-2003 and NCBI datasets. We observe that the baseline model trains faster than the GNN-based method, which can be explained by the non-overlap constraint induced by the GNN that favours low recall. On the other hand, the Num. Ov. of the graph-based approach remains low during training, especially for the GNNer-AT approach, while the baseline model increases at the first stage of training before gradually decreasing.

3.5 Limitations

There are several limitations to our approach. First, the addition of GNN does not completely remove the overlapping spans in contrast to heuristic approches. Moreover, the inclusion of GNN layer bring more comptation to the model which result into a slower model than the baseline span-based NER. In fact since, the overlaping span graph is dense (contains many egde), the model does not really benefit of efficient sparse operations of GNN layers.

3.6 Related works

Approaches for NER NER is an important tasks in Natural Language Processing and is used in many downstream information extraction applications. Usually, NER tasks are designed as sequence labelling [Chiu and Nichols, 2016, Huang et al., 2015, Ma and Hovy, 2016, Lample et al., 2016b, Akbik et al., 2018, Zaratiana et al., 2022b]. The goal is to predict BIO tags in which a word is labelled as B-tag if it is the beginning of an entity, I-tag if it is within but not the first in the entity and O for non-entity words. Recently, different approaches have been proposed to perform NER tasks that go beyond traditional sequence labelling. One approach that has been widely adopted is the span-based approach [Luan et al., 2018, 2019b, Wadden et al., 2019a, Xue et al., 2020] where the representation of each segment is computed using a neural network, then fed to a classifier. To prevent overlapping span, priors works either used heuristic decoding [Fu et al., 2021, Li et al., 2021b, Xia et al., 2019] or structured decoding using semi-CRFs [Sato et al., 2017, Ye and Ling, 2018]. However, to the best of our knowledge, no work have used GNN for the purpose of reducing span overlap for NER. Some work [Li et al., 2020] has also approached NER as a question answering task in which named entities are extracted by retrieving answer spans. In addition, with the growing popularity of prompt-based learning, recent work such as Cui et al. [2021c] considers NER as template filling by fine-tuning a BART [Lewis et al., 2019] encoder-decoder model. In contrast we focus on learning appropriate span representations.

GNN for NLP GNNs have gained a lot of popularity recently due to their powerful ability to represent arbitrary shapes of data [Hamilton et al., 2018, Wu et al., 2019, Hamilton, 2020]. Specifically, GNNs provide a way to inject prior knowledge into NLP systems through, for example, dependency graphs [Liu et al., 2018, Zhang et al., 2019], constituency graphs [Marcheggiani and Titov, 2020] or knowledge graphs [Sun et al., 2018, Lin et al., 2021]. As a result, GNNs have been widely applied to different NLP tasks such as Neural Machine Translation [Bastings et al., 2017, Beck et al., 2018], Semantic Parsing [Xu et al., 2018, Shao et al., 2020], Information Extraction [Fu et al., 2019b, Sun et al., 2019] and text classification

[Yao et al., 2018, Liu et al., 2020a]. More relevant to our work, DyGiE [Luan et al., 2019b, Wadden et al., 2019a] used GNNs to refine the span representation for joint NER and RE extraction, but in contrast, they learn their graph dynamically during training while we used a static span graph. For a detailed review of GNNs for NLP, please refer to Wu et al. [2021].

3.7 Conclusion

In this work, we proposed GNNer, an architecture for span-based Named Entity Recognition (NER) that utilizes graph neural networks to refine span representations using a span overlap graph. The resulting models significantly reduce the number of overlapping spans compared to the baseline approach and, in some cases, also improve performance. In the next chapter, we conduct a more comprehensive study of span-based NER and propose a new exact decoding method for the task.

Chapter 4

Preliminary Study on Span-based Named Entity Recognition

4.1 Context

In the previous chapter, we saw that structural constraints can be implicitly injected into span-based architectures using GNNs, but the resulting model is not scalable, as the overlap graph is dense, making the model undesirable in real-world scenarios. In this chapter, we study in detail different aspects of span-based NER, namely span representation, learning strategies, and decoding algorithms to avoid span overlap, and compare their performances. We also propose an exact algorithm that efficiently finds the set of non-overlapping spans that maximizes a global score, given a list of candidate entity spans. We performed our study on three benchmark NER datasets from different domains.

4.2 Introduction

Span-based Named Entity Recognition (NER) has recently gained popularity. Unlike traditional sequence tagging, which operates at the token level, span-based NER works directly at the span level. The main idea is to enumerate all possible contiguous sequence of tokens of an input text and predict their identity [Lee et al., 2017]. One of the major advantages of the span-based NER is that it can learn a rich representation of the span instead of only learning the representation of each token. In addition, a recent study by Fu et al. [2021] reveals that span-based NERs are better in a context with more OOV words and Li et al. [2021b] showed that span-based NERs are much better than sequence labelling in settings with unlabelled entities (missing entities due to annotation errors).

However, unlike sequence labelling, *unconstrained* span-based approaches tend to produce overlapping entities, which is undesirable for flat, non-overlapping NER tasks. To avoid overlap in span-based NER, two main approaches have been adopted in the literature. The first is the Semi-Markov conditional random field [Sarawagi and Cohen, 2005] that trains a globally normalized model and then uses a Viterbi algorithm to produce the optimal segmentation without span overlap, we call this approach *Semi-CRF*. The second algorithm is the one employed by Li et al. [2021b] for locally normalized span-based NER; it first eliminates all non-entity spans and deals with the overlap conflict by keeping the span with the highest prediction probability while eliminating the others. In this work, we call this approach *greedy decoding*.

In this chapter, we analyze and compare two formulations of span-based NER. The first is a *segmentation* model of the Semi-CRF; the second is the two-step pipeline of span filtering and decoding. In addition to greedy decoding, we propose an exact algorithm based on Maximum Weighted Independent Set (MWIS) [Hsiao et al., 1992, Pal and Bhattacharjee, 1996] on internal graphs. We build such graphs to encode the overlapping structure between spans. This formulation of the NER task is novel up to our knowledge. For completeness, we include in the comparison a token-based sequence labeling model with a linear-chain CRF.

In order to understand the effect of span representation, we explore different alternatives including max-pooling, convolution and endpoints (representing span by its extreme tokens) and show that endpoints are effective across models and datasets.

Our contributions can be summarized as follow:

- We propose an exact decoding algorithm to eliminate span overlap on locally trained models that overcomes the myopic bias of the greedy approach [Li et al., 2021b]. We present a detailed comparison with global models.
- We investigate different span representations for span-based NER when using pretrained Transformer models. Our experiment provide a confirmation that the endpoint representation, the currently dominant representation strategy is the most robust.
- We conduct few-shot performance analysis for different modeling. We found that classical sequence labeling models provide strong result for datasets with few entity types, while span-based approaches are better for larger type sets.

4.3 Span Representation

Given an input sequence $\boldsymbol{x} = [x_1, \dots, x_n]$, a span (i, j) is the contiguous segment of tokens $[x_i, \dots, x_j]$. The goal of representation is to compute an embedding vector for each span of an input text which can be used for downstream prediction tasks. We denote $\boldsymbol{h}_i \in \mathbb{R}^{d_h}$ the representation of the word at the position i and $\boldsymbol{s}_{ij} \in \mathbb{R}^{d_s}$ the representation of the span (i, j) with the width $k = j - i + 1$; here d_h, $d_s \in \mathbb{N}^+$ are respectively the embedding sizes for word and span representations. The token representations are computed using a BERT-based model [Devlin et al., 2019]. However, since BERT-based tokenization divides the input words into subwords, we take the first subword to represent the whole word, which has proven to be very competitive for several token classification tasks [Beltagy et al., 2019c]. In the following, we present different approaches for representing the spans.

Span representation	Num params.
Endpoints	$(2d_h + d_k)C$
Maxpool	$d_h C$
Convolution	$\frac{1}{2}d_h^2 K(K+1) + d_h C$
Convolution (shared)	$d_h^2 K + d_h C$
FirstToken	$d_h^2 K + d_h C$

Table 4.1: Number of parameters for different representation, without including the word representation layer which is the same for any approach. d_h, K and C are respectively the word embedding size, the maximum span width and the number of classes. Blue terms are parameters for computing span representations and Red terms denote number of parameters for the final layer.

Endpoints This representation consists in representing a span using the representation of the tokens of its right and left extremities, in addition to a span width feature. Specifically, the representation of the span (i, j), $\boldsymbol{s}_{ij}$ is computed as:

$$\boldsymbol{s}_{ij} := [\boldsymbol{h}_i; \boldsymbol{h}_j; \boldsymbol{w}_k] \tag{4.1}$$

where $\boldsymbol{w}_k$ is a learned vector of width k and $[;]$ denotes the concatenation operation. Endpoints have been widely used in previous works for span prediction tasks such as NER and coreference resolution [Lee et al., 2017, Luan et al., 2019b, Zhong and Chen, 2021b].

Max-pooling Since spans consist of a contiguous segment of tokens, pooling operations are a fairly natural way to compute their representations. In this context, we use an element-wise max-pooling operation to all tokens inside the span. Formally,

$$\boldsymbol{s}_{ij} := \text{MAX}([\boldsymbol{h}_i; \boldsymbol{h}_{i+1}; \ldots; \boldsymbol{h}_j]) \tag{4.2}$$

where MAX is the element-wise max pooling operation. Max-pooling has been previously used by Eberts and Ulges [2020] for joint entity and relation extraction.

Convolution Instead of simply applying the pooling operation, we explored aggregating tokens using learned filters via convolution. Specifically, representations of all spans of size k are computed simultaneously using a 1D convolution of kernel size k. To keep the number of parameters linear with respect to the maximum span width, we share the convolution weights across the different span widths.

$$\boldsymbol{s}_{ij} := \texttt{Conv1D}_k([\boldsymbol{h}_i; \boldsymbol{h}_{i+1}; \ldots; \boldsymbol{h}_j]) \tag{4.3}$$

Lei et al. [2021] used this convolutional approach to represent spans for keyphrase extraction.

FirstToken For this representation, we only use the start token along with span width information:

$$\boldsymbol{s}_{ij} := \boldsymbol{W}^{(k)}\boldsymbol{h}_i \tag{4.4}$$

where $\boldsymbol{W}^{(k)} \in \mathbb{R}^{d_h \times d_h}$ is the weight matrix associated with width k. Note that the computation of the representation of all spans for this approach can be done in parallel and in a single line of code using einsum operation Rogozhnikov [2022]. This representation was inspired by the synthetic attention from Tay et al. [2021], where the authors predict attention scores without pairwise interaction.

Number of parameters The number of parameters required for each span representation is shown in Table 4.1.

4.4 Span scores

We model the task of NER as assigning to each span (i, j) a label from a set of C different types that correspond to named-entity types and special `null` type, indicating that the span does not correspond to an entity. Label assignment is constrained so that no pair of overlapping spans have entity types (both different from `null`). We present two models to solve this structured prediction problem: a locally normalized approach with a zero-order scoring function which does not take into consideration the interactions between label assignment (§4.5); and a globally normalized approach with first-order scoring function which considers dependencies between pairs of consecutive spans (§4.6). Both formulations employ the following span scoring function. Given a span representation $\boldsymbol{s}_{ij}$, the logits $\boldsymbol{\phi}(i, j) \in \mathbb{R}^C$ for the C different labels are computed using a non-linear activation function followed by an affine transformation:

$$\boldsymbol{\phi}(i, j) = \boldsymbol{W}\texttt{ReLU}(\boldsymbol{s}_{ij}) + \boldsymbol{b}f \tag{4.5}$$

where $\boldsymbol{W} \in \mathbb{R}^{d_s \times C}$ is the final weight matrix, $\boldsymbol{b}f \in \mathbb{R}^C$ is the bias vector, and ReLU is the activation function. We denote by $\boldsymbol{\phi}(i, j, l) \in \mathbb{R}$ the (unnormalized) score of the label l for the span (i, j).

4.5 Locally Normalized Models

Under this approach, we perform span labeling in two steps, span classification followed by a decoding step.

4.5.1 Span Classification

Each span (i, j) is assigned its highest scoring label $\hat{l}_{ij} = \text{argmax}_l \boldsymbol{\phi}(i, j, l)$, and we denote $\hat{k}_{ij}$ the corresponding highest score. The set of spans classified as entities may contains overlapping spans, a decoding step is therefore required to select a subset with no overlaps. We learn the parameters[1] of this classifier under a locally normalized setup. The training's objective is to maximize the likelihood for every span label (up to a maximum lenght K) from the training data. The loss function is as follows:

$$\mathcal{L} = -\sum_{(i,j,l)\in\mathcal{T}} \log \frac{\exp \phi(i, j, l)}{\sum_{l'} \exp \phi(i, j, l')} \tag{4.6}$$

which is the well-known cross-entropy loss.

4.5.2 Greedy Decoding

Let $S = \{(i, j) : \hat{l}_{ij} \neq \texttt{null}\}$ be the set of spans classified as entities. The goal of decoding is to find the subset of S that maximizes a global score function:

$$\begin{aligned} E^* = &\arg\max_{E\subseteq S} \sum_{(i,j)\in E} \hat{k}_{ij} \\ \text{s.t. } &\forall e, e' \in E : \texttt{!overlap}(e, e') \\ &\forall u \notin E, \exists e \in E : \texttt{overlap}(e, u) \end{aligned} \tag{4.7}$$

where $\texttt{overlap}(e, e')$ is True if the spans e and e' overlap but are not equal. The first constraint in Eq. 4.7 ensures that the set E is *independent*, i.e. it doesn't contains overlapping spans; the second constraint ensures that it is *maximal*, i.e. adding any other span breaks the no-overlap constraint. Greedy decoding constructs an approximation to E^* by iteratively adding the highest-scoring entity not overlapping with any previously selected entity. This algorithm is efficient and has a complexity of $O(n \log n)$ with $n = |S|$.

4.5.3 Exact Decoding with MWIS

We define an *overlapping graph* as the graph G whose nodes are the elements of S and contains an edge between each pair of overlapping spans. Its adjacency matrix is defined as:

$$\boldsymbol{A}[e, e'] = \begin{cases} 1, & \text{if } \texttt{overlap}(e, e') \\ 0, & \text{otherwise} \end{cases} \tag{4.8}$$

We associate a weight to each node as provided by its label score $\boldsymbol{\phi}(i, j, \hat{l}_{ij})$. An exact solution to Eq. 4.7 is given by the Maximum Weight Independent Set (MWIS) of the overlapping graph. For general graphs, computing the MWIS is NP-Hard but since our graph

[1]The parameters include all weight matrices from span representation and scoring functions. We omit the parameters from the notation for simplicity.

can be seen as an *interval graph* (spans can be considered as intervals over their start and end positions), MWIS has a complexity of $O(n \log n)$ or $O(n)$ if the spans are sorted by their endpoint Hsiao et al. [1992].

4.5.4 Exhaustive Search Decoding

For efficient decoding, the scoring function in Eq. 4.7 decomposes as a sum over graph nodes. More complex scoring functions do not necessarily admit efficient decoding. Finding an optimal set under the mean scoring function for instance, that is $\frac{1}{|E|}\sum_{(i,j)\in E} \hat{k}_{ij}$, requires enumerating all possible candidates subsets of S, which is NP-Hard [Johnson et al., 1988, Raman et al., 2007] but feasible for reasonably small interval graphs. In this chapter, we experiment with this scoring functions but leave more complex ones for future work.

Decoding	Time Complexity
CRF	$O(L\|Y\|^2)$
Semi-CRF	$O(LK\|Y\|^2)$
Greedy Decoding	$O(n \log n)$
MWIS	$O(n \log n)$
Exhaustive Search (EXT)	$O(3^{n/3})$

Table 4.2: Complexity of various decoding algorithms with parameters: input length L, maximum segment width K, number of classes $|Y|$, and filtered spans $n \approx 0.15L$.

4.6 Globally normalized model

Under this approach, NER is modeled using a semi-Markov segmentation CRF introduced by Sarawagi and Cohen [2005]. The input sentence $\boldsymbol{x}$ is segmented into a labeled sequence of spans $\boldsymbol{y}$. Each segmentation is scored as:[2]

$$\Omega(\boldsymbol{y}) = \sum_{y_k=(i,j,l)} \boldsymbol{\phi}(i,j,l) + \boldsymbol{T}_{l',l} \tag{4.9}$$

with $y_k = (i,j,l)$ being the labeled span at position k. Unlike the scoring function in Eq. 4.7, the score here contains the transition scores from label l' at position $k-1$ to label l, in the *learnable* matrix $\boldsymbol{T}$.

Training The parameters of the model are learned to maximize the conditional probability of the gold segmentation in the training data. The probability of a segmentation is computed by globally normalizing the score: $P(\boldsymbol{y}|\boldsymbol{x}) = \exp\Omega(\boldsymbol{y}) - Z$, where Z is the log partition function $\log\sum_{\boldsymbol{y}\in\mathcal{Y}(\boldsymbol{x})} \exp\Omega(\boldsymbol{y})$, which sums over all possible segmentation $\mathcal{Y}(\boldsymbol{x})$. This normalization term can be computed in polynomial time using dynamic programming. Following [Sarawagi and Cohen, 2005], we assume that segments have strictly

[2]We drop the dependence on the input $\boldsymbol{x}$ for simplicity.

positive lengths, adjacent segments touch and we assume that non-entity spans have unit length. For instance, a segmentation of the sentence "Michael Jordan eats an apple ." would be Y*=[(0, 1, PER), (2, 2, O), (3, 3, O), (4, 4, O), (5, 5, O)].*

Decoding Selecting the most probable segmentation $\hat{\boldsymbol{y}} = \arg\max_{\boldsymbol{y} \in \mathcal{Y}(\boldsymbol{x})} \Omega(\boldsymbol{y})$ is efficiently performed using the segmental variant of the Viterbi algorithm [Sarawagi and Cohen, 2005].

4.7 Experimental Setup

4.7.1 Datasets

We evaluated our model on three benchmark datasets for Named Entity Recognition: Conll-2003 [Tjong Kim Sang and De Meulder, 2003], OntoNotes 5.0 [Weischedel et al., 2013] and TDM [Hou et al., 2021]. Conll-2003 is a dataset from the news domain that was designed for extracting entities such as Person, Location and Organisation. OntoNotes 5.0 is a large corpus comprising various genres of text including newswire, broadcast news and telephone conversation. It contains in total 18 different entity types such as Person, Organization, Location, Product or Date. TDM is a NER dataset that was recently published and it was designed for extracting Tasks, Datasets, and Metrics entities from Natural Language Processing chapters.

4.7.2 Implementation Details

Backbones For span encoding, we used RoBERTa-base [Liu et al., 2019b] for models trained on Conll-2003 and OntoNotes 5.0 because they come from general domains and we employed SciBERT [Beltagy et al., 2019c] for models trained on TDM, which is a scientific NER data set.

Baseline model We compare the span-based approaches to a sequence labelling BERT-CRF [Beltagy et al., 2019c], which we trained on our datasets.

Hyperparameters All models were trained using a single V100 GPU. We trained for up to 25 epochs using Adam [Kingma and Ba, 2017] as the optimizer with a learning rate of 1e-5. We opted for a batch size of 10 and used early stopping with a patience of 5 (on the F1-score) and keep the best model on the validation set for testing.

Libraries We implement our model with pytorch [Paszke et al., 2019b]. The pre-trained transformer models were loaded from the HuggingFace's Transformers [Wolf et al., 2020].

Model	**Span Representation**											
	Convolution			*Endpoints*			*Maxpool*			*FirstToken*		
	P	R	F	P	R	F	P	R	F	P	R	F
Conll-2003												
Local	91.40	89.86	90.62	91.07	90.48	90.77	90.52	90.52	90.52	90.34	89.25	89.79
+ Greedy	91.97	89.74	90.84	91.5	90.1	90.79	91.26	90.1	90.67	90.64	89.08	89.85
+ MWIS	91.97	89.76	**90.85**	91.5	90.11	90.8	91.22	90.09	90.65	90.64	89.08	89.85
+ EXT	91.97	89.75	**90.85**	91.54	90.11	**90.82**	91.33	90.11	**90.71**	90.66	89.09	**89.86**
Semi-CRF	89.45	88.99	89.22	89.64	89.23	89.43	89.48	88.82	89.15	89.5	89.17	89.33
OntoNotes 5.0												
Local	88.59	88.99	88.79	88.06	89.55	88.8	88.42	89.34	88.88	88.18	88.73	88.45
+ Greedy	89.3	88.62	**88.96**	88.93	89.0	88.96	89.38	88.83	89.11	88.8	88.22	88.51
+ MWIS	89.26	88.61	88.93	88.9	88.98	88.94	89.38	88.87	**89.13**	88.81	88.26	**88.53**
+ EXT	89.31	88.61	88.95	88.95	89.01	**88.98**	89.37	88.79	89.08	88.80	88.20	88.50
Semi-CRF	87.35	87.76	87.55	87.36	88.26	87.81	87.04	87.99	87.51	87.11	87.86	87.48
TDM												
Local	73.05	69.38	71.15	67.75	69.88	68.78	70.86	70.69	70.73	68.54	65.06	66.74
+ Greedy	75.86	68.28	**71.84**	75.12	67.82	71.26	73.24	69.43	71.26	69.82	64.40	66.99
+ MWIS	75.46	68.07	71.55	75.25	68.12	**71.48**	73.31	69.53	**71.34**	69.89	64.50	67.07
+ EXT	75.72	68.07	71.67	74.63	66.97	70.57	73.24	69.43	71.26	69.82	64.40	66.99
Semi-CRF	68.34	72.55	70.35	69.38	72.85	71.05	70.32	69.89	70.09	69.98	70.64	**70.31**

Table 4.3: This table reports the main results of our study. It shows the performance along different settings including the datasets, the training, decoding and span representations. We report the average across three seeds. **Bold** numbers indicate the best model/decoding for a fixed representation and underlined numbers indicate the best representation for a fixed model/decoding.

We employed AllenNLP [Gardner et al., 2018a] for data preprocessing and the seqeval library [Nakayama, 2018] for evaluating the baseline sequence labelling model. Our Semi-CRF implementation is based on pytorch-struct [Rush, 2020b].

4.8 Results

4.8.1 Span Representation

In the following, we analyze the performance of the span representations on both the local model and the Semi-CRF model, as shown in the table 4.3.

Local models On local models, we find that *Convolution*, *Endpoints* and *Maxpool* all got competitive results while *FirstToken* representation obtains a result one notch below the others. On both the conll and TDM datasets, *Convolution* performed the best, yet the endpoints performed only slightly worse. However, on OntoNotes, the *Maxpool* representation

Dataset	Model	#Examples						
		100	250	500	1000	2500	5000	All
Conll-2003	CRF	68.92	**77.49**	**82.05**	**85.38**	**88.50**	**89.40**	**90.96**
	Local	63.09	70.95	77.21	82.46	85.51	87.62	90.77
	+ Greedy	66.44	73.02	78.70	83.39	86.2	88.05	90.79
	+ MWIS	66.54	73.1	78.70	83.47	86.23	88.01	90.80
	+ EXT	65.54	72.53	78.49	83.35	86.10	88.00	90.82
	Semi-CRF	**69.21**	73.91	79.26	82.6	86.03	87.26	89.43
OntoNotes 5.0	CRF	61.0	69.59	74.17	77.18	78.86	81.08	88.36
	Local	60.23	68.13	73.33	76.69	81.32	82.49	88.80
	+ Greedy	62.60	70.20	74.95	77.46	**82.13**	83.09	88.96
	+ MWIS	**63.03**	**70.28**	**74.97**	**77.52**	82.08	**83.10**	88.94
	+ EXT	61.95	69.88	74.69	77.37	82.06	83.07	**88.98**
	Semi-CRF	63.02	69.46	72.79	77.03	80.33	81.97	87.81
TDM	CRF	**63.39**	**68.39**	**69.76**	–	–	–	**71.66**
	Local	54.87	63.1	67.08	–	–	–	68.78
	+ Greedy	55.94	65.28	67.64	–	–	–	71.26
	+ MWIS	57.04	65.22	67.60	–	–	–	71.48
	+ EXT	55.06	64.38	67.43	–	–	–	70.57
	Semi-CRF	60.49	65.06	66.52	–	–	–	71.05

Table 4.4: Few-shot performance. We report the average F1-score across three different seeds in all datasets and different training set sizes.

outperforms all other approaches, while the *Endpoints* and *Convolution* got very similar performance. Out of all the datasets, *FirstToken* had the lowest score.

Global models On Semi-CRF models, the *Endpoints* representation consistently achieves the best results across datasets. We also notice that the *FirstToken* representation has better result than *Maxpool* and *Convolution* on two datasets, Conll-2003 and TDM in this setting. The *Endpoints* representation is the most reliable overall, since it achieves robust performance regardless of the context in which it is used. However, for optimal performance and given a sufficient amount of compute resources, the span representation should be best tuned on a held-out set.

4.8.2 Comparison of Decoding for Local Models

Table 4.3 shows the performance results of the different decoding algorithms under different settings. For the local models, we can see that the application of decoding always improves the performance of the F1 score, by increasing the precision and by decreasing the recall score. However, there is no significant difference between the greedy decoding and the global decoding since the models are already well trained and thus, the overlap filtering does not make much difference in terms of quantitative results. We will provide more insight on decoding in the subsections 4.8.3 and 4.8.5.

4.8.3 Few-Shot Performance

We conducted a study to compare the performance of each model in a few-shot scenario. The evaluation was performed on the test set of each dataset using from 100 to the full training dataset. For this study, we used the Endpoints representation for spans because it is widely used and has shown good performance across different training and decoding schemes. The results of our few-shot evaluation are presented in Table 4.4.

Semi-CRF is better than the local spans-based approach when overlap filtering is not performed but the local approach performs better than Semi-CRF when the number of data become larger. Furthermore, while the difference between Greedy decoding and MWIS decoding is narrow in the high data regime, we can see that MWIS outperforms Greedy decoding in the low and very low data regime. Furthermore, we notice that the increase in performance by decoding is higher when a local model is training on a few datasets while the difference becomes less significant when the number of training data is large.

We find that the baseline sequence labelling, BERT-CRF approach is indeed competitive. It most of the time obtains a better performance on Conll-2003 and TDM datasets across any dataset sizes. However, the span-based approach is better on the OntoNotes 5.0 dataset. This can be explained by the fact that OntoNotes 5.0 contains 18 entity types and, therefore, the labelling approach would require 37 labels since it uses a BIO scheme, which makes the task much more difficult.

4.8.4 Analysis of Local Modeling

We previously found that decoding had little effect on our local model performance, especially for high resource datasets. We believe this is due to the fact that we were training with all negative samples (non-entity spans). As a result, the model was overconfident regarding non-entity spans (and not confident enough to predict entity spans) due to this unbalanced training. To resolve this issue, we propose three alternative training procedures to make the classifier leave more room for the decoder.

Negative sampling This approach randomly drops a percentage of the non-entity spans during training, but keeps all positive samples (entity spans). By training with fewer non-entity spans, we expect the model to be less confident and thus predict more entities. This negative sampling has been previously used by Li et al. [2021b] to avoid training NER models with unlabeled (or missing) entities.

	Conll			OntoNotes			TDM		
	P	R	F	P	R	F	P	R	F
Local	91.07	90.48	90.77	88.06	89.55	88.80	67.75	69.88	68.78
+ decoding	0.47	-0.37	+0.05	+0.89	-0.54	+0.18	+7.5	-1.76	+2.7
Neg. Sample	90.69	90.54	90.61	86.81	90.23	88.49	66.83	73.66	**70.01**
+ decoding	+0.84	-0.43	+0.21	+1.58	-0.98	+0.33	+7.7	-2.12	+2.97
Down-weighting	90.88	90.10	90.49	87.70	90.24	**88.95**	57.79	78.63	66.52
+ decoding	+0.48	-0.27	+0.1	+1.24	-0.72	+0.28	+13.77	-3.92	+6.57
Thresholding	90.80	90.96	**90.88**	87.49	88.81	88.14	63.99	74.56	68.85
+ decoding	+0.90	-0.41	+0.25	+1.00	-0.61	+0.21	+6.78	-2.70	+2.32

Table 4.5: Result for the local model when changing the training/loss. The best results before decoding are in **bold** and the best results after decoding are underlined. For this experiment, we use MWIS as decoding. We report the average over three seeds.

Down-weighing This method is similar to negative sampling, but instead of randomly eliminating negative samples, this approach retains all negative samples and down-weights their loss contribution while keeping loss for entity spans intact.

Thresholding This approach separates the span classifier into two models: a filtering model to classify whether a span is an entity or not, and a second an entity classification model to classify the entity type. During training, both models are trained end-to-end by multi-task learning with equally weighted losses. For prediction, span filtering is first performed and then the result is passed to the entity classification layer. By default, a span is passed into the entity classification layer if its probability of being an entity is greater than 0.5; however, we here adjust this threshold on the dev set and select the one with best F1 score.

The result from this analysis is show in the table 4.5. The results of this analysis show that, overall, the use of regularization techniques leads to a significant improvement in decoding accuracy for most datasets. As the most striking example, we can see that on the TDM dataset, the down-weighting approach which initially had a precision score of 57.79 was able to increase this score by 13.77 thanks to decoding improvements. Furthermore, it appears that the best approach according to these empirical results is the downw-eighting approach. Under this method, the decoder was most "successful" on both OntoNotes and TDM datasets, meaning it brought the largest improvements relatively to the performance of the local classifier before decoding.

4.8.5 Qualitative Comparison of Decoding

We performed a qualitative analysis to compare the three decoding approaches for local models. This study is presented in Figure 4.1, which shows the input text (truncated), the raw prediction with overlap, and the results after applying greedy decoding and the global

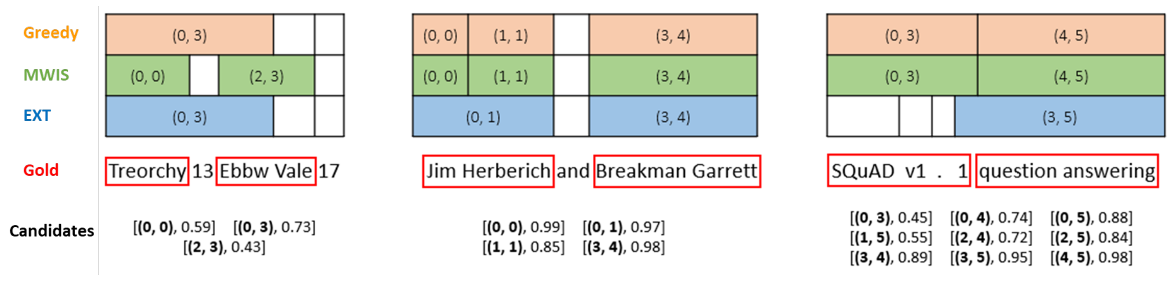


Figure 4.1: Shows how overlapping conflicts are handled by the different decoding algorithm on local span-based NER models. We only include overlaps involving at least three entities, because otherwise all decoding produce the same result.

decoding (MWIS and EXT). We only include overlaps involving more than two spans, because when two spans overlap, all algorithms take the span with the highest score. We can see that the greedy approach always retrieves the most probable entity since it iteratively selects the best spans that do not overlap with previously selected spans. However, this algorithm tends to suffer from a myopic bias. Second, the MWIS approach, which maximizes the sum of span scores, tends to select as many spans as possible, which means that it favours shorter spans over longer ones. Also, MWIS decoding has a slightly higher recall score most of the time than other decoding algorithms. Finally, EXT decoding, which selects the set of spans that maximizes the average score, tends to select the smallest number of spans, but the selected spans generally have a high score. In general, this decoding tends to favour precision over recall score.

4.9 Conclusion

In this study, we explored various span representations for Named Entity Recognition (NER) and identified the endpoint representation as the most robust. Additionally, we proposed a novel formulation of NER using overlapping graphs, accompanied by an exact and efficient decoding algorithm. This formulation effectively eliminates span overlap in locally trained models.

Furthermore, we performed a few-shot performance analysis of different modeling approaches. Our findings indicate that classical sequence labeling models deliver strong results for datasets with a few entity types, whereas span-based approaches are more effective for datasets with larger type sets. In the next chapter, we will introduce an extension to local span-based NER models that incorporates a globally normalized span selection framework, inspired by the Semi-Markov Conditional Random Field (CRF).

Chapter 5

Filtered Semi-Markov CRF

5.1 Context

Semi-Markov CRF has been proposed as an alternative to the traditional Linear Chain CRF for text segmentation tasks such as Named Entity Recognition (NER). Unlike CRF, which treats text segmentation as token-level prediction, Semi-CRF considers segments as the basic unit, making it more expressive. However, Semi-CRF suffers from two major drawbacks: (1) quadratic complexity over sequence length, as it operates on every span of the input sequence, and (2) inferior performance compared to CRF for sequence labeling tasks like NER. In this chapter, we introduce Filtered Semi-Markov CRF, a variant of Semi-CRF that addresses these issues by incorporating a filtering step to eliminate irrelevant segments, reducing complexity and search space. Our approach is evaluated on several NER benchmarks, where it outperforms both CRF and Semi-CRF while being significantly faster. The implementation of our method is available on Github.

5.2 Introduction

Sequence segmentation, the process of dividing a sequence into distinct, non-overlapping segments, has various applications, including Named Entity Recognition and Chinese Word Segmentation Tjong Kim Sang and De Meulder [2003], Li and Yuan [1998]. In the past, this task has been approached as a sequence labeling problem using pre-defined templates, such as the BIO and BILOU schemes Ratinov and Roth [2009a]. The Conditional Random Field (CRF) Lafferty et al. [2001] has become a popular method for sequence labeling problems due to its ability to model the dependency between adjacent token tags. However, the CRF model may not efficiently capture the underlying structure of the sequence, as it is limited to modeling relationships between individual tokens rather than segments.

The Semi-Markov CRF Sarawagi and Cohen [2005] has been proposed as a variant of the CRF, allowing for the incorporation of higher-level segment features, such as segment width. While the Semi-CRF allows for a more natural approach to sequence segmentation, it suffers from slower learning and inference due to its quadratic complexity with respect to the sequence length. Additionally, the Semi-CRF often underperforms CRF, showing only marginal improvements in some cases Liang [2005], Daumé and Marcu [2005], Andrew [2006], which can be attributed to the Semi-CRF's significantly larger solution space, complicating the search for optimal solutions.

To address the limitations of Semi-CRF, we propose Filtered Semi-CRF, which introduces a filtering step to prune irrelevant segments using a lightweight local segment classifier. By leveraging transformer-based features, such as BERT [Devlin et al., 2019], this classifier can identify high-quality candidate segments. Consequently, the task of the Semi-CRF is simplified to selecting the best segments from the pool of high-quality candidates. Our experiments demonstrate that this filtering step not only accelerates the decoding process but also improves the overall model performance.

Although pruning techniques have been applied to accelerate parsing algorithms [Roark and Hollingshead, 2008, Bodenstab et al., 2011], they often involve a trade-off between accuracy and inference speed. In contrast, our filtering approach is learned jointly and collaboratively with the Semi-CRF during training, resulting in a model that not only increases efficiency but also improves overall performance.

When evaluated on Named Entity Recognition, our model significantly outperforms both the CRF and Semi-CRF, achieving F1 score improvements of up to 2.5 and 1.1 points, respectively, on the CoNLL 2003 dataset. Additionally, our model also accelerates the decoding process to a speed that can be up to 20 times and 137 times faster than CRF and Semi-CRF, respectively.

5.3 Background

5.3.1 Probabilistic structured predictor

In this chapter, we aim to produce a structured output $\boldsymbol{y}$ given an input sequence $\boldsymbol{x}$. To assess the compatibility between the input and output, we employ a parameterized score function $\boldsymbol{S}_\theta(\boldsymbol{y}|\boldsymbol{x})$. The probability of a structure $\boldsymbol{y}$ given $\boldsymbol{x}$ is computed as follows:

$$p_\theta(\boldsymbol{y}|\boldsymbol{x}) = \frac{\exp \boldsymbol{S}_\theta(\boldsymbol{y}|\boldsymbol{x})}{\sum_{\boldsymbol{y}' \in \mathcal{Y}(\boldsymbol{x})} \exp \boldsymbol{S}_\theta(\boldsymbol{y}'|\boldsymbol{x})} \tag{5.1}$$

where $\mathcal{Y}(\boldsymbol{x})$ represents the set of all possible outputs for $\boldsymbol{x}$, and the denominator serves as a normalization constant, referred to as the partition function, denoted by $\mathcal{Z}_\theta(\boldsymbol{x})$.

Training During training, the goal is to update the model's parameters θ to maximize the likelihood of the training data. The loss function for a pair of data points $(\boldsymbol{x}, \boldsymbol{y})$ is computed as follows:

$$\begin{aligned}\mathcal{L}(\boldsymbol{x}, \boldsymbol{y}) &= -\log p_\theta(\boldsymbol{y}|\boldsymbol{x}) \\ &= -\boldsymbol{S}_\theta(\boldsymbol{y}|\boldsymbol{x}) + \log \mathcal{Z}_\theta(\boldsymbol{x})\end{aligned} \tag{5.2}$$

This loss function can be optimized using a stochastic gradient descent algorithm on the training data. Computing the partition function $\mathcal{Z}_\theta(\boldsymbol{x})$ can be challenging when the output space is large, but it can be calculated efficiently using dynamic programming in some cases.

Inference During inference, the goal is to produce the most likely output:

$$\boldsymbol{y}^* = \underset{\boldsymbol{y} \in \mathcal{Y}(\boldsymbol{x})}{\arg\max}\, \boldsymbol{S}_\theta(\boldsymbol{y}|\boldsymbol{x}) \tag{5.3}$$

All the models we present in this chapter follow this type of probabilistic modeling. For the remainder of this chapter, we omit the dependency on θ for better readability.

5.3.2 Linear Chain CRF

The Linear-Chain CRF [Lafferty et al., 2001] is a sequence labeling model that assigns a label to each token in the input sequence, taking into account dependencies between adjacent labels. The score function of the CRF has the following form:

$$\boldsymbol{S}(\boldsymbol{y}|\boldsymbol{x}) = \sum_{i=1}^{|\boldsymbol{x}|} \boldsymbol{\psi}(y_i|\boldsymbol{x}) + \sum_{i=2}^{|\boldsymbol{x}|} \boldsymbol{T}[y_{i-1}, y_i] \tag{5.4}$$

Here, $\boldsymbol{\psi}(y_i|\boldsymbol{x}) \in \mathbb{R}$ is the sequence label score at position i, and $\boldsymbol{T} \in \mathbb{R}^{|Y|\times|Y|}$ is a learnable label transition matrix. The partition function is computed using the Forward algorithm and the Viterbi algorithm [Rabiner, 1989] is used to determine the optimal labeling, both with a computational complexity of $\mathcal{O}(L|Y|^2)$.

5.3.3 Semi-Markov CRF

In this section, we revisit the Semi-Markov CRF algorithm introduced in the previous chapter and provide additional insights. The Semi-CRF, proposed by [Sarawagi and Cohen, 2005], operates at the segment level and allows for the modeling of features that cannot be captured by traditional linear-chain CRFs. It produces a segmentation $\boldsymbol{y}$ of length M for an input sequence $\boldsymbol{x}$ of length L ($L \geq M$). A segmentation $\boldsymbol{y} = \{s_1, \ldots, s_M\} \in \mathcal{Y}(\boldsymbol{x})$ satisfies the following properties:

- Each segment $s_k = (i_k, j_k, l_k) \in \boldsymbol{y}$ consists of a start position i_k, an end position j_k, and a label $l_k \in Y$.

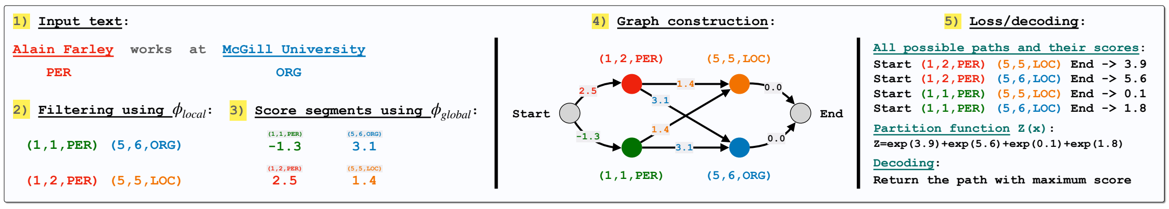

1) Input text:
Alain Farley works at McGill University
PER
ORG
2) Filtering using ϕ_{local}:
(1,1,PER) (5,6,ORG)
(1,2,PER) (5,5,LOC)
3) Score segments using ϕ_{global}:
(1,1,PER) -1.3
(5,6,ORG) 3.1
(1,2,PER) 2.5
(5,5,LOC) 1.4
4) Graph construction:
Start
End
(1,2,PER)
(5,5,LOC)
(1,1,PER)
(5,6,ORG)
2.5
-1.3
1.4
3.1
1.4
3.1
0.0
0.0
5) Loss/decoding:
All possible paths and their scores:
Start (1,2,PER) (5,5,LOC) End -> 3.9
Start (1,2,PER) (5,6,LOC) End -> 5.6
Start (1,1,PER) (5,5,LOC) End -> 0.1
Start (1,1,PER) (5,6,LOC) End -> 1.8
Partition function Z(x):
Z=exp(3.9)+exp(5.6)+exp(0.1)+exp(1.8)
Decoding:
Return the path with maximum score

Figure 5.1: Filtered Semi-CRF for NER. The model takes as text sequence and output the best entity segments.

- The segments have positive lengths and completely cover the input sequence positions $1, \dots, L$ *without overlapping*. In other words, the start and end positions satisfy $i_1 = 1$, $j_M = L$, and for every j_k and i_k we have $1 \leq i_k \leq j_k \leq L$, and $i_{k+1} = j_k + 1$.

