# Span-Oriented Information Extraction: A Unified Framework

Yifan Ding
Department of Computer Science and Engineering
University of Notre Dame
Notre Dame, IN, USA
yding4@nd.edu

Michael Yankoski
School of Data Science
College of William and Mary
Williamsburg, VA, USA
myankoski@wm.edu

Tim Weninger
Department of Computer Science and Engineering
University of Notre Dame
Notre Dame, IN, USA
tweninger@nd.edu

## ABSTRACT

Information Extraction (IE) encompasses a diverse array of tasks in Natural Language Processing (NLP), including Named Entity Recognition (NER), Entity Linking (EL), and Attribute Value Extraction (AVE), all of which aim to derive structured representations from unstructured text. Despite their shared goals, these tasks are often studied in isolation, leading to redundant methods and fragmented advances. This work introduces a unified framework for IE centered on the concept of a **Span**: a contiguous sequence of tokens annotated with one or more semantic labels. By adopting spans as the foundational unit of analysis, we unify a broad class of IE tasks under a common formulation. We formalize key dimensions of this span-oriented paradigm: task formulation via span identification and classification, evaluation through boundary- and label-sensitive metrics, representation strategies including token-to-span encodings, architectural components tailored to span extraction, and modeling strategies grounded in pre-trained language models. We also identify persistent challenges, such as boundary ambiguity and context-label mismatch, that cross tasks. Through this unified lens, we synthesize and standardize a field long divided by task-specific assumptions. The result is a cohesive framework that supports cross-task generalization, standardized evaluation, and principled model design, which lays the foundation for future advances in both span-based and general-purpose information extraction systems.

## 1. INTRODUCTION

Information Extraction (IE) is a foundational task in Natural Language Processing (NLP), concerned with transforming unstructured or semi-structured text into structured representations. IE systems power downstream applications such as question answering, knowledge graph construction, and document understanding, all of which require the identification of meaningful textual elements (*e.g.* entities, attributes, or quantities) and their association with appropriate labels or external references.

Despite these shared goals, IE tasks have historically been developed and studied in isolation. For instance, Named Entity Recognition (NER) is often framed as a sequential token labeling problem; Entity Linking (EL) connects textual mentions to knowledge base entries; and Attribute Value Extraction (AVE) identifies slot-filler pairs from product descriptions or tabular text. While these tasks exhibit significant overlap in formulation and modeling, the absence of a unifying perspective has led to fragmented approaches and duplicated methodological innovation.

At the heart of many IE tasks lies the concept of a *span*: a contiguous sequence of tokens that encodes a semantically coherent unit such as a named entity, numerical value, or product attribute. Advances in pre-trained language models, including BERT [19], GPT [11], T5 [79], and DeepSeek [32, 53], have prompted a rethinking of how such spans are extracted. Recent work in Unified Information Extraction (UIE) [56, 57] demonstrates that diverse IE tasks, such as NER, Relation Extraction, and Event Detection, can be reframed using shared paradigms like sequence-to-sequence generation, span classification, or span-based question answering. These developments challenge the traditional division of labor in IE and motivate a unified modeling perspective grounded in span prediction.

While recent research gestures toward unification, several gaps remain. Existing frameworks often emphasize high-level integration but lack a systematic breakdown of what can be unified and where task-specific distinctions persist. The precise modeling components that generalize across IE tasks, and those that require bespoke treatment, remain insufficiently analyzed. Moreover, existing surveys tend to focus narrowly on individual tasks, such as NER [69], EL [31], or the broader capabilities of large language models [118]. Few works synthesize these perspectives into a comprehensive framework for unifying IE.

To address this gap, we introduce a span-centric taxonomy for IE that organizes and analyzes core components shared across tasks. Our focus is on non-overlapping, single-span cases, which form the majority of practical IE use cases. More complex variants, such as nested spans, discontinuous spans, or multi-span extractions, can be viewed as natural extensions of this core formulation, but are outside the scope of this survey and left for future work.

This unified view enables a systematic comparison of task formulations, label representations, model architectures, and evaluation metrics across a diverse range of IE problems. In doing so, we aim to clarify the field's current state, reveal latent structure among IE tasks, and suggest pathways for consolidated innovation.

### 1.1 A Unified Framework

Unlike core NLP tasks such as tokenization, parsing, or

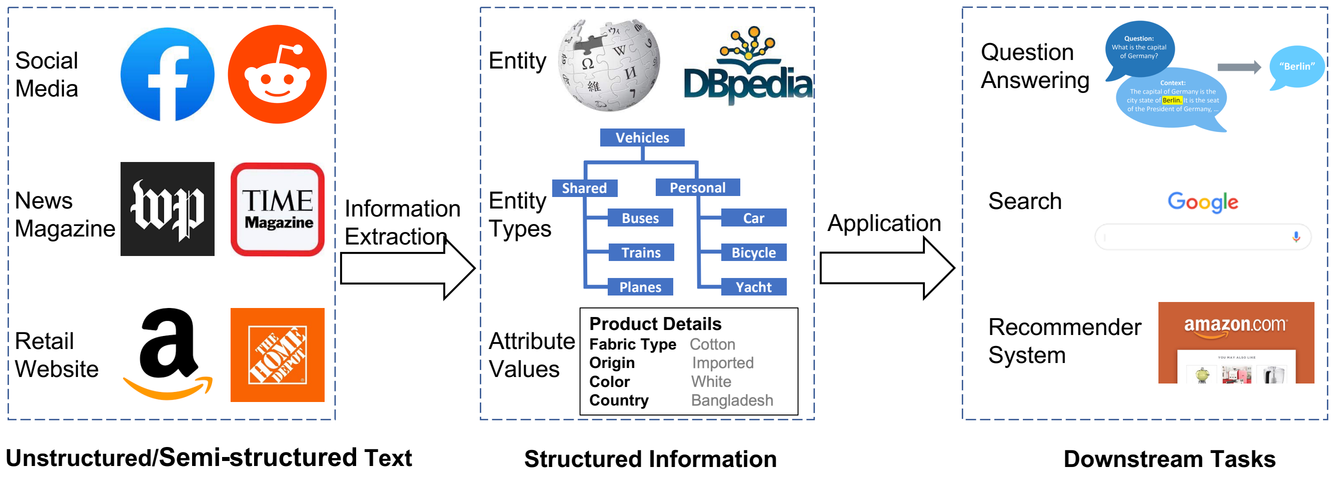

Figure 1: An overview of information extraction tasks. The goal of information extraction is to identify sub-sequences within unstructured or semi-structured text information and link them to certain class-labels, entities in a knowledge base, or other items within some structured database. This structured information plays a central role in many downstream applications such as question answering and recommender systems.

chunking, each of which has established formalism and tooling, the span representation does not have a clear formalism. The lack of a unified framework hampers model reuse and interoperability across tasks. For example. most IE pipelines still treat span detection and label assignment as one-directional operations. This limits their capacity to perform bidirectional reasoning about text and labels, which is a capability increasingly desirable in models that incorporate retrieval, prompting, or knowledge-grounded reasoning.