Consider a sentence from a Named Entity Recognition (NER) task: "*Alain Farley* works at *McGill University*". It would be segmented as $\boldsymbol{y}$=[(1,2,PER), (3,3,O), (4,4,O), (5,6,ORG)], considering assumption from Sarawagi and Cohen [2005] that non-entity segments (referred to as O or `null` segments) have unit length. Furthermore, the Semi-CRF score function is defined as follows:

$$\boldsymbol{S}(\boldsymbol{y}|\boldsymbol{x}) = \sum_{k=1}^{M} \boldsymbol{\phi}(s_k|\boldsymbol{x}) + \boldsymbol{T}[l_{k-1}, l_k] \tag{5.5}$$

Here, $\boldsymbol{\phi}(s_k|\boldsymbol{x}) \in \mathbb{R}$ represents the score of the k-th segment of $\boldsymbol{y}$, and $\boldsymbol{T}[l_{k-1}, l_k]$ denotes the label transition score. Additionally, $\boldsymbol{T}[l_0, l_1] = 0$. The partition function of the Semi-CRF can be computed in polynomial time using a modified version of the Forward algorithm and the segmental Viterbi algorithm is used to compute optimal segmentation. The computational complexity of the Semi-CRF increases quadratically with both the sequence length and the number of labels, *i.e* $\mathcal{O}(L^2|Y|^2)$.

5.3.4 Graph-based Formulation of Semi-CRF

In this section, we present a graph-based formulation of the Semi-CRF. As explained in § 5.3.3 , a sequence $\boldsymbol{x}$ of length L is divided into M labeled segments (i_k, j_k, l_k), with i_k, j_k and l_k denoting respectively the start position, end position and the label. We define a directed graph $\mathcal{G}(V_{\text{full}}, E)$, with V_{full} its set of nodes composed of all possible segments s_k of $\boldsymbol{x}$:

$$V_{\text{full}} = \bigcup_{i=1}^{L} \bigcup_{j=i}^{L} \bigcup_{l=1}^{|Y|} \{(i, j, l)\}, \tag{5.6}$$

and an edge $s_{k'} \to s_k \in E$ exists if and only if the start position of s_k immediately follows the end position of $s_{k'}$, i.e., $j_{k'} + 1 = i_k$. The weight of an edge $s_{k'} \to s_k$ is defined as:

$$w(s_{k'} \to s_k|\boldsymbol{x}) = \phi(s_k|\boldsymbol{x}) + \boldsymbol{T}[l_{k'}, l_k], \tag{5.7}$$

where $\phi(s_k|\boldsymbol{x})$ is the score of the segment s_k and $\boldsymbol{T}[l_{k'}, l_k]$ is the label transition score. Moreover, Any directed path $s_1, s_2, \ldots, s_M$ of the graph $\mathcal{G}$ corresponds to a valid segmentation of $\boldsymbol{x}$ if it verifies the segmentation properties described in § 5.3.3 . Additionally, the score of a valid path is computed as the sum of the edge scores, and is equivalent to the Semi-CRF score of the segmentation (*Eq.* 5.5):

$$\boldsymbol{S}(s_1, \ldots, s_M|\boldsymbol{x}) = \sum_{k=1}^{M} w(s_{k-1} \to s_k|\boldsymbol{x}) \tag{5.8}$$

$$= \sum_{k=1}^{M} \phi(s_k|\boldsymbol{x}) + \boldsymbol{T}[l_{k-1}, l_k] \tag{5.9}$$

The search for the best segmentation of the sequence $\boldsymbol{x}$ is equivalent to finding the maximal weighted path of the graph $\mathcal{G}$ that starts at $i_1 = 1$ and ends at $j_M = L$. This search can be done using a generic shortest path algorithm such as *Bellman-Ford*, whose complexity is of L^3. Nevertheless, taking into account the lattice structure of the problem, the Viterbi algorithm [Viterbi, 1967, Rabiner, 1989] can achieve this while reducing the complexity to L^2.

5.4 Filtered Semi-Markov CRF

In this section, we propose an alternative model to Semi-CRF, named Filtered Semi-CRF, which aims to address two fundamental weaknesses of the original model. First, the Semi-CRF is not well-suited for long texts due to its *quadratic complexity* and the prohibitively large search space. Secondly, in tasks such as Named Entity Recognition (NER), where certain segments are labeled as `null` (representing non-entity segments), the Semi-CRF graph can create *multiple redundant paths*, all leading to the same set of entities. For instance, consider the scenario described in § 5.3.3. In this scenario, multiple segmentations, such as $\boldsymbol{y}$=[(1,2,PER), (3,3,O), (4,4,O), (5,6,ORG)] or $\boldsymbol{y}$=[(1,2,PER), (3,4,O), (5,6,ORG)], would yield the same final set of labeled entities, specifically (1,2, PER) and (5,6,ORG) in this case. To remedy these shortcomings, our proposed model incorporates a filtering step that eliminates irrelevant segments using a lightweight local classifier. By leveraging transformer-based features, this classifier effectively selects high-quality candidate segments, significantly reducing the task of the Semi-CRF to merely choosing the best among already high-quality candidates.

5.4.1 Filtering

Local classifier We first define the local classifier ϕ_{local} as a model that assigns a score to a labelled segment $s = (i, j, l)$ given an input sequence $\mathbf{x}$:

$$\phi_{\text{local}}(s = (i, j, l)|\boldsymbol{x}) = \boldsymbol{w_l}^T f(\boldsymbol{h}_i, \ldots, \boldsymbol{h}_j) \tag{5.10}$$

where $\mathbf{h}_i \in \mathbb{R}^D$ is the token representation at position i (computed by a pretrained transformer such as BERT), and $\mathbf{w}_l \in \mathbb{R}^D$ is a learnable weight associated with the label l. The function f represents the segment featurizer, which aggregates token representations into a single feature representation. We found that a simple sum operation provides strong performance across settings.

Filtered graph The filtering consists in removing the segments $s_k = (i_k, j_k, l_k)$ for which $l_k = \arg\max_l \phi_{local}(i_k, j_k, l|x)$ and $l_k = \texttt{null}$:

$$V = \left\{ (i_k, j_k, l_k) \in V_{\text{full}} \;\middle|\; \begin{array}{l} l_k = \arg\max_l \phi_{local}(i_k, j_k, l|x) \\ \wedge \\ l_k \neq \texttt{null} \end{array} \right\} \tag{5.11}$$

This new set of filtered nodes V requires to define the set of edges E differently from the definition of § 5.3.4. Thus, we propose to define the edges following Liang et al. [1991]: $\forall(s_{k'}, s_k) \in V^2$, $s_{k'} \to s_k \in E$ if $j_{k'} < i_k$ and there is no $s_{k^*} \in V$ such that $j_{k'} < i_{k^*}$ and $i_{k^*} < j_k$. This definition means that $s_{k'} \to s_k$ is an edge if the start position of s_k follows the end position of $s_{k'}$, and that no other segment lies completely in between these two positions $(j_{k'}, i_k)$. This formulation generalizes the Semi-CRF to graphs with missing segments. However, with missing segments, the starting and ending positions of segmentations do not necessarily verify $i_1 = 1$ and $j_M = L$. Thus, we simply add two terminal nodes `start` and `end`, verifying:

$$\begin{cases} \texttt{start} \to s_k \in E \textit{ iff } \forall k' \neq \texttt{start}, s_{k'} \to s_k \notin E \\ s_k \to \texttt{end} \in E \textit{ iff } \forall k' \neq \texttt{end}, s_k \to s_{k'} \notin E \end{cases} \tag{5.12}$$

In this context, a segmentation is simply a path in the graph starting at `start` and ending at `end` node (see Figure 5.1). Referring back to the example in § 5.3.3, the correct segmentation of "*Alain Farley* works at *McGill University*" using the Filtered Semi-CRF would be $\boldsymbol{y}$=[`start`, (1, 2, PER), (5, 6, ORG), `end`], where all remaining part of the segmentation are considered as having `null` labels.

5.4.2 Segmentation scoring

In the filtered graph, the score of a segmentation, $\boldsymbol{y} = \{\texttt{start}, s_1, \ldots, s_M, \texttt{end}\}$ is computed by summing its edge scores as for the Semi-CRF described in § 5.3.4:

$$\begin{aligned} \boldsymbol{S}(\boldsymbol{y}|\boldsymbol{x}) &= \sum_{s_k \in \boldsymbol{y}} w(s_{k'} \to s_k|\boldsymbol{x}) \\ &= \sum_{s_k \in \boldsymbol{y}} \boldsymbol{\phi}_{global}(s_k|\boldsymbol{x}) + \boldsymbol{T}[l_{k'}, l_k] \end{aligned} \tag{5.13}$$

where ϕ_{global} is a model that computes score of the nodes/segments in the filtered graph, defined similarly as ϕ_{local} in § 5.4.1 and they share the same feature f. $\boldsymbol{T}[l_{k'}, l_k]$ represents the transition score between the adjacent labels. By default, we set $w(\texttt{start} \to s_1) = \phi_{global}(s_1|\boldsymbol{x})$ and $w(s_M \to \texttt{end}) = 0$. See figure 5.1 for a visual example.

5.5 Training

In this section, we present our FSemiCRF training which involves updating the whole model parameters to minimize the following loss function:

$$\mathcal{L} = \mathcal{L}_{local} + \mathcal{L}_{global} \tag{5.14}$$

Here, $\mathcal{L}_{local}$ and $\mathcal{L}_{global}$ represent the filtering loss and the segmentation loss, respectively.

5.5.1 Filtering loss

The filtering loss is the sum of the negative log-probability of all gold-labeled segments, V^*:

$$\begin{aligned}\mathcal{L}_{local} &= -\sum_{(i,j,l^*)\in V^*} \log p(i,j,l^*|\boldsymbol{x}) \\ &= -\sum_{(i,j,l)\in V^*} \log \frac{\exp \phi_{local}(i,j,l|\boldsymbol{x})}{\sum_{l'} \exp \phi_{local}(i,j,l'|\boldsymbol{x})}\end{aligned} \tag{5.15}$$

In practice, we assign a lower weight to the loss of null segments to account for the imbalanced nature of the task. For that, we down-weight the loss for the label $l = \texttt{null}$ by a ratio $\beta \in]0,1]$, tuned on the dev set.

5.5.2 Segmentation loss

The segmentation loss is the negative log-likelihood of the gold path $\boldsymbol{y}$ in the filtered graph:

$$\mathcal{L}_{global} = -\boldsymbol{S}(\boldsymbol{y}|\boldsymbol{x}) + \log \mathcal{Z}(\boldsymbol{x}) \tag{5.16}$$

$\boldsymbol{S}(\boldsymbol{y}|\boldsymbol{x})$ is the segmentation score as per § 5.4.2, and the partition function $\mathcal{Z}(\boldsymbol{x})$, the sum of exponentiated scores for all valid paths in the graph from `start` to `end`. It can be computed efficiently via a message-passing algorithm [Wainwright and Jordan, 2008]:

Algorithm 2: Computing $\mathcal{Z}(\boldsymbol{x})$

```
Topologically sort the nodes in $V$
$\alpha[\texttt{start}] = 1$ and $\alpha[k] = 0$ otherwise for $k \in V$
forall $k \neq \texttt{start}$ in $V$ do
    forall $k'$ such that $k' \to k \in E$ do
        $\alpha[k] \leftarrow \alpha[k] + \alpha[k'] \exp\{w(s_{k'} \to s_k)|\boldsymbol{x}\}$
$\mathcal{Z}(\boldsymbol{x}) = \alpha[\texttt{end}]$
```

In practice, this implementation of $\mathcal{Z}(\boldsymbol{x})$ can be unstable, thus, all computations were performed in log space to prevent issues of overflow or underflow. The complexity of the algorithm is $\mathcal{O}(|V|+|E|)$ as it performs a topological sort (which visits each node and edge once), and then iterates over each node and its incoming edges exactly once, performing constant time operations.

During training, we impose certain constraints to ensure that the gold segmentation $\boldsymbol{y}$ forms a valid path in the *filtered* graph (with nodes V), which is critical for maintaining a positive loss, i.e., $\log \mathcal{Z}(\mathrm{x}) > \boldsymbol{S}(\mathrm{y}|\mathrm{x})$: 1) All segments in V that do not overlap with at least one segment from the gold segmentation $\boldsymbol{y}$ are excluded. 2) All segments from the gold segmentation, even those not initially selected in the filtering step, are included in V.

5.5.3 Inference

During inference, the first step is to obtain the candidate segments V through filtering, and then constructing the graph $\mathcal{G}(V, E)$ (see § 5.4.1). The final results is obtained by identifying the path, from `start` to `end`, in the graph that has the highest score. We achieve this by using a max-sum dynamic programming algorithm, which has a similar structure to Algorithm 2:

```
Algorithm 3: Decoding
  Topologically sort the nodes in $V$
  $\delta[\texttt{start}] = 0$
  forall $k \neq \texttt{start}$ in $V$ do
      $\delta[k] = \max_{(k' \to k) \in E}^{k'} \delta[k'] + w(s_{k'} \to s_k|\boldsymbol{x})$
  $\boldsymbol{y}^* = \texttt{Traced}(\delta[\texttt{end}])$
```

The highest scoring path $\boldsymbol{y}^*$, represented by $\mathrm{argmax}_{\boldsymbol{y}} \boldsymbol{S}(\boldsymbol{y}|\boldsymbol{x})$, is identified by the path traced by $\delta[\texttt{end}]$, which can be obtained through backtracking. This algorithm has a computational complexity of $\mathcal{O}(|V|+|E|)$, the same as that of computing the partition function $\mathcal{Z}(\boldsymbol{x})$ in Algorithm 2.

5.5.4 Complexity analysis

In this section, we analyze the complexity of the algorithms 2 and 3, $\mathcal{O}(|V| + |E|)$, as a function of the input sequence length L. Note that the size of V does not depend on the number of labels $|Y|$ since there is at most one label per segment due to the filtering step in equation 5.11.

Proposition 5.5.1. *The number of nodes in a Semi-CRF graph (as described in § 5.3.4) with an input length of L is given by $\frac{L(L+1)}{2}$.*

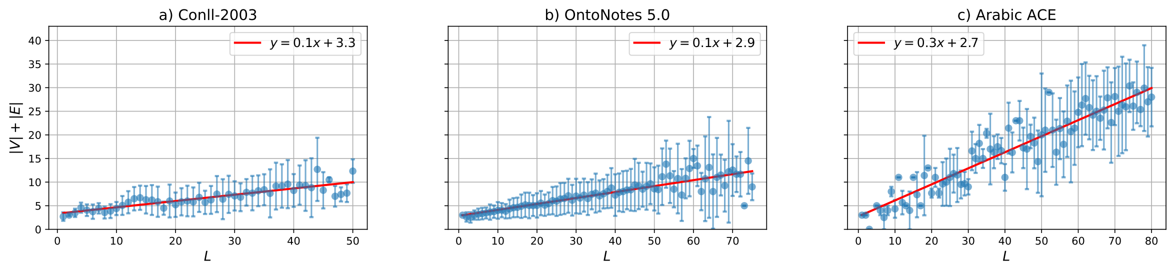


Figure 5.2: **Empirical complexity analysis**. We conducted an empirical complexity analysis using trained Filtered Semi-CRF models. The plot showcases the relationship between the size of the filtered graph ($|V| + |E|$) and the input sequence length L on three NER datasets. As the length of the input sequence increases, the graph size seems to grow in a linear fashion.

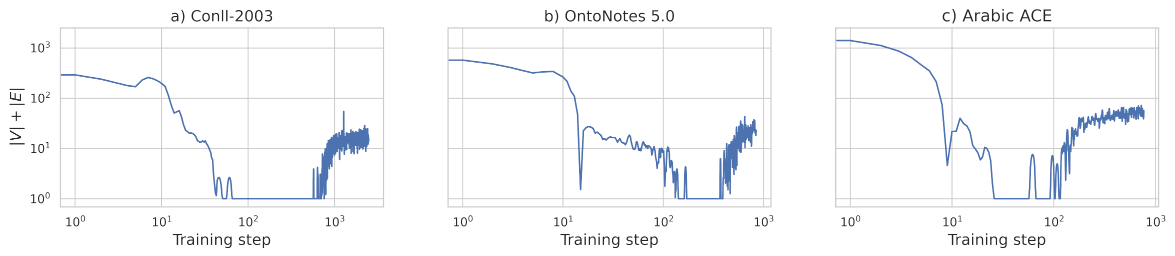


Figure 5.3: **Graph Size during Training.** The graph size ($|V| + |E| + 1$) undergoes three stages during training: 1) initially large when the filtering classifier is untrained, 2) decreasing in the second stage as most of segments in the training set have a `null` label (biasing the classifier toward this label), and 3) increasing again as the classifier improves, better aligning with the training dataset statistics.

Proof. Nodes are the enumeration of all segments (regardless of labels). Thus,

$$\begin{aligned} V = \bigcup_{i=1}^{L}\bigcup_{j=i}^{L}(i,j) \implies |V| &= \sum_{i=1}^{L}\sum_{j=i}^{L} 1 = \sum_{i=1}^{L}(L+1-i) \\ &= \sum_{i=1}^{L}(L+1) - \sum_{i=1}^{L} i = L(L+1) - \frac{L(L+1)}{2} \\ |V| &= \frac{L(L+1)}{2} \end{aligned} \tag{5.17}$$

□

Proposition 5.5.2. *The number of edges in a Semi-CRF graph (as described in § 5.3.4) with an input length of* L *is given by* $\frac{L(L-1)(L+1)}{6}$.

Proof. We know that in the complete segment graph

1. By definition, $(i_k, j_k) \rightarrow (i_{k'}, j_{k'}) \in E$ iff $j_k + 1 = i_{k'}$

2. There are j_k segments ending at j_k *i.e* $|\bigcup_{i=1}^{j_k}(i, j_k)| = j_k$

3. There are $L - j_k$ segments starting at $i_{k'}$ *i.e* $|\bigcup_{i=i_{k'}}^{L}(i_{k'}, i)| = L - i_{k'} + 1 = L - j_k$

From 1, 2 and 3, we can deduce that there is $j_k(L - j_k)$ segments starting at $i_{k'}$ and ending at j_k. Finally, the total number of edges of the graph is the sum over all j_k from 0 to L:

$$\begin{aligned}
|E| &= \sum_{j_k=1}^{L} j_k(L - j_k) = L\sum_{j_k=1}^{L} j_k - \sum_{j_k=1}^{L} j_k^2 \\
&= L\frac{L(L+1)}{2} - \frac{L(L+1)(2L+1)}{6} = L(L+1)(\frac{L}{2} - \frac{2L+1}{6}) \qquad (5.18)\\
|E| &= \frac{L(L+1)(L-1)}{6}
\end{aligned}$$

□

We employ these propositions to determine the complexity of the Filtered Semi-CRF model in the following.

Worst case complexity In the worst case scenario, the filtering model ϕ_{local} does not filter any segments, resulting in all segments being retained. By utilizing Propositions 3.1 and 3.2, we can deduce that in the worst case, $\mathcal{O}(|V|) = \mathcal{O}(L^2)$ and $\mathcal{O}(|E|) = \mathcal{O}(L^3)$. This implies that the complexity of our algorithm in the worst case is cubic with respect to the sequence length L, as $\mathcal{O}(|V| + |E|) = \mathcal{O}(L^3)$. However, it is worth noting that in this worst case scenario, the resulting graph is the Semi-CRF and the complexity can be reduced to L^2 by utilizing the Forward algorithm during training and the Viterbi algorithm during inference.

Best Case Complexity In the ideal scenario, the filtering process is optimal, resulting in the number of nodes in the graph $|V|$ being equal to the true number of non-`null` segments in the input sequence, denoted by $\mathcal{S}$. Furthermore, since $\mathcal{S}$ does not contain overlapping segments, $|\mathcal{S}| \leq L$ with $|\mathcal{S}| = L$ if all segments in $\mathcal{S}$ have unit length and cover the entire sequence, i.e., $\mathcal{S} = \{(i, i, l_i) | i = 1 \ldots L, l_i \neq \texttt{null}\}$. Additionally, $|E| = |\mathcal{S}| - 1 \leq L - 1$ as optimal filtering implies that the path number is unique. As a result, in this best case scenario, the complexity of the algorithm is linear with respect to the sequence length L, i.e., $\mathcal{O}(L)$.

Empirical Analysis In this study, we assess our model's empirical complexity by examining the correlation between the graph size ($|V| + |E|$) and the input sequence length L. We use three popular NER datasets for this analysis - CoNLL-2003, OntoNotes 5.0, and Arabic ACE. Our findings (shown in Figure 5.2) indicate a linear increase in the graph size as the sequence length increases. Interestingly, the graph size always stays smaller than the sequence length. This suggests that in practice, the computational complexity of the

Models	CoNLL-2003			OntoNotes 5.0			Arabic ACE		
	P	R	F	P	R	F	P	R	F
Yu et al. [2020b]	93.7	93.3	93.5	91.1	91.5	91.3	-	-	-
Yan et al. [2021a]	92.61	93.87	93.24	89.99	90.77	90.38	-	-	-
Zhu and Li [2022]	93.61	93.68	93.65	91.75	91.74	**91.74**	-	-	-
Shen et al. [2022]	93.29	92.46	92.87	91.43	90.73	90.96	-	-	-
Zaratiana et al. [2022a]	94.29	93.33	93.81	90.21	91.21	90.71	85.35	83.64	**84.49**
El Khbir et al. [2022]	-	-	-	-	-	-	84.42	84.05	84.23
	Our experiments								
CRF	93.29	92.21	92.75	89.00	90.16	89.57	82.79	84.44	83.61
Semi-CRF	92.37	90.49	91.42	88.91	89.78	89.34	82.97	84.24	83.60
+ Unit size `null`†	92.08	91.41	91.74	89.17	89.76	89.47	83.35	83.62	83.48
FSemiCRF	94.72	93.09	**93.89**	90.69	91.31	91.00	83.43	85.51	**84.46**
– w/o $\mathcal{L}_{global}$ (5.16)†	94.24	92.70	93.46	90.85	89.57	90.21	83.73	83.56	83.64

Table 5.1: Main Results. All English models employ `bert-large-cased` as token representations for English datasets, except [Yan et al., 2021a] that uses `bart-large`. † See the ablation study (§ 5.8) for details.

FSemiCRF model is at worst, $\mathcal{O}(L)$. However, during the initial stages of model training, the graph size may be large because the filtering model, which is responsible for reducing the graph size, is not fully trained, as depicted in Figure 5.3. But, the graph size decreases rapidly after a few training steps as the filtering classifier is improving.

5.6 Experimental setups

Datasets and evaluation We evaluate our models on on three diverse Named Entity Recognition (NER) datasets: CoNLL-2003 and OntoNotes 5.0, both English, and Arabic ACE. We adopt the standard NER evaluation methodology, calculating precision (P), recall (R), and F1-score (F), based on the exact match between predicted and actual entities.

Hyperparameters To produce contextual token representations, we used bert-large-cased for both CoNLL-2003 and OntoNotes 5.0 datasets, and bert-base-arabertv2 [Antoun et al., 2020] for Arabic ACE. For simplicity, we do not use auxiliary embeddings (eg. *character embeddings*). All models are trained with *Adam* optimizer [Kingma and Ba, 2017]. We employed a learning rate of `2e-5` for the pre-trained parameters and a learning rate of `5e-4` for the other parameters. We used a batch size of 8 and trained for a maximal epoch of 15. We keep the best model on the validation set for testing. In this work, for all segment-based model, we restrict the segment to a maximum width K to reduce complexity without harming the recall score on the training set (however some segments may be missed for the test set). By bounding the maximum width of the segments, we reduce the number of segments from L^2 to LK. Under this setup, the complexity of the Semi-Markov CRF becomes $O(LK|Y|^2)$. We implemented our model with PyTorch [Paszke et al., 2019a]. The pre-trained transformer models were loaded from HuggingFace's Transformers library, we used AllenNLP [Gardner et al., 2018b] for data preprocessing and the seqeval library [Nakayama, 2018] for evaluating the sequence labeling models. Our Semi-CRF implementation is based on pytorch-struct [Rush, 2020a]. We trained all the models on a server with V100 GPUs.

Baselines We compare our Filtered Semi-CRF model against the CRF [Lafferty et al., 2001] and Semi-CRF [Sarawagi and Cohen, 2005]. Additionally, we include results from previous studies: BiaffineNER [Yu et al., 2020b], BartNER [Yan et al., 2021a], Boundary Smoothing [Zhu and Li, 2022], PIQN [Shen et al., 2022], GSS [Zaratiana et al., 2022a] and ArabIE El Khbir et al. [2022]. For English datasets, models use `bert-large-case` (except BartNER with `bart-large`). Our model for Arabic data utilizes `bert-base-arabertv2`.

5.7 Main results

FSemiCRF vs. CRF and Semi-CRF As shown in Table 5.1, our FSemiCRF model outperforms both the CRF and Semi-CRF reference models in all datasets, validating its effectiveness. The Semi-CRF model, while providing competitive results, often lags behind, either matching or slightly underperforming the CRF model. This observation is in line with the findings of Liang [2005].

Comparison to pior works In our work, we mainly compare our approach with previous work that we consider comparable, i.e. that uses sentence-level context and the same backbone model. As shown in the Table 5.1, on all datasets, we found that our FSemiCRF achieves competitive results on all the datasets. For example, our approach outperforms a span-based model we proposed earlier Zaratiana et al. [2022a], which uses the Maximum weighted independent set to select the best spans.

5.8 Ablation study

Semi-CRF + Unit `null` We study a variation of the Semi-CRF that only allows for the use of `null` labels for unit length segments. To do this, we simply modify the original Semi-CRF by eliminating/masking segmentation paths that contain null segments with a size greater than one. The motivation for this study is to fix the *multiple redundant paths* problem of the Semi-CRF (§ 5.4). The results show that this approach improves performance on most of the datasets, but still does not perform as well as the other methods, thus

	CoNLL-2003 (|Y| = 4)			*OntoNotes 5.0 (|Y| = 18)*			*Arabic ACE (|Y| = 7)*		
	CRF	Semi-CRF	FSemiCRF	CRF	Semi-CRF	FSemiCRF	CRF	Semi-CRF	FSemiCRF
Scoring	3.9	3.9	3.9	4.8	4.9	4.9	8.1	8.3	8.3
Decoding	2.7	3.7	0.2	4.4	27.5	0.2	6.0	10.1	0.3
Decoding Speedup	1.3x	1.0x	**18.5x**	6.2x	1.0x	**137x**	1.7x	1.0x	**33.7x**
Overall	6.6	7.6	4.1	9.2	32.4	5.1	14.1	18.4	8.6
Overall Speedup	1.1x	1.0x	**1.8x**	3.5x	1.0x	**6.3x**	1.30x	1.0x	**2.1x**

Table 5.2: Inference Wall Clock Time (lower is better). Comparison of required wall-clock time for the scoring (tokens for CRF, segments for Semi-CRF/FSemiCRF) and decoding processes, measured in *milliseconds / sample.*

validating the importance of segment filtering.

FSemiCRF w/o global loss We investigate the impact of removing the global loss (*Eq.* 5.16) component on our FSemiCRF model, resulting in a local span classfication model. The results are presented in Table 5.1 (w/o $\mathcal{L}_{global}$), and show that even without the global loss component, the model still performs competitively. However, including global loss consistently improves the overall scores of FSemiCRF across all the datasets.

5.8.1 Efficiency analysis

This section focuses on the computational efficiency of different models, for both training and inference. For this experiment, all the models use a base size for the encoder to ensure a fair comparison.

Inference wall clock time The wall clock time analysis for scoring and decoding operations, summarized in Table 5.2, highlights subtle differences in scoring times across all models. However, when it comes to decoding, FSemiCRF significantly outperforms both CRF and Semi-CRF models on all datasets. Notably, FSemiCRF achieves a remarkable 137x speedup over Semi-CRF on the OntoNotes 5.0. Overall, FSemiCRF demonstrates superior performance, being up to 6x and 2x faster than CRF and Semi-CRF, respectively.

Training throughput Figure 6.6 presents the training throughput of the models, which measures the number of batches processed per second using a batch size of 8. It reveals that, in general, CRF is the fastest during training, with FSemiCRF following closely as the second fastest model. This can be attributed to the larger graph size of FSemiCRF during training, particularly in the early stages, which can potentially slow down the process, as discussed in the complexity analysis (5.5.4). However, the differences in training performance between the models are not as pronounced as during inference.

5.9 Related Work

Linear-chain CRF Numerous frameworks exist for text segmentation. The commonly used Linear-Chain CRF [Lafferty et al., 2001] treats this task as token-level prediction, training through sequence-level objectives and using the Viterbi algorithm [Viterbi, 1967, Forney, 2010] for decoding. Variants have evolved from using hand-crafted features [Lafferty et al., 2001, Gross et al., 2006, Roth and tau Yih, 2005] to automated feature learning through neural networks [Do and Artières, 2010, van der Maaten et al., 2011, Kim et al., 2015, Huang et al., 2015, Lample et al., 2016a]. Higher order dependencies (Markov order $N > 1$) have been explored for enhanced performance, but their adoption is limited due to complexity and marginal gains [Ye et al., 2009, Cuong et al., 2014].

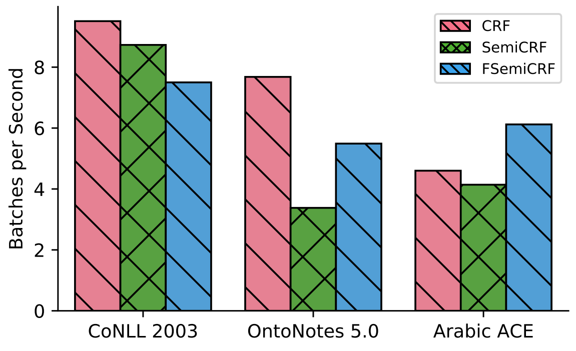


Figure 5.4: Training throughput in *batches per second.*

Semi-Markov CRF Semi-CRF [Sarawagi and Cohen, 2005] is an alternative operating at segment level, applied to tasks like Chinese word segmentation [Kong et al., 2016] and Named Entity Recognition [Sarawagi and Cohen, 2005, Andrew, 2006, Zhuo et al., 2016, Liu et al., 2016b, Ye and Ling, 2018]. It has the advantage of incorporating segment-level features but suffers from quadratic complexity and generally equivalent or marginally better performance than CRFs [Liang, 2005, Daumé and Marcu, 2005, Andrew, 2006].

Dynamic Programming Pruning Prior research has investigated the use of pruning techniques in dynamic programming to improve the efficiency of structured prediction tasks [Roark and Hollingshead, 2008, Rush and Petrov, 2012, Bodenstab et al., 2011, Vieira and Eisner, 2017]. These approaches aim to optimize runtime by selectively discarding hypotheses during inference. However, these methods often involve a trade-off between efficiency and performance. In contrast, our Filtered Semi-CRF model introduces a learned filtering step that collaboratively improves both efficiency and overall model performance.

Named Entity Recognition NER is an important task in Natural Language Processing and is used in many downstream information extraction applications such as relation extraction [Zaratiana et al., 2023b] and taxonomy construction [Zhang et al., 2018, Dauxais et al., 2022]. Usually, NER tasks are designed as sequence labelling [Huang et al., 2015, Lample et al., 2016a, Akbik et al., 2018] where the goal is to predict tagged sequence (eg. BIO tags). Recently, different approaches have been proposed to perform NER tasks that go beyond traditional sequence labelling. One approach that has been widely adopted is the span-based approach [Liu et al., 2016a, Fu et al., 2021, Li et al., 2021b, Zaratiana et al., 2022a,c,d, Lou et al., 2022, Corro, 2023] where the prediction is done in the span level instead of entity level. Futhermore, the use of the sequence-to sequence models for Named Entity Recognition has become popular recently. For instance, Yan et al. [2021a] uses the BART [Lewis et al., 2019] model to generate named entity using encoder-decoder with copy mechanism.

5.10 Conclusion

In this chapter, we introduce Filtered Semi-CRF, a novel algorithm for text segmentation tasks. By applying our method to NER, we show substantial performance gains over traditional CRF and Semi-CRF models on several datasets. Additionally, our algorithm exhibits improved efficiency, speed, and scalability compared to the baselines. As future work, we plan to investigate the extension of Filtered Semi-CRF to nested segment structures.

Chapter 6

GLiNER: Generalist Model for Named Entity Recognition using Bidirectional Transformer

6.1 Context

Named Entity Recognition (NER) is essential in various Natural Language Processing (NLP) applications. Traditional NER models are effective but limited to a set of predefined entity types. In contrast, Large Language Models (LLMs) can extract arbitrary entities through natural language instructions, offering greater flexibility. However, their size and cost, particularly for those accessed via APIs like ChatGPT, make them impractical in resource-limited scenarios. In this chapter, we introduce a compact NER model trained to identify any type of entity. Leveraging a bidirectional transformer encoder, our model, GLiNER, facilitates parallel entity extraction, an advantage over the slow sequential token generation of LLMs. Through comprehensive testing, GLiNER demonstrate strong performance, outperforming both ChatGPT and fine-tuned LLMs in zero-shot evaluations on various NER benchmarks.

6.2 Introduction

Named Entity Recognition plays a crucial role in various real-world applications, such as constructing knowledge graphs [Ye et al., 2022b, Zaratiana et al., 2024]. Traditional NER models are limited to a predefined set of entity types.Expanding the number of entity types can be beneficial for many applications, but it may require labeling data and retraining a model, which can be costly and time-consuming. The emergence of large language models (LLMs), like GPT-3 [Brown et al., 2020], has introduced a new era for Open NER by enabling the extraction of any entity type using only natural language instructions. However, powerful LLMs typically comprise billions of parameters and thus require substantial com-

```python
# Installation: pip install gliner
from gliner import GLiNER

# load model
model = GLiNER.from_pretrained("urchade/gliner_base")

# choose labels
labels = ["person", "organization", "date"]

text = "Bill Gates founded Microsoft on April 4, 1975."

entities = model.predict_entities(text, labels)

for entity in entities:
    print(entity["text"], "=>", entity["label"])

## Expected output:
# Bill Gates => person
# Microsoft => organization
# April 4, 1975 => date
```

Figure 6.1: **GLiNER for open type NER.** GLiNER is capable of identifying any entity type using a bidirectional transformer encoder (BERT-like). It provides a practical alternative to traditional NER models, which are limited to predefined entities, and Large Language Models (LLMs) that, despite their flexibility, are costly and large for resource-constrained scenarios. GLiNER can be easily installed via pip and its pretrained models are hosted on HuggingFace, ensuring easy accessibility. Moreover, it can run efficiently on CPU, making GLiNER especially suitable for environments with limited computing power.

puting resources. Although it is possible to access some LLMs via APIs [OpenAI, 2023b], using them at scale can incur high costs.

In spite of cost considerations, researchers have recently explored the fine-tuning of open-source language models such as LLaMa [Touvron et al., 2023] for NER. Wang et al. [2023b], for example, introduced InstructUIE, a fine-tuned FlanT5-11B model [Raffel et al., 2019, Chung et al., 2022] on existing information extraction (IE) datasets, achieving high performance in zero-shot settings. Sainz et al. [2024] proposed GoLLIE as an extension of InstructUIE which work by fine-tuning a CodeLLaMa [Rozière et al., 2023] using detailed annotation guidelines, obtaining significant performance improvements. Another recent proposal by Zhou et al. [2024], called UniversalNER, involves the fine-tuning of LLMs using diverse datasets from various domains, annotated with ChatGPT instead of relying on standard NER datasets. Remarkably, their approach not only replicates but also surpasses the original capability of ChatGPT when evaluated in zero-shot settings.

While these approaches have achieved remarkable results, they present certain limitations we seek to address. They use autoregressive language models, which can be slow due to sequential generation. Moreover, these models have several billion parameters, limiting their deployment in compute-limited scenarios.

In this chapter, we propose a model that addresses the above-mentioned problems. Instead of relying on large autoregressive models, we use a smaller and more compute-

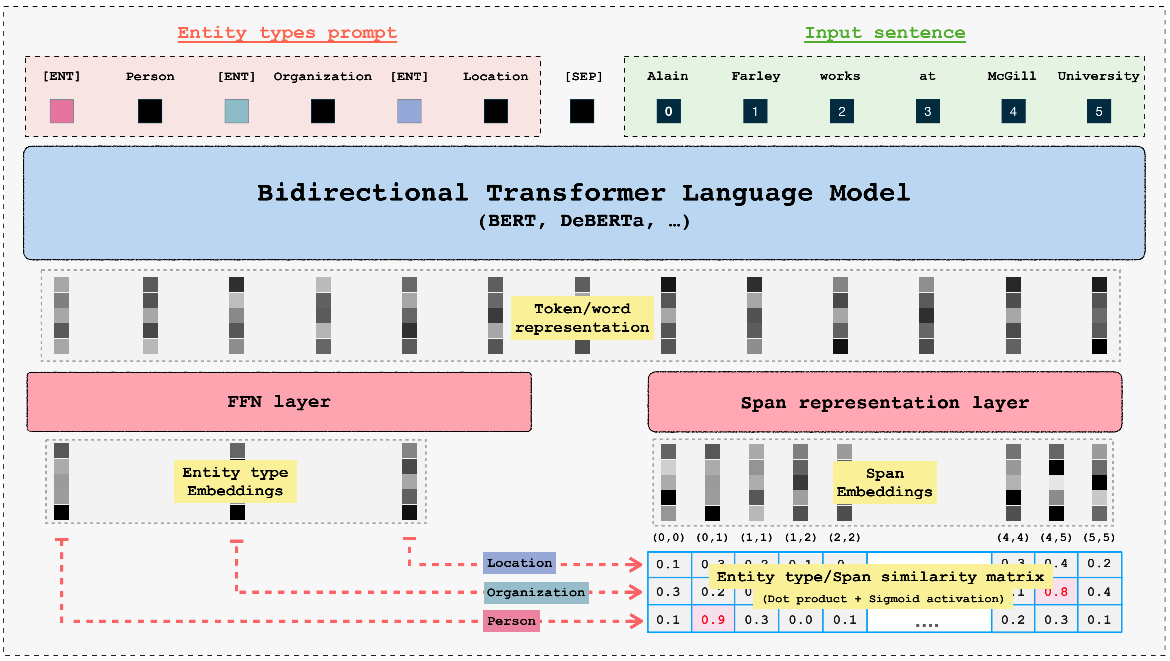


Figure 6.2: Model architecture. GLiNER employs a BiLM and takes as input entity type prompts and a sentence/text. Each entity is separated by a learned token [ENT]. The BiLM outputs representations for each token. Entity embeddings are passed into a FeedForward Network, while input word representations are passed into a span representation layer to compute embeddings for each span. Finally, we compute a matching score between entity representations and span representations (using dot product and sigmoid activation). For instance, in the figure, the span representation of (0, 1), corresponding to "Alain Farley," has a high matching score with the entity embeddings of "Person".

efficient Bidirectional Language Models (BiLM), such as BERT [Devlin et al., 2019] or DeBERTa [He et al., 2021]. The core concept of our model involves treating the task of Open NER as matching entity type embeddings to textual span representations in the latent space, rather than as a generation task. This approach naturally solves the scalability issues of autoregressive models and allows for bidirectional context processing, which enables richer representations. When trained on the Pile-NER dataset released by Zhou et al. [2024], which comprises texts from numerous domains and thousands of entity types, our model demonstrates impressive zero-shot performance. More specifilcally, it outperforms both ChatGPT and fine-tuned LLMs on various NER datasets without fine-tuning (Table 6.1). The robustness of our model is particularly highlighted by its capability to process languages that were not part of its training data. Notably, it outperforms ChatGPT in 8 out of 10 such languages, as detailed in Table 6.3.

6.3 Method

This section presents our model, GLiNER, which is trained to extract any type of entity using a Bidirectional Language Models. Our model has three main components: i) a pre-trained textual encoder (a BiLM such as BERT), ii) a span representation module which computes span embeddings from token embeddings, iii) an entity representation module which computes entity embeddings that the model seeks to extract. The goal is to have entity and span embeddings in the same latent space to assess their compatibility (degree of matching). The overall architecture of our model is depicted in Figure 6.2.

6.3.1 Architecture

Input format The input to our model comprises a unified sequence combining entity types (expressed in natural language) and the input text from which entities are to be extracted. The input format is as follows:

$$\boxed{\texttt{[ENT]}\ t_0\ \texttt{[ENT]}\ t_1\ \dots\ \texttt{[ENT]}\ t_{M-1}}\ \texttt{[SEP]}\ \boxed{x_0\ x_2\ \dots\ x_{N-1}}$$

[ENT] token represents a special token placed before each entity type and the [SEP] token functions as a delimiter, separating the sequence of entity types from the input text. They are initialized randomly at the start of training.

Token representation The token encoder processes the input to compute interactions between all tokens (both entity types and input text), producing contextualized representations. Let $\boldsymbol{p} = \{\boldsymbol{p}_i\}_0^{M-1} \in \mathbb{R}^{M \times D}$ denote the encoder's output for the [ENT] tokens, representing all entity types. Similarly, $\boldsymbol{h} = \{\boldsymbol{h}_i\}_0^{N-1} \in \mathbb{R}^{N \times D}$ denotes the representation of each word in the input text. For words tokenized into multiple subwords, we use the representation of the first subword, which is a standard choice in the NER literature.

Entity and Span Representation In our model, we aim to encode entity types and span embeddings into a unified latent space. The entity representation is computed by refining the initial representation $\boldsymbol{p}$ using a two-layer feedforward network, resulting in $\boldsymbol{q} = \{\boldsymbol{q}_i\}_0^{M-1} \in \mathbb{R}^{M \times D}$. The representation of a span starting at position i and ending at position j in the input text, $\boldsymbol{S}_{ij} \in \mathbb{R}^D$, is computed as:

$$\boldsymbol{S}_{ij} = \text{FFN}(\boldsymbol{h}_i \otimes \boldsymbol{h}_j) \tag{6.1}$$

Here, FFN denotes a two-layer feedforward network, and $\otimes$ represents the concatenation operation. Moreover, we set an upper bound to the length (K=12) of the span in order to keep linear complexity in the size of the input text, without harming recall.

Entity Type and Span Matching To evaluate whether a span (i, j) corresponds to entity type t, we calculate the following matching score:

$$\phi(i, j, t) = \sigma(\boldsymbol{S}_{ij}^T \boldsymbol{q}_t) \in \mathbb{R} \tag{6.2}$$

In this equation, σ denotes a sigmoid activation function. As we train with binary cross-entropy loss (see next sec. 6.3.2), $\phi(i, j, t)$ can be interpreted as the probability of the span (i, j) being of type t.

Model	Params	Movie	Restaurant	AI	Literature	Music	Politics	Science	Average
Vicuna-7B	7B	6.0	5.3	12.8	16.1	17.0	20.5	13.0	13.0
Vicuna-13B	13B	0.9	0.4	22.7	22.7	26.6	27.0	22.0	17.5
USM	0.3B	37.7	17.7	28.2	56.0	44.9	36.1	44.0	37.8
ChatGPT	–	5.3	32.8	52.4	39.8	66.6	68.5	**67.0**	47.5
InstructUIE	11B	**63.0**	21.0	49.0	47.2	53.2	48.1	49.2	47.2
UniNER-7B	7B	42.4	31.7	53.6	59.3	67.0	60.9	61.1	53.7
UniNER-13B	13B	48.7	36.2	54.2	60.9	64.5	61.4	63.5	55.6
GoLLIE	7B	**63.0**	**43.4**	**59.1**	62.7	67.8	57.2	55.5	58.0
GLiNER-S	50M	46.9	33.3	50.7	60.0	60.9	61.5	55.6	52.7
GLiNER-M	90M	42.9	37.3	51.8	59.7	69.4	68.6	58.1	55.4
GLiNER-L	0.3B	57.2	42.9	57.2	**64.4**	**69.6**	**72.6**	62.6	**60.9**

Table 6.1: Zero-Shot Scores on Out-of-Domain NER Benchmark. We report the performance of GLiNER with various deberta-v3 [He et al., 2021] model sizes. Results for Vicuna, ChatGPT, and UniNER are from Zhou et al. [2024]; USM and InstructUIE are from Wang et al. [2023b]; and GoLLIE is from Sainz et al. [2024].

6.3.2 Training

During training, our objective is to optimize model parameters to enhance the matching score for correct span-type pairs (positive pairs) and reduce it for incorrect pairs (negative pairs). A span (i, j) paired with an entity type t forms a positive pair ($s \in \mathcal{P}$) if the span is labeled with type t in the training data. Otherwise, it is a negative pair ($s \in \mathcal{N}$). The training loss for an individual example, comprising spans $\mathcal{S}$ and entity types $\mathcal{T}$, is defined as:

$$\begin{aligned}\mathcal{L}_{\text{BCE}} = -\sum_{s \in \mathcal{S} \times \mathcal{T}} & \mathbb{I}_{s \in \mathcal{P}} \log \phi(s) + \\ & \mathbb{I}_{s \in \mathcal{N}} \log (1 - \phi(s))\end{aligned} \tag{6.3}$$

The variable s represents a pair of span/entity type and $\mathbb{I}$ is an indicator function, which returns 1 when the specified condition is true and 0 otherwise. This loss function corresponds to binary cross-entropy.

6.3.3 Decoding algorithm

In the decoding phase, we employ a greedy span section that selects entity spans based on matching scores, to ensure task/dataset specific constraints. This strategy is applied independently to each sentence. Only, spans (i, j) with matching scores $\phi(i, j, c) > 0.5$ are considered for selection.

Flat NER: The algorithm chooses the highest-scoring non-overlapping span and continues this process until all spans are evaluated.

Nested NER: Similar to Flat NER, but the algorithm allows selection of fully nested spans within other entities while still avoiding partial overlaps.

Algorithm Efficiency: The decoding is implemented using a priority queue for spans, ensuring an $O(n \log n)$ complexity, with n being the number of candidate spans. Empirically, the size of n is usually lower than the input sequence length [Zaratiana et al., 2023a].

6.4 Experimental Setting

6.4.1 Training data

Our objective is to construct a versatile NER model capable of accurately identifying any entity types across different textual domains. To achieve this, it is essential that our training dataset includes a diverse range of entity types from various domains. For this, we utilize the training data released by Zhou et al. [2024], known as Pile-NER[1]. This dataset is derived from the Pile corpus [Gao et al., 2020], commonly used for pretraining large language models, and comprises text from diverse sources. More specifically, to construct the dataset Zhou et al. [2024] sampled 50,000 texts from the Pile data and employed ChatGPT to extract their associated entity types. Notably, they did not specify the entity types to the LLMs, aiming to extract a diverse range of entity types. They used the prompting approach shown in Figure 6.3.

Finally, after filtering bad outputs their datasets result in 44,889 passages containing in total 240k entity spans and 13k distinct entity types.

6.4.2 Hyperparameters

Our model, GLiNER, is trained on the Pile-NER dataset, which we described in the previous section. We use the deberta-v3 [He et al., 2021] as our backbone due to its proven empirical

[1]https://huggingface.co/datasets/Universal-NER/Pile-NER-type

```
System Message: You are a helpful information extraction
system.

Prompt: Given a passage, your task is to extract all
entities and identify their entity types. The output
should be in a list of tuples of the following format:
[("entity 1", "type of entity 1"), ... ].

Passage: {input_passage}
```

Figure 6.3: Prompting ChatGPT for entity extraction. This prompt was used Zhou et al. [2024] to construct the Pile-NER dataset.

performance. All non-pretrained layers have a width dimension of 768 and a dropout rate of 0.4. Regarding the training process, we employ the AdamW optimizer [Loshchilov and Hutter, 2017], setting a base learning rate of 1e-5 for pretrained layers (the transformer backbone) and 5e-5 for non-pretrained layers (FFN layers and span representation). We trained our models for a maximum of 30k steps, starting with a 10% warmup phase, followed by a decay phase using a cosine scheduler. The Pile-NER dataset natively contains only positive entities (i.e., entities that are present in the sentence), and we found it useful to include negative entity types during training. This is achieved by sampling random entities from other examples in the same batch. In addition, we follow the strategies outlined in Sainz et al. [2024] as a form of regularization, which includes *shuffling entity order* and *randomly dropping entities*. Furthermore, we limit the number of entity types to 25 per sentence during training. The larger variant of our model, GLiNER-L, takes 5 hours to train on an A100 GPU 80 GB memory.

6.4.3 Baselines

In our evaluation, we compare our model, GLiNER, with several recent models designed for Open NER. First, we examine chat models like **ChatGPT** and **Vicuna** [Chiang et al., 2023], which utilize the prompting from Ye et al. [2023]; we present their results as reported by Zhou et al. [2024].. We also compare our method to three recent Large Language Models (LLMs) that have been fine-tuned for NER: **InstructUIE** [Wang et al., 2023b], based on the FlanT5 11B model and fine-tuned on various NER datasets; **UniNER** [Zhou et al., 2024], which employs a LLaMa model fine-tuned on a dataset generated by ChatGPT; **GoLLIE** [Sainz et al., 2024], fine-tuned to adhere to detailed annotation guidelines for enhanced performance in unseen IE tasks, utilizing CodeLLama as its base model. Finally, we include **USM** [Lou et al., 2023] in our comparison, which is similar in size to ours but features a different architecture.

6.4.4 Evaluation

Datasets We primarily evaluate our model in a zero-shot (i.e, without fine-tuning on the target dataset) on common NER benchmarks, following previous works Wang et al. [2023b], Zhou et al. [2024]. The first is the *OOD NER Benchmark* (Table 6.1), which comprises seven diverse NER datasets from CrossNER [Liu et al., 2020b] and MIT [Liu et al., 2013]. This benchmark is typically used for evaluating out-of-domain generalization capabilities of NER models. The second benchmark consists of *20 NER datasets* (Table 6.2) from a wide range of domains, including biomedical, news articles, and tweets. These datasets are commonly used for training supervised NER models. Additionally, we evaluate our model on multilingual NER datasets (Table 6.3) for further investigation. For this purpose, we use the recently released *MultiCoNER* (Multilingual Complex NER) [Malmasi et al., 2022], which contains data in 11 languages across various domains.

Metric We adopt the standard NER evaluation methodology, calculating F1-score based on the exact match (span boundary and span type) between predicted and reference entities.

6.5 Results

6.5.1 Zero-shot on English datasets

In this section, we discuss the performance of our model in a zero-shot context, i.e., by only training on the Pile-NER dataset without further fine-tuning on target datasets.

OOD NER Benchmark We first evaluate our model on the OOD benchmark as reported in Table 6.1. We compare three different sizes of our model (small, medium, and large) against the baselines. The results demonstrate our model's impressive capability, irrespective of its size. For example, even our smallest model, with only 50M parameters, outperforms general-purpose models such as ChatGPT and Vicuna. It also shows better performance than the 11B InstructUIE, which has been instruction-tuned for the NER task. Furthermore, when compared to UniNER, which used the same training data as GLiNER, our medium-sized model (90M) achieves comparable results to UniNER-13B (55 F1 for both), despite being 140 times smaller. Meanwhile, our largest variant consistently outperforms UniNER by an average margin of 5 points.. Our best competitor, GoLLIE, which leads among the LLMs, achieves better performance than most of our models but is still less effective than GLiNER-L. When compared to USM, which has a comparable number of parameters to ours, our model demonstrates significantly superior performance, showing the superiority of our architecture.

20 NER Benchmark table 6.2 presents a comparison of our model against ChatGPT and UniNER across 20 diverse NER datasets. First, similar to the OOD benchmark, ChatGPT significantly lags behind fine-tuned models for NER. Furthermore, GLiNER achieves the high-

Dataset	**ChatGPT**	**UniNER-7B**	**GLiNER-L**
ACE05	26.6	**36.9**	27.3
AnatEM	30.7	25.1	**33.3**
bc2gm	40.2	46.2	**47.9**
bc4chemd	35.5	**47.9**	43.1
bc5cdr	52.4	**68.0**	66.4
Broad Tweeter	61.8	**67.9**	61.2
CoNLL03	52.5	**72.2**	64.6
FabNER	15.3	**24.8**	23.6
FindVehicle	10.5	22.2	**41.9**
GENIA	41.6	54.1	**55.5**
HarveyNER	11.6	18.2	**22.7**
MIT Movie	5.3	42.4	**57.2**
MIT Restaurant	32.8	31.7	**42.9**
MultiNERD	58.1	59.3	**59.7**
ncbi	42.1	60.4	**61.9**
OntoNotes	29.7	27.8	**32.2**
PolyglotNER	33.6	41.8	**42.9**
TweetNER7	40.1	**42.7**	41.4
WikiANN	52.0	55.4	**58.9**
WikiNeural	57.7	69.2	**71.8**
Average	36.5	45.7	**47.8**

Table 6.2: Zero-shot performance on 20 NER datasets. Results of ChatGPT and UniNER are reported from [Zhou et al., 2024].

est performance on 13 of these datasets, surpassing UniNER by an average of 2 points. This superior performance underscores GLiNER's robustness and adaptability across a broad spectrum of domains. However, a notable observation is that GLiNER underperforms compared to UniNER on tweet-based NER datasets. This highlights potential areas for improvement in GLiNER's ability to process informal, colloquial, or noisy data, typical of social media content.

6.5.2 Zero-Shot Multilingual Evaluation

In this section, we evaluate the performance of our model in a zero-shot context on unseen languages to assess its generalizability. This evaluation uses the MultiCONER [Malmasi et al., 2022], with results detailed in Table 6.3. Our model, GLiNER, is presented in two variants: **En**, which employs `deberta-v3-large` as its backbone, and **Multi**, which utilizes a multilingual version of deberta-v3 (`mdeberta-v3`). Both versions were fine-tuned on the Pile-NER dataset. For comparative purposes, we report results from ChatGPT

	Language	Sup.	ChatGPT	GLiNER	
				En	Multi
Latin	German	64.6	37.1	35.6	**39.5**
	English	62.7	37.2	**42.4**	41.7
	Spanish	58.7	34.7	38.7	**42.1**
	Dutch	62.6	35.7	35.6	**38.9**
Non-Latin	Bengali	39.7	23.3	0.89	**25.9**
	Persian	52.3	25.9	14.9	**30.2**
	Hindi	47.8	27.3	11.3	**27.8**
	Korean	55.8	**30.0**	20.5	28.7
	Russian	59.7	27.4	30.3	**33.3**
	Turkish	46.8	**31.9**	22.0	30.0
	Chinese	53.1	18.8	6.59	**24.3**
Average		54.9	29.9	23.6	**32.9**

Table 6.3: Zero-Shot Scores on Different Languages. The baseline, **Sup.**, is an XLM-R [Conneau et al., 2019] model fine-tuned on the training set of each language separately, as reported by Malmasi et al. [2022]. ChatGPT evaluation is taken from Lai et al. [2023]. GLiNER-En employs `deberta-v3-large`, and Multi uses `mdeberta-v3-base`.

and a supervised baseline, the latter being fine-tuned on the training set of each dataset using separate models.

Results As expected, the supervised baseline demonstrated superior performance, significantly outperforming the zero-shot models. Notably, among these models, GLiNER-Multi showed the most promising results, surpassing ChatGPT in the majority of languages. This is particularly noteworthy given that the fine-tuning dataset, Pile-NER, comprises solely English examples. While GLiNER-En generally underperformed compared to ChatGPT on average, it demonstrated competitive, and occasionally superior, performance in languages that utilize the Latin script, such as Spanish and German. However, its effectiveness was substantially less in non-Latin languages, with a marked underperformance in Bengali, where it achieved only a 0.89 F1 score.

6.5.3 In-domain Supervised tuning

In this section, we perform in-domain supervised fine-tuning (on 20 NER datasets) of our model to compare its capabilities against LLMs under this setup. Specifically, we compare our model against InstructUIE and UniNER, both of which have also been fine-tuned. The main difference is that UniNER has been pre-trained on the Pile-NER dataset before fine-tuning.

Dataset	**InstructUIE**	**UniNER-7B**	**GLiNER-L**	
	w/o	**w/**	**w/**	**w/o**
ACE05	79.9	**86.7**	82.8	81.3
AnatEM	88.5	88.5	**88.9**	88.4
bc2gm	80.7	82.4	**83.7**	82.0
bc4chemd	87.6	**89.2**	87.9	86.7
bc5cdr	89.0	**89.3**	88.7	88.7
Broad Twitter	80.3	81.2	82.5	**82.7**
CoNLL03	91.5	**93.3**	92.6	92.5
FabNER	78.4	**81.9**	77.8	74.8
FindVehicle	87.6	**98.3**	95.7	95.2
GENIA	75.7	77.5	**78.9**	77.4
HarveyNER	**74.7**	74.2	68.6	67.4
MIT Movie	89.6	**90.2**	87.9	87.5
MIT Restaurant	82.6	82.3	**83.6**	83.3
MultiNERD	90.3	93.7	**93.8**	93.3
ncbi	86.2	87.0	**87.8**	87.1
OntoNotes	88.6	**89.9**	89.0	88.1
PolyglotNER	53.3	**65.7**	61.5	60.6
TweetNER7	**65.9**	65.8	51.4	50.3
WikiANN	64.5	**84.9**	83.7	82.8
wikiNeural	88.3	**93.3**	91.3	91.4
Average	81.2	**84.8**	82.9	82.1

Table 6.4: In-domain Supervised fine-tuning. All the models are fine-tuned on the mix of all training data of the benchmark. **w/** indicates that the model was trained on the Pile-NER dataset before fine-tuning.

Training Setup For the supervised setting, we adhere to the same experimental setup as described in the main experiment (using deberta-v3 large). Regarding the training data, we follow the approach of InstructUIE: we randomly sample 10,000 data points for each dataset in the 20 NER benchmark. If a dataset does not contain 10,000 samples, we include all available data. We implement two variants of our model: the first one initializes the weights from our zero-shot model, which is a pretrained on the Pile-NER dataset. The second variant is trained without the Pile-NER dataset, same as InstructUIE.

Result Firstly, we observe that for the in-domain fine-tuning, our GLiNER model, pre-trained on Pile-NER, achieves slightly better results than the non-pretrained variant, with an average difference of 0.8. Moreover, our pretrained GLiNER model outperforms InstructUIE (with an average difference of 0.9) despite being fine-tuned on the same dataset, whereas InstructUIE is significantly larger (approximately 30 times so). This demonstrates

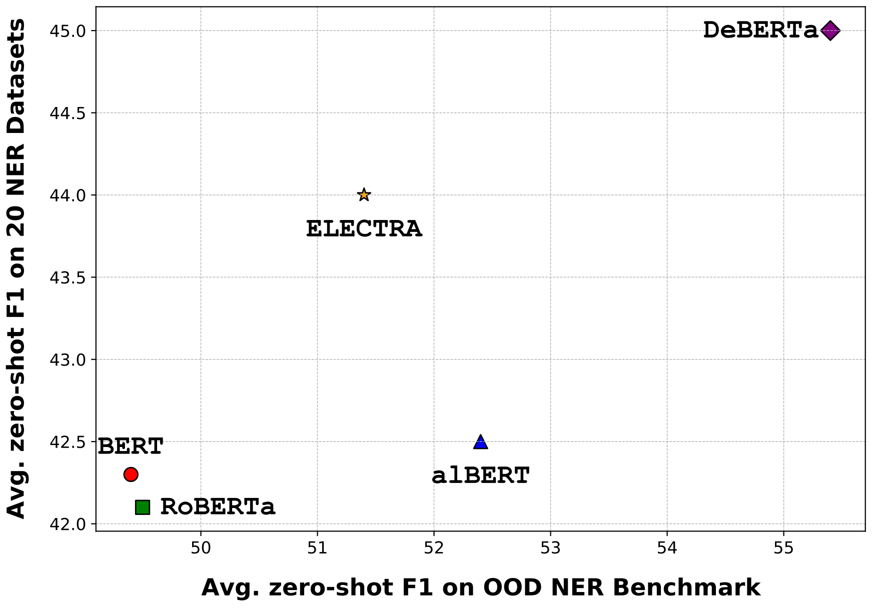


Figure 6.4: Zero-shot performance for different backbones. It reports the avg. results on 20 NER and OOD NER datasets

that our proposed architecture is indeed competitive. However, our model falls behind UniNER by almost 3 points. Nevertheless, our model still manages to achieve the best score in 7 out of 20 datasets.

6.6 Further analysis and ablations

In this section, we conduct different set of experiments to better investigate our model.

6.6.1 Effect of Different Backbones

In our work, we primarily utilize the deberta-v3 model as our backbone due to its strong empirical performance. However, we demonstrate here that our method is adaptable to a wide range of BiLMs.

Setup Specifically, we investigate the performance of our model using other popular BiLMs, including BERT [Devlin et al., 2019], RoBERTa [Liu et al., 2019a], AlBERT [Lan et al., 2019], and ELECTRA [Clark et al., 2020]. We also conducted experiments with XLNet [Yang et al., 2019] but did not achieve acceptable performance (achieving at most 3 F1 on the OOD benchmark) despite extensive hyperparameter tuning. For a fair comparison, we employed the base size (GLiNER-M) and tuned the learning rate for each model. We report the zero-shot results on both the OOD benchmark and the 20 NER benchmark in Figure 6.4.

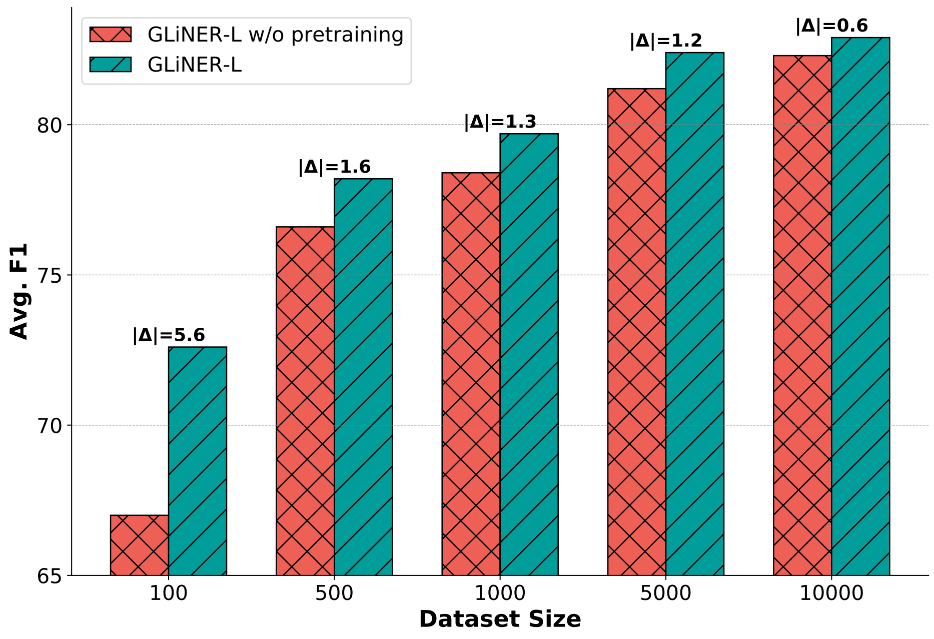


Figure 6.5: Supervised performance across different dataset sizes. The evaluation is conducted on the 20 NER datasets (in Table 6.4).

Result The results of our experiment, as shown in the Figure 6.4, clearly demonstrate the superiority of deberta-v3 over other pretrained BiLMs. It achieves the highest performance on both benchmarks by a clear margin. ELECTRA and AlBERT also show notable performance, albeit slightly lower, while BERT and RoBERTa lag behind with similar scores. However, it should be noted that all of the backbones we tested demonstrate strong performance compared to existing models. More specifically, even BERT-base, which ranks among the lower performers, achieves around 49 F1 on the OOD benchmark. This score is still 2 F1 points higher than the average for models like ChatGPT and InstructUIE.

6.6.2 Effect of Pretraining on In-domain Performance

In this section, we investigate the impact of pretraining on the Pile-NER dataset for supervised in-domain training on the 20 NER datasets, across various data sizes. The experiments range from 100 samples per dataset to 10,000 (full training setup). We use the same hyperparameters for all configurations. The results are reported in Figure 6.5.

Results As shown in the figure, models pretrained on Pile-NER consistently outperform their counterparts that are only trained on supervised data, indicating successful positive transfer. We further observe that the gain is larger when supervised data is limited. For instance, the difference in performance is 5.6 when employing 100 samples per dataset, and the gap becomes smaller as the size of the dataset increases.

Negative Samples	Prec	Rec	F1
0%	49.3	58.1	53.3
50%	**62.3**	**59.7**	**60.9**
75%	61.1	56.5	58.6

Table 6.5: Effect of negative entity types sampling.

6.6.3 Ablations

Negative Entity Sampling The original Pile-NER dataset, curated by Zhou et al. [2024], features passages with positive entity instances, i.e., entities that are directly present in the text. To better align training with real-world scenarios, where some entity types might be absent, we implemented negative entity sampling as mentioned in Section 6.4.2. We evaluate different sampling ratios: 0% (only positive entities), 50%, and 75% negative entities. table 6.5 shows that training with only positive entities results in lower precision but higher recall, indicating that the model often makes false positive errors. Conversely, using 75% negative entities increases precision but decreases recall, as the abundance of negatives makes the model more cautious, leading to missed correct entities. A 50% negative entity ratio proves to be the most effective, providing a balanced approach.

Entity type dropping In our experiments, we employed a strategy of randomly varying the number of entity prompts during training. This approach aimed to expose the model to different quantities of entity types in each training instance, thereby increasing its adaptability to handle scenarios with varying numbers of entities. The usage of this technique results in an average improvement of over 1.4 points in out-of-domain evaluation, as shown in the Figure 5.

6.7 Related Works

Named Entity Recognition NER is a well-established task in the field of NLP, with numerous applications. Initially, NER models relied on rule-based system [Weischedel et al., 1996] that were built using handcrafted algorithms and gazetteers [Mikheev et al., 1999, Nadeau et al., 2006, Zamin and Oxley, 2011]. However, these models had limitations in terms of scalability and adaptability to new domains or languages. To overcome these issues, machine learning approaches have been proposed [Lafferty et al., 2001]. In the early stages, NER tasks were designed as sequence labeling [Huang et al., 2015, Lample et al., 2016a, Akbik et al., 2018] where the objective was to predict tagged sequences (e.g., BILOU tags [Ratinov and Roth, 2009b]). Since then, several paradigm shifts have occurred: span-based approaches treating NER as span classification [Sarawagi and Cohen, 2004, Fu et al., 2021, Li et al., 2021b, Zaratiana et al., 2022a,c,d, 2023a]; NER being treated as a question answering problem [Li et al., 2019]; and even as a generation task [Yan et al., 2021b].

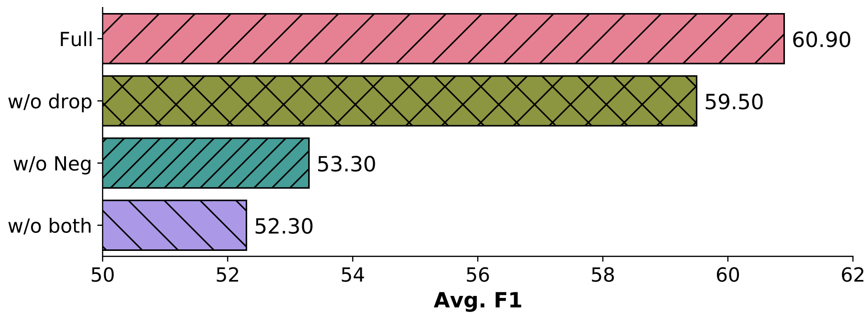


Figure 6.6: Randomly dropping entity types. We report the results with and without negative entity sampling.

Zero-shot learning for NER The advent of large-scale autoregressive models has recently transformed many paradigms in NLP through natural language prompting [Min et al., 2022, Wei et al., 2022, Qin et al., 2023]. This is also the case for NER [Li et al., 2022a, Ashok and Lipton, 2023a, Agrawal et al., 2022]. Others have fine-tuned these models for tasks to better align their capabilities with the requirements of entity recognition [Cui et al., 2021a, Zhou et al., 2024] or information extraction in general [Wu et al., 2020, Lou et al., 2023, Wang et al., 2023b, Sainz et al., 2024, Lu et al., 2022a, Geng et al., 2023]. This is sometime done through-instruction tuning [Mishra et al., 2021, Wang et al., 2022, Longpre et al., 2023].

6.8 Conclusion

In this chapter, we introduced GLiNER, a new method for identifying various types of entities in text using bidirectional language models. Our model not only outperforms state-of-the-art Large Language Models like ChatGPT in zero-shot scenarios but also offers a more resource-efficient alternative, crucial for environments with limited computing power. GLiNER is versatile, performing well in multiple languages, including those it wasn't trained on. In future work, we aim to further improve GLiNER's design for enhanced performance and to better adapt it for low-resource languages.

While our GLiNER model offers several advantages, it also has limitations that should be considered. One notable limitation is the model's inability to extract discontinuous entities. This constraint sets GLiNER apart from some Large Language Models (LLMs) that have this capability, potentially limiting its effectiveness in complex text scenarios where entities are not contiguous. Additionally, our evaluation methodology primarily relies on the exact matching metric. While this is a robust measure, it may not fully capture more nuanced aspects of the model's output. Subtleties such as partial matches or context-sensitive interpretations of entities are not adequately represented in this metric, suggesting that our evaluation might overlook some fine-grained characteristics of the model's performance.

In the upcoming chapters, we will address the problem of joint entity and relation extraction, where the model must not only extract entities but also identify the relation types between pairs of entities.

Part III

Structured Models for Joint Entity and Relation Extraction

Chapter 7

Autoregressive Text-To-Graph Generation for Joint Entity and Relation Extraction

7.1 Context

In this chapter, we propose a novel method for joint entity and relation extraction from unstructured text by framing it as a conditional sequence generation problem. In contrast to conventional generative information extraction models that are left-to-right token-level generators, our approach is *span-based*. It generates a linearized graph where nodes represent text spans and edges represent relation triplets. Our method employs a transformer encoder-decoder architecture with pointing mechanism on a dynamic vocabulary of spans and relation types. Our model can capture the structural characteristics and boundaries of entities and relations through span representations while simultaneously grounding the generated output in the original text thanks to the pointing mechanism. Evaluation on benchmark datasets validates the effectiveness of our approach, demonstrating competitive results.

7.2 Introduction

Joint entity and relation extraction is a fundamental task in Natural Language Processing (NLP), serving as the basis for various high-level applications such as Knowledge Graph construction [Ye et al., 2022b] and question answering [Chen et al., 2017]. Traditionally, this task was tackled via pipeline models that independently trained and implemented entity recognition and relation extraction, often leading to error propagation [Brin, 1999]. Deep learning has led to the creation of end-to-end models, allowing for the use of shared representations and joint optimization of loss functions for both tasks [Wadden et al., 2019b, Wang and Lu, 2020, Zhao et al., 2021b, Zhong and Chen, 2021a, Yan et al., 2021c]. Despite this advancement, these models essentially remain pipeline-based, with entity and relation predictions executed by separate classification heads, thereby ignoring potential interactions between these tasks.

Recent advancements have seen a shift towards “real” end-to-end solutions, where the prediction of entities and relations is intertwined, accomplished through autoregressive models. These models treat the joint entity-relation task as a process of generating plain text, employing *augmented languages* to encode and decode structural information [Paolini et al., 2021, Lu et al., 2022b, Liu et al., 2022, Fei et al., 2022]. While these models have achieved remarkable performance, we argue that they also expose room for improvement, especially in terms of grounding the output in the input text.

In this chapter, we present an autoregressive transformer encoder-decoder model that generates a linearized graph instead of generating plain text. Our model makes use of a pointing mechanism [Vinyals et al., 2017] on a dynamic vocabulary of spans and relations, providing explicit grounding in the original text. In fact, without grounding, models can generate output that are semantically coherent but contextually detached from the input. Our pointing mechanism mitigates this issue by ensuring that the decoder’s outputs, specifically the entity spans, are directly tied to the input text. Furthermore, by generating spans and relations directly from the text, rather than producing standalone plain text, our model encode the structural characteristics and boundaries of entities/spans more accurately, which can be missed by previous generative information extraction models. The cornerstone of our solution is the explicit enumeration of all spans[1] at the encoder’s output, making them readily available to the decoder. Although the number of spans can be extensive, we note that when bounding the span size, our model’s vocabulary is typically smaller than that of traditional language models [Devlin et al., 2019, Raffel et al., 2019], as discussed in subsequent sections.

Moreover, as previous generative IE models operate at the token level, they scatter the information regarding an entity’s span and its boundaries over multiple decoding steps. In contrast, generating an entity and its type in our approach is accomplished in a single decoding step, resulting in shorter sequence (Figure 7.5). Additionally, our method naturally ensures the well-formedness of the output while some generative IE models that produce text, often require non-regular constraints. As an example, in TANL Paolini et al. [2021], if a portion of the generated sentence has an invalid format, that segment is discarded. Such issues are readily addressed in our model since the vocabulary can be fully controlled.

We evaluated our model on three benchmark datasets for joint entity and relation extraction: CoNLL 2004, SciERC, and ACE 05. Our model demonstrated competitive performance on all datasets. Our contributions can be summarized as follows:

- We propose a novel method for joint entity and relation extraction by framing it as a conditional sequence generation problem. Our approach generates a linearized graph representation, where nodes represent entity spans and edges represent relation triplets.

- Our model employs a transformer encoder-decoder architecture with a pointing mech-

[1]Up to a certain length in practice.

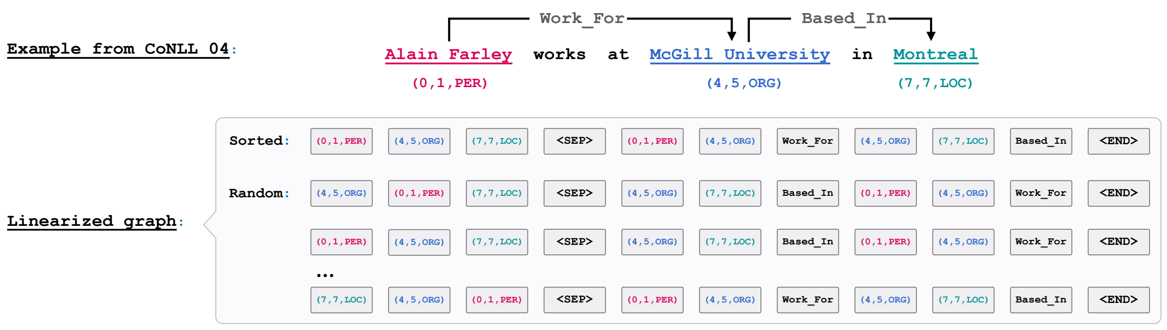

Example from CoNLL 04:
Work_For
Based_In
Alain Farley works at McGill University in Montreal
(0,1,PER)
(4,5,ORG)
(7,7,LOC)
Linearized graph:
Sorted: (0,1,PER) (4,5,ORG) (7,7,LOC) <SEP> (0,1,PER) (4,5,ORG) Work_For (4,5,ORG) (7,7,LOC) Based_In <END>
Random: (4,5,ORG) (0,1,PER) (7,7,LOC) <SEP> (4,5,ORG) (7,7,LOC) Based_In (0,1,PER) (4,5,ORG) Work_For <END>
(0,1,PER) (4,5,ORG) (7,7,LOC) <SEP> (4,5,ORG) (7,7,LOC) Based_In (0,1,PER) (4,5,ORG) Work_For <END>
...
(7,7,LOC) (4,5,ORG) (0,1,PER) <SEP> (0,1,PER) (4,5,ORG) Work_For (4,5,ORG) (7,7,LOC) Based_In <END>

Figure 7.1: Linearization for Information Graph Generation. The input text is mapped into an information extraction graph. The graph consists of entities and relation triplets, which are generated sequentially by first producing entity spans (represented by start word, end word, and entity type) followed by relation triplets (head entity, tail entity, and relation type).

anism on a dynamic vocabulary of spans and relation types. This allows to capture the structural characteristics and boundaries of entities and relations while grounding the generated output in the original text.

- We demonstrate the effectiveness of our approach through extensive evaluations on benchmark datasets, including CoNLL 2004, SciERC, and ACE 05. Our model achieves state-of-the-art results on CoNLL 2004 and SciERC, surpassing previous comparable models in terms of Entity F1 scores and Relation F1 scores.

7.3 Task Definition

We address the task of joint entity and relation extraction from text as a graph generation approach. Our proposed model generates nodes and edges as a single sequence, effectively integrating both entity and relation extraction into a unified framework. Formally, the task can be defined as follows: Given an input text sequence $\boldsymbol{x} = \{x_1, x_2, ..., x_L\}$, where x_i represents the i-th token in the sequence, our objective is to generate a linearized graph representation $\mathbf{y} = \{y_1, y_2, ..., y_M\}$, where y_j represents a token in the generated sequence. As shown in Figure 7.1, in Each token y_j can take one of three forms:

- **Entity span**: The token y_j represents an entity span, defined as $y_j = (s_j, e_j, t_j)$, where s_j and e_j denote the starting and ending positions of the entity span, and t_j denotes the type of the entity.
- **Relation type**: The token y_j represents a relation type between two entities, such as `Work_For` relation.
- **Special token**: The token y_j represents special tokens used in the generation process, such as <SEP> to separate entities and relations or the <END> to stop the generation.

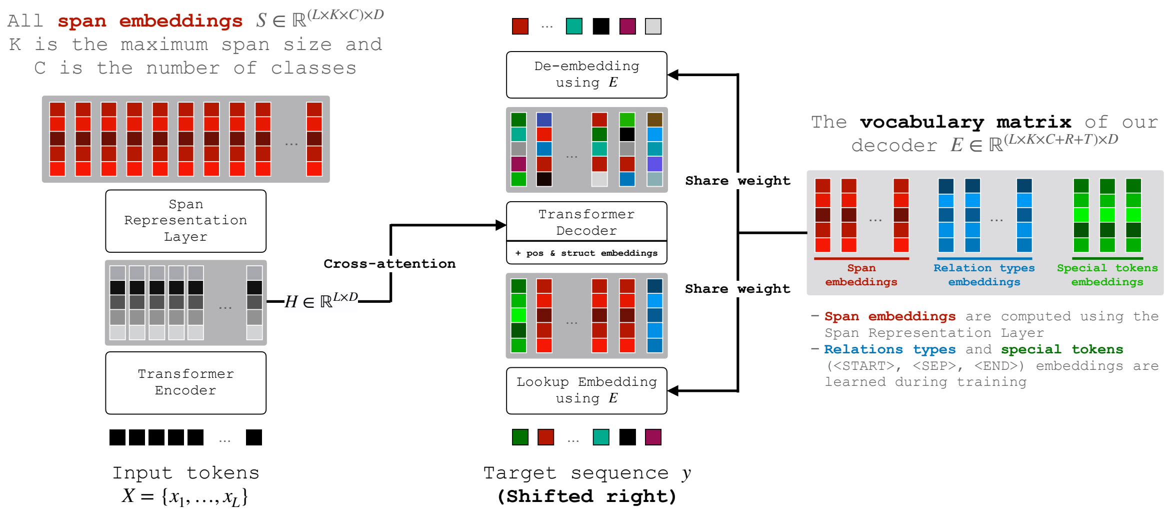

All span embeddings $S \in \mathbb{R}^{(L \times K \times C) \times D}$
K is the maximum span size and
C is the number of classes
Span Representation Layer
$H \in \mathbb{R}^{L \times D}$
Transformer Encoder
Input tokens
$X = \{x_1, \ldots, x_L\}$
Cross-attention
De-embedding using E
Transformer Decoder
+ pos & struct embeddings
Lookup Embedding using E
Target sequence y
(Shifted right)
Share weight
Share weight
The vocabulary matrix of our decoder $E \in \mathbb{R}^{(L \times K \times C+R+T) \times D}$
Span embeddings
Relation types embeddings
Special tokens embeddings
- Span embeddings are computed using the Span Representation Layer
- Relations types and special tokens (<START>, <SEP>, <END>) embeddings are learned during training

Figure 7.2: Illustration of the architecture of our model, ATG. (*left*) The Encoder takes in the input sequence X and generates representations of the tokens $\boldsymbol{H}$ and spans $\boldsymbol{S}$. (*middle*) The Decoder then generates the next token conditioned on the previous tokens and the input representation $\boldsymbol{H}$. (*right*) The vocabulary matrix used for decoding consists of the concatenation of span embeddings $\boldsymbol{S}$, learned relation type embeddings, and special token embeddings.

The template employed in our model, we refer to **ATG** (**A**utoregrestive **T**ext-to-**G**raph), is depicted in Figure 7.1. It starts by generating the entities, followed by a <SEP> token, and ends with the relation triplets, each consisting of head node, tail node and edge/relation type. During training, we try two distinct orderings for the graph linearization: *sorted* and *random*. As shown in the Figure 7.1, the *sorted* linearization organizes entities and relations based on their positions in the original text, while the *random* linearization randomly shuffles entities and relations order.

7.4 Model Architecture

Our model, ATG, employed an encoder-decoder architecture, which processes the input text sequence and produces a linearized graph as illustrated in the Figure 7.2.

7.4.1 Encoder

The encoder in ATG utilizes a transformer layer that takes an input text sequence $\mathbf{x}$ and outputs token representations $\mathbf{H} \in \mathbb{R}^{L \times D}$, where D is the model dimension.

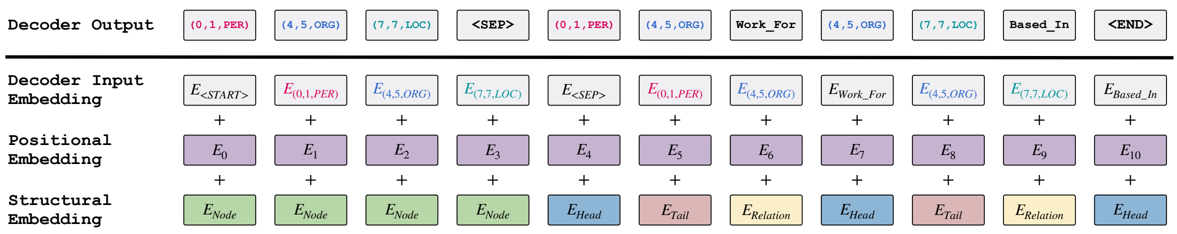


Figure 7.3: Input/ouptut of the decoder. The process starts with the special token <START> and continues until the <END> token is generated. To separate the generation of nodes and edges, a special token <SEP> is used. At each position, the decoder takes in the sum of the embedding of the current token, absolute position embedding, and structural embedding.

7.4.2 Vocabulary Construction

Dynamic vocabulary To enable the pointing mechanism in our decoder, we construct a dynamic vocabulary matrix $\mathbf{E}$ that includes embeddings for spans, special tokens, and relation types. While special tokens and relation type embeddings are randomly initialized and updated during training, the span embeddings are dynamically computed [Zaratiana et al., 2023a], *i.e* their representations depend on the input sequence. More specifically, the embedding of a span $(\text{start}, \text{end}, \text{type})$ is computed as follows:

$$\mathbf{S}_{[\text{start},\text{end},\text{type}]} = \mathbf{W}_{\text{type}}^{T}[\mathbf{h}_{\text{start}} \odot \mathbf{h}_{\text{end}}] \tag{7.1}$$

In this equation, $[\odot]$ represents a concatenation operation; $\mathbf{h}_{\text{start}}$ and $\mathbf{h}_{\text{end}}$ denote the representations of tokens at the start and end positions, respectively. $\mathbf{W}_{\text{type}} \in \mathbb{R}^{2D \times D}$ is a weight matrix associated with the entity type (*i.e*, there is a $\mathbf{W}_{\text{type}}$ for each entity types in a datasets). Finally, the vocabulary embedding matrix $\mathbf{E}$ is formed by stacking all the span embeddings $\mathbf{S}$, special token embeddings $\mathbf{T}$, and relation type embeddings $\mathbf{R}$.

Vocabulary size The size of the vocabulary matrix $\mathbf{E} \in \mathbb{R}^{V \times D}$ is $V = L \times K \times C + R + T$, where L represents the sequence length, K the maximum span size, C the number of entity types, R the number of relations, and T the number of special tokens (<START>, <END>, and <SEP>). Let's take the CoNLL 2004 dataset as an example to illustrate this. This dataset has the following characteristics: $K = 12$, $C = 4$, $R = 5$, and $T = 3$. Considering a sentence of length 114 (which is the maximum length in the training set), the resulting vocabulary size would be 5480. This size is considerably smaller when compared with the vocabulary size of a typical language model, which usually hovers around 30,000 distinct tokens.