The goal of Span-Oriented Information Extraction (SIE) is to reinterpret the fragmented landscape of IE through a unifying lens: that of span prediction. We structure this paper around three perspectives that correspond to the primary dimensions of SIE: (1) information extraction *tasks* as span prediction problems, (2) span-based *evaluation*, (3) *model architectures* designed to extract and label spans, and finally, because this is a unified model, (4) *transformations* between model architectures.

### 1.1.1 Tasks as Span Prediction

At the heart of SIE is the proposition that a wide range of IE tasks can be unified by treating them as span prediction problems. Tasks such as Named Entity Recognition (NER), Entity Linking (EL), and Attribute Value Extraction (AVE) may differ in surface formulation, but they all involve identifying spans and associating them with predefined labels.

Recasting IE tasks in terms of span prediction reveals their structural commonalities. For instance, both NER and AVE can be modeled as identifying text spans and assigning class labels (e.g., `PERSON`, `PRICE`); EL additionally maps spans to knowledge base entries. Even question answering tasks like Machine Reading Comprehension (MRC) can be reframed as extracting an answer span from a given context. The variation across tasks is often limited to differences in span classes, input conditioning, or whether the span is known or latent. By shifting the focus from bespoke task formulations to a unified span framework, SIE enables methodological reuse and cross-task generalization.

### 1.1.2 Evaluation through Span-Based Alignment

A unified perspective on IE tasks calls for a corresponding unified evaluation methodology. Under SIE, evaluation is framed in terms of span identification and classification accuracy. This enables consistent benchmarking across tasks, facilitating direct comparison of systems regardless of whether the task is framed as entity recognition, slot filling, or span-based question answering.

Span-level evaluation emphasizes the quality of predicted spans in terms of boundary correctness and label accuracy. Common metrics such as span-level F1 score and overlap-based precision-recall can be applied across all tasks reimagined under SIE. This approach not only simplifies comparative analysis but also draws attention to shared challenges such as handling ambiguous boundaries, overlapping spans, or label sparsity. Section 3 of this paper elaborates on a standardized evaluation framework grounded in span-level metrics.

### 1.1.3 Taxonomy of Information Extraction Features

Span-oriented information extraction models rely on a range of features that encode information at different levels of granularity. These features provide the representational foundation for span identification and classification.

We organize these features into three categories:

- **Token-Level Features**: These include contextualized token embeddings from pre-trained language models (*e.g.* BERT, RoBERTa), part-of-speech tags, or syntactic cues. Such features capture local semantics and serve as the base layer for span construction.
- **Span-Level Features**: These involve representations derived from contiguous token sequences, such as start and end indicators, span length, pooled embeddings (*e.g.* mean or max over token embeddings), and span position encodings. Span-level features are essential for tasks like classification or scoring.
- **Class-Level Features**: These refer to representa-

tions of span labels themselves, such as class embeddings, textual descriptions, or prototype vectors, especially useful in open-world or zero-shot settings where labels are semantically meaningful.

Our goal is to integrate these feature types into a unified span representation, which then feeds into downstream prediction modules. The selection and combination of these features significantly affect a model's ability to generalize across span types, domains, and tasks.

#### *1.1.4 Model Transformations among Information Extractors*

Having established the core features used in span-oriented IE, we now consider the transformations these features undergo within different model architectures. Despite the surface diversity in IE systems, they often follow a shared transformation pipeline from raw input to labeled spans.

We identify three key components of this pipeline:

1. **Training Objectives**: Objectives range from span classification (predicting a label for each span) to span boundary detection (predicting start/end indices) to joint models. Classical sequence labeling (*e.g.* BIO tagging) and modern span selection techniques (*e.g.* span scoring or span ranking) fall into this space.
2. **Inference Strategies**: Inference may involve enumerating all candidate spans up to a maximum width, scoring each span-label pair, and selecting those above a confidence threshold. More advanced systems use constrained decoding or joint inference over multiple spans.
3. **Span/Label Duality**: A core challenge across tasks is the mismatch between token-level representations and span-level outputs. Additionally, many tasks, especially in zero-shot or open-world settings, require a more flexible alignment between surface spans and semantic labels. This duality is exemplified by cases like Wikification, where a string (*e.g.* `Tim Cook`) must be matched with a canonical entity (`wiki/Tim_Cook`) that encodes rich contextual meaning.

This span/label duality suggests that spans should be treated not merely as outputs, but as bidirectional constructs: they are both the textual surface forms and the semantic identifiers. Embracing this dual nature can lead to more flexible and generalizable IE models, especially when paired with techniques such as prompt tuning, generative decoding, and retrieval-based linking.

## 1.2 Research Implications and Applications

By unifying a diverse array of Information Extraction (IE) tasks under the span-oriented framework, this work resolves long-standing fragmentation in the field. Prior surveys often focus narrowly—on Named Entity Recognition, Relation Extraction, or generative models—without connecting task-specific innovations to broader trends. In contrast, the Span-Oriented Information Extraction (SIE) perspective offers a cohesive foundation for evaluating, comparing, and extending IE systems. It supports standardized benchmarking through consistent span-level metrics, enabling more rigorous evaluation across tasks such as NER, EL, AVE, and MRC. The framework also facilitates cross-task transfer: techniques for handling span boundaries, ambiguity, or open-set classification can be reused across domains, accelerating innovation. As large language models (LLMs) continue to blur the lines between extraction and generation, SIE complements these systems by offering structure, interpretability, and factual grounding—especially in hybrid architectures that combine symbolic extraction with generative reasoning. Finally, SIE benefits multiple audiences: it clarifies research boundaries for NLP practitioners, supports general-purpose model development for the broader AI community, and simplifies instruction for educators by offering a single, principled formulation of IE. In doing so, it lays the groundwork for a more unified, scalable, and interpretable future for information extraction.