7.4.3 Decoder

The decoder is a causal transformer trained to predict the next token in the sequence, akin to traditional language modeling. However, it is important to note that the vocabulary of our decoder consists of entity spans, relation types, and special tokens, rather than plain text. The decoder conditions its predictions on the previously generated tokens $y_{<j}$ using self-attention and on the input token representations $\mathbf{H}$ using cross-attention. This enables the decoder to attend to relevant information from both the previously generated tokens and the input text. Through attention visualizations, as depicted in Figure 7.8 and 7.9, we observed that the model effectively harnesses both sources of information. Finally, The training objective aims to maximize the following conditional probability:

$$p(\boldsymbol{y}|\boldsymbol{x}) = \prod_{j=1}^{M} p(y_j|\boldsymbol{y}_{<j}, \boldsymbol{H}) \tag{7.2}$$

This is achieved during the training by minimizing the negative log-likelihood of a reference sequence obtained by linearizing the reference IE graph. Details about the decoder input and output are given in the subsequent paragraphs.

Decoder input embedding The embedding step feeds the previous decoder outputs $y_1, \ldots, y_{i-1}$ into the model using the vocabulary matrix $\mathbf{E}$, along with *positional* and *structural* embeddings as shown in Figure 7.3. This process can be expressed as follows:

$$\begin{aligned} \boldsymbol{z}_1, \ldots, \boldsymbol{z}_{i-1} = &\boldsymbol{E}[y_1, \ldots, y_{i-1}] \\ &+\boldsymbol{E}_{pos}[1, \ldots, i-1] \\ &+\boldsymbol{E}_{struct}[y_1, \ldots, y_{i-1}] \end{aligned} \tag{7.3}$$

Here, $\boldsymbol{E}[y_1, \ldots, y_{i-1}]$ corresponds to the token embeddings, which may corresponds to spans, relation types, or special tokens embeddings depending on the nature of $y_{<i}$. The matrix $\boldsymbol{E}_{pos}$ represents absolute positional embedding. It allows to capture the positional information of the decoder outputs from y_1 to y_{i-1}. Additionally, the structural embedding $\boldsymbol{E}_{struct}$ serve as indicators to guide the model in generating specific elements. In particular, they provide information about whether the model should generate a Node (before <SEP> tokens), Tail, Head nodes, or Relation types (as illustrated in Figure 7.3). Both the absolute position encoding and the structural embedding are randomly initialized and updated during training. In summary, by combining these embeddings, ATG can capture the semantic, positional and structural information about the linearized graph.

Decoder output We define $\tilde{\boldsymbol{z}}_i$ as the hidden state at the last position of the decoder output sequence obtained by feeding the previous output embedding and the encoder outputs $\boldsymbol{H}$ (for cross-attention) to the decoder, *i.e*,

$$\tilde{\boldsymbol{z}}_i = \mathcal{D}ecoder(\boldsymbol{z}_1, \ldots, \boldsymbol{z}_{i-1}; \boldsymbol{H})[-\mathbf{1}] \tag{7.4}$$

Then, to compute the probability distribution over the dynamic vocabulary for generating the next token, y_i, our model employs the softmax function on the dot product between the dynamic vocabulary embedding matrix $\boldsymbol{E}$ and $\tilde{\boldsymbol{z}}_i$:

$$p(y_i|\boldsymbol{y}_{<i}, \boldsymbol{H}) = \frac{\exp \boldsymbol{E}^T \tilde{\boldsymbol{z}}_i}{\sum_{k=1}^{V} (\exp \boldsymbol{E}^T \tilde{\boldsymbol{z}}_i)_k} \tag{7.5}$$

The probabilities generated by this formulation allow the model to select the appropriate token from the vocabulary for generating a span, special token, or relation type.

Constrained decoding During inference, we sample from the model by enforcing constraints that preserve the well-formedness of the output graph. More specifically, during inference, we feed our model with the <START> token, and the generation process is guided by state-transition constraints as outlined in Figure 7.4. This structured approach ensures that each step in the generation aligns with the defined template, thereby maintaining the well-formedness of the output and allowing the production of valid IE graphs. All generations start with the start token **Start** state and continue until the <END> token is sampled. In practice, we also add other constraints: in state **1**, we prevent the repetition of already generated spans, and in state **3**, we ensure the tail span is different from the head. Furthermore, it is also possible to incorporate domain knowledge into the prediction. For instance, if the type of a head entity is PER and the tail entity is ORG, the relation can be constrained to be Work_For (CoNLL 04 dataset).

7.4.4 Training with Sentence Augmentation

In our work, we observed oversmoothing [Kulikov et al., 2022], where the model prematurely generates the <EOS>, *i.e* a bias towards short sequences [Murray and Chiang, 2018, Xuewen et al., 2021]. We found this bias to harm the recall of the tasks as the generation terminates before predicting all the entities/relations. To counteract this, we propose sentence augmentation, drawing inspiration from Pix2seq [Chen et al., 2022], who encounter a similar problem for their generative object detection model. This approach forms an augmented training sample s_{aug} by randomly concatenating sentences from the training set $\mathcal{D} = \{s_1, s_2, ..., s_N\}$:

$$s_{\text{aug}} = \bigoplus_{k=1}^{n} s_{i_k}, \quad i_k \sim \mathcal{U}(1, N), n \sim \mathcal{U}(1, B) \tag{7.6}$$

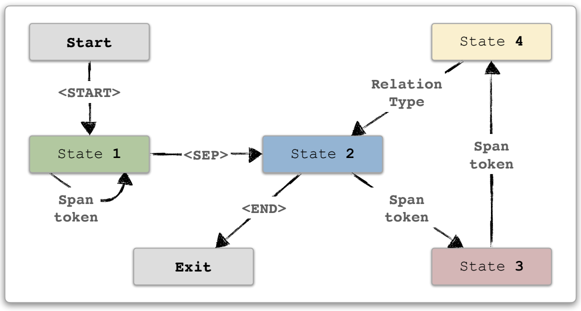
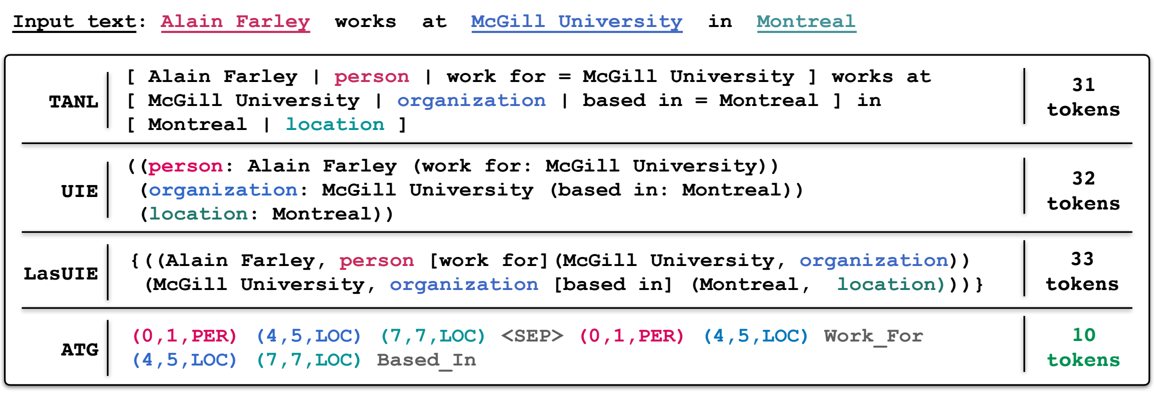


Figure 7.4: State-Transition diagram for constrained decoding. This diagram illustrates the state-based decision process used during the inference phase, which ensures the generation of a correct graph. Each state is represented by a node, and directed edges indicate valid actions. We use the same color code as the *structural embedding* in Figure 7.3.

Here, $\mathcal{U}$ denotes the uniform distribution, and B is a hyperparameter indicating the maximum number of sentences that can be concatenated. By applying this sentence augmentation technique during training, the model is exposed to diverse and longer output sequence lengths, reducing the risk of premature generation of the <EOS> token and thus improving recall. We perform an ablation study of the effect of sentence augmentation in our experiments section, showing that it largely improves the overall performance of our model.



Input text: Alain Farley works at McGill University in Montreal

TANL	`[ Alain Farley | person | work for = McGill University ] works at [ McGill University | organization | based in = Montreal ] in [ Montreal | location ]`	31 tokens
UIE	`((person: Alain Farley (work for: McGill University)) (organization: McGill University (based in: Montreal)) (location: Montreal))`	32 tokens
LasUIE	`{((Alain Farley, person [work for](McGill University, organization)) (McGill University, organization [based in] (Montreal, location)))}`	33 tokens
ATG	`(0,1,PER) (4,5,LOC) (7,7,LOC) <SEP> (0,1,PER) (4,5,LOC) Work_For (4,5,LOC) (7,7,LOC) Based_In`	10 tokens

Figure 7.5: Linearization for different models. In contrast to existing approaches (*TANL* [Paolini et al., 2021], *UIE* [Lu et al., 2022b], *LasUIE* [Fei et al., 2022]), our proposed model, *ATG*, generates spans (along with relation/special tokens) instead of text tokens, which allows for a shorter output sequence, richer (span-level) representation and fully controlled decoding.

7.5 Experimental Setup

7.5.1 Datasets

We evaluated our model on three benchmark English datasets for joint entity-relation extraction, namely SciERC [Luan et al., 2018], CoNLL 2004 [Carreras and Màrquez, 2004], and ACE 05 [Walker et al., 2006a]. The statistics of the dataset is reported on Table 9.4.

Dataset	$\|\mathcal{E}\|$	$\|\mathcal{R}\|$	# Train	# Dev	# Test
ACE05	7	6	10,051	2,424	2,050
CoNLL 04	4	5	922	231	288
SciERC	6	7	1,861	275	551

Table 7.1: The statistics of the datasets. We use ACE04, ACE05, SciERC, and CoNLL 04 for evaluating end-to-end relation extraction.

Models	SciERC			ACE 05			CoNLL 2004		
	ENT	**REL**	**REL+**	**ENT**	**REL**	**REL+**	**ENT**	**REL**	**REL+**
DYGIE++ [Wadden et al., 2019b]	67.5	<u>48.4</u>	–	88.6	63.4	–	–	–	–
Tab-Seq [Wang and Lu, 2020]	–	–	–	89.5	–	64.3	90.1	73.8	73.6
PURE [Zhong and Chen, 2021a]	66.6	48.2	35.6	88.7	66.7	63.9	–	–	–
PFN [Yan et al., 2021c]	66.8	–	<u>38.4</u>	89.0	–	**66.8**	–	–	–
UniRE [Wang et al., 2021]	<u>68.4</u>	–	36.9	89.9	–	66.0	–	–	–
TablERT [Ma et al., 2022]	–	–	–	87.8	65.0	61.8	**90.5**	73.2	72.2
	GENERATIVE								
HySPA [Ren et al., 2021b]	–	–	–	88.9	68.2	–	–	–	–
TANL [Paolini et al., 2021]	–	–	–	89.0	–	63.7	90.3	–	70.0
ASP [Liu et al., 2022] †	–	–	–	**91.3**	*72.7*	*70.5*†	<u>90.3</u>	–	*76.3*
UIE [Lu et al., 2022b]	–	–	36.5	–	–	<u>66.6</u>	–	75.0	–
LasUIE [Fei et al., 2022]	–	–	–	–	–	66.4	–	75.3	–
ATG (*Our model*)	**69.7**	**51.1**	**38.6**	<u>90.1</u>	**68.7**	66.2	**90.5**	**78.5**	**78.5**

Table 7.2: Comparison of our proposed model with state-of-the-art methods. Results are reported in terms of Entity (ENT) F1, Relation (REL) F1, and Strict Relation (REL+) F1 scores. The best scores are shown in bold, and the second-best scores are underlined. † *Italic scores use undirected evaluation for relation extraction and thus are not strictly comparable to our results.*

ACE 05 is collected from a variety of domains, such as newswire, online forums and broadcast news. It provides a diverse set of entity types such as Persons (PER), Locations (LOC), Geopolitical Entities (GPE), and Organizations (ORG), along with intricate relation types that include ART (Artifact relationships), GEN-AFF (General affiliations), and PER-SOC (Personal social relationships). This dataset is particularly notable for its complexity and wide coverage of entity and relation types, making it a robust benchmark for evaluating the performance of IE models.

CoNLL 2004 is an annotated corpus collected from newswires and focuses on general entities such as People, Organizations, and Locations, and relations like Work_For and Live_in.

SciERC is a dataset that comes with entity, coreference, and relation annotations for a collection of documents from 500 AI paper abstracts. The dataset defines scientific term types and relation types specifically designed for AI domain knowledge graph construction.

7.5.2 Evaluation Metrics

For the NER task, we adopt a span-level evaluation requiring precise entity boundaries and type predictions. To evaluate relations, we use two metrics: (1) Boundaries evaluation (**REL**) necessitates the correct prediction of entity boundaries and relation types; (2) Strict evaluation (**REL+**) additionally require accurate entity type prediction. We report the micro-averaged F1 score.

7.5.3 Baselines

We succinctly and briefly describe here the baseline that we compared with our model, which we separate into two categories: Span-based/table-filling and generative IE.

Span-based and table-filling *DyGIE++* [Wadden et al., 2019b] is a model that uses a pre-trained transformer to compute contextualized representations and employs graph propagation to update the representations of spans for prediction. *Tab-Seq* [Wang and Lu, 2020] tackles the task of joint information extraction by treating it as a table filling problem. *PURE* [Zhong and Chen, 2021a] is a pipeline model for the information extraction task that learns distinct contextual representations for entities and relations. *PFN* [Yan et al., 2021c] introduces methods that model two-way interactions between tasks by partitioning and filtering features. *UniRE* [Wang et al., 2021] proposes a joint entity and relation extraction model that eliminates the separation of label spaces for entity detection and relation classification. Their model uses a unified classifier to predict labels for each cell in a table of word pairs. In *TablERT* [Ma et al., 2022], entities and relations are treated as tables, and the model utilizes two-dimensional CNNs to effectively capture and model local dependencies.

Generative IE *HySPA* [Ren et al., 2021b] is a model for text-to-graph extraction that has linear space and time complexity using a Hybrid span generator. *TANL* [Paolini et al., 2021] treat the joint IE task as translation from plain text to augmented natural languages by fine-tuning a T5 model [Raffel et al., 2019]. This model has been further extended by *UIE* [Lu et al., 2022b] and *LasUIE* [Fei et al., 2022], which both proposed better linearization and additional pretraining to enhance results. Finally, *ASP* [Liu et al., 2022] handles entity and relation extraction by encoding the target structure as a series of structure-building actions, using a conditional language model to predict these actions.

7.5.4 Hyperparameter Settings

Our model, ATG, employs a transformer encoder-decoder [Vaswani et al., 2017c] architecture. We train it for a maximum of 70k steps using AdamW [Loshchilov and Hutter, 2017] optimizer. We use learning rate warmup for the first 10% of training and then decay to 0. The base learning rates are 3e-5 for the encoder, 7e-5 for the decoder, and 1e-4 for other projection layers. Unlike other generative IE models that utilize pretrained encoder-decoder architectures, often relying on large models such as T5 [Raffel et al., 2019], we initialize ATG's encoder with pre-trained transformer encoders, while the decoder is randomly initialized. In our preliminary experiments, we observed that initializing ATG with a pretrained encoder-decoder led to suboptimal performance. We hypothesize that this

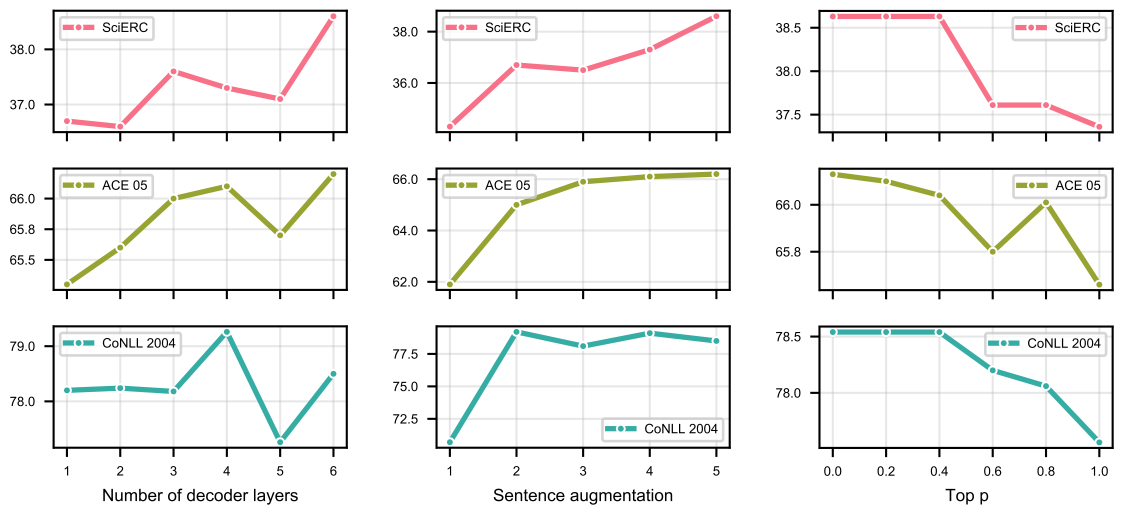


Figure 7.6: Investigation of the effect of different choices on model performance (REL+). (*Left*) Effect of the number of decoder layers, (*Center*) Impact of Sentence Augmentation, (*Right*) Study of different values of top-p for Nucleus Sampling.

is due to our decoder's utilization of a dynamic vocabulary, whereas existing pretrained encoder-decoder models have a fixed token vocabulary, which creates a large discrepancy. We use DeBERTa [He et al., 2021] for Conll-04 and ACE 05 and SciBERT [Beltagy et al., 2019a] for SciERC dataset. Across all configurations, the number of decoder layers is set to 6, though we noted that even a single layer can be enough in certain cases. The sentence augmentation hyperparameter B is set to 5.

7.6 Main Results

Table 7.2 presents the main results of our experiments, along with comparable approaches from the literature. ATG demonstrates strong performance across all datasets. On the SciERC dataset, ATG achieves the highest scores across all metrics. It outperforms the second-best result by 0.2 in **REL+** and surpasses the best generative approach, UIE [Lu et al., 2022b], by 2.1 points. On ACE 05, ATG provide a competitive performance, securing the second-highest scores. The reported top-performing model, ASP [Liu et al., 2022], operates under a relaxed, undirected relation evaluation, thereby limiting a fair comparison of results [Taillé et al., 2021]. On the CoNLL 2004 dataset, ATG exhibits its superiority by outperforming the second-best result by 2.2 in terms of **REL+**. Overall, across all three datasets, our proposed model either holds the top position or showcases strong competitive performance.

7.7 Ablation Studies

Number of decoder layers The number of decoder layers impact is illustrate on the Figure 7.6. It has a varying impact on performance across the datasets. In SciERC, increasing the number of decoder layers leads to a gradual improvement in the performance, reaching a peak of 38.6 at 6 layer. For ACE 05, the score shows a slight improvement from 65.3 to 66.2 as the number of decoder layers increases from 1 to 6. For the CoNLL 2004 dataset, the score fluctuates with different numbers of decoder layers, achieving already strong performance with only a single layer. Overall, the choice of the number of decoder layers can have a noticeable impact on **REL+** performance but the effect may vary across datasets.

Sentence augmentation The effect of sentence augmentation size on **REL+** performance is illustrated in Figure 7.6. The results reveal that increasing the number of sentence augmentations always improves performance across all datasets, except for CoNLL, where achieving state-of-the-art (SOTA) results is possible with just a size of 2. However, the absence of sentence augmentation leads to a significant decrease in **REL+**, proving its importance.

Nucleus sampling The impact of different top p values in nucleus sampling [Holtzman et al., 2019] on the performance (REL+) is shown in Figure 7.6. The scores across all datasets demonstrate a relatively stable trend, with minor variations observed as the top p value changes. This can be attributed to the application of constrained decoding, which ensures that the output remains well-formed. However, the lowest values of top p, corresponding to greedy decoding, consistently deliver the best performance.

Positional and structural smbeddings Table 7.3 illustrates the importance of positional and structural encoding on the Relation F1 score. When employing both encoding, **ATG** achieves the best performance across all datasets (38.6 for SciERC, 66.2 for ACE 05, and 78.5 for CoNLL 2004). Excluding positional encoding causes only slight performance drops, since the span representations may contain some positional information. Omitting structural encoding leads to similar, but slightly larger drops. Finally, when both are removed, the scores decrease the most, indicating their importance for the task.

	SciERC	ACE 05	CoNLL 2004
Full	**38.6**	**66.2**	**78.5**
- *Pos*	<u>36.4</u>	<u>66.0</u>	78.3
- *Struct*	36.1	65.8	<u>78.4</u>
- *Both*	35.4	65.4	78.0

Table 7.3: Effect of positional (*Pos*) and structural embedding (*Struct*) on REL+.

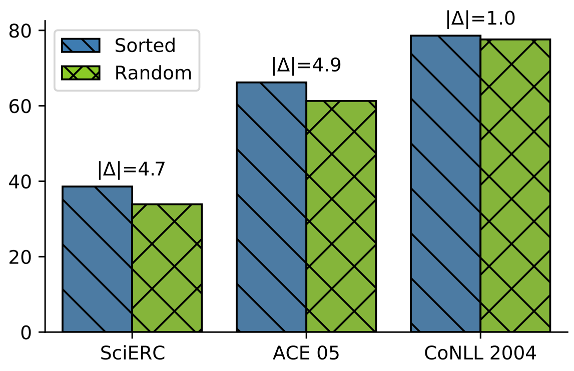


Figure 7.7: Impact of sequence ordering on REL+.

Sequence ordering Figure 7.7 compares the effects of sorted and random sequence ordering across different datasets. The results clearly show that the sorted ordering approach consistently outperforms the random one. The difference in performance is particularly significant on SciERC and ACE 05, with improvements of 4.7 and 4.9, respectively. On the CoNLL 04 dataset, although the sorted approach still leads, the difference narrows to 1. Interestingly, we initially hypothesized that random ordering would deliver better performance, given that any generation errors in a sorted order could be difficult to rectify.

7.8 Interpretability Analysis

7.8.1 Attention Maps

Here we analyze the attention of the model during the decoding step, which allow us to explain some of the model's decision. We investigate both the self-attention (Fig. 7.8) and cross-attention (Fig. 7.9).

Self-attention The self-attention map, shown in Figure 7.8, depicts the distribution of attention across preceding tokens during generation. One notable observation is the model's tendency to focus on the head and tail entities that comprise the relation when predicting relation types. For example, when predicting the `Work_For` relation, the model allocates most of its attention weight to the tokens `(0,1,Peop)` and `(4,5,Org)`.

Cross-attention The cross-attention map in Figure 7.9 indicates the specific areas in the input sequence that the decoded tokens attend to during generation. For entity labels in the output sequence such as `(0,1,Peop)`, `(4,5,Org)`, and `(7,7,Loc)`, we can observe higher attention scores for the words `Alain`, `McGill`, and `Montreal`, respectively, in the input sequence. This indicates that the model tends to focuses on the beginning of each entity span when generating these entities in the output sequence. Furthermore, when predicting tail entities for relations, significant attention is directed toward

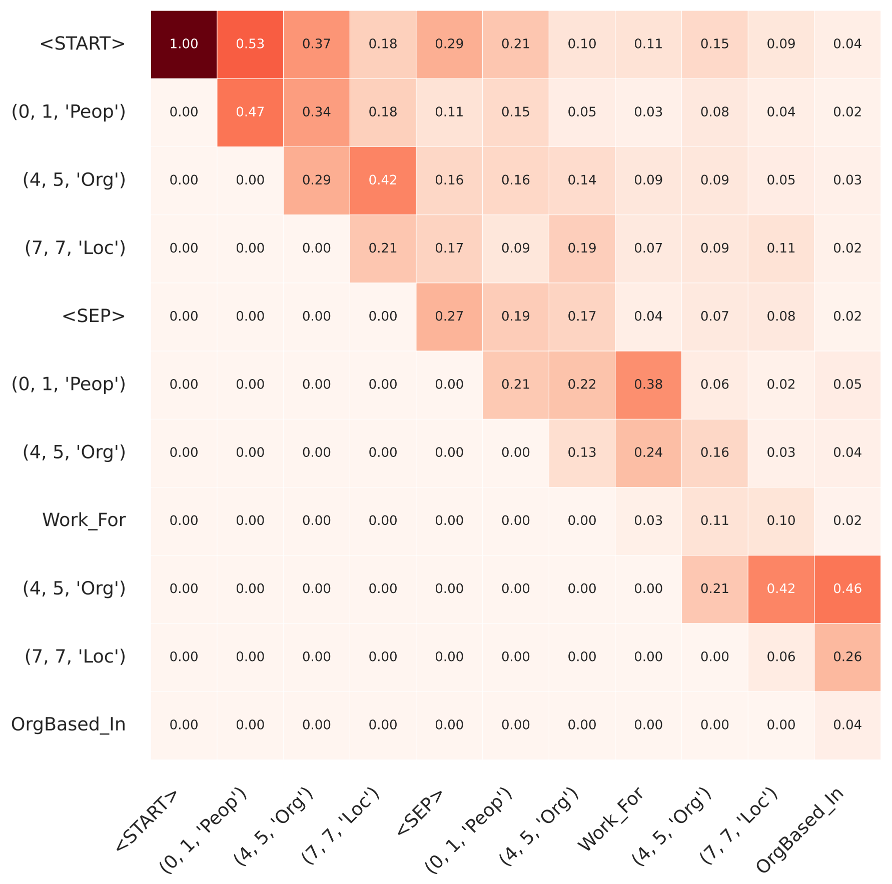


Figure 7.8: Decoder Self-Attention Visualization. This figure illustrates the attention patterns among elements in the generated sequence.

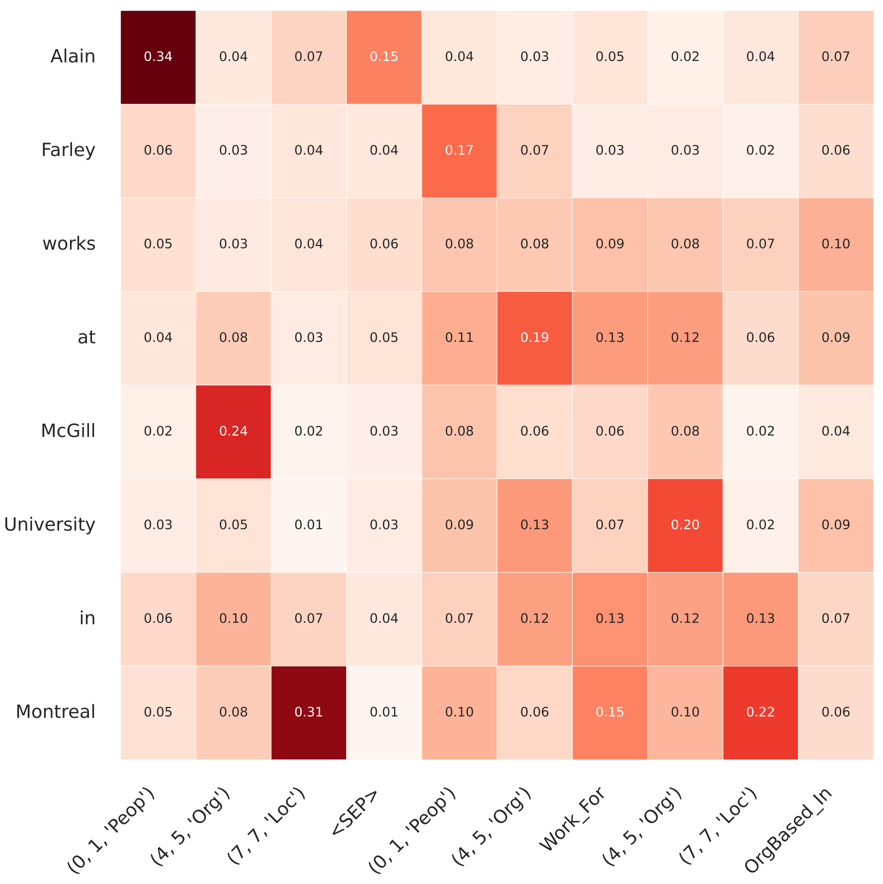
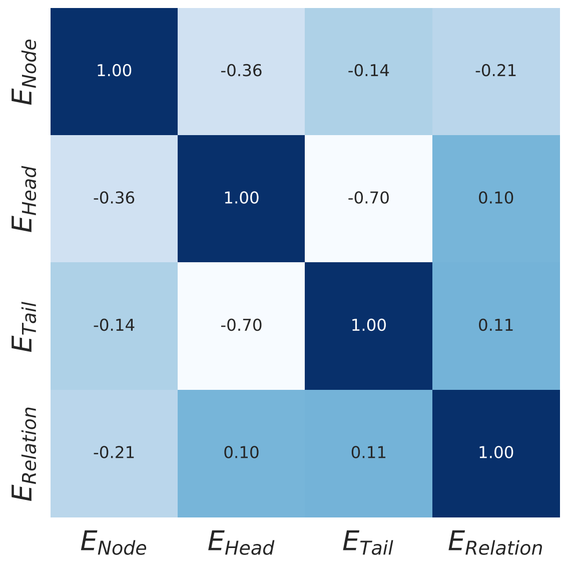


Figure 7.9: Decoder Cross-Attention Visualization. This map shows how each target element in the decoder interacts with and utilizes the original input text.

the prepositions 'at' and 'in' in the input sequence. This suggests that the model has learned to associate these prepositions with specific relations between entities.

7.8.2 Learned Structure Embedding

To investigate the impact of the learned structure embedding on model performance, we analyze the similarity between the structure embeddings learned during training, depicted in Figure 7.10. Notably, we observe a consistent pattern across datasets: the embeddings for the Head and Tail exhibit a high negative correlation. This finding may suggest that the model learns to differentiate between the Head and Tail entities, capturing their distinct characteristics.



	E_{Node}	E_{Head}	E_{Tail}	$E_{Relation}$
E_{Node}	1.00	-0.36	-0.14	-0.21
E_{Head}	-0.36	1.00	-0.70	0.10
E_{Tail}	-0.14	-0.70	1.00	0.11
$E_{Relation}$	-0.21	0.10	0.11	1.00

Figure 7.10: Structure embedding similarity. This map shows the cosine similarity between pairs of structure embedding.

However, we do not have a clear interpretation of this phenomenon. Additionally, Figure 7.11 illustrates structure embedding values over 512 dimensions. Noticeably, E_{Head} and E_{Tail} show higher values, indicating that predicting head and tail entities may be the most challenging for the model.

7.9 Background Works

Classification-based IE In the field of information extraction (IE), traditional pipeline models have been used, consisting of separate stages for entity recognition and relation extraction [Roth and Yih, 2004]. Entity recognition is performed to identify mentioned entities [Chiu and Nichols, 2015, Lample et al., 2016a, Zaratiana et al., 2022c,d, 2023a], followed by relation extraction to determine the relationships between these entities [Zelenko et al.,

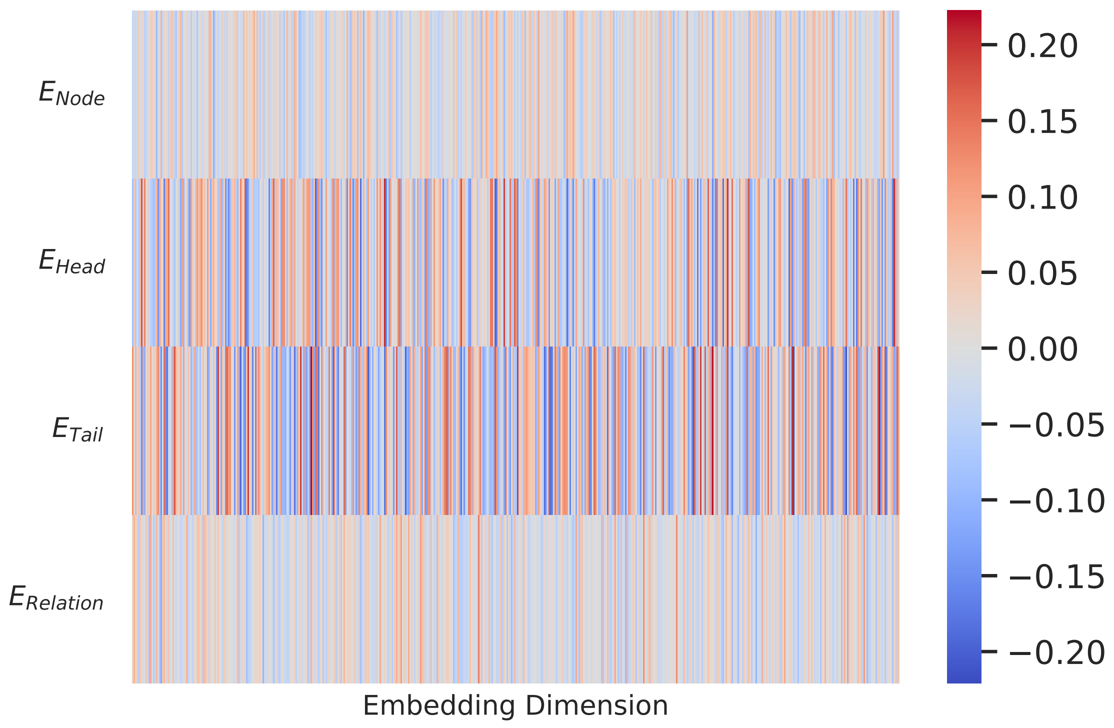


Figure 7.11: Structure embedding values. This shows the values taken by the learned structure embeddings.

2002a, Bach and Badaskar, 2007, Lin et al., 2016, Wu et al., 2017]. However, this approach suffers from error propagation, where mistakes in entity recognition can negatively impact the accuracy of relation extraction [Brin, 1999, Roth and Yih, 2004, Nadeau and Sekine, 2007]. To address these challenges, there has been a shift towards end-to-end models that jointly optimize both entity recognition and relation extraction. This joint optimization aims to harness the interplay between the two tasks, thereby enhancing overall performance [Sun et al., 2021a, Zhao et al., 2021b, Ye et al., 2022a]. Noteworthy directions in this domain include table-filling methods [Wang and Lu, 2020, Ma et al., 2022], span pair classification [Eberts and Ulges, 2019, Wadden et al., 2019b], set prediction [Sui et al., 2020], augmented sequence tagging mechanisms [Ji et al., 2020], fine-grained triplet classification [Shang et al., 2022], and the use of unified labels for the task [Wang et al., 2021].

Generative IE Recent advancements in generative Information Extraction (IE) emphasize the use of language models (LMs) to produce entities and relations, either as text or as a sequence of actions [Paolini et al., 2021, Lu et al., 2022b, Nayak and Ng, 2020, Liu et al., 2022, Fei et al., 2022, Wan et al., 2023]. Typically, these models employ pretrained encoder-decoder architectures, such as T5 [Raffel et al., 2019] or BART [Lewis et al., 2020], to encode an input text and subsequently decode it into a structured output. Their primary advantage over non-generative methods is their ability to seamlessly integrate tasks by treating them as a unified generation process. A comprehensive review of this approach is available in [Ye et al., 2022b, Xu et al., 2023]. Generative models have also found applications in other

IE tasks, including entity linking [Cao et al., 2021], event extraction [Li et al., 2021a], and document-level IE [Giorgi et al., 2022].

Constrained Decoding In the generative IE paradigm, the model can in principle generate any sequence over the LM's vocabulary if the decoder is not constrained in some way. One might resort to *controlled* to bias the model and guide the generation [Li et al., 2022b, Kumar et al., 2022, Amini et al., 2023]. These approaches still generate a sequence over LM's vocabulary that needs to be mapped to an output graph using some kind of a *parser* that analyzes the output and extracts a well-formed structure [Paolini et al., 2021]. Another approach incorporates constraints explicitly into the decoding algorithm to restrict the LM's vocabulary to allowed tokens. For instance, Cao et al. [2021] and Josifoski et al. [2022] use constrained beam search to force the output to a set of allowed entities and relations from a knowledge base schema. Another solution is to build a custom decoder that is constrained to a tailored vocabulary and decoding algorithm guaranteed to produce a linearized well-formed structure. Liu et al. [2022] and Lu et al. [2022b] use a specialized decoder with some sort of explicit grammar over a specific vocabulary to restrict the output to valid sequences. Our work falls in this category which can be formalized as generating words from a formal language described by a grammar over a specialized alphabet [Willard and Louf, 2023, Geng et al., 2023].

7.10 Conclusion

In conclusion, our autoregressive text-to-graph framework for joint entity and relation extraction has demonstrated its effectiveness in achieving state-of-the-art or competitive results on multiple benchmark datasets. By directly generating a linearized graph representation instead of plain text, ATG successfully captures the structural characteristics, boundaries, and interactions of entities and relations. Moreover, the pointing mechanism on dynamic vocabulary provides robust grounding in the original text, which also allows our model to be fully controllable using grammar-based decoding.

In the next chapter, we propose an extension to this model by framing the joint information extraction task as graph structure learning. This approach allows for more efficient inference by eliminating the slow autoregressive method.

Chapter 8

Structure-Aware Model for Joint Entity and Relation Extraction

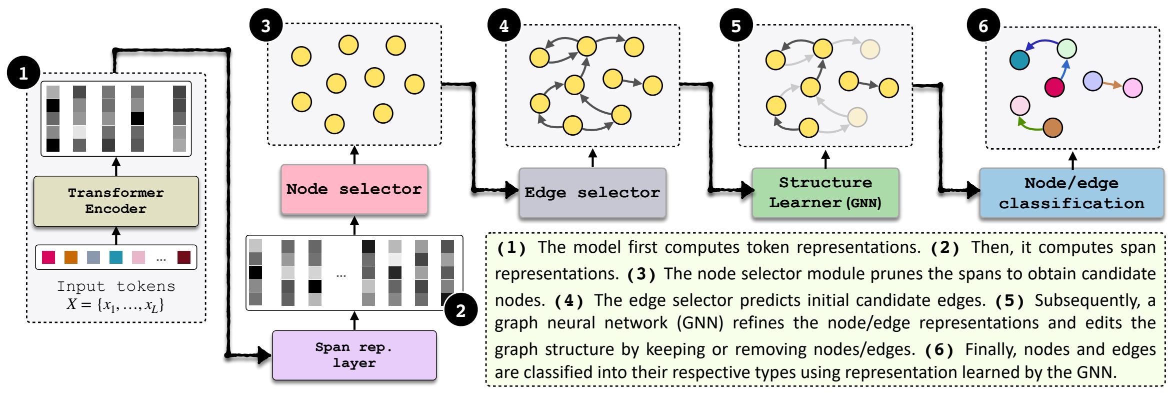
8.1 Context

In this chapter, we propose a novel approach to joint information extraction task by formulating it as graph structure learning (GSL). By formulating IE as GSL, we enhance the model's ability to dynamically refine and optimize the graph structure during the extraction process. This formulation allows for better interaction and structure-informed decisions for entity and relation prediction, in contrast to previous models that have separate or untied predictions for these tasks. When compared against state-of-the-art baselines on joint entity and relation extraction benchmarks, our model achieves competitive results.

8.2 Introduction

Information extraction is a fundamental task in NLP with many crucial real-world applications, such as knowledge graph construction. Early systems for this task were rule-based with manually coded rules [Appelt et al., 1993, Riloff, 1993], which are time-consuming and offer low performance. Methods based on machine learning have been proposed [Zelenko et al., 2002b, Jiang and Zhai, 2007], usually implementing pipeline approaches, with entity and relation models trained separately [Roth and Yih, 2004, Rosenfeld and Feldman, 2007]. The emergence of deep learning has enabled the training of joint IE models end-to-end through multitask learning, benefiting from rich features learned from self-supervised language models [Peters et al., 2018, Devlin et al., 2019].

Span-based approaches were proposed [Dixit and Al-Onaizan, 2019, Eberts and Ulges, 2019, Ji et al., 2020], which first classify spans as entities and then predict the relations by classifying all pairs of span entities. While this approach benefits from rich span representations computed by pretrained transformers, it overlooks potential interactions between entities and relations, as relations cannot influence entity types since entities are predicted first. As an alternative, table-filling approaches [Gupta et al., 2016, Wang and Lu, 2020, Ren et al., 2021a, Yan et al., 2023] have been proposed to perform joint predictions, using unified labels for the task, unlike span-based approaches that assume a prediction order. While it obtains competitive performance, by not explicitly modeling the graph nature of the task, table-filling methods might miss out on capturing more complex structural dependencies. More recently, autoregressive approaches have gained popularity, treating this task as linearised graph generation [Paolini et al., 2021]. Several approaches have been proposed, either by fine-tuning [Lu et al., 2022b, Liu et al., 2022, Fei et al., 2022] or by prompting large language models in zero or few-shot [Wadhwa et al., 2023, Geng et al., 2023, Han et al., 2023]. However, these models exhibit slow inference due to autoregressive generation and are prone to issues such as hallucination, where the model generates plausible but incorrect or irrelevant information [Guerreiro et al., 2023, Manakul et al., 2023].



(1) The model first computes token representations. **(2)** Then, it computes span representations. **(3)** The node selector module prunes the spans to obtain candidate nodes. **(4)** The edge selector predicts initial candidate edges. **(5)** Subsequently, a graph neural network (GNN) refines the node/edge representations and edits the graph structure by keeping or removing nodes/edges. **(6)** Finally, nodes and edges are classified into their respective types using representation learned by the GNN.

Figure 8.1: Model architecture. Please refer to the text box in the figure for an explanation of the different steps.

In this chapter, we propose a new paradigm for the IE task by treating it as a graph structure learning problem [Kipf et al., 2018, Franceschi et al., 2019, Jin et al., 2020, Zhao et al., 2021a, Li et al., 2023b], enabling more robust graph representation and thus structure-informed prediction. This method begins by creating an initial, imperfect graph from the text, where nodes represent textual spans and edges represent the relationships between these spans. Subsequently, the structure learner performs two operations: 1) it first enriches the representation of the elements of the graph using Graph Neural Networks (GNN) [Kipf and Welling, 2016, Hamilton et al., 2017], and then 2) it performs edit operations on the current graph, by either *keeping* or *dropping* elements (node or edge) to recover the final graph structure. Our structure learner is similar to the Graph Edit Network Paassen et al. [2021], except that our model does not have an *adding* operation as we assume that the initial graph contains them, and all edit operations are performed in a single step in our case. Furthermore, our structure learner takes advantage of recent advancements in GNN literature by employing the Token Graph Transformer (TokenGT) [Kim et al., 2022], a highly expressive model for graph-structured tasks. We found that it performs significantly better than standard message-passing GNNs [Gilmer et al., 2017] such as graph convolution network (GCN) [Kipf and Welling, 2016] and graph attention network (GAT) [Veličković et al.,

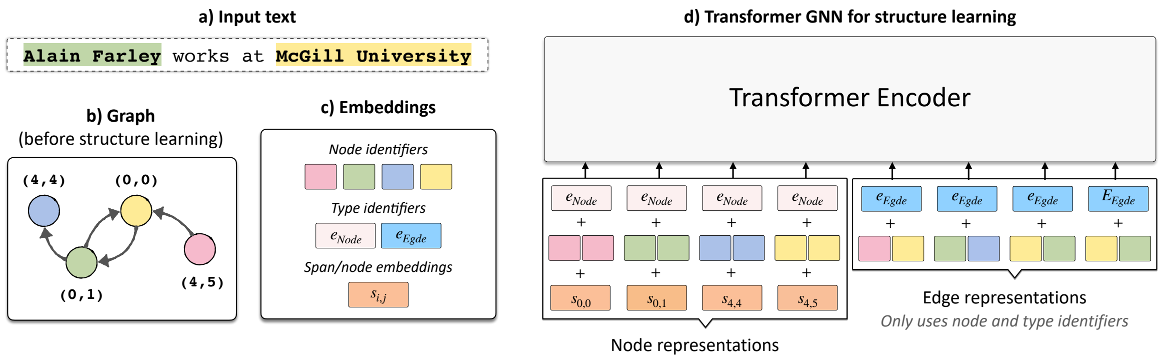


Figure 8.2: Graph transformer: Our model first constructs an initial graph from the input text (b) (Section 8.3.2). A transformer then processes nodes and edges of the graphs to refine their representation (d) (Section 8.4.1). Edge representation only uses node and type identifiers (without using edge-specific features) to enforce the transformer to use the graph structure for representation computation.

2018], especially since our graph is noisy and highly heterogeneous (i.e., contains many types of nodes and edges). Finally, our model performs node and edge classification on the final graph structure. When evaluated on benchmark datasets for joint IE, we found that our model achieves competitive results compared to strong baselines.

8.3 Input Graph Modeling

In this section, we provide a detailed explanation of the architecture of our proposed model, as illustrated in Figure 8.2.

8.3.1 Span Representation

The first step of our model consists of converting the input token $\{x_i\}_{i=1}^{L}$ into a set of contextualised embeddings $\{\boldsymbol{h}_i\}_{i=1}^{L} \in \mathbb{R}^D$. In this work, token embeddings are computed using a pretrained transformer encoder [Devlin et al., 2019]. Then, the representation of a span starting at position i and ending at position j is computed as follow:

$$\boldsymbol{s}_{ij} = \text{FFN}([\boldsymbol{h}_i^s; \boldsymbol{h}_j^e]) \tag{8.1}$$

where, $\boldsymbol{h}_i^s$ and $\boldsymbol{h}_j^e$ are the embeddings of the start and end word respectively, and FFN is a two-layer feed-forward network. To prevent quadratic complexity, we restrict the maximum width of the spans to a fixed number ($< L$).

8.3.2 Graph Construction

Here, we describe the initial graph construction process in our model, which corresponds to steps 2 and 3 of Figure 8.2. The objective is to select an appropriate number of nodes and edges before the structure learning phase. Including an excessive number of nodes and edges can hinder scalability, while opting for too few may lead to recall issues, potentially omitting critical nodes and edges. Hence, the goal is to strike a balance that ensures efficiency without sacrificing the incorporation of essential graph components.

Node selection In this step, the aim is to select relevant text spans to serve as nodes in the initial graph. For each span (i, j) within the span set $\mathcal{S}$, the model computes a score:

$$\texttt{sel\_node}((i,j)) = \sigma(\boldsymbol{w}_n^T \boldsymbol{s}_{ij}) \tag{8.2}$$

where $\boldsymbol{w}_n \in \mathbb{R}^{D\times 1}$ is a learned weight matrix, and σ the sigmoid function. Finally, the top-K spans that have the highest `sel_node` score are selected as nodes. We denote these nodes as $\mathcal{V}$.

Edge selection This layer prunes the edges of the fully connected graph formed by all the nodes in $\mathcal{V}$. It first computes the score of each potential directed edge $(i, j) \to (k, l)$ – with both source and target nodes belonging to the previously computed $\mathcal{V}$:

$$\texttt{sel\_edge}((i,j),(k,l)) = \boldsymbol{w}_e^T [\boldsymbol{s}_{ij}; \boldsymbol{s}_{kl}] \tag{8.3}$$

where $\boldsymbol{w}_e \in \mathbb{R}^{2D\times 1}$ is a learned weight matrix. The top-K edges with the highest scores are then selected; we denote them as $\mathcal{E}$.

Initial graph formation After the node $\mathcal{V}$ and edge $\mathcal{E}$ selections, we construct the initial graph, denoted as $G = (\mathcal{V}, \mathcal{E})$, illustrated in step 4 of Figure 8.2. This graph forms the basis for the subsequent structure learning phase, whose objective is to refine and edit $\mathcal{G}$ into the final IE graph.

8.4 Structure Learning

The goal of structure learning is to modify the structure of the graph constructed previously to produce the final IE graph. This process unfolds in two main steps: Utilizing a Graph Neural Network (GNN), the structure learner first enriches the representations of nodes and edges in $\mathcal{G}$, leveraging the existing graph's information. Then, the structure learner performs editing operations on $\mathcal{G}$, using the learned structure-aware representations.

8.4.1 Graph Representation Learning

This layer aims to enrich the representations of nodes and edges using information from the previously constructed graph $\mathcal{G}$. Typically, graph representation learning employs message-passing Graph Neural Networks (MPGNNs), which aggregate information from neighboring nodes. However, we find this approach yields suboptimal performance due to the noisy and heterogeneous nature of our input graph. Moreover, standard message-passing GNNs often encounter challenges such as oversmoothing [Chen et al., 2019] and oversquashing [Alon and Yahav, 2021]. Given these limitations, we use TokenGT [Kim et al., 2022], which treats nodes and edges of the graph as independent tokens, and feeds these tokens as input to a standard Transformer [Vaswani et al., 2017b]. TokenGT has been proven to be more expressive than message-passing GNNs and provides stronger empirical performance. It effectively addresses the shortcomings of MPGNNs and facilitates long-range interactions thanks to global attention mechanisms.

Graph Tokenization Let's consider two nodes $n = (i, j)$ and $m = (k, l)$, representing spans, and a directed edge $n \rightarrow m$ in graph $\mathcal{G}$. The representation of the nodes n and m are computed as:

$$\begin{aligned} \boldsymbol{z}_n &= \boldsymbol{s}_n + [\boldsymbol{p}_n; \boldsymbol{p}_n] \\ \boldsymbol{z}_m &= \boldsymbol{s}_m + [\boldsymbol{p}_m; \boldsymbol{p}_m] \end{aligned} \tag{8.4}$$

and the representation of the edge $n \rightarrow m$ is:

$$\boldsymbol{z}_{n,m} = [\boldsymbol{p}_n; \boldsymbol{p}_m] \tag{8.5}$$

Where $\mathbf{s}_n$ and $\mathbf{s}_m$ ($\in \mathbb{R}^D$) are the representations of the spans (as computed in Equation 8.1), and $\mathbf{p}_n$ and $\mathbf{p}_m$ ($\in \mathbb{R}^{D/2}$) correspond to node identifiers. The node identifiers are initialised with orthonormal vectors (following [Kim et al., 2022]) and then updated during training. During forward passes, they are randomly assigned to each node of the graph. We found that freezing the node identifiers can also work but provides suboptimal results. Furthermore, as noted in Equation 8.5, the edge representation consists solely of node identifiers. By doing so (not providing edge-specific features), we ensure the transformer layer purely uses the graph structure to represent the edges, which it does surprisingly well. However, explicitly integrating edge features can sometimes enhance the results.

Transformer layer The transformer layer takes as input the node and edge tokens computed previously. Following [Kim et al., 2022], we augment the input with learned token type embeddings $\boldsymbol{e}_{Node}$ and $\boldsymbol{e}_{Edge}$ ($\in \mathbb{R}^D$), which specifies whether a token represents a node or an edge. Then, we linearly project the token embeddings to match the dimensions of the transformer layer:

$$\begin{aligned} \boldsymbol{z}_n^{(0)} &= \boldsymbol{W}_{in}^T(\boldsymbol{z}_n + \boldsymbol{e}_{Node}) \\ \boldsymbol{z}_{n,m}^{(0)} &= \boldsymbol{W}_{in}^T(\boldsymbol{z}_{n,m} + \boldsymbol{e}_{Edge}) \end{aligned} \tag{8.6}$$

where, $\boldsymbol{W}_{in} \in \mathbb{R}^{D \times D}$ is the input projection weight. Then, the stacked nodes and edges, denoted as $\boldsymbol{Z}^{(0)} \in \mathbb{R}^{(|\mathcal{V}|+|\mathcal{E}|) \times D}$ are fed into an L-layer transformer to produce the final representation $\boldsymbol{Z}^{(L)}$.

8.4.2 Graph editing

In this stage, the goal is to obtain the final graph structure (IE graph), comprising nodes and their connectivity. This is accomplished using the representations $\boldsymbol{Z}^{(L)}$ learned by the graph transformer in the preceding subsection. We use a similar approach to the Graph Edit Network (GEN) [Paassen et al., 2021]. Following GEN, our layer either keeps or removes elements from the graph. However, in contrast to GEN, we do not introduce new elements, assuming that the initial graph $\mathcal{G}$ already contains all necessary nodes and edges. Consequently, since no new nodes are added, graph editing can be executed in a single stage.

Edit layer The graph editing process computes the probability for each node ($p_{\text{keep}}(n)$) and edge ($p_{\text{keep}}(n, m)$) regarding whether they should be kept or removed using a linear layer:

$$\begin{aligned} p_{\text{keep}}(n) &= \sigma(\boldsymbol{w}_k^T \boldsymbol{z}_n^{(L)}) \\ p_{\text{keep}}(n, m) &= \sigma(\boldsymbol{w}_k^T \boldsymbol{z}_{n,m}^{(L)}) \end{aligned} \tag{8.7}$$

where $\boldsymbol{w}_k \in \mathbb{R}^D$ is a learnable weight shared by the edges and the nodes.

Final graph structure To predict the final graph structure, we select the final nodes $\mathcal{V}_f$ and final edges $\mathcal{E}_f$, according to their *keep* probability, as follows:

$$\begin{aligned} \mathcal{V}_f &= \{n \in \mathcal{V} : p_{\text{keep}}(n) > 0.5\} \\ \mathcal{E}_f &= \{(n \rightarrow m) \in \mathcal{E} : p_{\text{keep}}(n, m) > 0.5\} \end{aligned} \tag{8.8}$$

Depending on the specific text-graph dataset used, it might be necessary to ensure that the final nodes $\mathcal{V}_f$ do not have overlapping spans. To address this, we employ a greedy algorithm that iteratively selects the highest-scoring node while respecting this constraint. Additionally, although rare, there may be instances where an edge $(n \rightarrow m)$ is selected in $\mathcal{E}f$, while either n or m is not in $\mathcal{V}_f$. In such cases, we simply discard the edge to maintain consistency in the graph structure.

8.4.3 Classification

The final step for obtaining the IE graph involves labeling the nodes and edges of the graph. We do so by computing classification scores for nodes and edges, denoted as y_n and $y_{n,m}$

respectively, using two independent feed-forward networks:

$$\begin{aligned} \boldsymbol{y}_n &= \mathrm{FFN}_{\mathcal{V}}(\boldsymbol{z}_n^{(L)}) \in \mathbb{R}^{|\mathcal{C}|} \\ \boldsymbol{y}_{n,m} &= \mathrm{FFN}_{\mathcal{E}}(\boldsymbol{z}_{n,m}^{(L)}) \in \mathbb{R}^{|\mathcal{R}|} \end{aligned} \tag{8.9}$$

where $|\mathcal{C}|$ is the number of node types, and $|\mathcal{R}|$ is the number of entity types.

8.5 Training

Our model is trained using multitask learning [Caruana, 1997], by jointly optimising the loss for the different components, treated as *independent classifiers* [Punyakanok et al., 2005]. The total loss function, $\mathcal{L}_{total}$, for training our model is computed as:

$$\mathcal{L}_{total} = \mathcal{L}_{\mathcal{V}} + \mathcal{L}_{\mathcal{E}} + \mathcal{L}_{edit} + \mathcal{L}_{cls} \tag{8.10}$$

This equation sums up the node selection loss ($\mathcal{L}_{\mathcal{V}}$), edge selection loss ($\mathcal{L}_{\mathcal{E}}$), edit losses ($\mathcal{L}_{edit}$), and final node/edge classification losses ($\mathcal{L}_{cls}$).

8.6 Experimental setup

8.6.1 Datasets

We evaluated our model on three datasets for joint entity-relation extraction, namely SciERC [Luan et al., 2018], CoNLL04 [Carreras and Màrquez, 2004], and ACE 05 [Walker et al., 2006a]. We describe them in the following and provide details and statistics about the datasets in Table 9.4.

ACE 05 is collected from a variety of domains, such as newswires, online forums, and broadcast news. It provides a very diverse set of entity types, such as Persons ("*PER*"), Locations ("*LOC*"), Geopolitical Entities ("*GPE*"), and Organizations ("*ORG*"), as well as complex types of relationships between them, including General Affiliations ("*GEN-AFF*"), Personal Social Relationships ("*PER-SOC*"), among others. This dataset is particularly notable for its complexity and wide coverage of entity and relation types, making it a robust benchmark for evaluating the performance of joint information extraction models.

CoNLL-2004 is a popular benchmark dataset for entity-relation extraction in English. It focuses on general entities, such as People, Organizations, and Locations, and simple relation types, such as "*Work_For*" and "*Live_In*".

Dataset	$\lvert\mathcal{E}\rvert$	$\lvert\mathcal{R}\rvert$	# Train	# Dev	# Test
ACE05	7	6	10,051	2,424	2,050
CoNLL 2004	4	5	922	231	288
SciERC	6	7	1,861	275	551

Table 8.1: Number of entity labels, relation labels, and sentences in each dataset.

Hyperparameter	ACE 05	CoNLL 2004	SciERC
Backbone	ALB	ALB	SciB
GNN Layers	2	2	2
Optimizer	AdamW	AdamW	AdamW
lr backbone	1e-5	3e-5	3e-5
lr others	5e-5	5e-5	5e-5
Weight Decay	1e-2	1e-4	1e-4
FNN Dropout	0.1	0.1	0.1
Hidden Size	768	768	768
Training Steps	120k	50k	30k
Warmup	5000	5000	3000
Batch size	8	8	8
Span length	12	12	12

Table 8.2: Hyperparameters Recapitulation. ALB denotes `albert-xxlarge-v1` and SciB denotes `scibert_scivocab_uncased`.

SciERC is specifically designed for the AI domain. It includes entity and relation annotations from a collection of documents from 500 AI paper abstracts. It contains scientific entity types and relation types and is primarily intended for scientific knowledge graph construction.

8.6.2 Evaluation Metrics

For the named entity recognition (NER) task, we use span-level evaluation, demanding precise entity boundary and type predictions. For evaluating relations, we employ two metrics: (1) Boundary Evaluation (**REL**), which requires correct prediction of entity boundaries and relation types, and (2) Strict Evaluation (**REL+**), which also necessitates correct entity type prediction. We report the micro-averaged F1 score following previous works.

		ACE 05			CoNLL 2004			SciERC		
Model	Backbone	Entity	REL	REL+	Entity	REL	REL+	Entity	REL	REL+
DYGIE++ [Wadden et al., 2019b]	BB & SciB	88.6	63.4	–	–	–	–	67.5	48.4	–
Tab-Seq [Wang and Lu, 2020]	ALB	89.5	–	64.3	90.1	73.8	73.6	–	–	–
PURE [Zhong and Chen, 2021a]	ALB & SciB	89.7	69.0	65.6	–	–	–	66.6	48.2	35.6
PFN [Yan et al., 2021c]	ALB & SciB	89.0	–	66.8	–	–	–	66.8	–	38.4
UniRE [Wang et al., 2021]	ALB & SciB	89.9	–	66.0	–	–	–	68.4	–	36.9
TablERT [Ma et al., 2022]	ALB	87.8	65.0	61.8	**90.5**	73.2	72.2	–	–	–
UTC-IE [Yan et al., 2023]	ALB & SciB	89.9	–	**67.8**	–	–	–	69.0	–	38.8
Our model	ALB & SciB	89.8	68.4	66.4	89.6	76.5	76.5	**69.2**	**50.6**	39.1
+ *Edge features*	ALB & SciB	89.8	68.0	66.0	90.2	**76.6**	**76.6**	68.0	50.4	**39.4**

Table 8.3: Results for different approaches and datasets. "Entity" refers to the F1 score for entity recognition, "REL" for relaxed relation extraction, and "REL+" for strict relation extraction. The "Backbone" column indicates the underlying architecture for each model (ALB for `albert-xxlarge-v1`, BB for `bert-base-cased`, and SciB for `scibert-base-uncased`).

8.6.3 Hyperparameters

In this study, we implemented our model using ALBERT [Lan et al., 2019] for the ACE 05 and CoNLL 2004 datasets, and SciBERT [Beltagy et al., 2019b] for the SciERC dataset, aligning with previous works. For all the models, we used the AdamW optimizer, and the learning rates and the number of training steps were tuned differently according to the dataset. We used two layers for the transformer layer, using the standard variant from Vaswani et al. [2017b] for structure learning. For the top-K value used for node and edge selection, we set it to the same length as the input sequence, which proved satisfactory in preliminary experiments. We detail the hyperparameters in Table 8.2. Our model was implemented using PyTorch and trained on a server equipped with A100 GPUs.

8.6.4 Baselines

We primarily compare our model, with comparable approaches from the literature in terms of model size. **DyGIE++** [Wadden et al., 2019b] is a model that uses a pretrained transformer to compute contextualised representations and enriches the representations of spans using graph propagation.**PURE** [Zhong and Chen, 2021a] is a pipeline model for the information extraction task that learns distinct contextual representations for entities and relations. **PFN** [Yan et al., 2021c] introduces methods that model two-way interactions between the task by partitioning and filtering features. **UniRE** [Wang et al., 2021] proposes a joint entity and relation extraction model that uses a unified label space for entity and relation classification. **Tab-Seq** [Wang and Lu, 2020] tackles the task of joint information extraction by treating it as a table-filling problem. Similarly, in **TablERT** [Ma et al., 2022], entities and relations are treated as tables, and the model utilizes two-dimensional CNNs to effectively capture and model local dependencies within these table-like structures. Finally, **UTC-IE** [Yan et al., 2023] treats the task as token-pair classification. It incorporates Plusformer to facilitate axis-aware interactions through plus-shaped self-attention and local interactions using CNNs over token pairs.

8.7 Results and Analysis

8.7.1 Main results

The main results of our experiments are reported in Table **??**. For our model, we report two variants: with and without edge features. Interestingly, there is only a marginal distinction between these variants, suggesting that utilizing node identifiers alone adequately represents edges for the graph transformer layer. Compared to state-of-the-art baselines, our proposed model achieves the highest performance on both CoNLL 2004 and SciERC datasets. Notably, our model outperforms the best-performing approach on CoNLL by more than 3 points in relation F1. Finally, while our model's performance on ACE 05 exhibits slightly lower results in relation evaluation, its entity evaluation performance matches state-of-the-art results.

Dataset	**Setting**	**ENT**	**REL**	**REL+**
ACE 05	`Trans`	**89.8**	**68.4**	**66.4**
	GCN	88.3	55.3	52.1
	GAT	88.3	54.5	52.1
	SAGE	57.2	56.8	53.2
CoNLL 04	`Trans`	89.6	**76.5**	**76.5**
	GCN	**89.7**	72.4	72.4
	GAT	86.1	70.6	70.6
	SAGE	88.8	72.2	72.2
SciERC	`Trans`	**69.2**	**50.6**	**39.1**
	GCN	49.8	32.7	17.5
	GAT	30.9	31.6	15.0
	SAGE	36.2	37.1	19.3

Table 8.4: Comparison of graph transformers against message-passing based GNN: Graph Convolution Network (GCN), Graph Attention Network (GAT) and GraphSAGE (SAGE).

8.7.2 Comparison with Message Passing GNN

In our study, we primarily utilize a transformer to learn the structure of the information extraction graph. In this section, we contrast it with the traditional Message Passing Graph

Neural Network (MPGNN), where each node in the graph can only communicate messages to its immediate neighboring nodes for each layer. To achieve this, we substitute the transformer layer with MPGNN to learn the representation of the spans, i.e., nodes. The numerical results of this comparative analysis are presented in Table 9.7.

Message Passing First, let's briefly outline the concept of message passing. During each iteration of message passing in a GNN, a hidden embedding $\boldsymbol{z}_n$ corresponding to each node n is updated based on information aggregated from n's incoming nodes $(m|m \to n \in \mathcal{E})$ in the graph, denoted as $\mathcal{N}(n)$. A message passing update can be expressed as follows:

$$\boldsymbol{z}_n^{(k+1)} = \psi\left(\boldsymbol{z}_n^{(k)}, \bigoplus_{m \in \mathcal{N}(n)} \phi(\boldsymbol{z}_n^{(k)}, \boldsymbol{z}_m^{(k)})\right) \tag{8.11}$$

Where ψ and ϕ are learnable functions, and $\bigoplus_m$ represents a permutation-invariant operation such as summation or max pooling. In our study, we explore three variants of the message-passing GNN: Graph Convolutional Network (GCN) [Kipf et al., 2018], Graph Attention Network (GAT) [Veličković et al., 2018], and GraphSAGE (SAGE) [Hamilton et al., 2017]. The only distinction among these variants lies in the aggregation function ϕ for message passing and the permutation-invariant operation $\bigoplus$ in Equation 8.11. For all variants, we utilized the available implementations in the PyTorch Geometric library [Fey and Lenssen, 2019].

Results The results for each variant are reported in Table 9.7. The transformer variant consistently outperforms the MPGNN baselines by a significant margin in terms of relation evaluation. For instance, in terms of *REL+* metric, it outperforms the variants by more than 10 points on ACE 05, more than 4 points on CoNLL 04, and over 20 points on SciERC. The results of MPGNN on CoNLL 04 are relatively more competitive compared to other datasets, possibly due to its simplicity (generic entity and relation types), requiring less reasoning for the model. Our main intuition for the poor performance of the MPGNN approach is as follows: (1) Our constructed graph is highly noisy at both node and edge levels. The functionality of MPGNN, which forces neighborhood direct connections, makes it difficult to distinguish noise from valid signals. In contrast, the transformer layer allows all nodes and edges to have a global overview of the entire graph, facilitating the discrimination between signal and noise. (2) Additionally, the information extraction graph is highly heterogeneous (many node and edge types), while MPGNNs are more suited for homogeneous and homophilic graphs. (3) Finally, MPGNNs are prone to problems such as oversmoothing and oversquashing, which can result in suboptimal learned representations. Transformers are less prone to these problems.

8.7.3 Attention Analysis

In Figure 9.4, we illustrate the attention map generated by our graph transformers. The resulting map exhibits intuitive patterns: first, we observe that nodes assign high attention

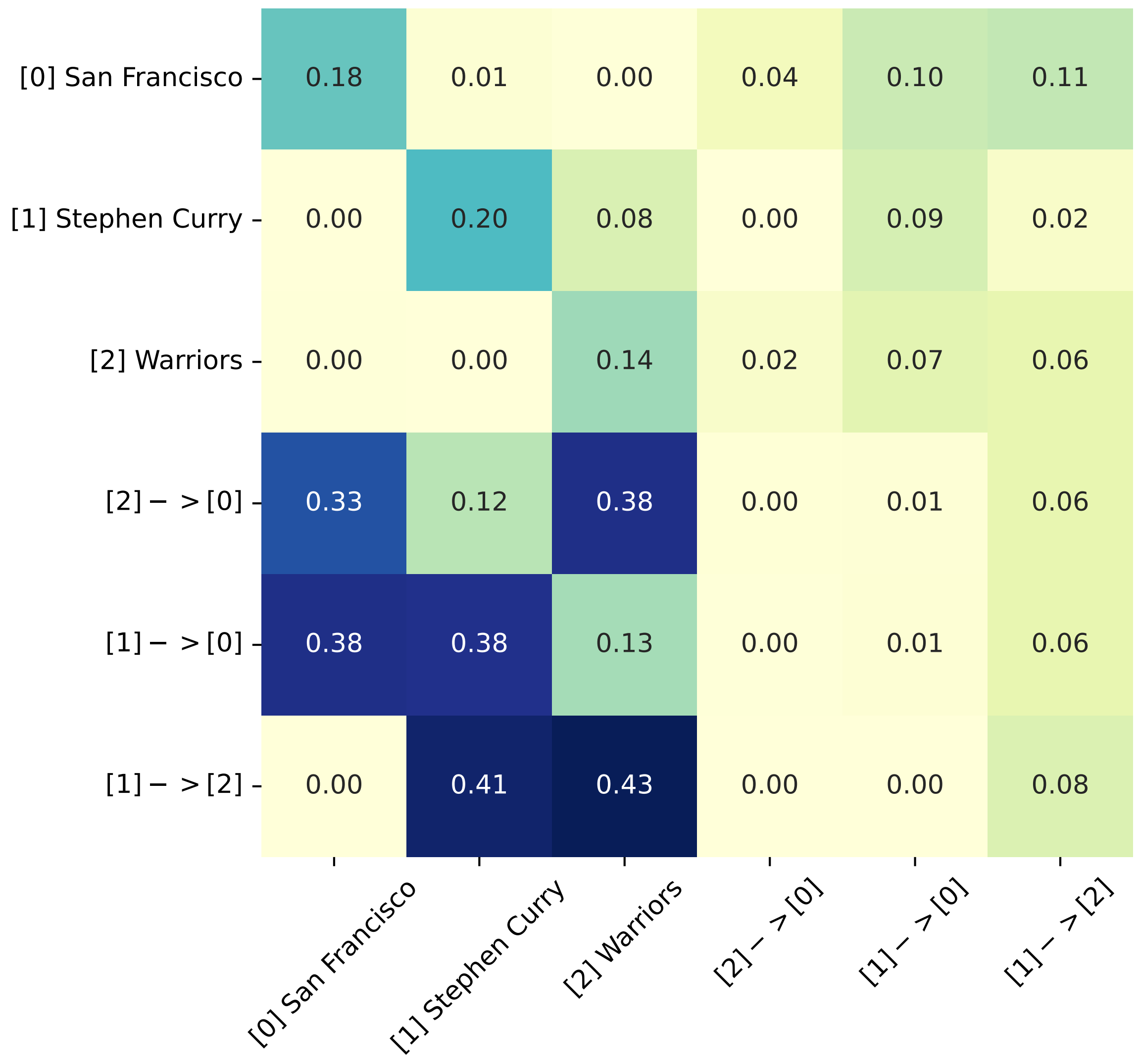


Figure 8.3: Attention visualization in Graph Transformer. In this figure, we illustrate the attention for the input "*Stephen Curry plays for the Warriors in San Francisco*". The nodes are represented by their identifier and text spans and edges are represented by their corresponding (directed) pairs of node identifier. We illustrate only the top 3 nodes and top 3 edge for better visibility.

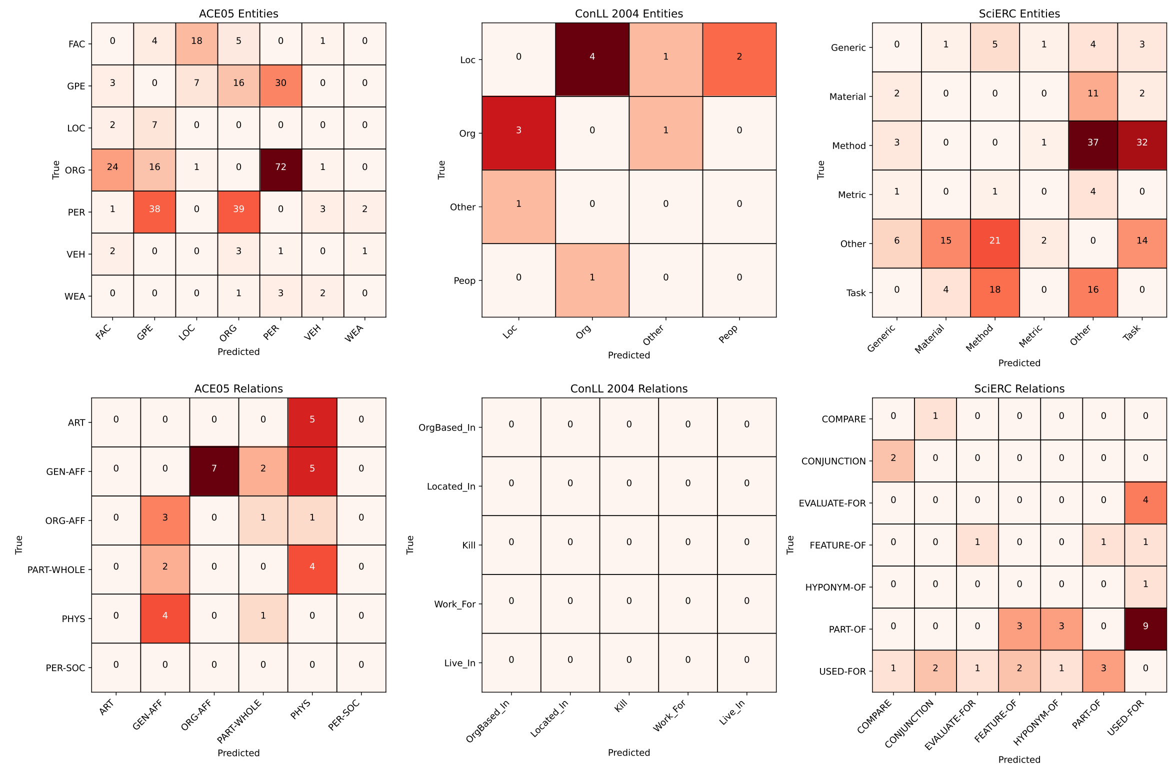


Figure 8.4: Entity and Relation confusion matrix.

weight to themselves, indicating that a significant part of the information is already encoded within the span representation. In addition, spans also show considerable attention to edges, indicating that the graph structure may be useful to improve node representation to some extent. Furthermore, we also observe interesting patterns concerning relations. Notably, relations that are solely based on node identifiers can precisely attend to their corresponding head and tail nodes. For instance, the edge "$[2]->[0]$" assigns the highest attention to the nodes with identifier [2] and [0], which corresponds to "*Warriors*" and "*San Francisco*" respectively. This observation applies to the other edges. This suggests that by solely using identifiers to represent edges, the transformer layer effectively reconstructs the graph structure.

8.8 Error Analysis

We conduct a detailed error analysis on the test sets, evaluating our model's performance and pinpointing improvement areas (see Figure 8.4).

Entity errors Our analysis of entity extraction errors is illustrated in Figure 8.5. For ACE 05, errors primarily come from correct span identification but incorrect label prediction. Notably, a significant portion of these errors involves pronominal entities like *we*, *us*, *it*, and *our*. In contrast, we observe few labeling errors with the CoNLL 2004 dataset, due to its simple entity types and predominantly exact span identification. However, in SciERC, an important number of misclassifications occur, primarily attributed to underspecified relation types such as "*Generic*" and "*OtherScientificTerm*", as well as the confusion between "*Method*" and "*Task*" entity types.

Relation errors For relations, we concentrate on analysing labeling errors, assuming spans are predicted correctly. In the ACE 05 dataset, errors involving GEN-AFF represent 40% of errors, often confused with ORG-AFF due to their semantic similarities, which can even challenge human discernment. In SciERC, 70% of labeling errors involve PART-OF and USED-FOR types, which are often confused with each other. Finally, for the CoNLL 2003 dataset, there are no labeling errors for the relations; the errors are mainly false positives and false negatives. This highlights that this dataset is less challenging, but also that our model is strong.

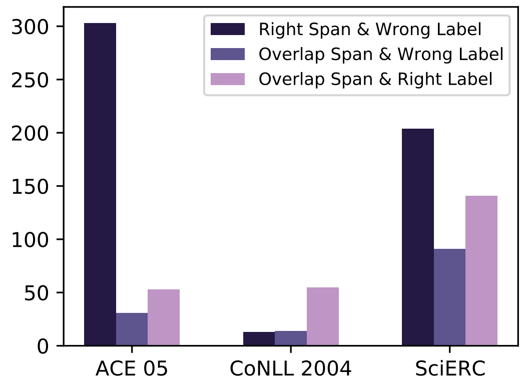


Figure 8.5: Common Errors in Entity Recognition. This caption highlights prevalent model errors found within false positives.

8.9 Related Works

Joint IE The field of information extraction (IE) has evolved from traditional pipeline models, which handle entity recognition [Chiu and Nichols, 2015] and relation extraction [Zelenko et al., 2002a, Bach and Badaskar, 2007, Lin et al., 2016] sequentially, to end-to-end models. These approaches aim to mitigate error propagation [Brin, 1999, Nadeau and Sekine, 2007] by jointly optimizing entity and relation extraction [Roth and Yih, 2004, Sun et al., 2021a], enhancing the interaction between the two tasks and overall performance. Proposed approaches include table-filling methods [Wang and Lu, 2020, Ma et al., 2022], span pair classification [Eberts and Ulges, 2019, Wadden et al., 2019b, Ye et al., 2022a], set prediction [Sui et al., 2020], augmented sequence tagging [Ji et al., 2020], and the use of unified labels for the task [Wang et al., 2021, Yan et al., 2023]. Additionally, the usage of generative models [OpenAI, 2023a] has become popular for this task, where input texts are encoded and decoded into augmented language [Paolini et al., 2021]. Some of these approaches conduct fine-tuning on labeled datasets [Lu et al., 2022b, Fei et al., 2022], while others use prompting techniques on large language models such as ChatGPT [Wadhwa et al., 2023]. Diverging from these approaches, our proposed model tackles the joint IE task as Graph Structure Learning, where the structure of information extraction is first inferred, followed by the prediction of node and edge types to more effectively incorporate structural information.

GSL Graph Structure Learning is a crucial task aimed at jointly optimizing the graph structure and downstream performance during training [Zhu et al., 2022, Zhou et al., 2023]. In this context, graphs may be incomplete, and in some cases, missing entirely, as they can be in our study. Different models employ various approaches to infer edges between nodes. For example, a metric learning approach utilizes a metric function on pairwise node embeddings to derive edge weights [Li et al., 2018, Yu et al., 2020a, Zhang and Zitnik, 2020]. Others utilize more expressive neural networks to infer edge weights based on node representations [Zheng et al., 2020, Luo et al., 2021, Veličković et al., 2020, Sun et al., 2021b]. Alternatively, they may treat the adjacency matrix as learnable parameters and directly optimize them along with GNN parameters [Gao et al., 2019, Jin et al., 2020]. In this chapter, we approach joint entity and relation extraction as Graph Structure Learning (GSL) by first inferring the structure of the graph, where text spans are nodes and relations are edges, and then predicting the types of its elements.

8.10 Conclusion

In this work, we approached the task of joint entity and relation extraction as Graph Structure Learning. Our methodology involves predicting an initial and noisy graph, followed by leveraging cutting-edge techniques from the Graph Neural Network (GNN) literature to refine the graph representations. This refinement process results in the production of the final information extraction graph through graph editing. When evaluated on common IE benchmarks, our model demonstrates competitive performance compared to state-of-the-art approaches.

In the next chapter, we propose a novel architecture for the joint Information Extraction (IE) task, where spans and relation extraction are enhanced through higher-level interactions enabled by an attention mechanism. Additionally, we introduce several decoding algorithms that incorporate domain- and task-specific constraints into joint IE models, thereby making their outputs more reliable.

Chapter 9

Enriched Representation and Globally Constrained Inference for Entity and Relation Extraction

9.1 Context

Despite recent progress in joint entity and relation extraction, existing approaches often fall short in two key aspects: richness of representation and coherence in output structure. These models often rely on handcrafted heuristics for computing entity and relation representations, potentially leading to loss of crucial information. Furthermore, they disregard task and/or dataset-specific constraints, resulting in output structures that lack coherence. In our work, we introduce *EnriCo*, which mitigates these shortcomings. Firstly, to foster rich and expressive representation, our model leverage attention mechanisms that allow both entities and relations to dynamically determine the pertinent information required for accurate extraction. Secondly, we introduce a series of decoding algorithms designed to infer the highest scoring solutions while adhering to task and dataset-specific constraints, thus promoting structured and coherent outputs. Our model demonstrates competitive performance compared to baselines when evaluated on Joint IE datasets.

9.2 Introduction

Joint entity and relation extraction is a pivotal task in Natural Language Processing (NLP), aiming to identify entities (such as "*Person*" or "*Organization*") within raw text and to discern the relationships between them (such as "*Work_for*"). This process forms the cornerstone for numerous applications, including the construction of Knowledge Graphs [Nickel et al., 2016]. Traditionally, this task is tackled via pipeline models that independently trained and implemented entity recognition and relation extraction, often leading to error propagation [Brin, 1999, Nadeau and Sekine, 2007]. The advent of deep learning has facilitated the

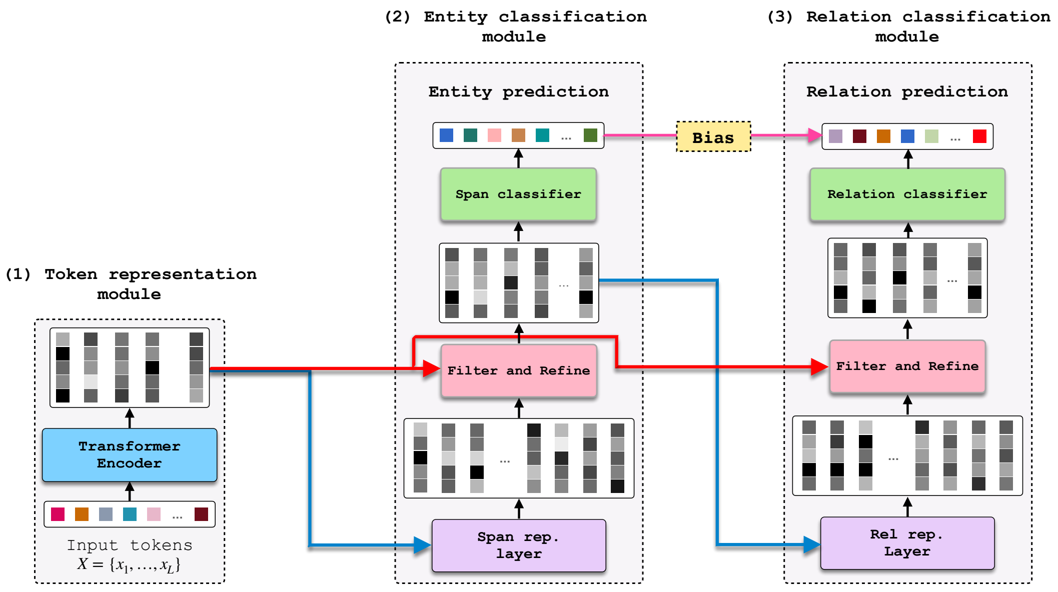


Figure 9.1: The model consists of three key components: (1) Word Representation, responsible for computing word embeddings for each word in the input sentence. (2) Entity Classification Module, which calculates, prunes, enriches span representations, and classifies them. (3) Relation Classification Module, which similarly calculates, prunes, enriches span representations, and classifies them. The pruning and enrichment of entity and relation representations are performed by a "Filter and Refine" layer, as described in Section 9.3.5 and illustrated in Figure 9.2.

development of end-to-end and multitask models for this task, enabling the utilization of shared representations and the simultaneous optimization of loss functions for both tasks. [Wadden et al., 2019b, Wang and Lu, 2020, Zhao et al., 2021b, Zhong and Chen, 2021a, Yan et al., 2021c]. Despite this advancement, these models essentially remain pipeline-based, with entity and relation predictions executed by separate classification heads, thereby ignoring potential interactions between these tasks.

While end-to-end models have been proposed [Lin et al., 2020], they often resort to hand-coded operations like concatenation for computing entity and relation representations, thereby limiting their flexibility. Moreover, these representations ignore potential inter-span and inter-relation interactions, as well as their interactions with the input text. Integrating these interactions could enrich the representations by preserving valuable contextual information overlooked during pooling operations. Moreover, existing approaches tend to overlook the structured nature of the output. In many real-world scenarios, the relationships between entities follow certain patterns or constraints, which may vary depending on the domain or dataset. However, current models typically treat entity and relation extraction as separate classification tasks without considering these constraints explicitly. Consequently, the extracted entities and relations may lack coherence or violate domain-specific rules, limiting the utility of the extracted knowledge.