## 1.3 Formal Definition of a Span

Let a document $d$ be represented as a sequence of tokens:

$$d = \langle t_1, t_2, \dots, t_k \rangle,$$

where each $t_i$ is a token and $k = \ell(d)$ denotes the total number of tokens in the document. A **span** is formally defined as a tuple $s = (b, e, c)$, where $b$ and $e$ are token indices such that $1 \leq b \leq e \leq \ell(d)$, and $c$ is a class label or set of labels assigned to the span. The indices $(b, e)$ define a contiguous subsequence of tokens in $d$, corresponding to a surface form in the original text.

A span thus encapsulates a dual structure:

- A *surface form*: the contiguous subsequence $\langle t_b, \dots, t_e \rangle$, which serves as the textual expression of the information unit.
- A *semantic label* or set of labels $c \in C$, denoting the type or role assigned to the surface form (*e.g.* `PERSON`, `LOCATION`, `PRICE`).

The task of Span-Oriented Information Extraction (SIE) can then be described as identifying the correct set of spans $\{s_1, s_2, \dots, s_n\}$ from $d$, including both accurate boundary identification $(b, e)$ and appropriate label assignment $c$.

We distinguish between two settings based on the nature of the label space $C$:

1. **Closed-world IE**: The label set $C$ is predefined and finite. Each span must be assigned one or more labels from this fixed set. This setting is common in tasks such as NER and AVE.
2. **Open-world IE**: The label set $C$ is not fixed in advance. Labels may be dynamically assigned from an open vocabulary, external knowledge base, or generative component. This formulation appears in tasks such as zero-shot entity linking or question answering.

This formalization is consistent with implementations in popular NLP libraries such as SpaCy [35] and Stanford CoreNLP [61], both of which treat spans as first-class objects that associate contiguous text with semantic roles.

By establishing spans as the central abstraction, we unify a wide range of IE tasks, regardless of their original formulation, under a common representation that supports comparison, composition, and generalization.

## 1.4 Road Map

The remainder of this paper develops a unified framework for span-oriented information extraction by examining the

Table 1: Overview of Span-Oriented Information Extraction Tasks

**Apple** CEO **Tim Cook** sold his **Texas** house.

| Task Name | Span Req. | Span Class | Span Class Example |
|---|---|---|---|
| Entity Disambiguation (ED) | ✓ | Entity | wiki/Tim_Cook, wiki/Texas |
| Entity Linking (EL) | | Entity | `wiki/Tim_Cook`, `wiki/Texas` |
| Entity Typing (ET) | ✓ | Fine-grained Type | Businessman, State |
| Named Entity Recognition (NER) | | Coarse-grained Type | `PER`, `LOC` |
| Attribute Value Extraction (AVE) | | Attributes | CEO: `Tim Cook` |
| Machine Reading Comprehension (MRC) | | Reading Question | `Tim Cook` |

field along three core dimensions: task formulation, evaluation methodology, and model design. We begin by surveying major IE tasks such as Named Entity Recognition, Entity Linking, and Machine Reading Comprehension and demonstrate how they can be recast as instances of span identification and labeling. We then introduce a span-based evaluation framework that addresses boundary fuzziness and label ambiguity, supporting more consistent and informative comparisons across tasks. Finally, we present a two-part model analysis: first, a taxonomy of the token-, span-, and class-level features that underpin most systems; and second, a view of IE models as transformation pipelines that convert these features into labeled spans. Together, these perspectives reveal a high degree of structural similarity across IE systems, supporting the case for a unified span-oriented paradigm.

## 2. TYPES OF SPAN-ORIENTED INFORMATION EXTRACTION TASKS

There are a variety of information extraction tasks that are defined based upon the needs of the system and the data available. These tasks have been given various names and descriptions in the literature, but they all share the same basic definition of a span.

Given a sentence in some document, for example the sentence, "Apple CEO Tim Cook sold his Texas house", different varieties of information extraction would seek to label different sub-sequences having one or more classes, *e.g.* Apple as a company, Texas a US state. Among these varieties are: (1) Entity disambiguation (ED) [34], (2) Entity Linking (EL) [39], (3) Entity Typing (ET) [36], (4) Named Entity Recognition (NER) [96], (5) Attribute Value Extraction (AVE) [122], (6) Machine Reading Comprehension (MRC) [83].

Table 1 provides a non-exhaustive list of different information extraction tasks and a specific example of that task.

### 2.1 Entity Disambiguation

In cases where the surface form is given, either through text matching or some other entity identification task, what remains is to match the surface form with the appropriate class label. In the following examples we identify the surface form with a beginning and end token-index of the input where the first token is indexed at 0.

Entity disambiguation is so named because the task is mostly to determine which specific entity, if there are many similarly-named entities, that surface form represents. To do this, ED systems rely on the context to make their decisions. The difficulty of ED is that entities typically have a giant number of classes. Subsequently, rare and infrequent entities are difficult to disambiguate [5, 86, 88, 105].

| | |
|---|---|
| TASK: | Entity Disambiguation |
| INPUT: | Apple CEO Tim Cook sold his Texas house |
| INPUT: | $[(2, 3, ?), (6, 6, ?)]$ |
| OUTPUT: | [(0, 0, `wiki/Apple_Inc.`), (2, 3, `wiki/Tim_Cook`), (6, 6, `wiki/Texas`)] |

### 2.2 Entity Linking

In cases where the surface form is not pre-defined, the position of the spans and the class of the span (*i.e.* the entity) must both be extracted. Compared to ED, EL is much more difficult. EL requires jointly identifying non-standard surface forms from the input text and assigning the correct labels. As a result ED systems are typically more precise, while EL systems observe redundant performance drops in the same datasets [12, 39, 86, 97].

| | |
|---|---|
| TASK: | Entity Linking |
| INPUT: | Apple CEO Tim Cook sold his Texas house |
| OUTPUT: | [(0, 0, `wiki/Apple_Inc.`), (1, 1, `wiki/Chief_executive_officer`), (2, 3, `wiki/Tim_Cook`), (6, 6, `wiki/Texas`)] |

The EL task essentially performs the surface form identification sub-task and the ED task simultaneously. This provides greater freedom to the system so that additional context of any found-span might be used to find more spans. As a result, EL systems generally have greater coverage, but at the expense of precision.

### 2.3 Entity Typing

There are also cases where the end-user does not seek a specific entity-entry in some knowledge base, but rather seeks to know the specific (*i.e.* fine-grained) types of entities that are resident in some span [17, 71].

| | |
|---|---|
| TASK: | Entity Typing |
| INPUT: | Apple CEO Tim Cook sold his Texas house |
| INPUT: | $[(0, 0, ?), (2, 3, ?), (6, 6, ?)]$ |
| OUTPUT: | [(0, 0, Company), (2, 3, Businessman), (6, 6, State)] |

The ET task is a slightly relaxed form of the ED task, where the number of classes is not as large, but can still be extensive depending on the type-granularity that the user seeks. Like in the ED task, because the beginning and ending indexes of the spans are provided as input, this task typically has high precision at the expense of coverage. However, evaluation of the ET task can be difficult because properly aligning the right span-label that matches the ground-truth granularity can be difficult. Indeed, previous work has found that reasonable (and sometimes arguably more-correct span-labels) are often counted as incorrect in ET evaluation [17].

## 2.4 Named Entity Recognition

One of the first information extraction tasks from the MUC workshops described above was the NER task. This task seeks to identify entities from a sentence and, almost as a bonus, also labels the entities into one of three or four broad types; typically `PER`, `ORG`, and `LOC` representing person, organization, and location entities respectively.

| | |
|---|---|
| TASK: | Named Entity Recognition |
| INPUT: | Apple CEO Tim Cook sold his Texas house |
| OUTPUT: | [(0, 0, `ORG`), (2, 3, `PER`), (6, 6, `LOC`)] |

The primary difficulty of the NER task is the identification of the span's starting and ending indexes; once the surface form can be identified, the class label is rather straightforward because the class label set is typically very small and contains only coarse-grained entity types. However, because NER systems are typically trained on a small set of coarse-grained entity types, many spans that would be easily discovered with EL systems (*e.g.* animals, technology, works of art) are not easily binned into one of the coarse types.

## 2.5 Attribute Value Extraction

In instances where many sentences describe a general set of items, like, for example, descriptions of episodes on a streaming service, item descriptions in an online game, or groceries available on an e-commerce website, the entities themselves are not of interest. Instead, the AVE task seeks to extract the collective values corresponding to attributes of interest from the descriptive document(s). Nevertheless, AVE can still be views as a span-oriented information extraction task. Continuing the running example, if we imagine a collection of sentences discussing technology news, then the AVE task might extract the corresponding technology leaders as follows:

| | |
|---|---|
| TASK: | Attribute Value Extraction |
| INPUT: | Satya Nadella says that Microsoft products will soon connect to OpenAI. Apple CEO Tim Cook sold his Texas house. Jensen Huang, head of NVIDIA, announces the launch of DGX GH200. |
| OUTPUT: | [(0, 1, CEO), (13, 14, CEO), (19, 20, CEO)] |

## 2.6 Machine Reading Comprehension

In some cases users seek to extract spans related to some free-text question. Although the MRC task is unlike many span-oriented information extraction tasks, it still requests the same fundamental output: a span of tokens and a class.

| | |
|---|---|
| TASK: | Machine Reading Comprehension |
| INPUT: | Apple CEO Tim Cook sold his Texas house. |
| INPUT: | Q: Who is the CEO of Apple? |
| OUTPUT: | [(2, 3, `Who is the CEO of Apple?`)] |

Like in the AVE task, the MRC task contains a beginning- and ending-index and as well as a class. In this particular case, the class is a direct restating of the input question. This is an important consideration. This may be best explained with an analogy to the NER task. If we re-consider the NER task to be an MRC task, then the question asked of the NER system is the entity-label:

| | |
|---|---|
| TASK: | Named Entity Recognition |
| INPUT: | Apple CEO Tim Cook sold his Texas house |
| INPUT: | Q: PER |
| OUTPUT: | [(2, 3, `PER`)] |

Here we see that the answer to the MRC task is not the class label, but is instead denoted by the span indices which reveal the answer to be `Tim Cook`, who is in Person.

The above list of span-oriented information extraction tasks is by no means exhaustive, but these examples are meant to be representative of our philosophy: by re-imagining information extraction tasks as systems that output spans, then these systems can be considered natural analogs of one another. With this in mind, the means by which these various information extraction systems are evaluated can be viewed from a more coherent perspective too.

## 3. EVALUATION OF SPAN-ORIENTED INFORMATION EXTRACTION

The most common way to evaluate information extraction systems is to use the standard precision, recall and $F_1$ metrics [89]. However, any metric that evaluates spans deserve a more-thorough consideration. This is because the groundtruth span may not exactly line up with the predicted span, yet still be close-enough to warrant a true-positive judgment. Likewise, the class-label(s) within the predicted spans might not exactly match the groundtruth span, yet still be close-enough to warrant leniency. Because of these consideration several metrics and metric-variants have been developed to handle these difficult cases.

The $F_\beta$ metric was originally intended as a way to balance the precision and recall (*i.e.* coverage) of machine learning system. When the $\beta = 1$ then the precision and recall values are evenly weighted in the $F$-score. Lower values for $\beta$ give more weight to the precision metric and *vice versa.* Unless otherwise specified, systems typically set $\beta = 1$ yielding the well-known $F_1$-score.

The $F_1$-score uses precision and recall metrics, which themselves require some notion of binary true and false predictions. As applied to span-oriented information extraction, this creates a rigid requirement that any token can belong to at most one span and that a span must exactly match the groundtruth to be considered a true-positive instance.

In the simple case where there are only labels (*e.g.* yes/no, on/off), the $F_1$-score provides a meaningful, although rigid evaluation metric. However, as noted above, most of the class-sets in information extraction are enormous, having thousands (or hundreds of thousands) of classes. In such cases, a decision needs to be made on how to calculate certain mean-averages. The two most common decisions are called (1) the micro $F_1$-score and (2) the macro $F_1$-score.

### 3.1 Micro $F_1$-Score

The micro $F_1$ score is widely recognized as the standard evaluation metric for rigid NER. The notion of rigidity in this instance denotes that any token of the input document can belong to at most one span and that a true-positive instance must match the groundtruth span exactly.