In this work, we address these limitations by proposing *EnriCo*, a novel framework for joint entity and relation extraction. *EnriCo* aims to provide richer representation and promote coherence in output structures by leveraging attention mechanisms [Vaswani et al., 2017c] and incorporating task and dataset-specific constraints during decoding. To enhance representation richness, *EnriCo* employs attention mechanisms that allow entities and relations to dynamically attend to relevant parts of the input text (Fig. 9.4). This allows for richer and more expressive representations by preserving valuable contextual information that may be overlooked by traditional pooling operations. In addition, it also incorporates span and relation-level interactions, enabling each candidate entity or relation to update its representation based on the presence and characteristics of other candidate entities or relations in the text. This fosters a more holistic understanding of the relationships between different spans and relations, helping to resolve ambiguities and improve extraction accuracy (Table 9.7). To address the computational complexity arising from a large number of spans and relations, the model integrates a filtering layer to prune candidates, retaining only the most relevant ones, enabling efficient processing without sacrificing accuracy. Previous works have also leveraged rich high-level interactions. For instance, Zaratiana et al. [2022c] employed span-level interaction, yet their application is confined to entity recognition and lacks incorporation of pruning, making it computationally inefficient due to large number of spans. Similarly, Zhu et al. [2023] proposed span-to-token interaction for NER, but our work extends this approach to the relation (span-pair) level.

Finally, to ensure structural coherence of the output, we introduce a series of decoding algorithms to boost model performance by integrating task-specific and dataset-specific constraints. To achieve this, we formulate entity and relation prediction using an Answer Set Programming (ASP) solver, enabling the derivation of exact solutions. Experiments across benchmark datasets demonstrate the efficacy and performance of our proposed model.

9.3 Architecture

In this section, we present the architecture of our proposed model, *EnriCo*, for joint entity and relation extraction. The overall architecture comprises three main components: (1) Word Representation, (2) Entity Classification, and (3) Relation Classification modules. Figure 9.1 illustrates the architecture, offering a visual overview of how these modules interact.

9.3.1 Token Representation

The primary purpose of this module is to generate word embeddings from the input sentences. For that, we use a transformer layer that takes an input text sequence $\mathbf{x}$ and outputs token representations $\mathbf{H} \in \mathbb{R}^{L \times D}$, where D is the model dimension. In practice, this component is a pretrained transformer encoder such as BERT [Devlin et al., 2019].

9.3.2 Entity Module

In the Entity Module, the objective is to identify and classify spans in the input text as entities. A span refers to a contiguous sequence of words in the text that represents a candidate entity. Each entity is defined by its start and end positions within the input sentence, as well as its associated entity type. For example, the spans "Alain Farley" and "Montreal" could be classified as entities of type "Person" and "Location", respectively.

Span representation To compute span representations, we first enumerate all possible spans from the input sentence (up to a maximum span length in practice). Then, for each span, we concatenate the embeddings of its start and end words to compute a span representation. More formally, the span representation $\mathbf{S}$ for a span starting at word i and ending at word j is given by:

$$\mathbf{S}_{ij} = \boldsymbol{w}_{ent}^T(\mathbf{h}_i^s \oplus \mathbf{h}_j^e) \tag{9.1}$$

where $\mathbf{h}_i^s$ and $\mathbf{h}_j^e$ are the embeddings of the start and end words, $\boldsymbol{w}_{ent} \in \mathbb{R}^{2D\times D}$ is a learned weight matrix, and $\oplus$ denotes concatenation. In total, we compute $L \times M$ span vectors (we mask invalids), where L represents the sentence length and M represents the maximum span length, thus $\mathbf{S} \in \mathbb{R}^{LM\times D}$. The spans are then passed into a *Filter and Refine* (Sec. 9.3.5) layer to prune the number of spans to K and update their representation, resulting in $\mathbf{S}_f \in \mathbb{R}^{K\times D}$. The span representation $\mathbf{S}_f$ will serve for both span classification in the next paragraph and the relation representation in Sec. 9.3.3.

Span classification For span classification, we feed the representation of the filtered span $\mathbf{S}_f$ into a feed-forward network to obtain the span classification score:

$$\mathbf{Y}^{ent} = \text{FFN}(\mathbf{S}_f) \in \mathbb{R}^{K\times|\mathcal{E}|} \tag{9.2}$$

where $|\mathcal{E}|$ corresponds to the number of entity types, including the `non-entity` type.

9.3.3 Relation Module

In the Relation Module, the goal is to classify pairs of spans in the input text as specific relations. For instance, when presented with two spans "Alain Farley" and "McGill University", this module has to predict the relation between them, such as "*Work_for*" in this case.

Relation representation To compute the representation of a relation between two spans (i, j) and (k, l), we simply concatenate their respective span representations using

$$\mathbf{R}_{ij|kl} = \boldsymbol{w}_{rel}^T(\mathbf{S}_{f_{ij}}^{head} \oplus \mathbf{S}_{f_{kl}}^{tail}) \tag{9.3}$$

where $\mathbf{S}_{f_{ij}}^{head}$ and $\mathbf{S}_{f_{kl}}^{tail}$ are the span representations for the spans (i, j) and (k, l) respectively and $\boldsymbol{w}_{rel} \in \mathbb{R}^{2D\times D}$ is a learned weight matrix. This operation results in $K \times K$ candidate relations, corresponding to all pairs of candidate entities. Similarly to the entities, we process the relation representations through a *Filter and Refine* (Sec. 9.3.5) layer to reduce their quantity to K, thereby updating their representation, which results in $\mathbf{R}_f \in \mathbb{R}^{K \times D}$.

Relation classification Finally, we compute the relation classification score for each relation representation using a feed-forward network:

$$\mathbf{Y}^{rel} = \mathbf{FFN}(\mathbf{R}_f) \in \mathbb{R}^{K \times |\mathcal{R}|} \tag{9.4}$$

where $|\mathcal{R}|$ represents the number of relation types, including the no-relation type.

9.3.4 Entity-Relation Biases

To facilitate a more nuanced interaction between entity and relation prediction, our model incorporates a bias score for each combination of (head and tail) entity types and relation type (see Fig. 9.3 for an illustrative example). In the training phase, these bias scores are learned and seamlessly integrated into the relation score. Specifically, we augment the relation logits (all $y_r^{rel} \in \mathbf{Y}^{rel}$) by incorporating information about the predicted head entity type $h \in \mathcal{E}$ and the tail entity type $t \in \mathcal{E}$ in the following manner:

$$y_{rht}^{rel} = y_r^{rel} + \mathbf{b}(h, t, r) \tag{9.5}$$

where $\mathbf{b}(h, t, r) \in \mathbb{R}$, a learned bias score for the triplet (h, t, r) defined as follow:

$$\boldsymbol{b}(h, t, r) = \boldsymbol{\phi}(h, t, r) + \boldsymbol{\phi}(h, r) + \boldsymbol{\phi}(t, r) + \boldsymbol{\phi}(h, t) \tag{9.6}$$

In the above, we use the Gumbel-Softmax trick [Jang et al., 2017] to predict discrete entity types h and t, enabling gradient-based optimization of the whole process. The term $\boldsymbol{\phi}(h, t, r)$ captures the joint affinity score between a specific head, tail, and relation type. For instance, if the head entity is *Person* and the tail entity is *Organization*, the relation score would be higher for *works_for* than *born_in*. Meanwhile, $\boldsymbol{\phi}(h, r)$ and $\boldsymbol{\phi}(t, r)$ capture the general tendencies for entities (head or tail) of certain types to engage in specific relations. Lastly, $\boldsymbol{\phi}(h, t)$ capture any intrinsic compatibility between an head and tail types. Furthermore, another utility of the bias term is that it allows to incorporate domain constraints by manually assigning large negative values to invalid triples (see Table 9.1 and 9.3).

9.3.5 Filter and Refine

In this section, we detail the *Filter and Refine* layer (see Figure 9.2), a crucial component utilized in both the entity and relation blocks. The purpose of this layer is to prune the candidate elements (entities or relations) and then enhance their representations. Let $\mathbf{Z} \in \mathbb{R}^{N \times D}$ denote the matrix containing the representations of either entities or relations (i.e., $\mathbf{S}$ or $\mathbf{R}$), where N represents the number of entities or relations, and D is the dimension of the model.

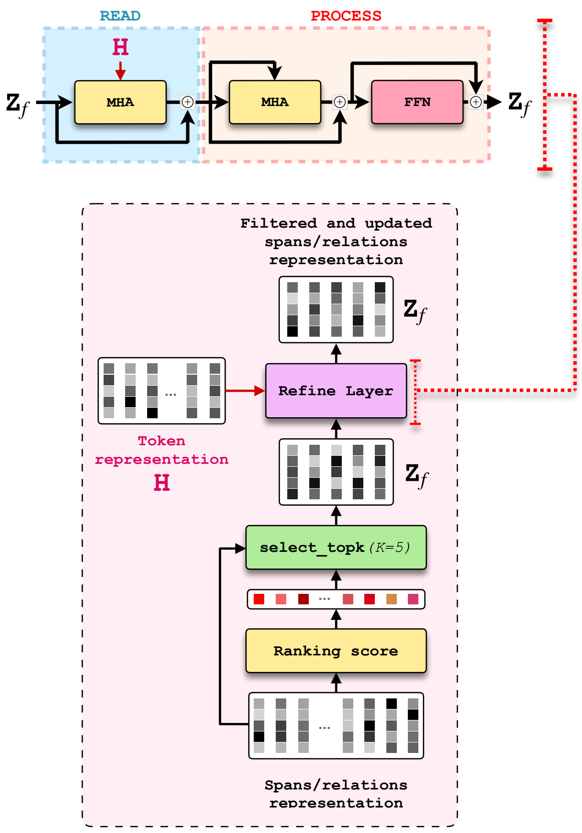


Figure 9.2: Filter and Refine layer.

Filtering mechanism The filtering first computes ranking scores for each element in $\mathbf{Z}$ using a FFN:

$$\mathbf{F} = \text{FFN}(\mathbf{Z}) \in \mathbb{R}^{N \times 1} \tag{9.7}$$

Let $\texttt{argtopK}(\mathbf{F})$ denote the indices of the top K elements in the vector $\mathbf{F}$. Then, the filtered set $\mathbf{Z}_f \in \mathbb{R}^{K \times D}$ is defined as:

$$\begin{aligned} \mathbf{Z}_f &= \texttt{select\_topk}(\mathbf{Z}, \mathbf{F}) \\ &= [\mathbf{Z}[i] \mid i \in \texttt{argtopK}(\mathbf{F})] \end{aligned} \tag{9.8}$$

This equation defines $\mathbf{Z}_f$ as the subset of $\mathbf{Z}$ containing only those elements that are ranked within the top K according to their scores.

Refine mechanism The refine module updates the representations of $\boldsymbol{Z}_f$ using two layers: READ and PROCESS. The READ layer updates each element in $\boldsymbol{Z}_f$ by incorporating information from the original token representation $\boldsymbol{H}$, using multi-head attention:

$$\boldsymbol{Z}_f = \boldsymbol{Z}_f + \text{MHA}(\boldsymbol{Z}_f, \boldsymbol{H}) \tag{9.9}$$

where $\text{MHA}(\mathbf{Z}_f, \mathbf{H})$ represents a multi-head attention mechanism with Queries $\mathbf{Z}_f$ and Keys and Values $\mathbf{H}$. This operation proves beneficial as some information may be lost during the

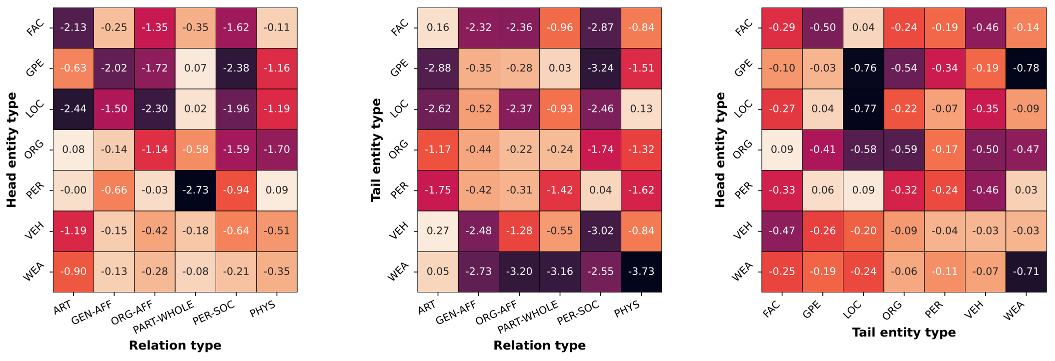


Figure 9.3: Biases value (Sec. 9.3.4) for ACE 05 dataset. This figure shows the values of learned biases for different associations of entity and relation types. (*left*) $\phi(h, r)$, bias scores between head entity type and relation type. (*middle*) $\phi(t, r)$, bias scores between tail entity type and relation type. (*right*) $\phi(h, t)$, bias scores between head entity type and tail entity type.

hand-crafted representation computation via concatenation (Equations 9.1 and 9.3). Allowing the entity and relation representations to attend to the original input sequence enables them to dynamically gather crucial information, thereby enhancing overall performance (see ablation study in Table 9.7). Furthermore, this introduces an additional layer of interpretability to our model, as illustrated in Figure 9.4. Additionally, the PROCESS layer updates the representations by enabling each element ($\in \boldsymbol{Z}_f$) to aggregate information from others ($\in \boldsymbol{Z}_f$):

$$\begin{aligned}\boldsymbol{Z}_f &= \boldsymbol{Z}_f + \text{MHA}(\boldsymbol{Z}_f, \boldsymbol{Z}_f),\\ \boldsymbol{Z}_f &= \boldsymbol{Z}_f + \text{FFN}(\boldsymbol{Z}_f)\end{aligned} \tag{9.10}$$

where $\texttt{MHA}(\mathbf{Z}_f, \mathbf{Z}_f)$ is a multi-head self-attention layer, and $\texttt{FFN}(\mathbf{Z}_f)$ is a feed-forward network. While *inter-span* interactions have been explored in previous works [Zaratiana et al., 2022c, Floquet et al., 2023], we are the first to employ this mechanism at the relation level.

9.3.6 Training

During training, our model employs multi-task learning by jointly minimizing the filtering and classification losses. We utilize a pairwise ranking loss with margin for the filtering: Usunier et al. [2009]:

$$\mathcal{L}_f = \sum_{p=1}^{N}\sum_{n=1}^{N} \max(0, \boldsymbol{F}_n - \boldsymbol{F}_p + \alpha) \cdot \delta(y_p, y_n) \tag{9.11}$$

In this equation, $\mathbf{F}_p$ and $\mathbf{F}_n$ represent the filtering scores for positive and negative samples (computed in Sec. 9.3.5), respectively. The term α is the margin, and $\delta(y_p, y_n)$ is an indicator function that is active when $y_p = 1$ and $y_n = 0$. This loss function encourages the model to prioritize positive samples over negative ones. This loss is applied at both the entity and relation levels. For the classification, we minimize the negative log-likelihood of the gold label spans and relations (on $\mathbf{Y}^{ent}$ and $\mathbf{Y}^{rel}$). Finally, the total loss function is a sum of all losses:

$$\mathcal{L}_{total} = \mathcal{L}_f^{ent} + \mathcal{L}_f^{rel} + \mathcal{L}_{cl}^{ent} + \mathcal{L}_{cl}^{rel} \tag{9.12}$$

Here, $\mathcal{L}_{total}$ is the sum of filtering losses ($\mathcal{L}_f^{ent}+\mathcal{L}_f^{rel}$) and classification losses ($\mathcal{L}_{cl}^{ent}+\mathcal{L}_{cl}^{rel}$) for entities and relations. To maintain simplicity, we do not add weighting terms for individual losses.

9.4 Decoding

In this section, we details the different decoding algorithm we employed in this chapter. The role of decoding is to produce the final output, which comprises the prediction of entity types (span prediction) and relation types (span pair prediction).

9.4.1 Unconstrained Decoding

Our baseline is *unconstrained decoding*, which corresponds to the raw predictions of the model for both entities and relations. The predictions for entities are obtained as follows:

$$E_p = \left\{ (i,j,c) \;\middle|\; \begin{array}{l} c = \arg\max_{c'} \boldsymbol{Y}_{ijc'}^{ent} \\ c \neq \texttt{non-entity} \end{array} \right\} \tag{9.13}$$

where $\mathbf{Y}_{ijc'}^{ent}$ is the score of the spans (i,j) having entity type $c' \in \mathbb{R}^{|\mathcal{E}|}$ (see the computation of span classification score in Equation 9.2). Furthermore, the prediction of the relations are obtained as follows:

$$R_p = \left\{ (h,t,r) \;\middle|\; \begin{array}{l} r = \arg\max_{r'} \boldsymbol{Y}_{htr'}^{rel} \\ r \neq \texttt{no-relation} \end{array} \right\} \tag{9.14}$$

where $\boldsymbol{Y}_{htr'}^{rel}$ is the score of the pairs of span h and t having relation type $r' \in \mathbb{R}^{|\mathcal{R}|}$ (the computation of relation classification score is in equation 9.4).

9.4.2 Constrained Decoding

Motivations The *unconstrained decoding* we describe before, does not consider the task-specific which are crucial for producing well-formed and coherent outputs. For instance, the Joint IE task has the following constraints:

- **Unique Type Assignment:** Each entity and relation must have a unique type assigned to it. (*Trivial*)

- **Non-overlapping Entity Spans:** Predicted entity spans must not overlap with each other.
- **Consistency:** A valid relation can only be formed by two valid entities, *i.e.*, a relation cannot be formed by a non-entity span.

Moreover, each dataset may have its specific constraints. For instance, in the *CoNLL 2004* dataset, if the head entity is *people* and the tail is *Org*, the relation type should be *work_for* (or non-relation) (see Table 9.1 for an exhaustive list).

Head	Tail	Relation
`Peop`	`Org`	*Work_For*
`Peop`	`Loc`	*Live_in*
`Org`	`Loc`	*OrgBased_in*
`Loc`	`Loc`	*Located_in*
`Peop`	`Peop`	*Kill*

Table 9.1: CoNLL 2004 dataset constraints.

Inference with ASP In our work, we formulate the decoding problem using *ASP* (Answer Set Programming) [Brewka et al., 2011, Gebser et al., 2014], a form of declarative programming oriented towards combinatorial search problems. This framework is particularly suitable for our task, as it allows for the integration of various constraints in a straightforward manner. We implement three decoding variants: Joint, which jointly optimizes the global score for entities and relations; Entity First, which first finds the optimal solution for entities and then for relations conditioned by predicted entities; and Relation First, which initially finds the optimal solution for relations and then for entities given the relations. For these decodings, we integrate both task-specific (described above) and dataset constraints (Table 9.1 and 9.3).

Fast variant While *ASP* provides strong performance, we find it is slow in practice. To address this, we propose a more scalable solution, which is equivalent to the Entity First variant of ASP, described before. Firstly, we predict candidate entities E_p using Equation 9.13. Then, we search for the optimal solution $\hat{E}_p$, which is a subset of E_p with no overlapping spans and the maximum score:

$$\hat{E}_p = \arg\max_{E \in \Psi(E_p)} \sum_{(i,j,c) \in E} \boldsymbol{Y}^{ent}_{ijc} \tag{9.15}$$

where $\Psi(E_p)$ contains all possible solution. The solution to this problem is provided by Zaratiana et al. [2022d], who transform the problem into a weighted graph search to derive *exact solution.* Then, once the entities are determined, the goal is to predict the types of each candidate relation based on these predicted entities. A key assumption is that the type of one relation is independent of others, provided the entities are known (i.e there is no *inter-relation* constraints). Therefore, we can predict each relation types (for all $y^{rel} \in \boldsymbol{Y}^{rel}$) independently as follow:

$$r = \arg\max_{r \in \mathcal{R}} y^{rel}_r + \mathbf{b}(h, t, r) \tag{9.16}$$

where h and t are respectively the type of head and tail entities. The bias term $\mathbf{b}(h,t,r)$ add entity prediction information in the relation facilitate the integration of constraints into the prediction. It does so by assigning a negative infinity value to any invalid entity-relation type associations, as dictated by the specific dataset constraint (Table 9.1 and 9.3). As shown in the table 9.2, this algorithm is significantly faster than ASP-based approached, while allowing the adherence to constraints.

	ASP-based	Fast variant
Joint	6.7	-
Relation first	7.4	-
Entity first	5.5	21.7

Table 9.2: Decoding speed in sentence per second. All decoding can be implemented using ASP solver. Entity first variant can be implemented without ASP resulting in faster decoding.

9.5 Experimental Setup

9.5.1 Datasets

We evaluated our model on three datasets for joint entity-relation extraction, namely SciERC [Luan et al., 2018], CoNLL04 [Carreras and Màrquez, 2004], and ACE 05 [Walker et al., 2006a]. We provide details and statistics about the datasets in the Table 9.4 and the description of entity and relation types in Table 2.1.

ACE 05 is collected from a variety of domains, such as newswire, online forums and broadcast news. It provides a diverse set of entity types such as Persons (PER), Locations (LOC), Geopolitical Entities (GPE), and Organizations (ORG), along with intricate relation types that include Artifact relationships (ART), General affiliations (GEN-AFF), and Personal social relationships (PER-SOC). This dataset is particularly notable for its complexity and wide coverage of entity and relation types, making it a robust benchmark for evaluating the performance of Joint IE models.

Dataset	$\|\mathcal{E}\|$	$\|\mathcal{R}\|$	# Train	# Dev	# Test
ACE05	7	6	10,051	2,424	2,050
CoNLL 2004	4	5	922	231	288
SciERC	6	7	1,861	275	551

Table 9.4: The statistics of the datasets. We use ACE04, ACE05, SciERC, and CoNLL 2004 for evaluating end-to-end relation extraction.

CoNLL04 is a popular benchmark dataset for entity-relation extraction in English. It focuses on general entities such as People, Organizations, and Locations. The dataset primarily includes simple and generic relations like *Work_For* and *Live_in*.

Head	**Tail**	**Relations**
PER	FAC	*ART, PHYS*
PER	LOC	*PHYS, GEN-AFF*
PER	GPE	*PHYS, ORG-AFF, GEN-AFF*
PER	PER	*PER-SOC, GEN-AFF*
PER	ORG	*ORG-AFF, GEN-AFF*
PER	WEA	*ART*
PER	VEH	*ART*
FAC	FAC	*PART-WHOLE, PHYS*
FAC	GPE	*PART-WHOLE, PHYS*
FAC	LOC	*PART-WHOLE, PHYS*
GPE	FAC	*PART-WHOLE, PHYS, ART*
GPE	GPE	*PART-WHOLE, PHYS, ORG-AFF*
GPE	LOC	*PART-WHOLE, PHYS*
GPE	ORG	*ORG-AFF*
GPE	WEA	*ART*
GPE	VEH	*ART*
LOC	FAC	*PART-WHOLE, PHYS*
LOC	GPE	*PART-WHOLE, PHYS*
LOC	LOC	*PART-WHOLE, PHYS*
ORG	ORG	*PART-WHOLE, ORG-AFF*
ORG	GPE	*PART-WHOLE, ORG-AFF, GEN-AFF*
ORG	WEA	*ART*
ORG	VEH	*ART*
ORG	FAC	*ART*
ORG	LOC	*GEN-AFF*
VEH	VEH	*PART-WHOLE*
WEA	WEA	*PART-WHOLE*

Table 9.3: ACE 05 dataset constraints.

SciERC dataset is specifically designed for the AI domain. It includes entity and relation annotations from a collection of documents from 500 AI paper abstracts. It contains entity types such as Task, Method, Metric and relation types such as *Use-for*, *Part-of* and *Compare*. SciERC is particularly suited for constructing knowledge graphs in the AI domain.

9.5.2 Dataset Constraints

In this section, we discuss the dataset constraints used in our work. For the CoNLL 2004 dataset, the constraints are based on the seminal work of Roth and Yih Roth and Yih [2004], which we report in Table 9.1. The constraints for this dataset are relatively simple, allowing only five triplet combinations, for instance, (Peop, Org, Work_For). For the ACE 05 dataset, no constraints were publicly available. Thus, we decided to design the constraints manually by examining the annotation guidelines provided by the Linguistic Data Consortium dataset's annotation guidelines [1], resulting in the set of constraints reported in Table 9.3. As shown in the table, the task for the ACE 05 dataset is highly complex, with more than 40 possible triples compared to CoNLL 2004, which only has 5. Finally, for SciERC, we do not include dataset-specific constraints as the annotation guideline is not detailed enough to permit that, and the presence of ill-defined entities such as *Generic* and *Other-ScientificTerm* makes it difficult (see Table 2.1).

9.5.3 Evaluation Metrics

For the named entity recognition (NER) task, we use span-level evaluation, demanding precise entity boundary and type predictions. In evaluating relations, we employ two metrics: (1) Boundary Evaluation (**REL**), which requires correct prediction of entity boundaries and relation types, and (2) Strict Evaluation (**REL+**), which also necessitates correct entity type prediction. We report the micro-averaged F1 score following previous works.

9.5.4 Hyperparameters

In this study, we implemented our model using BERT [Devlin et al., 2019] or ALBERT [Lan et al., 2019] for the CoNLL 2004 and ACE 05 datasets. For the SciERC dataset, we opted for SciBERT [Beltagy et al., 2019a], aligning with previous works. We detail the hyperparameters in Table 8.2. Our model was implemented using PyTorch and trained on a server equipped with A100 GPUs.

9.5.5 Baselines

We primarily compare our model, *EnriCo*, with comparable approaches from the literature in terms of model size. **DyGIE++** [Wadden et al., 2019b] is a model that uses a pretrained

[1]https://www.ldc.upenn.edu/collaborations/past-projects/ace/annotation-tasks-and-specifications

		ACE 05			CoNLL 2004			SciERC		
Model	Backbone	Entity	REL	REL+	Entity	REL	REL+	Entity	REL	REL+
DYGIE++ [Wadden et al., 2019b]	BB & SciB	88.6	63.4	–	–	–	–	67.5	48.4	–
Tab-Seq [Wang and Lu, 2020]	ALB	89.5	–	64.3	90.1	73.8	73.6	–	–	–
PURE [Zhong and Chen, 2021a]	ALB & SciB	89.7	69.0	65.6	–	–	–	66.6	48.2	35.6
PFN [Yan et al., 2021c]	ALB & SciB	89.0	–	66.8	–	–	–	66.8	–	38.4
UniRE [Wang et al., 2021]	ALB & SciB	89.9	–	66.0	–	–	–	68.4	–	36.9
TablERT [Ma et al., 2022]	ALB	87.8	65.0	61.8	**90.5**	73.2	72.2	–	–	–
UTC-IE [Yan et al., 2023]	ALB & SciB	89.9	–	**67.8**	–	–	–	69.0	–	38.8
UIE [Lu et al., 2022b]	T5	–	–	66.6	–	75.0	–	–	–	36.5
ChatGPT [Wadhwa et al., 2023]	(*Few-shot*)	–	9.04	–	–	76.5	–	–	17.92	–
EnriCo (*Ours*)	ALB & SciB	**90.1**	**69.1**	67.6	89.8	**76.6**	**76.6**	**69.3**	**50.5**	**40.2**
	BL	88.8	67.2	64.7	89.9	73.5	73.5	–	–	–

Table 9.5: Main results. *Entity* refers to the F1 score for entity recognition, *REL* for relaxed relation extraction, and *REL+* for strict relation extraction. The *Backbone* column indicates the underlying architecture for each model (ALB for `albert-xxlarge-v1` [Lan et al., 2019], BL for `bert-large-cased` [Devlin et al., 2019], and SciB for `scibert-base-uncased` [Beltagy et al., 2019a]).

transformer to compute contextualized representations and employs graph propagation to update the representations of spans for prediction. **PURE** [Zhong and Chen, 2021a] is a pipeline model for the information extraction task that learns distinct contextual representations for entities and relations. **PFN** [Yan et al., 2021c] introduces methods that model two-way interactions between the task by partitioning and filtering features. **UniRE** [Wang et al., 2021] proposes a joint entity and relation extraction model that uses a unified label space for entity and relation classification. **Tab-Seq** [Wang and Lu, 2020] tackles the task of joint information extraction by treating it as a table-filling problem. Similarly, in **TablERT** [Ma et al., 2022], entities and relations are treated as tables, and the model utilizes two-dimensional CNNs to effectively capture and model local dependencies within these table-like structures. Finally, **UTC-IE** [Yan et al., 2023] treats the task as token-pair classification. It incorporates Plusformer to facilitate axis-aware interactions through plus-shaped self-attention and local interactions via Convolutional Neural Networks over token pairs. We also included evaluations of generative approaches for information extraction, comprising **UIE** [Lu et al., 2022b], which fine-tunes a T5 model for information extraction, and **ChatGPT** [Wadhwa et al., 2023] prompted using few-shot demonstrations.

9.6 Results and Analysis

9.6.1 Main Results

The main results of our experiments are reported in Table 2. On ACE 05, our model obtains the highest results in entity evaluation and is second in relation prediction, slightly under-performing *UTC-IE*. On CoNLL 2004, our model surpasses the best *non-generative*

Setting	**ACE 05**			**CoNLL 2004**			**SciERC**		
	ENT	REL	REL+	ENT	REL	REL+	ENT	REL	REL+
`Unconstr.`	60.2	68.6	67.1	60.6	76.3	76.1	**69.4**	**51.4**	39.7
`Joint`	90.1	68.9	67.5	**89.8**	76.6	76.6	69.3	50.8	40.5
`Ent First`	**90.2**	**69.1**	**67.6**	**89.8**	**76.7**	**76.7**	69.3	50.5	40.2
`Rel First`	90.0	68.7	66.9	89.7	76.5	76.5	**69.4**	51.1	**40.7**

Table 9.6: Performance Comparison of Decoding Algorithms. We compare unconstrained and constrained approaches.

baseline by a large margin. Specifically, it obtains 76.6 on relation evaluation, achieving a +3 F1 improvement compared to *Tab-Seq*. Similarly, on SciERC, it also obtains strong results for both entities and relations, outperforming *UTC-IE* by 0.3 and 1.4 on entity and relation F1, respectively. Furthermore, our model also show competitive performance against generative models, *UIE* and *ChatGPT*. On CoNLL 2004, *ChatGPT* performs quite well due to the simplicity of relations in this dataset. However, on more complex datasets (SciERC and ACE 05), its performance is far behind, showing the benefits of fine-tuning task-specific models for the task. Overall, our model showcases strong performance across all datasets, demonstrating the utility of our proposed framework.

9.6.2 Decoding Algorithms

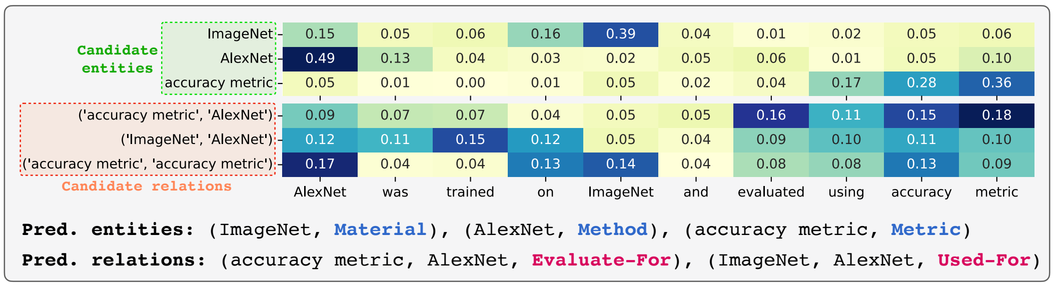
In Table 9.6, we report the performance of our model using different decoding algorithms described in Section 9.4. We observe that, as expected, *unconstrained decoding* is the least competitive, except on SciERC where we did not apply a domain constraint. In particular, *unconstrained decoding* performance on entity recognition can be very poor, especially for ACE 05 and CoNLL 2004, where it falls behind the constrained method by almost 30 points in terms of F1, mainly due to span boundary and span overlap errors. For relation extraction, constrained decoding can improve by up to 0.5, 0.6, and 1.0 points in terms of the F1 score on ACE 05, CoNLL 2004, and SciERC, respectively. These results demonstrate that structural and domain constraints are important not only for improving coherence but also for performance. Furthermore, we notice that the performance difference between different constrained decoding methods (*Joint*, *Entity First*, and *Relation First*) is minimal across datasets. However, *Entity First* is the most beneficial one as it can be implemented efficiently without the need for using an *ASP* solver, making it up to 3x to 4x faster than other alternatives (Table 9.2).

9.6.3 Refine Layer Ablation

We perform an ablation analysis in Table 9.7 to assess the effectiveness of the refine layer, specifically examining the contributions of the entity-level and relation-level refine layers described in Section 9.3.5. To ensure a fair comparison, we maintain a similar number

Setting	**ACE 05**			**CoNLL 2004**			**SciERC**		
	ENT	REL	REL+	ENT	REL	REL+	ENT	REL	REL+
`Full.`	**90.1**	**69.1**	**67.6**	**89.8**	**76.6**	**76.6**	69.3	**50.5**	**40.2**
`No ent.`	89.7	67.6	65.9	89.7	76.1	76.1	**69.8**	50.4	39.8
`No rel.`	**90.2**	68.1	66.5	**89.8**	76.4	76.4	69.0	50.3	39.6
`No both`	89.7	67.6	65.7	89.5	75.7	75.7	68.0	50.0	38.8

Table 9.7: Ablation experiments. With and without refine layer at the entity/relation level.



Pred. entities: (ImageNet, **Material**), (AlexNet, **Method**), (accuracy metric, **Metric**)
Pred. relations: (accuracy metric, AlexNet, **Evaluate-For**), (ImageNet, AlexNet, **Used-For**)

Figure 9.4: Attention visualization. This illustrate the attention score of candidate entities and candidate relations with the input sequence, averaged across attention heads.

of parameters for all compared variants. In general, our model with the full configuration—incorporating both entity and relation level interactions—achieves the most competitive scores across the datasets. However, removing either the entity or relation level interaction does not significantly impact performance, whereas removing both leads to a more substantial drop in performance.

9.6.4 Attention Visualization

In Figure 9.4, we present the attention visualization of the READ module for entities and relations, highlighting their interaction with the input sequence. This visualization depicts the attention scores averaged across all attention heads. The examples illustrated demonstrate that each span generally attends most to its corresponding position in the input text. However, intriguingly, we also observe attention to certain clue words such as “on” and “using”, which may contribute to type prediction. For relations, attention is directed to both head and tail spans constituting the relation. However, contextual information beyond the spans is attended to; for example, the word “evaluated” receives significant attention from the (“accuracy metric”, “AlexNet”) relation, indicating the Evaluate-For relation between the two spans. Similarly, in the same line, the word “trained” is highly attended to by the (“ImageNet”, “AlexNet”) pair.

9.7 Related Works

Joint IE The field of information extraction (IE) has evolved from traditional pipeline models, which sequentially handle entity recognition [Chiu and Nichols, 2015, Lample et al., 2016a] and relation extraction [Zelenko et al., 2002a, Bach and Badaskar, 2007, Lin et al., 2016, Wu et al., 2017], to end-to-end models. These approaches aim to mitigate error propagation [Brin, 1999, Nadeau and Sekine, 2007] by jointly optimizing entity and relation extraction [Roth and Yih, 2004, Fu et al., 2019a, Sun et al., 2021a, Ye et al., 2022a], enhancing the interaction between the two task and overall performance. Proposed approaches include table-filling methods [Wang and Lu, 2020, Ma et al., 2022], span pair classification [Eberts and Ulges, 2019, Wadden et al., 2019b], set prediction [Sui et al., 2020], augmented sequence tagging [Ji et al., 2020] and the use of unified labels for the task [Wang et al., 2021, Yan et al., 2023]. In addition, recently, the usage of generative models [OpenAI, 2023a] has become popular for this task, where input texts are encoded and decoded into augmented language [Paolini et al., 2021]. Some of these approaches conduct fine-tuning on labeled datasets [Lu et al., 2022b, Fei et al., 2022], and others use prompting techniques on large language models such as ChatGPT [Wadhwa et al., 2023].

Higher-order attention Recent works have proposed higher-order interactions for structured prediction models. For instance, Floquet et al. [2023] employed span-level attention for parsing, utilizing linear transformers to circumvent quadratic complexity of dot-product attention. The work of Zaratiana et al. [2022c] employed span-level Graph Attention Networks [Velickovic et al., 2017] to enhance span representations for Named Entity Recognition (NER), using overlap information as edges. However, their approach is slow and takes huge memory due to the substantial size of the overlap graph, characterized by numerous nodes and edges. In our work, we address this challenge by implementing a filtering mechanism to alleviate computational inefficiencies. Similarly, Ji et al. [2023] leveraged span-level attention by restricting the number of attended spans for each span using predefined heuristic. In contrast, our proposed method dynamically selects them. Additionally, Zhu et al. [2023] utilized span-to-token attention for Named Entity Recognition (NER). Our model extends their approach by incorporating both span- and relation-level interaction.

9.8 Conclusion

In summary, this chapter introduces *EnriCo*, a novel model crafted for joint entity and relation extraction tasks. By integrating span-level and relation-level attention mechanisms, our model fosters richer representations of spans and their interactions. The incorporation of a filtering mechanism efficiently manages computational complexity, while the integration of learned biases and constraint-based decoding further enhances the precision of model predictions. Experimental evaluations across benchmark datasets demonstrate the efficacy and performance of our proposed model.

Chapter 10

Conclusion and Perspectives

Summary of Contributions

In this thesis, we explored approaches for entity and relation extraction. Our contributions were centered around, firstly, decoding algorithms which ensure that the output of the model follows some predefined structural constraints. Secondly, we also contributed to the development of novel architectures, which are either structurally motivated (such as GNNer and GraphER) or expressive of model interactions, for instance, through rich interactions (EnriCO, GLiNER).

Named Entity Recognition

For named entity recognition, we investigated span-based models with a focus on improving the structure of their output by preventing overlapping spans. Initially, we examined the potential of modifying the model architecture to eliminate overlaps, particularly by incorporating graph neural networks to integrate overlapping information into span representations. Although this approach reduced the number of overlapping spans, it did so at the expense of increased complexity. The dense overlap graph of spans proved unsuitable for modern GNN libraries. Furthermore, we discovered that while adding overlap information through GNNs marginally enhances performance, it may also negatively impact recall by excessively limiting overlaps, thus constraining the model's span prediction capabilities.

To address these issues, we developed a new decoding method for span-based models that optimizes a given score, serving as a more structured alternative to the commonly used greedy decoding. This exact decoding method utilizes the maximum weighted independent set algorithm, enabling efficient output of a solution that maximizes the sum of scores. Additionally, we proposed an exhaustive enumeration approach, allowing practitioners to maximize an arbitrary score function, such as average confidence.

Moreover, we introduced the filtered semi-Markov CRF (Fsemi-CRF) model, identifying and addressing the weaknesses of standard semi-Markov CRF models, such as quadratic complexity and poor empirical performance relative to local span-based models. The Fsemi-CRF model leverages a strong local model to select high-quality candidate spans and employs a generalized semi-Markov CRF on these filtered spans. During training, both the local model and semi-CRF are jointly optimized using a multitask loss. The decoding stage involves a two-step process: filtering candidates with a local model followed by graph search algorithms like Bellman-Ford to determine the optimal set of entity spans. This model demonstrates robust performance on benchmarks by integrating the strengths of both local span-based and semi-CRF models.

Generalist and Ligthweight Model for NER

In this thesis, we introduced GLiNER, an efficient and general model for named entity recognition. By aligning entity type representations with span representations in a latent space and training with synthetic data encompassing thousands of distinct entity types, GLiNER achieves impressive performance across NER benchmarks without the need for further fine-tuning. Our findings demonstrate that a well-designed BERT-based architecture can rival large language models like GPT in zero-shot scenarios.

Joint Entity and Relation Extraction

For relation extraction, we proposed three innovative and structurally motivated architectures. Firstly, we proposed the ATG framework employs a generative encoder-decoder approach, setting it apart from other generative models by using spans, rather than word tokens, as the basic units for generating nodes in the output graph. This span-based approach significantly shortens the output sequence compared to text-based models, while also ensuring the sequence forms a valid and coherent graph.

We further introduced the GraphER architecture, which frames the task of joint entity and relation extraction as a graph structure learning (GSL) problem. Given the inherent nature of the task as a graph prediction exercise—where entities are nodes and relations are edges connecting node pairs—we found that transformers are more effective than traditional message-passing GNNs for tasks involving structured graph data.

Lastly, we developed EnriCO, a specialized architecture for joint entity and relation extraction (JERE), which facilitates rich interactions among input tokens, spans, and relations through an attention mechanism. This mechanism enables each model component to selectively access relevant information for accurate predictions. EnriCO also integrates joint decoding to enforce structural coherence in the output information extraction graph, utilizing Answer Set Programming to ensure globally optimal entity-relation pairings.

Future directions and limitations

Generalist Model for Named Entity Recognition

Our proposed model, GLiNER, offers an efficient alternative to large language models for zero-shot NER. However, there are still many ways to improve it.

Context Length The current versions of GLiNER are limited to 512 subtokens, which restricts the number of entities and the length of text that can be processed by the model. This limitation can impact the model's ability to fully capture and understand context in longer documents, potentially reducing the accuracy of entity recognition. A future direction could involve using a backbone that supports a longer maximum context size, allowing GLiNER to process more extended texts and identify a greater number of entities within a single pass. This enhancement would open up opportunities for applying GLiNER to more complex and extensive documents, such as legal texts, scientific papers, or comprehensive reports, where understanding the broader context is crucial for accurate NER.

More Detailed Instructions GLiNER has been trained on a dataset with entity types that are brief and narrowly defined. In the future, it could be fine-tuned to extract entities described by more detailed instructions, beyond just entity type names. This presents an opportunity to enhance the model's flexibility and accuracy by allowing it to understand and extract entities from more complex and nuanced descriptions. Such an improvement could enable GLiNER to handle a wider variety of NER tasks, particularly in specialized domains where entity definitions may be more intricate and context-dependent.

Domain-Specific Models GLiNER has been trained using data from a diverse range of domains, making it a powerful general model for NER. However, in some cases, domain-specific models are beneficial as they offer significantly better performance within their specific domains, which can be crucial in certain scenarios. In the future, it would be interesting to develop specialized versions of GLiNER across various domains such as medicine, law, social media, and finance.