Formally, for each class $c^* \in C$ and for a groundtruth set $s = (b, e, c) \in S$ and for a set of predicted span instances $\hat{s} = (\hat{b}, \hat{e}, \hat{c}) \in \hat{S}$, the number of true positives ($\mathrm{TP}_{c^*}$) is:

$$\mathrm{TP}_{c^*} = \sum_{(b,e,c)\in S} \sum_{(\hat{b},\hat{e},\hat{c})\in\hat{S}} \mathbb{1}\left(b = \hat{b} \wedge e = \hat{e} \wedge c = \hat{c} = c^*\right) \quad (1)$$

Likewise we count the number of false negatives ($\mathrm{FN}_{c^*}$) and false positives $\mathrm{FP}_{c^*}$ as:

$$\mathrm{FN}_{c^*} = \sum_{(b,e,c)\in S} \left(1 - \sum_{(\hat{b},\hat{e},\hat{c})\in\hat{S}} \mathbb{1}\left(b = \hat{b} \wedge e = \hat{e} \wedge c = \hat{c} = c^*\right)\right) \quad (2)$$

$$\mathrm{FP}_{c^*} = \sum_{(\hat{b},\hat{e},\hat{c})\in\hat{S}} \left(1 - \sum_{(b,e,c)\in S} \mathbb{1}\left(b = \hat{b} \wedge e = \hat{e} \wedge c = \hat{c} = c^*\right)\right) \quad (3)$$

Note that some tasks, like MRC and AVE sometimes relax the indicator function $\mathbb{1}(\cdot)$ so that the beginning- and ending-indexes need not exactly match, but that the tokens denoted by these indexes match: $\mathbb{1}(\mathrm{substr}(b, e) = \mathrm{substr}(\hat{b}, \hat{e}) \wedge c = \hat{c} = c^*)$. For example, in MRC we do not need to know that `Tim Cook` begins and ends with tokens 2 and 3, only that the tokens between indexes 2 and 3 match the groundtruth answer.

Then, to obtain the micro $F_1$-scores the TPs, FNs, and FPs are summed across the various classes and substituted into the standard precision and recall metrics to obtain micro-precision and micro-recall.

$$\begin{aligned} \text{micro-Prec} &= \frac{\sum_{c^*\in C} \mathrm{TP}_{c^*}}{\sum_{c^*\in C}(\mathrm{TP}_{c^*} + \mathrm{FP}_{c^*})}; \\ \text{micro-Rec} &= \frac{\sum_{c^*\in C} \mathrm{TP}_{c^*}}{\sum_{c^*\in C}(\mathrm{TP}_{c^*} + \mathrm{FN}_{c^*})}; \\ \text{micro-}F_1 &= \frac{2 \times \text{micro-Prec} \times \text{micro-Rec}}{\text{micro-Prec} + \text{micro-Rec}} \end{aligned} \quad (4)$$

## 3.2 Macro $F_1$-Score

The micro $F_1$-score is a natural way to sum up the successes and errors of the model predictions. However, this simple solution can be easily swayed in the likely case that the class labels are unbalanced, *i.e.* certain labels occur much more frequently than others. In this situation, a class-based precision and recall measurement can be calculated as follows:

$$\begin{aligned} \mathrm{Prec}_{c^*} &= \frac{\mathrm{TP}_{c^*}}{\mathrm{TP}_{c^*} + \mathrm{FP}_{c^*}}; \\ \mathrm{Rec}_{c^*} &= \frac{\mathrm{TP}_{c^*}}{\mathrm{TP}_{c^*} + \mathrm{FN}_{c^*}}; \\ F_{1,c^*} &= \frac{2 \times \mathrm{Prec}_{c^*} \times \mathrm{Rec}_{c^*}}{\mathrm{Prec}_{c^*} + \mathrm{Rec}_{c^*}} \end{aligned} \quad (5)$$

Then, the overall macro $F_1$-score is the arithmetic mean of $F_1$ scores across all the individual classes as follows:

$$\text{macro-}F_1 = \frac{\sum_{c^*\in C} F_{1,c^*}}{|C|} \quad (6)$$

## 3.3 Exact Match Evaluation

The macro and micro metrics described above require exact matches of the beginning-index, ending-index, and the class $\mathbb{1}(b = \hat{b} \wedge e = \hat{e} \wedge c = \hat{c} = c^*)$ in order to count towards a true positive instance.

Also note that the string matching function $\mathbb{1}(\mathrm{substr}(b, e) = \mathrm{substr}(\hat{b}, \hat{e}) \wedge c = \hat{c} = c^*)$ commonly used in the AVE and MRC tasks do not require exact matches of the indices, but do require exact matches of the sub-sequences represented by the indexes.

| | |
|---|---|
| TASK: | NER/MRC |
| INPUT: | Apple CEO Tim Cook sold his Texas house Tim Cook announces new M2 chip. |
| GT: | [(2, 3, `PER`)] |
| OUTPUT: | [(8, 9, `PER`)] |

In the example directly above the extracted span (8, 9, `PER`) does match ground truth span in the MRC and AVE task, but would not match the ground truth (GT) span in NER, EL, ET, and ED tasks.

## 3.4 Relaxed Match Evaluation

Exact matching requirements are often criticized for imposing too strict of a requirement onto the system. It is often the case that a sub-sequence or super-sequence of the ground truth span is an equally valid match. Likewise, in fine-grained ET or ED tasks, a close, but still inexact match between the predicted class $\hat{c}$ and the ground truth class $c^*$ could also be equally valid (and our experience shows that sometimes the predicted match is arguably better than the ground truth match) [22]. To allay this criticism, the use of relaxed (*i.e.* partial) span matching is also used in evaluation [25, 29, 39, 88]. Under a relaxed span matching regime, if a predicted span has the same class as one of the ground truth spans and the span indices intersect, then that prediction is counted as a true positive.

| | |
|---|---|
| TASK: | NER |
| INPUT: | Apple CEO Tim Cook sold his Texas house Tim Cook announces new M2 chip. |
| GT: | [(2, 3, `PER`)] |
| OUTPUT: | [(3, 6, `PER`)] |

In the example directly above, the extracted span representing the subsequence `Cook sold his Texas` would count as a true positive for the ground truth span representing the sequence `Tim Cook`. This additional tolerance almost always results in a positive performance shift; however, partial matching may sometimes be too lenient—as shown in the example above. Therefore, strict span matching remains the standard regime used to evaluate span-oriented information extraction tasks.

Table 2: Matrix of Information Extraction Tasks by their Transformation Type

| Transformation | NER | ED | EL | ET | AVE | MRC |
|---|---|---|---|---|---|---|
| Sequential Labeling | [19, 37, 59, 108] | | [5, 15, 39, 97] | | [106, 112, 122] | [20, 38, 92] |
| Token Prototype | [36] | [10, 110] | [10, 28, 107, 108, 110] | [60] | [112, 114] | |
| Token-pair Classification | [48] | | | | | |
| Span Classification | [3, 52, 91, 95, 115, 123] | [1, 15, 28, 80, 81, 105, 107] | [16, 39, 97] | [17, 18, 68] | [23] | [6, 66, 113] |
| Span Locating | [51, 90] | [7, 30] | [117] | | [23, 100] | [85, 120] |
| Span Generation | [26, 58, 104, 111] | [12] | [12] | [21] | | [46, 76, 101] |

## 4. TAXONOMY OF INFORMATION EXTRACTION FEATURES

In this section, we begin to consider how information extraction systems use natural language to create a model from which information can be extracted. To that end, we will summarize the features commonly gleaned from (1) tokens, (2) spans, and (3) span classes.

Because natural language is digitally represented as a sequence of bytes in its most basic form, we consider that to be the lowest-level representation of written (digital) language. From that form natural language tokenizers turn bytes into words (or sub-words) from which sentences, Tweets, paragraphs, posts, articles, stories, and narratives are formed. Information extraction tasks typically operate at the token-level; by our definition, these systems output a span, which is a sequence of one or more words and a class. Each of these levels: the token, the span, and the class all have information that can be used in the construction of a natural language model. This section will briefly highlight each.

## 4.1 Token Features

Most natural language is grouped into tokens—typically words. These tokens are fundamental element in communication; dictionaries, for example, are one source of understanding for these tokens, as are encyclopedias and thesauri. The same is true in natural language processing. Because tokens are the basic elements, crafting the corresponding token features to be flexible and generalized is an important consideration for most tasks.

### *4.1.1 Linguistic Token Features*

Many token features are linguistic in their nature. For example, part-of-speech tags are one of the earliest token features used to distinguish word classes (*e.g.* noun, verb, adjective, and adverb) [40]. The abstract syntax tree is another linguistic feature that transforms plain text into a self-referential tree structure [70]. These approaches produce symbolic properties and are easily interpreted by human beings. However, linguistic features have three significant limitations. First, they do not directly provide information of interest in to most practical applications. For example, although knowing whether a token is a noun or a verb or modifies some other token can be useful in downstream tasks, this is not directly useful in many applications. Second, training linguistic models requires an enormous amount of expert human annotations [14, 62]. In the decades since linguistic token features were first proposed, many datasets have been created, but these features constantly require updating. Third, even perfect labels result in performance limitations on many IE tasks [14, 107, 108]. This is because shallow, token-based information represents a limited view of the deeper intent and meaning within natural language [47, 89].

### *4.1.2 Pre-trained Token Features from Language Models*

With the development of language models (LMs), pre-trained word embeddings have become a primary source of token features. The goal of any LM (large or small) is to model the probability distribution over sequences of tokens. That is, given a document $d$ composed of a sequence of tokens $d = \langle t_1, t_2, \ldots, t_c, \ldots, t_{\ell(d)} \rangle$, an LM provides for the estimation of the probability distribution of any token $t_c$ by utilizing other contextual words in the sequence as follows:

$$p\left(t_c | t_1, t_2, \ldots, t_{c-2}, t_{c-1}, t_{c+1}, t_{c+2}, \ldots, t_{\ell(d)}\right) \tag{7}$$

Constructing LMs has been one of the most fundamental and important tasks for the NLP community. So called *large* LMs (LLMs) are able to scale because they are trained in a self-supervised regime without any human annotation at all. As a result, LLMs have shown the ability to learn token features from a wide variety of documents from different domains. Early pre-trained word embeddings were based on the bag-of-word or skip-gram models; which is best represented by word2vec [65], GloVe [72] and fasttext [9]. With the development of text Transformers, self-supervised token features has been widely adopted and even supplanted most alternatives in natural language processing. The two most representative projects in this category are (1) the GPT series having a left-to-right language model [11, 77, 78] and (2) the BERT series with masked language modeling (MLM) [19]. The broad pre-training with self-supervised labels that these LLMs undergo is typically sufficient for most tokens to obtain robust features [19, 65]. Therefore these pre-trained LLMs can also be adapted to tackle information extraction tasks. This is also important because information extraction tasks typically have limited and biased training data [25, 96], and it is difficult and even inapplicable to employ self-supervised training to the information extraction tasks [11, 19, 58].

#### 4.1.3 *Character Features*

Our definition of a span uses tokens as the base type. However, in some cases the tokens are unable to provide a useful properties of characters. In these cases, it may be beneficial to consider individual characters as extra supplemental features. These character features can then be used as extra learning parameters to improve performance in some specific instances including Chinese language modeling [64,94], and in fields that commonly use acronyms and initialisms like chemistry [24,102], biology [45], and law [13].

#### 4.1.4 *Token Sequences*

Because natural language is (digitally) expressed in a sequence of bytes or tokens, there has been a large effort to model these sequences [47, 86]. Arguably the most well known model for token sequences is the transformers [19,98], although other architectures like the recurrent neural network (RNN) [33, 50], convolutional neural network (CNN) [44,93], and the point network [99,116] have been used as well. Alternatives to these neural models tend to use probability graphic models to model token dependencies with implementations such as the hidden Markov model (HMM) [67], the maximum entropy Markov model (MeMMs) [63] and the conditional random field (CRF) [41].

### 4.2 Span Features

#### 4.2.1 *Span Embeddings*

A span defined as sequence of one or more tokens may therefore have a variety of representations. Typically, span embeddings are built on top of token features. For example, Chen et al. directly used the word embedding of the first token to represent the span features [15], and Tan et al. used the concatenation of word embeddings of the first and last tokens [95]. Otherwise, single-pass frameworks use an averaging over a pool of token embeddings to form span embeddings [4,5]. The PURE model further learns a length embedding as part of span embeddings [123]. The W2NER model uses a convolutional layer and an LSTM jointly to fuse token embeddings into span embeddings [48].

In addition to formulating span embeddings based on token embeddings, another approach is to create span embeddings from scratch as a different pre-training task. Early work on this trajectory extends the word2vec idea to learn span embeddings from contextual span-token and span-span contextual correlations [107,109]. Later work in this area extended the mask language model of BERT to generate span embeddings as a co-training task within language modeling [10,16,108,110].