Limitations of GLiNER The current GLiNER model uses a cross-encoder approach, where a transformer encoder processes both entity types and input text in a unified manner. While effective, this approach has several drawbacks. One significant limitation is that it requires processing both the entity types and the text together, which can be inefficient and restrict scalability. A potential improvement would involve adopting a bi-encoder architecture, where entity types and text are processed separately. This would allow for the pre-computation of entity type embeddings, enabling the model to scale more effectively to thousands of entity types. Another limitation of the original GLiNER model is its sensitivity to label interaction. The model's performance can be affected by the order of entity types in the prompt, with shuffling potentially altering predictions. Additionally, adding new labels to the prompt can influence the model's accuracy. To address these issues, we plan to explore methods to reduce unwanted label interactions, such as modifying the attention mechanism to prevent these interactions, thereby making the model more robust and consistent in its predictions.

General and Structured Model for Information Extraction

My long-term goal is to develop an efficient, lightweight, and general model for information extraction, akin to GLiNER but applicable more broadly. With the foundational work presented in our models such as ATG, GraphER, and EnriCO, we have established a solid architectural base for joint information extraction tasks. We envision these models being adaptable to operate conditioned on a user-provided knowledge graph schema, thereby generating a custom graph from unstructured textual content. Below is my roadmap to achieve this goal:

- **Architecture**: Modify the architectures of ATG, GraphER, or EnriCO to enable the output of user-specified entity and relation types. In this case, entity and relation embeddings could be computed similarly to how it is done in GLiNER.
- **Decoding**: Implement or adapt the decoding strategy from the EnriCO model, where the coherence of entity and relation associations is constrained by domain knowledge from knowledge graph schema. Decoding could use a combinatorial approach such as Answer Set Programming (ASP) to produce optimal outputs or more efficient alternative using greedy decoding.
- **Synthetic Data**: The method to train this model involves generating synthetic data using powerful large language models like OpenAI's GPT-4 or Anthropic's Claude. The model could be prompted to simultaneously generate text, knowledge graph schema, and output graphs.

I am convinced that the development of this model could profoundly influence the field of information extraction. The ability to produce a knowledge graph as an output holds vast potential for real-world applications, transforming raw data into structured, actionable insights. However, this would require building richer datasets with diverse types of entities and relations, as well as varied text lengths.

Bibliography

M. Agrawal, S. Hegselmann, H. Lang, Y. Kim, and D. A. Sontag. Large language models are few-shot clinical information extractors. In *Conference on Empirical Methods in Natural Language Processing*, 2022. URL https://api.semanticscholar.org/CorpusID:249062918.

A. Akbik, D. Blythe, and R. Vollgraf. Contextual string embeddings for sequence labeling. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 1638–1649, Santa Fe, New Mexico, USA, Aug. 2018. Association for Computational Linguistics. URL https://aclanthology.org/C18-1139.

U. Alon and E. Yahav. On the bottleneck of graph neural networks and its practical implications, 2021.

Y. Altun, M. Johnson, and T. Hofmann. Investigating loss functions and optimization methods for discriminative learning of label sequences. In *Proceedings of the 2003 Conference on Empirical Methods in Natural Language Processing*, pages 145–152, 2003. URL https://aclanthology.org/W03-1019.

A. Amini, L. Du, and R. Cotterell. Structured voronoi sampling. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. URL https://openreview.net/forum?id=vf77fTbgG3.

G. Andrew. A hybrid Markov/semi-Markov conditional random field for sequence segmentation. In *Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing*, pages 465–472, Sydney, Australia, July 2006. Association for Computational Linguistics. URL https://aclanthology.org/W06-1655.

W. Antoun, F. Baly, and H. Hajj. AraBERT: Transformer-based model for Arabic language understanding. In *Proceedings of the 4th Workshop on Open-Source Arabic Corpora and Processing Tools, with a Shared Task on Offensive Language Detection*, pages 9–15, Marseille, France, May 2020. European Language Resource Association. ISBN 979-10-95546-51-1. URL https://aclanthology.org/2020.osact-1.2.

D. E. Appelt, J. R. Hobbs, J. Bear, D. J. Israel, and M. Tyson. Fastus: A finite-state processor for information extraction from real-world text. In *International Joint Conference on Artificial Intelligence*, 1993. URL https://api.semanticscholar.org/CorpusID:11268011.

D. Ashok and Z. C. Lipton. Promptner: Prompting for named entity recognition. *ArXiv*, abs/2305.15444, 2023a. URL https://api.semanticscholar.org/CorpusID:258887456.

D. Ashok and Z. C. Lipton. Promptner: Prompting for named entity recognition, 2023b.

G. Attardi, D. Sartiano, and M. Simi. Biaffine dependency and semantic graph parsing for EnhancedUniversal dependencies. In *Proceedings of the 17th International Conference on Parsing Technologies and the IWPT 2021 Shared Task on Parsing into Enhanced Universal Dependencies (IWPT 2021)*, pages 184–188, Online, Aug. 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.iwpt-1.19. URL https://aclanthology.org/2021.iwpt-1.19.

N. Bach and S. Badaskar. A review of relation extraction. 2007. URL https://api.semanticscholar.org/CorpusID:9249117.

J. Bastings, I. Titov, W. Aziz, D. Marcheggiani, and K. Sima'an. Graph convolutional encoders for syntax-aware neural machine translation. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1957–1967, Copenhagen, Denmark, Sept. 2017. Association for Computational Linguistics. doi: 10.18653/v1/D17-1209. URL https://aclanthology.org/D17-1209.

D. Beck, G. Haffari, and T. Cohn. Graph-to-sequence learning using gated graph neural networks, 2018.

I. Beltagy, K. Lo, and A. Cohan. Scibert: A pretrained language model for scientific text. In *EMNLP 2019*, 2019a.

I. Beltagy, K. Lo, and A. Cohan. SciBERT: A pretrained language model for scientific text. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3615–3620, Hong Kong, China, Nov. 2019b. Association for Computational Linguistics. doi: 10.18653/v1/D19-1371. URL https://aclanthology.org/D19-1371.

I. Beltagy, K. Lo, and A. Cohan. Scibert: A pretrained language model for scientific text, 2019c.

O. Bender, F. J. Och, and H. Ney. Maximum entropy models for named entity recognition. In *Proceedings of the Seventh Conference on Natural Language Learning at HLT-NAACL 2003*, pages 148–151, 2003. URL https://aclanthology.org/W03-0420.

N. Bodenstab, A. Dunlop, K. Hall, and B. Roark. Beam-width prediction for efficient context-free parsing. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 440–449, Portland, Oregon, USA, June 2011. Association for Computational Linguistics. URL https://aclanthology.org/P11-1045.

G. Brewka, T. Eiter, and M. Truszczynski. Answer set programming at a glance. *Communications of the ACM*, 54:92 – 103, 2011. URL https://api.semanticscholar.org/CorpusID:17746168.

S. Brin. Extracting patterns and relations from the world wide web. In *The World Wide Web and Databases*, 1999. ISBN 978-3-540-48909-2.

T. B. Brown, B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, S. Agarwal, A. Herbert-Voss, G. Krueger, T. J. Henighan, R. Child, A. Ramesh, D. M. Ziegler, J. Wu, C. Winter, C. Hesse, M. Chen, E. Sigler, M. Litwin, S. Gray, B. Chess, J. Clark, C. Berner, S. McCandlish, A. Radford, I. Sutskever, and D. Amodei. Language models are few-shot learners. *ArXiv*, abs/2005.14165, 2020. URL `https://api.semanticscholar.org/CorpusID:218971783`.

R. Bunescu and M. Paşca. Using encyclopedic knowledge for named entity disambiguation. In *11th Conference of the European Chapter of the Association for Computational Linguistics*, pages 9–16, Trento, Italy, Apr. 2006. Association for Computational Linguistics. URL `https://aclanthology.org/E06-1002`.

M. Candito. Auxiliary tasks to boost biaffine semantic dependency parsing. In *Findings of the Association for Computational Linguistics: ACL 2022*, pages 2422–2429, Dublin, Ireland, May 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.findings-acl.190. URL `https://aclanthology.org/2022.findings-acl.190`.

N. D. Cao, G. Izacard, S. Riedel, and F. Petroni. Autoregressive entity retrieval. In *International Conference on Learning Representations*, 2021. URL `https://openreview.net/forum?id=5k8F6UU39V`.

X. Carreras and L. Màrquez. Introduction to the CoNLL-2004 shared task: Semantic role labeling. In *Proceedings of the Eighth Conference on Computational Natural Language Learning (CoNLL-2004) at HLT-NAACL 2004*, pages 89–97, Boston, Massachusetts, USA, May 6 - May 7 2004. Association for Computational Linguistics. URL `https://aclanthology.org/W04-2412`.

R. Caruana. Multitask learning. *Machine Learning*, 28:41–75, 1997.

D. Chen, A. Fisch, J. Weston, and A. Bordes. Reading Wikipedia to answer open-domain questions. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1870–1879, Vancouver, Canada, July 2017. Association for Computational Linguistics. doi: 10.18653/v1/P17-1171. URL `https://aclanthology.org/P17-1171`.

D. Chen, Y. Lin, W. Li, P. Li, J. Zhou, and X. Sun. Measuring and relieving the over-smoothing problem for graph neural networks from the topological view, 2019.

T. Chen, S. Saxena, L. Li, D. J. Fleet, and G. Hinton. Pix2seq: A language modeling framework for object detection. In *International Conference on Learning Representations*, 2022. URL `https://openreview.net/forum?id=e42KbIw6Wb`.

W.-L. Chiang, Z. Li, Z. Lin, Y. Sheng, Z. Wu, H. Zhang, L. Zheng, S. Zhuang, Y. Zhuang, J. E. Gonzalez, I. Stoica, and E. P. Xing. Vicuna: An open-source chatbot impressing gpt-4 with 90%* chatgpt quality, March 2023. URL `https://lmsys.org/blog/2023-03-30-vicuna/`.

H. L. Chieu and H. T. Ng. Named entity recognition: A maximum entropy approach using global information. In *COLING 2002: The 19th International Conference on Computational Linguistics*, 2002. URL https://aclanthology.org/C02-1025.

H. L. Chieu and H. T. Ng. Named entity recognition with a maximum entropy approach. In *Proceedings of the Seventh Conference on Natural Language Learning at HLT-NAACL 2003*, pages 160–163, 2003. URL https://aclanthology.org/W03-0423.

N. A. Chinchor. Overview of MUC-7. In *Seventh Message Understanding Conference (MUC-7): Proceedings of a Conference Held in Fairfax, Virginia, April 29 - May 1, 1998*, 1998. URL https://aclanthology.org/M98-1001.

J. P. C. Chiu and E. Nichols. Named entity recognition with bidirectional lstm-cnns. *Transactions of the Association for Computational Linguistics*, 2015.

J. P. C. Chiu and E. Nichols. Named entity recognition with bidirectional lstm-cnns, 2016.

H. W. Chung, L. Hou, S. Longpre, B. Zoph, Y. Tay, W. Fedus, Y. Li, X. Wang, M. Dehghani, S. Brahma, A. Webson, S. S. Gu, Z. Dai, M. Suzgun, X. Chen, A. Chowdhery, A. Castro-Ros, M. Pellat, K. Robinson, D. Valter, S. Narang, G. Mishra, A. Yu, V. Zhao, Y. Huang, A. Dai, H. Yu, S. Petrov, E. H. Chi, J. Dean, J. Devlin, A. Roberts, D. Zhou, Q. V. Le, and J. Wei. Scaling instruction-finetuned language models, 2022.

K. Clark, M.-T. Luong, Q. V. Le, and C. D. Manning. Electra: Pre-training text encoders as discriminators rather than generators, 2020.

N. Collier, T. Ohta, Y. Tsuruoka, Y. Tateisi, and J.-D. Kim. Introduction to the bio-entity recognition task at JNLPBA. In *Proceedings of the International Joint Workshop on Natural Language Processing in Biomedicine and its Applications (NLPBA/BioNLP)*, pages 73–78, Geneva, Switzerland, Aug. 28th and 29th 2004. COLING. URL https://aclanthology.org/W04-1213.

A. Conneau, K. Khandelwal, N. Goyal, V. Chaudhary, G. Wenzek, F. Guzmán, E. Grave, M. Ott, L. Zettlemoyer, and V. Stoyanov. Unsupervised cross-lingual representation learning at scale. In *Annual Meeting of the Association for Computational Linguistics*, 2019. URL https://api.semanticscholar.org/CorpusID:207880568.

C. Corro. A dynamic programming algorithm for span-based nested named-entity recognition in $o(n^2)$. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 10712–10724, Toronto, Canada, July 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.acl-long.598. URL https://aclanthology.org/2023.acl-long.598.

S. Cucerzan. Large-scale named entity disambiguation based on Wikipedia data. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 708–716, Prague, Czech Republic, June 2007. Association for Computational Linguistics. URL https://aclanthology.org/D07-1074.

L. Cui, Y. Wu, J. Liu, S. Yang, and Y. Zhang. Template-based named entity recognition using bart. In *Findings*, 2021a. URL `https://api.semanticscholar.org/CorpusID:235313658`.

L. Cui, Y. Wu, J. Liu, S. Yang, and Y. Zhang. Template-based named entity recognition using BART. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 1835–1845, Online, Aug. 2021b. Association for Computational Linguistics. doi: 10.18653/v1/2021.findings-acl.161. URL `https://aclanthology.org/2021.findings-acl.161`.

L. Cui, Y. Wu, J. Liu, S. Yang, and Y. Zhang. Template-based named entity recognition using bart, 2021c.

H. Cunningham, D. Maynard, K. Bontcheva, and V. Tablan. GATE: an architecture for development of robust HLT applications. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 168–175, Philadelphia, Pennsylvania, USA, July 2002. Association for Computational Linguistics. doi: 10.3115/1073083.1073112. URL `https://aclanthology.org/P02-1022`.

N. V. Cuong, N. Ye, W. S. Lee, and H. L. Chieu. Conditional random field with high-order dependencies for sequence labeling and segmentation. *Journal of Machine Learning Research*, 15(28):981–1009, 2014. URL `http://jmlr.org/papers/v15/cuong14a.html`.

H. Daumé and D. Marcu. Learning as search optimization: approximate large margin methods for structured prediction. *Proceedings of the 22nd international conference on Machine learning*, 2005.

Y. Dauxais, U. Zaratiana, M. Laneuville, S. D. Hernandez, P. Holat, and C. Grosman. Towards automation of topic taxonomy construction. In *Advances in Intelligent Data Analysis XX: 20th International Symposium on Intelligent Data Analysis, IDA 2022, Rennes, France, April 20–22, 2022, Proceedings*, page 26–38, Berlin, Heidelberg, 2022. Springer-Verlag. ISBN 978-3-031-01332-4. doi: 10.1007/978-3-031-01333-1_3. URL `https://doi.org/10.1007/978-3-031-01333-1_3`.

L. Derczynski, K. Bontcheva, and I. Roberts. Broad Twitter corpus: A diverse named entity recognition resource. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 1169–1179, Osaka, Japan, Dec. 2016. The COLING 2016 Organizing Committee. URL `https://aclanthology.org/C16-1111`.

T. Derr, Y. Ma, and J. Tang. Signed graph convolutional network, 2018.

J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics. doi: 10.18653/v1/N19-1423. URL `https://aclanthology.org/N19-1423`.

S. Ding, G. Cong, C.-Y. Lin, and X. Zhu. Using conditional random fields to extract contexts and answers of questions from online forums. In *Proceedings of ACL-08: HLT*, pages 710–718, Columbus, Ohio, June 2008. Association for Computational Linguistics. URL https://aclanthology.org/P08-1081.

K. Dixit and Y. Al-Onaizan. Span-level model for relation extraction. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5308–5314, Florence, Italy, July 2019. Association for Computational Linguistics. doi: 10.18653/v1/P19-1525. URL https://aclanthology.org/P19-1525.

T. M. T. Do and T. Artières. Neural conditional random fields. In *AISTATS*, 2010.

C. dos Santos, B. Xiang, and B. Zhou. Classifying relations by ranking with convolutional neural networks. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 626–634, Beijing, China, July 2015. Association for Computational Linguistics. doi: 10.3115/v1/P15-1061. URL https://aclanthology.org/P15-1061.

T. Dozat and C. D. Manning. Deep biaffine attention for neural dependency parsing. In *International Conference on Learning Representations*, 2017. URL https://openreview.net/forum?id=Hk95PK9le.

R. I. Doğan, R. Leaman, and Z. Lu. Ncbi disease corpus: A resource for disease name recognition and concept normalization. *Journal of Biomedical Informatics*, 47:1–10, 2014. ISSN 1532-0464. doi: https://doi.org/10.1016/j.jbi.2013.12.006. URL https://www.sciencedirect.com/science/article/pii/S1532046413001974.

M. Eberts and A. Ulges. Span-based joint entity and relation extraction with transformer pre-training. In *European Conference on Artificial Intelligence*, 2019. URL https://api.semanticscholar.org/CorpusID:202583766.

M. Eberts and A. Ulges. Span-based joint entity and relation extraction with transformer pre-training. *ArXiv*, abs/1909.07755, 2020.

M. Eberts and A. Ulges. An end-to-end model for entity-level relation extraction using multi-instance learning. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 3650–3660, Online, Apr. 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.eacl-main.319. URL https://aclanthology.org/2021.eacl-main.319.

N. El Khbir, N. Tomeh, and T. Charnois. ArabIE: Joint entity, relation and event extraction for Arabic. In *Proceedings of the The Seventh Arabic Natural Language Processing Workshop (WANLP)*, pages 331–345, Abu Dhabi, United Arab Emirates (Hybrid), Dec. 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.wanlp-1.31. URL https://aclanthology.org/2022.wanlp-1.31.

J. L. Elman. Finding structure in time. *Cognitive science*, 14(2):179–211, 1990.

H. Fei, S. Wu, J. Li, B. Li, F. Li, L. Qin, M. Zhang, M. Zhang, and T.-S. Chua. LasUIE: Unifying information extraction with latent adaptive structure-aware generative language model. In *Advances in Neural Information Processing Systems*, 2022. URL https://openreview.net/forum?id=a8qX5RG36jd.

M. Fey and J. E. Lenssen. Fast graph representation learning with pytorch geometric, 2019.

N. Floquet, N. Tomeh, J. Le Roux, and T. Charnois. Attention sur les spans pour l'analyse syntaxique en constituants. In *Actes de CORIA-TALN 2023. Actes de la 30e Conférence sur le Traitement Automatique des Langues Naturelles (TALN), volume 2 : travaux de recherche originaux – articles courts*, pages 37–45, Paris, France, 6 2023. ATALA. URL https://aclanthology.org/2023.jeptalnrecital-short.4.

J. G. Forney. Viterbi algorithm. In *Encyclopedia of Machine Learning*, 2010.

L. Franceschi, M. Niepert, M. Pontil, and X. He. Learning discrete structures for graph neural networks. In *Proceedings of the 36th International Conference on Machine Learning*, 2019.

J. Fu, X. Huang, and P. Liu. SpanNER: Named entity re-/recognition as span prediction. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 7183–7195, Online, Aug. 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.acl-long.558. URL https://aclanthology.org/2021.acl-long.558.

T.-J. Fu, P.-H. Li, and W.-Y. Ma. Graphrel: Modeling text as relational graphs for joint entity and relation extraction. In *Annual Meeting of the Association for Computational Linguistics*, 2019a.

T.-J. Fu, P.-H. Li, and W.-Y. Ma. GraphRel: Modeling text as relational graphs for joint entity and relation extraction. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 1409–1418, Florence, Italy, July 2019b. Association for Computational Linguistics. doi: 10.18653/v1/P19-1136. URL https://aclanthology.org/P19-1136.

R. Gaizauskas and Y. Wilks. Information extraction: Beyond document retrieval. In *International Journal of Computational Linguistics & Chinese Language Processing, Volume 3, Number 2, August 1998*, pages 17–60, Aug. 1998. URL https://aclanthology.org/O98-4002.

L. Gao, S. R. Biderman, S. Black, L. Golding, T. Hoppe, C. Foster, J. Phang, H. He, A. Thite, N. Nabeshima, S. Presser, and C. Leahy. The pile: An 800gb dataset of diverse text for language modeling. *ArXiv*, abs/2101.00027, 2020. URL https://api.semanticscholar.org/CorpusID:230435736.

X. Gao, W. Hu, and Z. Guo. Exploring structure-adaptive graph learning for robust semi-supervised classification, 2019.

M. Gardner, J. Grus, M. Neumann, O. Tafjord, P. Dasigi, N. Liu, M. Peters, M. Schmitz, and L. Zettlemoyer. Allennlp: A deep semantic natural language processing platform, 2018a.

M. Gardner, J. Grus, M. Neumann, O. Tafjord, P. Dasigi, N. F. Liu, M. Peters, M. Schmitz, and L. Zettlemoyer. AllenNLP: A deep semantic natural language processing platform. In *Proceedings of Workshop for NLP Open Source Software (NLP-OSS)*, pages 1–6, Melbourne, Australia, July 2018b. Association for Computational Linguistics. doi: 10.18653/v1/W18-2501. URL https://aclanthology.org/W18-2501.

M. Gebser, R. Kaminski, B. Kaufmann, and T. Schaub. Clingo= asp+ control: Preliminary report. *arXiv preprint arXiv:1405.3694*, 2014.

S. Geng, M. Josifoski, M. Peyrard, and R. West. Grammar-constrained decoding for structured NLP tasks without finetuning. In H. Bouamor, J. Pino, and K. Bali, editors, *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 10932–10952, Singapore, Dec. 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.emnlp-main.674. URL https://aclanthology.org/2023.emnlp-main.674.

J. Gilmer, S. S. Schoenholz, P. F. Riley, O. Vinyals, and G. E. Dahl. Neural message passing for quantum chemistry. In D. Precup and Y. W. Teh, editors, *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 1263–1272. PMLR, 06–11 Aug 2017. URL https://proceedings.mlr.press/v70/gilmer17a.html.

J. Giorgi, G. D. Bader, and B. Wang. A sequence-to-sequence approach for document-level relation extraction. In *Workshop on Biomedical Natural Language Processing*, 2022. URL https://api.semanticscholar.org/CorpusID:247939302.

S. Gross, O. Russakovsky, C. B., and S. Batzoglou. Training conditional random fields for maximum labelwise accuracy. In B. Schölkopf, J. Platt, and T. Hoffman, editors, *Advances in Neural Information Processing Systems*, volume 19. MIT Press, 2006. URL https://proceedings.neurips.cc/paper/2006/file/24b43fb034a10d78bec71274033b4096-Paper.pdf.

R. Guan, K. L. Man, F. Chen, S. Yao, R. Hu, X. Zhu, J. Smith, E. G. Lim, and Y. Yue. Findvehicle and vehiclefinder: A ner dataset for natural language-based vehicle retrieval and a keyword-based cross-modal vehicle retrieval system. *arXiv preprint arXiv:2304.10893*, 2023.

N. M. Guerreiro, P. Colombo, P. Piantanida, and A. Martins. Optimal transport for unsupervised hallucination detection in neural machine translation. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 13766–13784, Toronto, Canada, July 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.acl-long.770. URL https://aclanthology.org/2023.acl-long.770.

D. Gupta, A. Ekbal, and P. Bhattacharyya. A deep neural network based approach for entity extraction in code-mixed Indian social media text. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*, Miyazaki, Japan, May 2018. European Language Resources Association (ELRA). URL `https://aclanthology.org/L18-1278`.

P. Gupta, H. Schütze, and B. Andrassy. Table filling multi-task recurrent neural network for joint entity and relation extraction. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 2537–2547, Osaka, Japan, Dec. 2016. The COLING 2016 Organizing Committee. URL `https://aclanthology.org/C16-1239`.

W. L. Hamilton. Graph representation learning. *Synthesis Lectures on Artificial Intelligence and Machine Learning*, 14(3):1–159, 2020. doi: 10.2200/S01045ED1V01Y202009AIM046.

W. L. Hamilton, Z. Ying, and J. Leskovec. Inductive representation learning on large graphs. In *Neural Information Processing Systems*, 2017. URL `https://api.semanticscholar.org/CorpusID:4755450`.

W. L. Hamilton, R. Ying, and J. Leskovec. Representation learning on graphs: Methods and applications, 2018.

R. Han, T. Peng, C. Yang, B. Wang, L. Liu, and X. Wan. Is information extraction solved by chatgpt? an analysis of performance, evaluation criteria, robustness and errors. *ArXiv*, abs/2305.14450, 2023. URL `https://api.semanticscholar.org/CorpusID:258865200`.

P. He, J. Gao, and W. Chen. Debertav3: Improving deberta using electra-style pre-training with gradient-disentangled embedding sharing. *ArXiv*, abs/2111.09543, 2021. URL `https://api.semanticscholar.org/CorpusID:244346093`.

S. Hochreiter and J. Schmidhuber. Long short-term memory. In *Neural computation*, volume 9, pages 1735–1780. MIT Press, 1997.

A. Holtzman, J. Buys, M. Forbes, and Y. Choi. The curious case of neural text degeneration. *ArXiv*, abs/1904.09751, 2019.

Y. Hou, C. Jochim, M. Gleize, F. Bonin, and D. Ganguly. TDMSci: A specialized corpus for scientific literature entity tagging of tasks datasets and metrics. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 707–714, Online, Apr. 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.eacl-main.59. URL `https://aclanthology.org/2021.eacl-main.59`.

J. Y. Hsiao, C. Y. Tang, and R. S. Chang. An efficient algorithm for finding a maximum weight 2-independent set on interval graphs. *Information Processing Letters*, 43(5):229–235, 1992. ISSN 0020-0190. doi: https://doi.org/10.1016/0020-0190(92)90216-I. URL `https://www.sciencedirect.com/science/article/pii/002001909290216I`.

Z. Huang, W. Xu, and K. Yu. Bidirectional lstm-crf models for sequence tagging, 2015.

E. Jang, S. Gu, and B. Poole. Categorical reparameterization with gumbel-softmax. In *International Conference on Learning Representations*, 2017. URL https://openreview.net/forum?id=rkE3y85ee.

B. Ji, J. Yu, S. Li, J. Ma, Q. Wu, Y. Tan, and H. Liu. Span-based joint entity and relation extraction with attention-based span-specific and contextual semantic representations. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 88–99, Barcelona, Spain (Online), Dec. 2020. International Committee on Computational Linguistics. doi: 10.18653/v1/2020.coling-main.8. URL https://aclanthology.org/2020.coling-main.8.

P. Ji, S. Yang, and K. Tu. Improving span representation by efficient span-level attention. In H. Bouamor, J. Pino, and K. Bali, editors, *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 11184–11192, Singapore, Dec. 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.findings-emnlp.747. URL https://aclanthology.org/2023.findings-emnlp.747.

J. Jiang and C. Zhai. A systematic exploration of the feature space for relation extraction. In *Human Language Technologies 2007: The Conference of the North American Chapter of the Association for Computational Linguistics; Proceedings of the Main Conference*, pages 113–120, Rochester, New York, Apr. 2007. Association for Computational Linguistics. URL https://aclanthology.org/N07-1015.

V. Jijkoun, J. Mur, and M. de Rijke. Information extraction for question answering: Improving recall through syntactic patterns. In *COLING 2004: Proceedings of the 20th International Conference on Computational Linguistics*, pages 1284–1290, Geneva, Switzerland, aug 23–aug 27 2004. COLING. URL https://aclanthology.org/C04-1188.

W. Jin, Y. Ma, X. Liu, X. Tang, S. Wang, and J. Tang. Graph structure learning for robust graph neural networks, 2020.

D. S. Johnson, M. Yannakakis, and C. H. Papadimitriou. On generating all maximal independent sets. *Information Processing Letters*, 27(3):119–123, 1988.

M. Josifoski, N. De Cao, M. Peyrard, F. Petroni, and R. West. GenIE: Generative information extraction. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 4626–4643, Seattle, United States, July 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.naacl-main.342. URL https://aclanthology.org/2022.naacl-main.342.

N. Kambhatla. Combining lexical, syntactic, and semantic features with maximum entropy models for information extraction. In *Proceedings of the ACL Interactive Poster and Demonstration Sessions*, pages 178–181, Barcelona, Spain, July 2004. Association for Computational Linguistics. URL https://aclanthology.org/P04-3022.

J. Kim, D. T. Nguyen, S. Min, S. Cho, M. Lee, H. Lee, and S. Hong. Pure transformers are powerful graph learners. In A. H. Oh, A. Agarwal, D. Belgrave, and K. Cho, editors, *Advances in Neural Information Processing Systems*, 2022. URL `https://openreview.net/forum?id=um2BxfgkT2_`.

J.-D. Kim, T. Ohta, Y. Tateisi, and J. Tsujii. Genia corpus—a semantically annotated corpus for bio-textmining. *Bioinformatics*, 19(suppl_1):i180–i182, 2003.

Y.-B. Kim, K. Stratos, and R. Sarikaya. Pre-training of hidden-unit CRFs. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 192–198, Beijing, China, July 2015. Association for Computational Linguistics. doi: 10.3115/v1/P15-2032. URL `https://aclanthology.org/P15-2032`.

D. P. Kingma and J. Ba. Adam: A method for stochastic optimization, 2017.

T. Kipf and M. Welling. Semi-supervised classification with graph convolutional networks. *ArXiv*, abs/1609.02907, 2016. URL `https://api.semanticscholar.org/CorpusID:3144218`.

T. Kipf and M. Welling. Semi-supervised classification with graph convolutional networks. *ArXiv*, abs/1609.02907, 2017.

T. Kipf, E. Fetaya, K.-C. Wang, M. Welling, and R. Zemel. Neural relational inference for interacting systems. *arXiv preprint arXiv:1802.04687*, 2018.

L. Kong, C. Dyer, and N. A. Smith. Segmental recurrent neural networks. *CoRR*, abs/1511.06018, 2016.

M. Krallinger, O. Rabal, F. Leitner, M. Vazquez, D. Salgado, Z. Lu, R. Leaman, Y. Lu, D. Ji, D. M. Lowe, et al. The chemdner corpus of chemicals and drugs and its annotation principles. *Journal of cheminformatics*, 7(1):1–17, 2015.

I. Kulikov, M. Eremeev, and K. Cho. Characterizing and addressing the issue of oversmoothing in neural autoregressive sequence modeling. In *Proceedings of the 2nd Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics and the 12th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1115–1124, Online only, Nov. 2022. Association for Computational Linguistics. URL `https://aclanthology.org/2022.aacl-main.82`.

S. Kumar, B. Paria, and Y. Tsvetkov. Gradient-based constrained sampling from language models. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 2251–2277, Abu Dhabi, United Arab Emirates, Dec. 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.emnlp-main.144. URL `https://aclanthology.org/2022.emnlp-main.144`.

J. D. Lafferty, A. McCallum, and F. C. N. Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the Eighteenth International Conference on Machine Learning*, ICML '01, page 282–289, San Francisco, CA, USA, 2001. Morgan Kaufmann Publishers Inc. ISBN 1558607781.

V. D. Lai, N. T. Ngo, A. P. B. Veyseh, H. Man, F. Dernoncourt, T. Bui, and T. H. Nguyen. Chatgpt beyond english: Towards a comprehensive evaluation of large language models in multilingual learning. *ArXiv*, abs/2304.05613, 2023. URL https://api.semanticscholar.org/CorpusID:258079179.

G. Lample, M. Ballesteros, S. Subramanian, K. Kawakami, and C. Dyer. Neural architectures for named entity recognition. In *NAACL 2016*, 2016a.

G. Lample, M. Ballesteros, S. Subramanian, K. Kawakami, and C. Dyer. Neural architectures for named entity recognition. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 260–270, San Diego, California, June 2016b. Association for Computational Linguistics. doi: 10.18653/v1/N16-1030. URL https://aclanthology.org/N16-1030.

Z. Lan, M. Chen, S. Goodman, K. Gimpel, P. Sharma, and R. Soricut. Albert: A lite bert for self-supervised learning of language representations. *ArXiv*, abs/1909.11942, 2019. URL https://api.semanticscholar.org/CorpusID:202888986.

Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

K. Lee, L. He, M. Lewis, and L. Zettlemoyer. End-to-end neural coreference resolution. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 188–197, Copenhagen, Denmark, Sept. 2017. Association for Computational Linguistics. doi: 10.18653/v1/D17-1018. URL https://aclanthology.org/D17-1018.

Y. Lei, C. Hu, G. Ma, and R. Zhang. Keyphrase extraction with incomplete annotated training data. In *Proceedings of the Seventh Workshop on Noisy User-generated Text (W-NUT 2021)*, pages 26–34, Online, Nov. 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.wnut-1.4. URL https://aclanthology.org/2021.wnut-1.4.

M. Lewis, Y. Liu, N. Goyal, M. Ghazvininejad, A. Mohamed, O. Levy, V. Stoyanov, and L. Zettlemoyer. Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension, 2019.

M. Lewis, Y. Liu, N. Goyal, M. Ghazvininejad, A. Mohamed, O. Levy, V. Stoyanov, and L. Zettlemoyer. BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7871–7880, Online, July 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.acl-main.703. URL https://aclanthology.org/2020.acl-main.703.

D. Li, B. Hu, and Q. Chen. Prompt-based text entailment for low-resource named entity recognition. *ArXiv*, abs/2211.03039, 2022a. URL https://api.semanticscholar.org/CorpusID:252819341.

G. Li, P. Wang, and W. Ke. Revisiting large language models as zero-shot relation extractors. In H. Bouamor, J. Pino, and K. Bali, editors, *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 6877–6892, Singapore, Dec. 2023a. Association for Computational Linguistics. URL `https://aclanthology.org/2023.findings-emnlp.459`.

H. Li and B. Yuan. Chinese word segmentation. In *Proceedings of the 12th Pacific Asia Conference on Language, Information and Computation*, pages 212–217, Singapore, Feb. 1998. Chinese and Oriental Languages Information Processing Society. doi: http://hdl.handle.net/2065/12081. URL `https://aclanthology.org/Y98-1020`.

J. Li, Y. Sun, R. J. Johnson, D. Sciaky, C.-H. Wei, R. Leaman, A. P. Davis, C. J. Mattingly, T. C. Wiegers, and Z. Lu. Biocreative v cdr task corpus: a resource for chemical disease relation extraction. *Database*, 2016, 2016.

R. Li, S. Wang, F. Zhu, and J. Huang. Adaptive graph convolutional neural networks, 2018.

S. Li, H. Ji, and J. Han. Document-level event argument extraction by conditional generation. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 894–908, Online, June 2021a. Association for Computational Linguistics. doi: 10.18653/v1/2021.naacl-main.69. URL `https://aclanthology.org/2021.naacl-main.69`.

X. Li, J. Feng, Y. Meng, Q. Han, F. Wu, and J. Li. A unified mrc framework for named entity recognition. *ArXiv*, abs/1910.11476, 2019. URL `https://api.semanticscholar.org/CorpusID:204902024`.

X. Li, J. Feng, Y. Meng, Q. Han, F. Wu, and J. Li. A unified mrc framework for named entity recognition, 2020.

X. Li, J. Thickstun, I. Gulrajani, P. S. Liang, and T. B. Hashimoto. Diffusion-lm improves controllable text generation. In *Advances in Neural Information Processing Systems*, 2022b. URL `https://proceedings.neurips.cc/paper_files/paper/2022/file/1be5bc25d50895ee656b8c2d9eb89d6a-Paper-Conference.pdf`.

Y. Li, lemao liu, and S. Shi. Empirical analysis of unlabeled entity problem in named entity recognition. In *International Conference on Learning Representations*, 2021b. URL `https://openreview.net/forum?id=5jRVa89sZk`.

Z. Li, L. Wang, X. Sun, Y. Luo, Y. Zhu, D. Chen, Y. Luo, X. Zhou, Q. Liu, S. Wu, L. Wang, and J. X. Yu. Gslb: The graph structure learning benchmark, 2023b.

P. Liang. Semi-supervised learning for natural language. 2005.

Y. Liang, S. Dhall, and S. Lakshmivarahan. On the problem of finding all maximum weight independent sets in interval and circular-arc graphs. In *[Proceedings] 1991 Symposium on Applied Computing*, pages 465–470, 1991. doi: 10.1109/SOAC.1991.143921.

Y. Lin, S. Shen, Z. Liu, H. Luan, and M. Sun. Neural relation extraction with selective attention over instances. In *Annual Meeting of the Association for Computational Linguistics*, 2016.

Y. Lin, H. Ji, F. Huang, and L. Wu. A joint neural model for information extraction with global features. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7999–8009, Online, July 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.acl-main.713. URL `https://aclanthology.org/2020.acl-main.713`.

Y. Lin, Y. Meng, X. Sun, Q. Han, K. Kuang, J. Li, and F. Wu. Bertgcn: Transductive text classification by combining gcn and bert, 2021.

F. Liu, J. Zhao, B. Lv, B. Xu, and H. Yu. Product named entity recognition based on hierarchical hidden Markov model. In *Proceedings of the Fourth SIGHAN Workshop on Chinese Language Processing*, 2005. URL `https://aclanthology.org/I05-3006`.

J. Liu, P. Pasupat, D. S. Cyphers, and J. R. Glass. Asgard: A portable architecture for multilingual dialogue systems. *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 8386–8390, 2013. URL `https://api.semanticscholar.org/CorpusID:14903208`.

T. Liu, Y. E. Jiang, N. Monath, R. Cotterell, and M. Sachan. Autoregressive structured prediction with language models. In *Findings of the Association for Computational Linguistics: EMNLP 2022*, pages 993–1005, Abu Dhabi, United Arab Emirates, Dec. 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.findings-emnlp.70. URL `https://aclanthology.org/2022.findings-emnlp.70`.

X. Liu, Z. Luo, and H. Huang. Jointly multiple events extraction via attention-based graph information aggregation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1247–1256, Brussels, Belgium, Oct.-Nov. 2018. Association for Computational Linguistics. doi: 10.18653/v1/D18-1156. URL `https://aclanthology.org/D18-1156`.

X. Liu, X. You, X. Zhang, J. Wu, and P. Lv. Tensor graph convolutional networks for text classification, 2020a.

Y. Liu, W. Che, J. Guo, B. Qin, and T. Liu. Exploring segment representations for neural segmentation models. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence*, IJCAI'16, page 2880–2886. AAAI Press, 2016a. ISBN 9781577357704.

Y. Liu, W. Che, J. Guo, B. Qin, and T. Liu. Exploring segment representations for neural segmentation models. In *IJCAI*, 2016b.

Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, and V. Stoyanov. Roberta: A robustly optimized bert pretraining approach. *ArXiv*, abs/1907.11692, 2019a. URL `https://api.semanticscholar.org/CorpusID:198953378`.

Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, and V. Stoyanov. Roberta: A robustly optimized bert pretraining approach, 2019b.

Z. Liu, Y. Xu, T. Yu, W. Dai, Z. Ji, S. Cahyawijaya, A. Madotto, and P. Fung. Crossner: Evaluating cross-domain named entity recognition. In *AAAI Conference on Artificial Intelligence*, 2020b. URL `https://api.semanticscholar.org/CorpusID:227736891`.

S. Longpre, L. Hou, T. Vu, A. Webson, H. W. Chung, Y. Tay, D. Zhou, Q. V. Le, B. Zoph, J. Wei, and A. Roberts. The flan collection: Designing data and methods for effective instruction tuning. In *International Conference on Machine Learning*, 2023. URL `https://api.semanticscholar.org/CorpusID:256415991`.

I. Loshchilov and F. Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations*, 2017. URL `https://api.semanticscholar.org/CorpusID:53592270`.

C. Lou, S. Yang, and K. Tu. Nested named entity recognition as latent lexicalized constituency parsing. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 6183–6198, Dublin, Ireland, May 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.acl-long.428. URL `https://aclanthology.org/2022.acl-long.428`.

J. Lou, Y. Lu, D. Dai, W. Jia, H. Lin, X. Han, L. Sun, and H. Wu. Universal information extraction as unified semantic matching. In *AAAI Conference on Artificial Intelligence*, 2023. URL `https://api.semanticscholar.org/CorpusID:255546103`.

Y. Lu, Q. Liu, D. Dai, X. Xiao, H. Lin, X. Han, L. Sun, and H. Wu. Unified structure generation for universal information extraction. In *Annual Meeting of the Association for Computational Linguistics*, 2022a. URL `https://api.semanticscholar.org/CorpusID:247619149`.

Y. Lu, Q. Liu, D. Dai, X. Xiao, H. Lin, X. Han, L. Sun, and H. Wu. Unified structure generation for universal information extraction. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 5755–5772, Dublin, Ireland, May 2022b. Association for Computational Linguistics. doi: 10.18653/v1/2022.acl-long.395. URL `https://aclanthology.org/2022.acl-long.395`.

Y. Luan, L. He, M. Ostendorf, and H. Hajishirzi. Multi-task identification of entities, relations, and coreference for scientific knowledge graph construction. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3219–3232, Brussels, Belgium, Oct.-Nov. 2018. Association for Computational Linguistics. doi: 10.18653/v1/D18-1360. URL `https://aclanthology.org/D18-1360`.

Y. Luan, D. Wadden, L. He, A. Shah, M. Ostendorf, and H. Hajishirzi. A general framework for information extraction using dynamic span graphs. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 3036–3046, Minneapolis, Minnesota, June 2019a. Association for Computational Linguistics. doi: 10.18653/v1/N19-1308. URL https://aclanthology.org/N19-1308.

Y. Luan, D. Wadden, L. He, A. Shah, M. Ostendorf, and H. Hajishirzi. A general framework for information extraction using dynamic span graphs, 2019b.

D. Luo, W. Cheng, W. Yu, B. Zong, J. Ni, H. Chen, and X. Zhang. Learning to drop: Robust graph neural network via topological denoising. In *Proceedings of the 14th ACM International Conference on Web Search and Data Mining*, WSDM '21, page 779–787, New York, NY, USA, 2021. Association for Computing Machinery. ISBN 9781450382977. doi: 10.1145/3437963.3441734. URL https://doi.org/10.1145/3437963.3441734.