#### 4.2.2 *Span Sequences*

Just as token features can be modelled as token sequences, span embeddings can likewise be modelling as a sequence of spans. Understanding span sequences is critical in many information extraction tasks. For example, in entity disambiguation one of the primary features used to select span candidates is the context-tokens and other neighbor spans that surround the span in the same sentence. For example, the tokens `Apple` and `CEO` in the running example in Section 2 could help to disambiguate `Tim Cook` as `wiki/Tim\_Cook` from some another entry with a similar name. Likewise, the identification of `wiki/Tim_Cook` may aid in the recognition that `Apple` refers to `wiki/Apple_Inc.` and not some other entity. Formally, most previous frameworks consider the training objective of a common classification model [28,39,73,74] adapted in Eq.(8):

$$g(\mathbf{s},\ \mathbf{c}) = \sum_{i=1}^{n} \Phi(s_i, c_i) + \sum_{i<j} \Psi(c_i, c_j) \tag{8}$$

where the contextual span scores $\Phi(s_i, c_i)$ for each span $s_i$ and class $s_i$ and the span-span correlation scores $\Psi(c_i, c_j)$ for the predicted classes $c_i$ and $c_j$ are both used to train the model. Modelling span sequences tends to work well when the number of classes is small. In this scenario, adjacent and correlated tokens are typically sufficient to determine the span class. However, when the number of classes is large like in the fine-grained ET task and the EL task, contextual words may not be enough to determine the corresponding span classes [28,39,80,81].

### 4.3 Span Class Representations

Most information extraction models produce spans where the class element is a simple id or perhaps a pointer (*e.g.* `PER`, `businessman`, `wiki/Texas`). However, many span classes contain metadata such as text descriptions and even relationship information (*e.g.* in the case of Wikipedia). These span classes can be used to add additional context to possibly improve performance. Yet another option is to use pre-trained methods to obtain representations for each span's class. For example, the TagMe system used hyperlinks among pages to learn a class representation [27]; likewise, wikipedia2vec [107] and deep-ed [28] extended word2vec to learn token and class correlations. In a similar way, ERNIE [119] and LUKE [108] extended the BERT masking language model to entity disambiguation in order to obtain better token and span class representations.

## 5. MODEL TRANSFORMATIONS AMONG INFORMATION EXTRACTORS

Having previously identified the various information extraction tasks and their features, the next piece of the puzzle is to describe how spans are transformed by different information extraction models. This section presents a different, yet unifying, perspective on information extraction by considering the various *transformations* that a span undergoes for different information extraction tasks.

In their attempt to tackle different information extraction tasks, different information extraction models employ various *transformations* to the spans. These transformations can be grouped into six types of transformations: (*5.1.1*) sequential labeling, (*5.1.2*) token prototyping, (*5.1.3*) token-pair transformation, (*5.1.4*) span classification, (*5.1.5*) span locating, and (*5.1.6*) span generation.

As we shall see, unifying these different tasks reveals the importance of the various transformations. For example, the sequential labelling transformation commonly used in the NER task appears to be vastly different than the two-step transformation used in the EL task. However, as we shall see, because these tasks share the same input and output, these transformations do naturally generalize to other transformations and are actually swappable. Despite their interchangeability, the taxonomy of different transformations

TASK: NER + Token-Pair Classification

| INDEX | INPUT: | 0 Apple | 1 CEO | 2 Tim | 3 Cook | 4 sold | 5 his | 6 Texas | 7 house |
|---|---|---|---|---|---|---|---|---|---|
| 0 | Apple | 🟦 NNW | 🟦 | 🟦 | 🟦 | 🟦 | | | |
| 1 | CEO | 🟥 | 🟦 | 🟦 | 🟦 | 🟦 | 🟦 | | |
| 2 | Tim | 🟥 | 🟥 | 🟦 | 🟦 NNW | 🟦 | 🟦 | 🟦 | |
| 3 | Cook | 🟥 | 🟥 | 🟥 THW | 🟦 | 🟦 | 🟦 | 🟦 | 🟦 |
| 4 | sold | 🟥 | 🟥 | 🟥 | 🟥 | 🟦 | 🟦 | 🟦 | 🟦 |
| 5 | his | | 🟥 | 🟥 | 🟥 | 🟥 | 🟦 | 🟦 | 🟦 |
| 6 | Texas | | | 🟥 | 🟥 | 🟥 | 🟥 | 🟦 NNW | 🟦 |
| 7 | house | | | | 🟥 | 🟥 | 🟥 | 🟥 | 🟦 |

OUTPUT: [(0, 0, `ORG`), (2, 3, `PER`), (6, 6, `LOC`)]

Figure 2: Token Pair Transformation

(see Tab. 2 for details) does come with trade offs for different tasks. For example, the sequential labelling transformation decomposes span-labels into token-labels where each token assigned a label. This transformation ignores token locality features, which could be important in accurately finding span boundaries, but is nevertheless fast and easy to train. In this section we describe different model transformations and briefly discuss their trade offs.

## 5.1 Transformation Approaches

### *5.1.1 Sequential labeling*

Sequential labeling (*i.e.* token classification) is the most traditional and common transformation used in NER [37] as well as other information extraction tasks [5, 112, 122]. The core idea of sequential labelling is to directly transform spans into token-wise classes labeled with the Begin, Inside, Outside, End (BIOE) schema, where each label represents a token that begins, is inside of, ends, or is outside of some span. This schema has been expanded to also include other labels, like Left and Right (L/R), to represent tokens to the left and right of a span [47].

TASK: NER + Sequential Labelling

| | | | | |
|---|---|---|---|---|
| INDEX: | 0 | 1 | 2 | 3 |
| INPUT: | Apple | CEO | Tim | Cook |
| TRANS: | B-ORG | O | B-PER | E-PER |
| INDEX: | 4 | 5 | 6 | 7 |
| INPUT: | sold | his | Texas | house |
| TRANS: | O | O | B-LOC | O |

OUTPUT: [(0, 0, `ORG`), (2, 3, `PER`), (6, 6, `LOC`)]

Continuing the example above, the sequential labelling transformation uses the B and E labels to identify the beginning and end of the span encompassing `Tim Cook` one token at a time. Those tokens that are outside of the span are labeled with O. Then, during inference, token classes are first labeled and then span labels are obtained by concatenating one or more continuous tokens belonging to the same class. Note that the I and E labels can be missing in a span when a span has only one or two tokens.

### *5.1.2 Token Prototyping*

Token prototyping considers each span to be a sequence of tokens, which is then mapped to the same class [36,112,114]. Unlike sequential labelling, which labels tokens one at a time, token prototypes (*e.g.* `PER`, Businessman) are computed with locality and clustering based objectives. During inference, the tokens classes are represented as prototypes and embeddings are computed for each token. Then tokens that are close in the embedding space are clustered together and the corresponding classes are obtained by selecting the closest prototype.