X. Ma and E. Hovy. End-to-end sequence labeling via bi-directional lstm-cnns-crf, 2016.

Y. Ma, T. Hiraoka, and N. Okazaki. Joint entity and relation extraction based on table labeling using convolutional neural networks. In *Proceedings of the Sixth Workshop on Structured Prediction for NLP*, pages 11–21, Dublin, Ireland, May 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.spnlp-1.2. URL https://aclanthology.org/2022.spnlp-1.2.

S. Malmasi, A. Fang, B. Fetahu, S. Kar, and O. Rokhlenko. Multiconer: A large-scale multilingual dataset for complex named entity recognition. In *International Conference on Computational Linguistics*, 2022. URL https://api.semanticscholar.org/CorpusID:251953674.

J. Maloney and M. Niv. TAGARAB: A fast, accurate Arabic name recognizer using high-precision morphological analysis. In *Computational Approaches to Semitic Languages*, 1998. URL https://aclanthology.org/W98-1002.

P. Manakul, A. Liusie, and M. Gales. SelfCheckGPT: Zero-resource black-box hallucination detection for generative large language models. In H. Bouamor, J. Pino, and K. Bali, editors, *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 9004–9017, Singapore, Dec. 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.emnlp-main.557. URL https://aclanthology.org/2023.emnlp-main.557.

D. Marcheggiani and I. Titov. Graph convolutions over constituent trees for syntax-aware semantic role labeling. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 3915–3928, Online, Nov. 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.emnlp-main.322. URL https://aclanthology.org/2020.emnlp-main.322.

I. Melnyk, P. Dognin, and P. Das. Knowledge graph generation from text. In *Findings of the Association for Computational Linguistics: EMNLP 2022*, pages 1610–1622, Abu Dhabi, United Arab Emirates, Dec. 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.findings-emnlp.116. URL https://aclanthology.org/2022.findings-emnlp.116.

A. Mikheev, M. Moens, and C. Grover. Named entity recognition without gazetteers. In *Conference of the European Chapter of the Association for Computational Linguistics*, 1999. URL `https://api.semanticscholar.org/CorpusID:7332330`.

S. Min, X. Lyu, A. Holtzman, M. Artetxe, M. Lewis, H. Hajishirzi, and L. Zettlemoyer. Rethinking the role of demonstrations: What makes in-context learning work? *ArXiv*, abs/2202.12837, 2022. URL `https://api.semanticscholar.org/CorpusID:247155069`.

S. Mishra, D. Khashabi, C. Baral, and H. Hajishirzi. Cross-task generalization via natural language crowdsourcing instructions. In *Annual Meeting of the Association for Computational Linguistics*, 2021. URL `https://api.semanticscholar.org/CorpusID:237421373`.

D. Mollá, M. van Zaanen, and D. Smith. Named entity recognition for question answering. In *Proceedings of the Australasian Language Technology Workshop 2006*, pages 51–58, Sydney, Australia, Nov. 2006. URL `https://aclanthology.org/U06-1009`.

K. Murray and D. Chiang. Correcting length bias in neural machine translation. In *Proceedings of the Third Conference on Machine Translation: Research Papers*, pages 212–223, Brussels, Belgium, Oct. 2018. Association for Computational Linguistics. doi: 10.18653/v1/W18-6322. URL `https://aclanthology.org/W18-6322`.

D. Nadeau and S. Sekine. A survey of named entity recognition and classification. *Lingvisticae Investigationes*, 2007.

D. Nadeau, P. D. Turney, and S. Matwin. Unsupervised named-entity recognition: Generating gazetteers and resolving ambiguity. In *Canadian Conference on AI*, 2006. URL `https://api.semanticscholar.org/CorpusID:17071982`.

H. Nakayama. seqeval: A python framework for sequence labeling evaluation, 2018. URL `https://github.com/chakki-works/seqeval`. Software available from https://github.com/chakki-works/seqeval.

T. Nayak and H. T. Ng. Effective modeling of encoder-decoder architecture for joint entity and relation extraction. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, New York, NY, USA, February 7-12, 2020*, pages 8528–8535. AAAI Press, 2020. URL `https://ojs.aaai.org/index.php/AAAI/article/view/6374`.

H. T. Ng, J. L. P. Kwan, and Y. Xia. Question answering using a large text database: A machine learning approach. In *Proceedings of the 2001 Conference on Empirical Methods in Natural Language Processing*, 2001. URL `https://aclanthology.org/W01-0509`.

T. H. Nguyen and R. Grishman. Relation extraction: Perspective from convolutional neural networks. In *Proceedings of the 1st Workshop on Vector Space Modeling for Natural Language Processing*, pages 39–48, Denver, Colorado, June 2015. Association for Computational Linguistics. doi: 10.3115/v1/W15-1506. URL `https://aclanthology.org/W15-1506`.

M. Nickel, K. Murphy, V. Tresp, and E. Gabrilovich. A review of relational machine learning for knowledge graphs. *Proceedings of the IEEE*, 104(1):11–33, 2016. doi: 10.1109/JPROC.2015.2483592.

Y. Niu and G. Hirst. Analysis of semantic classes in medical text for question answering. In *Proceedings of the Conference on Question Answering in Restricted Domains*, pages 54–61, Barcelona, Spain, July 2004. Association for Computational Linguistics. URL `https://aclanthology.org/W04-0509`.

M. E. Okurowski. Information extraction overview. In *TIPSTER TEXT PROGRAM: PHASE I: Proceedings of a Workshop held at Fredricksburg, Virginia, September 19-23, 1993*, pages 117–121, Fredericksburg, Virginia, USA, Sept. 1993. Association for Computational Linguistics. doi: 10.3115/1119149.1119164. URL `https://aclanthology.org/X93-1012`.

OpenAI. Gpt-4 technical report. 2023a. URL `https://api.semanticscholar.org/CorpusID:257532815`.

OpenAI. Gpt-4 technical report, 2023b.

L. Ouyang, J. Wu, X. Jiang, D. Almeida, C. L. Wainwright, P. Mishkin, C. Zhang, S. Agarwal, K. Slama, A. Ray, J. Schulman, J. Hilton, F. Kelton, L. Miller, M. Simens, A. Askell, P. Welinder, P. Christiano, J. Leike, and R. Lowe. Training language models to follow instructions with human feedback, 2022. URL `https://arxiv.org/abs/2203.02155`.

B. Paassen, D. Grattarola, D. Zambon, C. Alippi, and B. E. Hammer. Graph edit networks. In *International Conference on Learning Representations*, 2021. URL `https://openreview.net/forum?id=dlEJsyHGeaL`.

M. Pal and G. Bhattacharjee. A sequential algorithm for finding a maximum weight k-independent set on interval graphs. *International Journal of Computer Mathematics*, 60 (3-4):205–214, 1996.

X. Pan, B. Zhang, J. May, J. Nothman, K. Knight, and H. Ji. Cross-lingual name tagging and linking for 282 languages. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1946–1958, Vancouver, Canada, July 2017. Association for Computational Linguistics. doi: 10.18653/v1/P17-1178. URL `https://aclanthology.org/P17-1178`.

G. Paolini, B. Athiwaratkun, J. Krone, J. Ma, A. Achille, R. ANUBHAI, C. N. dos Santos, B. Xiang, and S. Soatto. Structured prediction as translation between augmented natural languages. In *International Conference on Learning Representations*, 2021. URL `https://openreview.net/forum?id=US-TP-xnXI`.

A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Köpf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala. Pytorch: An imperative style, high-performance deep learning library. *ArXiv*, abs/1912.01703, 2019a.

A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Köpf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala. Pytorch: An imperative style, high-performance deep learning library, 2019b.

F. Peng and A. McCallum. Accurate information extraction from research papers using conditional random fields. In *Proceedings of the Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics: HLT-NAACL 2004*, pages 329–336, Boston, Massachusetts, USA, May 2 - May 7 2004. Association for Computational Linguistics. URL `https://aclanthology.org/N04-1042`.

G. Petasis, F. Vichot, F. Wolinski, G. Paliouras, V. Karkaletsis, and C. D. Spyropoulos. Using machine learning to maintain rule-based named-entity recognition and classification systems. In *Proceedings of the 39th Annual Meeting of the Association for Computational Linguistics*, pages 426–433, Toulouse, France, July 2001. Association for Computational Linguistics. doi: 10.3115/1073012.1073067. URL `https://aclanthology.org/P01-1055`.

M. E. Peters, M. Neumann, M. Iyyer, M. Gardner, C. Clark, K. Lee, and L. Zettlemoyer. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 2227–2237, New Orleans, Louisiana, June 2018. Association for Computational Linguistics. doi: 10.18653/v1/N18-1202. URL `https://aclanthology.org/N18-1202`.

A.-M. Popescu and O. Etzioni. Extracting product features and opinions from reviews. In *Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing*, pages 339–346, Vancouver, British Columbia, Canada, Oct. 2005. Association for Computational Linguistics. URL `https://aclanthology.org/H05-1043`.

A.-M. Popescu, B. Nguyen, and O. Etzioni. OPINE: Extracting product features and opinions from reviews. In *Proceedings of HLT/EMNLP 2005 Interactive Demonstrations*, pages 32–33, Vancouver, British Columbia, Canada, Oct. 2005. Association for Computational Linguistics. URL `https://aclanthology.org/H05-2017`.

V. Punyakanok, D. Roth, W. tau Yih, and D. Zimak. Learning and inference over constrained output. In *International Joint Conference on Artificial Intelligence*, 2005.

S. Pyysalo and S. Ananiadou. Anatomical entity mention recognition at literature scale. *Bioinformatics*, 30(6):868–875, 2014.

C. Qin, A. Zhang, Z. Zhang, J. Chen, M. Yasunaga, and D. Yang. Is chatgpt a general-purpose natural language processing task solver? *ArXiv*, abs/2302.06476, 2023. URL `https://api.semanticscholar.org/CorpusID:256827430`.

L. Rabiner. A tutorial on hidden markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286, 1989. doi: 10.1109/5.18626.

C. Raffel, N. M. Shazeer, A. Roberts, K. Lee, S. Narang, M. Matena, Y. Zhou, W. Li, and P. J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *J. Mach. Learn. Res.*, 21:140:1–140:67, 2019. URL https://api.semanticscholar.org/CorpusID:204838007.

V. Raman, S. Saurabh, and S. Sikdar. Efficient exact algorithms through enumerating maximal independent sets and other techniques. *Theory of Computing Systems*, 41(3):563–587, 2007.

L. Ratinov and D. Roth. Design challenges and misconceptions in named entity recognition. In *Proceedings of the Thirteenth Conference on Computational Natural Language Learning (CoNLL-2009)*, pages 147–155, Boulder, Colorado, June 2009a. Association for Computational Linguistics. URL https://aclanthology.org/W09-1119.

L.-A. Ratinov and D. Roth. Design challenges and misconceptions in named entity recognition. In *Conference on Computational Natural Language Learning*, 2009b. URL https://api.semanticscholar.org/CorpusID:1859014.

F. Reiss, S. Raghavan, R. Krishnamurthy, H. Zhu, and S. Vaithyanathan. An algebraic approach to rule-based information extraction. *2008 IEEE 24th International Conference on Data Engineering*, pages 933–942, 2008. URL https://api.semanticscholar.org/CorpusID:5540395.

F. Ren, L. Zhang, S. Yin, X. Zhao, S. Liu, B. Li, and Y. Liu. A novel global feature-oriented relational triple extraction model based on table filling. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 2646–2656, Online and Punta Cana, Dominican Republic, Nov. 2021a. Association for Computational Linguistics. doi: 10.18653/v1/2021.emnlp-main.208. URL https://aclanthology.org/2021.emnlp-main.208.

L. Ren, C. Sun, H. Ji, and J. Hockenmaier. HySPA: Hybrid span generation for scalable text-to-graph extraction. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 4066–4078, Online, Aug. 2021b. Association for Computational Linguistics. doi: 10.18653/v1/2021.findings-acl.356. URL https://aclanthology.org/2021.findings-acl.356.

E. Riloff. Automatically constructing a dictionary for information extraction tasks. In *Proceedings of the Eleventh National Conference on Artificial Intelligence*, AAAI'93, page 811–816. AAAI Press, 1993. ISBN 0262510715.

B. Roark and K. Hollingshead. Classifying chart cells for quadratic complexity context-free inference. In *Proceedings of the 22nd International Conference on Computational Linguistics (Coling 2008)*, pages 745–752, Manchester, UK, Aug. 2008. Coling 2008 Organizing Committee. URL https://aclanthology.org/C08-1094.

A. Rogozhnikov. Einops: Clear and reliable tensor manipulations with einstein-like notation. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=oapKSVM2bcj.

B. Rosenfeld and R. Feldman. Using corpus statistics on entities to improve semi-supervised relation extraction from the web. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 600–607, Prague, Czech Republic, June 2007. Association for Computational Linguistics. URL `https://aclanthology.org/P07-1076`.

D. Roth and W. tau Yih. Integer linear programming inference for conditional random fields. *Proceedings of the 22nd international conference on Machine learning*, 2005.

D. Roth and W.-t. Yih. A linear programming formulation for global inference in natural language tasks. In *Proceedings of the Eighth Conference on Computational Natural Language Learning (CoNLL-2004) at HLT-NAACL 2004*, pages 1–8, Boston, Massachusetts, USA, May 6 - May 7 2004. Association for Computational Linguistics. URL `https://aclanthology.org/W04-2401`.

B. Rozière, J. Gehring, F. Gloeckle, S. Sootla, I. Gat, X. Tan, Y. Adi, J. Liu, T. Remez, J. Rapin, A. Kozhevnikov, I. Evtimov, J. Bitton, M. P. Bhatt, C. C. Ferrer, A. Grattafiori, W. Xiong, A. D'efossez, J. Copet, F. Azhar, H. Touvron, L. Martin, N. Usunier, T. Scialom, and G. Synnaeve. Code llama: Open foundation models for code. *ArXiv*, abs/2308.12950, 2023. URL `https://api.semanticscholar.org/CorpusID:261100919`.

A. Rush. Torch-struct: Deep structured prediction library. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 335–342, Online, July 2020a. Association for Computational Linguistics. doi: 10.18653/v1/2020.acl-demos.38. URL `https://aclanthology.org/2020.acl-demos.38`.

A. Rush and S. Petrov. Vine pruning for efficient multi-pass dependency parsing. In *The 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL '12)*, page Best Paper Award, 2012. URL `http://petrovi.de/data/naacl12.pdf`.

A. M. Rush. Torch-struct: Deep structured prediction library, 2020b.

O. Sainz, I. García-Ferrero, R. Agerri, O. L. de Lacalle, G. Rigau, and E. Agirre. GoLLIE: Annotation guidelines improve zero-shot information-extraction. In *The Twelfth International Conference on Learning Representations*, 2024. URL `https://openreview.net/forum?id=Y3wpuxd7u9`.

S. Sarawagi. Information extraction. *Foundations and Trends® in Databases*, 1(3):261–377, 2008. ISSN 1931-7883. doi: 10.1561/1900000003. URL `http://dx.doi.org/10.1561/1900000003`.

S. Sarawagi and W. W. Cohen. Semi-markov conditional random fields for information extraction. In *Neural Information Processing Systems*, 2004. URL `https://api.semanticscholar.org/CorpusID:14036493`.

S. Sarawagi and W. W. Cohen. Semi-markov conditional random fields for information extraction. In L. Saul, Y. Weiss, and L. Bottou, editors, *Advances in Neural Information Processing Systems*, volume 17. MIT Press, 2005. URL https://proceedings.neurips.cc/paper/2004/file/eb06b9db06012a7a4179b8f3cb5384d3-Paper.pdf.

M. Sato, H. Shindo, I. Yamada, and Y. Matsumoto. Segment-level neural conditional random fields for named entity recognition. In *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 97–102, Taipei, Taiwan, Nov. 2017. Asian Federation of Natural Language Processing. URL https://aclanthology.org/I17-2017.

Y.-M. Shang, H. Huang, and X.-L. Mao. Onerel: Joint entity and relation extraction with one module in one step. In *AAAI Conference on Artificial Intelligence*, 2022. URL https://api.semanticscholar.org/CorpusID:247362852.

B. Shao, Y. Gong, W. Qi, G. Cao, J. Ji, and X. Lin. Graph-based transformer with cross-candidate verification for semantic parsing. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(05):8807–8814, Apr. 2020. doi: 10.1609/aaai.v34i05.6408. URL https://ojs.aaai.org/index.php/AAAI/article/view/6408.

Y. Shen, X. Wang, Z. Tan, G. Xu, P. Xie, F. Huang, W. Lu, and Y. T. Zhuang. Parallel instance query network for named entity recognition. In *ACL*, 2022.

L. Smith, L. K. Tanabe, C.-J. Kuo, I. Chung, C.-N. Hsu, Y.-S. Lin, R. Klinger, C. M. Friedrich, K. Ganchev, M. Torii, et al. Overview of biocreative ii gene mention recognition. *Genome biology*, 9(2):1–19, 2008.

W. M. Soon, H. T. Ng, and D. C. Y. Lim. A machine learning approach to coreference resolution of noun phrases. *Computational Linguistics*, 27(4):521–544, 2001. doi: 10.1162/089120101753342653. URL https://aclanthology.org/J01-4004.

J. Straková, M. Straka, and J. Hajic. Neural architectures for nested NER through linearization. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5326–5331, Florence, Italy, July 2019. Association for Computational Linguistics. doi: 10.18653/v1/P19-1527. URL https://aclanthology.org/P19-1527.

D. Sui, Y. Chen, K. Liu, J. Zhao, X. Zeng, and S. Liu. Joint entity and relation extraction with set prediction networks. *IEEE transactions on neural networks and learning systems*, PP, 2020. URL https://api.semanticscholar.org/CorpusID:226237654.

C. Sun, Y. Gong, Y. Wu, M. Gong, D. Jiang, M. Lan, S. Sun, and N. Duan. Joint type inference on entities and relations via graph convolutional networks. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 1361–1370, Florence, Italy, July 2019. Association for Computational Linguistics. doi: 10.18653/v1/P19-1131. URL https://aclanthology.org/P19-1131.

H. Sun, B. Dhingra, M. Zaheer, K. Mazaitis, R. Salakhutdinov, and W. Cohen. Open domain question answering using early fusion of knowledge bases and text. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4231–4242, Brussels, Belgium, Oct.-Nov. 2018. Association for Computational Linguistics. doi: 10.18653/v1/D18-1455. URL `https://aclanthology.org/D18-1455`.

K. Sun, R. Zhang, S. Mensah, Y. Mao, and X. Liu. Progressive multi-task learning with controlled information flow for joint entity and relation extraction. *Proceedings of the AAAI Conference on Artificial Intelligence*, 35(15):13851–13859, May 2021a. doi: 10.1609/aaai.v35i15.17632. URL `https://ojs.aaai.org/index.php/AAAI/article/view/17632`.

Q. Sun, J. Li, H. Peng, J. Wu, X. Fu, C. Ji, and P. S. Yu. Graph structure learning with variational information bottleneck, 2021b.

B. Taillé, V. Guigue, G. Scoutheeten, and P. Gallinari. Let's stop incorrect comparisons in end-to-end relation extraction!, 2021.

Y. Tay, D. Bahri, D. Metzler, D.-C. Juan, Z. Zhao, and C. Zheng. Synthesizer: Rethinking self-attention in transformer models. *ArXiv*, abs/2005.00743, 2021.

E. F. Tjong Kim Sang and F. De Meulder. Introduction to the CoNLL-2003 shared task: Language-independent named entity recognition. In *Proceedings of the Seventh Conference on Natural Language Learning at HLT-NAACL 2003*, pages 142–147, 2003. URL `https://aclanthology.org/W03-0419`.

H. Touvron, T. Lavril, G. Izacard, X. Martinet, M.-A. Lachaux, T. Lacroix, B. Rozière, N. Goyal, E. Hambro, F. Azhar, A. Rodriguez, A. Joulin, E. Grave, and G. Lample. Llama: Open and efficient foundation language models. *ArXiv*, abs/2302.13971, 2023. URL `https://api.semanticscholar.org/CorpusID:257219404`.

T.-H. Tsai, C.-W. Wu, and W.-L. Hsu. Using maximum entropy to extract biomedical named entities without dictionaries. In *Companion Volume to the Proceedings of Conference including Posters/Demos and tutorial abstracts*, 2005. URL `https://aclanthology.org/I05-2046`.

Y. Tsuruoka and J. Tsujii. Boosting precision and recall of dictionary-based protein name recognition. In *Proceedings of the ACL 2003 Workshop on Natural Language Processing in Biomedicine*, pages 41–48, Sapporo, Japan, July 2003. Association for Computational Linguistics. doi: 10.3115/1118958.1118964. URL `https://aclanthology.org/W03-1306`.

N. Usunier, D. Buffoni, and P. Gallinari. Ranking with ordered weighted pairwise classification. In *Proceedings of the 26th Annual International Conference on Machine Learning*, ICML '09, page 1057–1064, New York, NY, USA, 2009. Association for Computing Machinery. ISBN 9781605585161. doi: 10.1145/1553374.1553509. URL `https://doi.org/10.1145/1553374.1553509`.

K. van Deemter and R. Kibble. What is coreference, and what should coreference annotation be? In *Coreference and Its Applications*, 1999. URL https://aclanthology.org/W99-0213.

L. van der Maaten, M. Welling, and L. K. Saul. Hidden-unit conditional random fields. In *AISTATS*, 2011.

A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin. Attention is all you need, 2017a.

A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. u. Kaiser, and I. Polosukhin. Attention is all you need. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017b. URL https://proceedings.neurips.cc/paper_files/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf.

A. Vaswani, N. M. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin. Attention is all you need. In *NIPS*, 2017c.

P. Velickovic, G. Cucurull, A. Casanova, A. Romero, P. Lio', and Y. Bengio. Graph attention networks. *ArXiv*, abs/1710.10903, 2017. URL https://api.semanticscholar.org/CorpusID:3292002.

P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Liò, and Y. Bengio. Graph attention networks. *arXiv preprint arXiv:1710.10903*, 2018.

P. Velickovic, G. Cucurull, A. Casanova, A. Romero, P. Lio', and Y. Bengio. Graph attention networks. *ArXiv*, abs/1710.10903, 2018.

P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Liò, and Y. Bengio. Graph attention networks. In *International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=rJXMpikCZ.

P. Veličković, L. Buesing, M. C. Overlan, R. Pascanu, O. Vinyals, and C. Blundell. Pointer graph networks, 2020.

T. Vieira and J. Eisner. Learning to prune: Exploring the frontier of fast and accurate parsing. *Transactions of the Association for Computational Linguistics*, 5:263–278, 2017.

O. Vinyals, M. Fortunato, and N. Jaitly. Pointer networks, 2017.

A. Viterbi. Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. *IEEE Transactions on Information Theory*, 13(2):260–269, 1967. doi: 10.1109/TIT.1967.1054010.

D. Wadden, U. Wennberg, Y. Luan, and H. Hajishirzi. Entity, relation, and event extraction with contextualized span representations. *ArXiv*, abs/1909.03546, 2019a.

D. Wadden, U. Wennberg, Y. Luan, and H. Hajishirzi. Entity, relation, and event extraction with contextualized span representations. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 5784–5789, Hong Kong, China, Nov. 2019b. Association for Computational Linguistics. doi: 10.18653/v1/D19-1585. URL `https://aclanthology.org/D19-1585`.

S. Wadhwa, S. Amir, and B. Wallace. Revisiting relation extraction in the era of large language models. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 15566–15589, Toronto, Canada, July 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.acl-long.868. URL `https://aclanthology.org/2023.acl-long.868`.

M. J. Wainwright and M. I. Jordan. Graphical models, exponential families, and variational inference. *Foundations and Trends® in Machine Learning*, 1(1–2):1–305, 2008. ISSN 1935-8237. doi: 10.1561/2200000001. URL `http://dx.doi.org/10.1561/2200000001`.

C. Walker, S. Strassel, J. Medero, and K. Maeda. Ace 2005 multilingual training corpus, Feb 2006a. URL `https://catalog.ldc.upenn.edu/LDC2006T06`.

C. Walker, S. Strassel, J. Medero, and K. Maeda. Ace 2005 multilingual training corpus. *Linguistic Data Consortium, Philadelphia*, 57:45, 2006b.

Z. Wan, F. Cheng, Z. Mao, Q. Liu, H. Song, J. Li, and S. Kurohashi. GPT-RE: In-context learning for relation extraction using large language models. In H. Bouamor, J. Pino, and K. Bali, editors, *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 3534–3547, Singapore, Dec. 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.emnlp-main.214. URL `https://aclanthology.org/2023.emnlp-main.214`.

J. Wang and W. Lu. Two are better than one: Joint entity and relation extraction with table-sequence encoders. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1706–1721, Online, Nov. 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.emnlp-main.133. URL `https://aclanthology.org/2020.emnlp-main.133`.

L. Wang, Z. Cao, G. de Melo, and Z. Liu. Relation classification via multi-level attention CNNs. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1298–1307, Berlin, Germany, Aug. 2016. Association for Computational Linguistics. doi: 10.18653/v1/P16-1123. URL `https://aclanthology.org/P16-1123`.

M. Wang and C. D. Manning. Effect of non-linear deep architecture in sequence labeling. In *Proceedings of the Sixth International Joint Conference on Natural Language Processing*, pages 1285–1291, Nagoya, Japan, Oct. 2013. Asian Federation of Natural Language Processing. URL `https://aclanthology.org/I13-1183`.

S. Wang, X. Sun, X. Li, R. Ouyang, F. Wu, T. Zhang, J. Li, and G. Wang. Gpt-ner: Named entity recognition via large language models, 2023a.

X. Wang, W. Zhou, C. Zu, H. Xia, T. Chen, Y. Zhang, R. Zheng, J. Ye, Q. Zhang, T. Gui, J. Kang, J. Yang, S. Li, and C. Du. Instructuie: Multi-task instruction tuning for unified information extraction. *ArXiv*, abs/2304.08085, 2023b. URL https://api.semanticscholar.org/CorpusID:258179792.

Y. Wang, C. Sun, Y. Wu, H. Zhou, L. Li, and J. Yan. UniRE: A unified label space for entity relation extraction. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 220–231, Online, Aug. 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.acl-long.19. URL https://aclanthology.org/2021.acl-long.19.

Y. Wang, S. Mishra, P. Alipoormolabashi, Y. Kordi, A. Mirzaei, A. Arunkumar, A. Ashok, A. S. Dhanasekaran, A. Naik, D. Stap, E. Pathak, G. Karamanolakis, H. G. Lai, I. Purohit, I. Mondal, J. Anderson, K. Kuznia, K. Doshi, M. Patel, K. K. Pal, M. Moradshahi, M. Parmar, M. Purohit, N. Varshney, P. R. Kaza, P. Verma, R. S. Puri, R. Karia, S. K. Sampat, S. Doshi, S. D. Mishra, S. Reddy, S. Patro, T. Dixit, X. Shen, C. Baral, Y. Choi, N. A. Smith, H. Hajishirzi, and D. Khashabi. Super-naturalinstructions: Generalization via declarative instructions on 1600+ nlp tasks. In *Conference on Empirical Methods in Natural Language Processing*, 2022. URL https://api.semanticscholar.org/CorpusID:253098274.

J. Wei, X. Wang, D. Schuurmans, M. Bosma, E. H. hsin Chi, F. Xia, Q. Le, and D. Zhou. Chain of thought prompting elicits reasoning in large language models. *ArXiv*, abs/2201.11903, 2022. URL https://api.semanticscholar.org/CorpusID:246411621.

R. Weischedel, S. Boisen, D. Bikel, R. Bobrow, M. Crystal, W. Ferguson, A. Wechsler, and The PLUM Research Group. Progress in information extraction. In *TIPSTER TEXT PROGRAM PHASE II: Proceedings of a Workshop held at Vienna, Virginia, May 6-8, 1996*, pages 127–138, Vienna, Virginia, USA, May 1996. Association for Computational Linguistics. doi: 10.3115/1119018.1119050. URL https://aclanthology.org/X96-1028.

R. Weischedel, M. Palmer, M. Marcus, E. Hovy, S. Pradhan, L. Ramshaw, N. Xue, A. Taylor, J. Kaufman, M. Franchini, et al. Ontonotes release 5.0 ldc2013t19. *Linguistic Data Consortium, Philadelphia, PA*, 23:170, 2013.

B. T. Willard and R. Louf. Efficient guided generation for large language models, 2023.

T. Wolf, L. Debut, V. Sanh, J. Chaumond, C. Delangue, A. Moi, P. Cistac, T. Rault, R. Louf, M. Funtowicz, J. Davison, S. Shleifer, P. von Platen, C. Ma, Y. Jernite, J. Plu, C. Xu, T. L. Scao, S. Gugger, M. Drame, Q. Lhoest, and A. M. Rush. Huggingface's transformers: State-of-the-art natural language processing, 2020.

L. Wu, F. Petroni, M. Josifoski, S. Riedel, and L. Zettlemoyer. Zero-shot entity linking with dense entity retrieval. In *EMNLP*, 2020.

L. Wu, Y. Chen, K. Shen, X. Guo, H. Gao, S. Li, J. Pei, and B. Long. Graph neural networks for natural language processing: A survey, 2021.

Y. Wu, D. Bamman, and S. J. Russell. Adversarial training for relation extraction. In *EMNLP 2017*, 2017.

Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, and P. S. Yu. A comprehensive survey on graph neural networks, 2019.

C. Xia, C. Zhang, T. Yang, Y. Li, N. Du, X. Wu, W. Fan, F. Ma, and P. Yu. Multi-grained named entity recognition. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 1430–1440, Florence, Italy, July 2019. Association for Computational Linguistics. doi: 10.18653/v1/P19-1138. URL https://aclanthology.org/P19-1138.

D. Xu, W. Chen, W. Peng, C. Zhang, T. Xu, X. Zhao, X. Wu, Y. Zheng, and E. Chen. Large language models for generative information extraction: A survey, 2023.

K. Xu, L. Wu, Z. Wang, M. Yu, L. Chen, and V. Sheinin. Exploiting rich syntactic information for semantic parsing with graph-to-sequence model, 2018.

M. Xue, B. Yu, Z. Zhang, T. Liu, Y. Zhang, and B. Wang. Coarse-to-fine pre-training for named entity recognition, 2020.

S. Xuewen, H. Heyan, J. Ping, and T. Yi-Kun. Reducing length bias in scoring neural machine translation via a causal inference method. In *Proceedings of the 20th Chinese National Conference on Computational Linguistics*, pages 874–885, Huhhot, China, Aug. 2021. Chinese Information Processing Society of China. URL https://aclanthology.org/2021.ccl-1.78.

H. Yan, T. Gui, J. Dai, Q. Guo, Z. Zhang, and X. Qiu. A unified generative framework for various NER subtasks. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 5808–5822, Online, Aug. 2021a. Association for Computational Linguistics. doi: 10.18653/v1/2021.acl-long.451. URL https://aclanthology.org/2021.acl-long.451.

H. Yan, T. Gui, J. Dai, Q. Guo, Z. Zhang, and X. Qiu. A unified generative framework for various ner subtasks, 2021b.

H. Yan, Y. Sun, X. Li, Y. Zhou, X. Huang, and X. Qiu. UTC-IE: A unified token-pair classification architecture for information extraction. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 4096–4122, Toronto, Canada, July 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.acl-long.226. URL https://aclanthology.org/2023.acl-long.226.

Z. Yan, C. Zhang, J. Fu, Q. Zhang, and Z. Wei. A partition filter network for joint entity and relation extraction. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 185–197, Online and Punta Cana, Dominican Republic, Nov. 2021c. Association for Computational Linguistics. doi: 10.18653/v1/2021.emnlp-main.17. URL https://aclanthology.org/2021.emnlp-main.17.

Z. Yang, Z. Dai, Y. Yang, J. G. Carbonell, R. Salakhutdinov, and Q. V. Le. Xlnet: Generalized autoregressive pretraining for language understanding. In *Neural Information Processing Systems*, 2019. URL https://api.semanticscholar.org/CorpusID:195069387.

L. Yao, C. Mao, and Y. Luo. Graph convolutional networks for text classification, 2018.

D. Ye, Y. Lin, P. Li, and M. Sun. Packed levitated marker for entity and relation extraction. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 4904–4917, Dublin, Ireland, May 2022a. Association for Computational Linguistics. doi: 10.18653/v1/2022.acl-long.337. URL https://aclanthology.org/2022.acl-long.337.

H. Ye, N. Zhang, H. Chen, and H. Chen. Generative knowledge graph construction: A review. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 1–17, Abu Dhabi, United Arab Emirates, Dec. 2022b. Association for Computational Linguistics. doi: 10.18653/v1/2022.emnlp-main.1. URL https://aclanthology.org/2022.emnlp-main.1.

J. Ye, X. Chen, N. Xu, C. Zu, Z. Shao, S. Liu, Y. Cui, Z. Zhou, C. Gong, Y. Shen, J. Zhou, S. Chen, T. Gui, Q. Zhang, and X. Huang. A comprehensive capability analysis of gpt-3 and gpt-3.5 series models. *ArXiv*, abs/2303.10420, 2023. URL https://api.semanticscholar.org/CorpusID:257632113.

N. Ye, W. Lee, H. Chieu, and D. Wu. Conditional random fields with high-order features for sequence labeling. In Y. Bengio, D. Schuurmans, J. Lafferty, C. Williams, and A. Culotta, editors, *Advances in Neural Information Processing Systems*, volume 22. Curran Associates, Inc., 2009. URL https://proceedings.neurips.cc/paper/2009/file/94f6d7e04a4d452035300f18b984988c-Paper.pdf.

Z. Ye and Z.-H. Ling. Hybrid semi-Markov CRF for neural sequence labeling. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 235–240, Melbourne, Australia, July 2018. Association for Computational Linguistics. doi: 10.18653/v1/P18-2038. URL https://aclanthology.org/P18-2038.

D. Yu, R. Zhang, Z. Jiang, Y. Wu, and Y. Yang. Graph-revised convolutional network, 2020a.

J. Yu, B. Bohnet, and M. Poesio. Named entity recognition as dependency parsing. In *ACL*, 2020b.

J. Yu, B. Bohnet, and M. Poesio. Named entity recognition as dependency parsing. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 6470–6476, Online, July 2020c. Association for Computational Linguistics. doi: 10.18653/v1/2020.acl-main.577. URL https://aclanthology.org/2020.acl-main.577.

N. Zamin and A. Oxley. Building a corpus-derived gazetteer for named entity recognition. In *International Conference on Software Engineering and Computer Systems*, 2011. URL https://api.semanticscholar.org/CorpusID:27060266.

U. Zaratiana, N. Elkhbir, P. Holat, N. Tomeh, and T. Charnois. Global span selection for named entity recognition. In *Proceedings of the Workshop on Unimodal and Multimodal Induction of Linguistic Structures (UM-IoS)*, pages 11–17, Abu Dhabi, United Arab Emirates (Hybrid), Dec. 2022a. Association for Computational Linguistics. doi: 10.18653/v1/2022.umios-1.2. URL `https://aclanthology.org/2022.umios-1.2`.

U. Zaratiana, P. Holat, N. Tomeh, and T. Charnois. Hierarchical transformer model for scientific named entity recognition. *ArXiv*, abs/2203.14710, 2022b.

U. Zaratiana, N. Tomeh, P. Holat, and T. Charnois. GNNer: Reducing overlapping in span-based NER using graph neural networks. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics: Student Research Workshop*, pages 97–103, Dublin, Ireland, May 2022c. Association for Computational Linguistics. doi: 10.18653/v1/2022.acl-srw.9. URL `https://aclanthology.org/2022.acl-srw.9`.

U. Zaratiana, N. Tomeh, P. Holat, and T. Charnois. Named entity recognition as structured span prediction. In *Proceedings of the Workshop on Unimodal and Multimodal Induction of Linguistic Structures (UM-IoS)*, pages 1–10, Abu Dhabi, United Arab Emirates (Hybrid), Dec. 2022d. Association for Computational Linguistics. doi: 10.18653/v1/2022.umios-1.1. URL `https://aclanthology.org/2022.umios-1.1`.

U. Zaratiana, N. Tomeh, N. El Khbir, P. Holat, and T. Charnois. Filtered semi-Markov CRF. In H. Bouamor, J. Pino, and K. Bali, editors, *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 222–235, Singapore, Dec. 2023a. Association for Computational Linguistics. doi: 10.18653/v1/2023.findings-emnlp.17. URL `https://aclanthology.org/2023.findings-emnlp.17`.

U. Zaratiana, N. Tomeh, P. Holat, and T. Charnois. An autoregressive text-to-graph framework for joint entity and relation extraction. In *ICML 2023 Workshop on Structured Probabilistic Inference & Generative Modeling*, 2023b. URL `https://openreview.net/forum?id=5sLIuaW5wT`.

U. Zaratiana, N. Tomeh, P. Holat, and T. Charnois. An autoregressive text-to-graph framework for joint entity and relation extraction. *Proceedings of the AAAI Conference on Artificial Intelligence*, 38(17):19477–19487, Mar. 2024. doi: 10.1609/aaai.v38i17.29919. URL `https://ojs.aaai.org/index.php/AAAI/article/view/29919`.

D. Zelenko, C. Aone, and A. Richardella. Kernel methods for relation extraction. In *Journal of machine learning research*, 2002a.

D. Zelenko, C. Aone, and A. Richardella. Kernel methods for relation extraction. In *Proceedings of the 2002 Conference on Empirical Methods in Natural Language Processing (EMNLP 2002)*, pages 71–78. Association for Computational Linguistics, July 2002b. doi: 10.3115/1118693.1118703. URL `https://aclanthology.org/W02-1010`.

D. Zeng, K. Liu, S. Lai, G. Zhou, and J. Zhao. Relation classification via convolutional deep neural network. In *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers*, pages 2335–2344, Dublin, Ireland, Aug. 2014. Dublin City University and Association for Computational Linguistics. URL `https://aclanthology.org/C14-1220`.

C. Zhang, F. Tao, X. Chen, J. Shen, M. Jiang, B. Sadler, M. Vanni, and J. Han. Taxogen: Unsupervised topic taxonomy construction by adaptive term embedding and clustering. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, KDD '18, page 2701–2709, New York, NY, USA, 2018. Association for Computing Machinery. ISBN 9781450355520. doi: 10.1145/3219819.3220064. URL `https://doi.org/10.1145/3219819.3220064`.

C. Zhang, Q. Li, and D. Song. Aspect-based sentiment classification with aspect-specific graph convolutional networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 4568–4578, Hong Kong, China, Nov. 2019. Association for Computational Linguistics. doi: 10.18653/v1/D19-1464. URL `https://aclanthology.org/D19-1464`.

X. Zhang and M. Zitnik. Gnnguard: Defending graph neural networks against adversarial attacks, 2020.

J. Zhao, X. Wang, C. Shi, B. Hu, G. Song, and Y. Ye. Heterogeneous graph structure learning for graph neural networks. *Proceedings of the AAAI Conference on Artificial Intelligence*, 35(5):4697–4705, May 2021a. doi: 10.1609/aaai.v35i5.16600. URL `https://ojs.aaai.org/index.php/AAAI/article/view/16600`.

S. Zhao. Named entity recognition in biomedical texts using an HMM model. In *Proceedings of the International Joint Workshop on Natural Language Processing in Biomedicine and its Applications (NLPBA/BioNLP)*, pages 87–90, Geneva, Switzerland, Aug. 28th and 29th 2004. COLING. URL `https://aclanthology.org/W04-1216`.

T. Zhao, Z. Yan, Y. Cao, and Z. Li. A unified multi-task learning framework for joint extraction of entities and relations. *Proceedings of the AAAI Conference on Artificial Intelligence*, 35(16):14524–14531, May 2021b. doi: 10.1609/aaai.v35i16.17707. URL `https://ojs.aaai.org/index.php/AAAI/article/view/17707`.

C. Zheng, B. Zong, W. Cheng, D. Song, J. Ni, W. Yu, H. Chen, and W. Wang. Robust graph representation learning via neural sparsification. In H. D. III and A. Singh, editors, *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 11458–11468. PMLR, 13–18 Jul 2020. URL `https://proceedings.mlr.press/v119/zheng20d.html`.

Z. Zhong and D. Chen. A frustratingly easy approach for entity and relation extraction. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 50–61, Online, June 2021a. Association for Computational Linguistics. doi: 10.18653/v1/2021.naacl-main.5. URL `https://aclanthology.org/2021.naacl-main.5`.

Z. Zhong and D. Chen. A frustratingly easy approach for entity and relation extraction, 2021b.

G. Zhou and J. Su. Named entity recognition using an HMM-based chunk tagger. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 473–480, Philadelphia, Pennsylvania, USA, July 2002. Association for Computational Linguistics. doi: 10.3115/1073083.1073163. URL `https://aclanthology.org/P02-1060`.

W. Zhou, S. Zhang, Y. Gu, M. Chen, and H. Poon. UniversalNER: Targeted distillation from large language models for open named entity recognition. In *The Twelfth International Conference on Learning Representations*, 2024. URL `https://openreview.net/forum?id=r65xfUb76p`.

Z. Zhou, S. Zhou, B. Mao, X. Zhou, J. Chen, Q. Tan, D. Zha, Y. Feng, C. Chen, and C. Wang. Opengsl: A comprehensive benchmark for graph structure learning, 2023.

E. Zhu and J. Li. Boundary smoothing for named entity recognition. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 7096–7108, Dublin, Ireland, May 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.acl-long.490. URL `https://aclanthology.org/2022.acl-long.490`.

E. Zhu, Y. Liu, and J. Li. Deep span representations for named entity recognition. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 10565–10582, Toronto, Canada, July 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.findings-acl.672. URL `https://aclanthology.org/2023.findings-acl.672`.

Y. Zhu, W. Xu, J. Zhang, Y. Du, J. Zhang, Q. Liu, C. Yang, and S. Wu. A survey on graph structure learning: Progress and opportunities, 2022.

J. Zhuo, Y. Cao, J. Zhu, B. Zhang, and Z. Nie. Segment-level sequence modeling using gated recursive semi-Markov conditional random fields. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1413–1423, Berlin, Germany, Aug. 2016. Association for Computational Linguistics. doi: 10.18653/v1/P16-1134. URL `https://aclanthology.org/P16-1134`.