TASK: NER + Token Prototyping

| | | | | |
|---|---|---|---|---|
| INDEX: | 0 | 1 | 2 | 3 |
| INPUT: | Apple | CEO | Tim | Cook |
| TRANS: | `ORG` | O | `PER` | `PER` |
| INDEX: | 4 | 5 | 6 | 7 |
| INPUT: | sold | his | Texas | house |
| TRANS: | O | O | `LOC` | O |

OUTPUT: [(0, 0, `ORG`), (2, 3, `PER`), (6, 6, `LOC`)]

Because this is a token-oriented approached, the tokens `Tim` and `Cook`, from the example above, are both individually assigned the `PER` label. Of course, this provides some ambiguity: it is unclear whether `Tim Cook` is a single person or two persons `Tim` and `Cook`. Typically, a post-processing step combines labels of the same type into a single multi-token span, but this isn't always desirable.

### *5.1.3 Token-pair Classification*

In token-pair classification, and as the name suggests, span labels are transformed into relationships between two tokens. During inference, each token-pair is labeled as one of an assortment of classes that describes the relationship between the two words within the same span as in illustrated in Fig *4.2.1*.

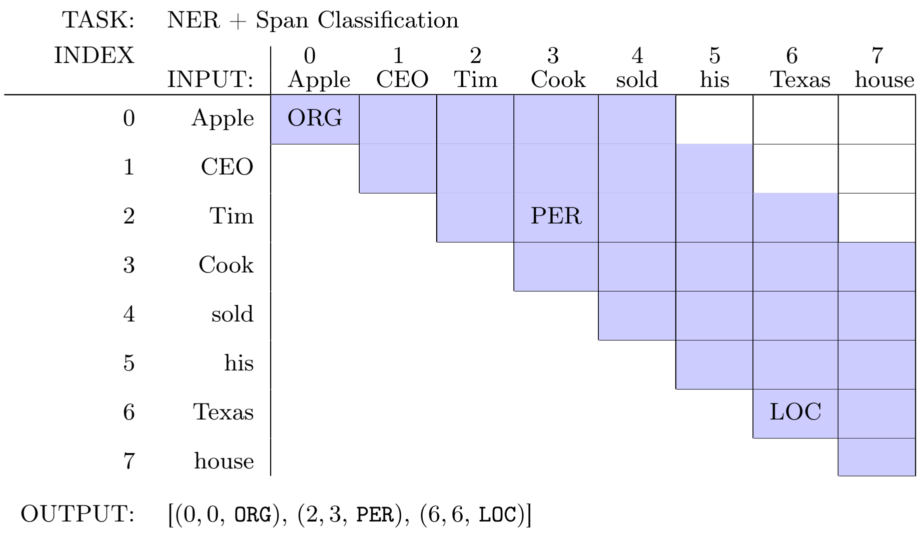
TASK: NER + Span Classification
INDEX
INPUT:
0 Apple
1 CEO
2 Tim
3 Cook
4 sold
5 his
6 Texas
7 house
ORG
PER
LOC
OUTPUT: [(0, 0, ORG), (2, 3, PER), (6, 6, LOC)]

Figure 3: Token Span Transformation

For example, the W2NER model uses labels *Next-Neighboring-Word* (NNW) to describe the relationship between pairs of words within a single span [48]. Applying this model to the running example produces the transformation above where blue cells represent the window size permitted by the model. Here the token-pair `Tim`–`Cook` is labeled with an NNW class describing `Cook` as the next-neighboring-word of `Tim`.

Although this transformation produces a sparse matrix with $\ell^2$ possible labels, it does permit non-contiguous dependency references to be labels produced in certain circumstances. Typically these additional labels are labeled as a *Tail-Head-Word* (THW) in the bottom-diagonal. The above illustration shows an example THW that, depending on the task, might be expanded THW-`PER` to indicate that the span refers to a person.

### 5.1.4 *Span Classification*

The core idea of the span classification transformation has two parts: (1) span candidate generation and (2) span label assignment. There are many ways to generate span candidates. The simplest is to enumerate all the possible spans up to certain window-lengths (*i.e.* n-grams) [91,123]. The window-length is normally a constant which is typically less than 5 in most tasks. Another way to generate spans is to learn a specific span generation model. For example, many of the token-oriented transformation approaches can generate span candidates with high-probability boundaries [95, 121]. In another vein, the Ask-and-Verify model uses a machine reading comprehension module to generate span candidates by finding potential boundaries from tokens with high predicted probabilities [23]. Span candidates can also be generated utilizing external data and models. For example, most existing entity disambiguation methods utilize rule based methods such as string match and frequency statistics [4, 5, 16, 28, 39, 43, 97]. Another popular way is to use retrieval models like TF-IDF [3], BM25 [55], phrase-mining [87,115], or dense retrieval [105] among many others.

After span candidates are generated, the goal of the following span label assignment step is to find a mapping function to select spans from the most promising span candidates and provide a class label. One common method is to employ a span-oriented classifier to distinguish positive span candidates from negative span candidates by screening all the span candidates [23,91,95,121]. One particularly compelling model, Locate-and-Label, which was inspired by two-stage object detection methods in computer vision also considers partially overlapped span candidates as positive samples as long as the intersection over union (IoU) is larger than a certain threshold value [90].

Continuing the running example, in Fig. *5.1.1* we utilize a upper triangular matrix with blue color to represent valid span candidates and the positive span is labeled with corresponding classes.

### 5.1.5 *Span Locating*

Another transformation approach is called span locating. The goal of this transformation is to consider an input sentence and relevant classes as a context-query pair and then find the corresponding span boundaries within the original input sentence [51]. The pipeline of this transformation is similar to machine reading comprehension (MRC), which is used to find answers in the context to the corresponding questions [82, 83]. Unlike in the span classification transformation where span candidates are generated, selected and labeled, in span locating the decision making process happens in reverse order: first the class label is determined and then the proper span holding that label is found.

Given a class label there are two typical ways to identify the span(s). First, given a input sentence with $\ell$ tokens, one way is to employ two $\ell$-class classifiers to predict the span boundaries (*i.e.* the start and end tokens) [23]. The second way is to employ two binary classifiers, one to predict whether each token is a starting token or not, and the other to determine whether each token is the end token or not, with the obvious restriction that the beginning token must precede the end token [51]. The span locating transformation illustrated in the example shows that first the `PER` and `LOC` classes are identified either via a input or another model. Then the transformation seeks to identify the boundary to-

Table 3: Trade-offs of Transformation Approaches

| | Complexity | Class Distr. | Features | | | Special Cases | |
|---|---|---|---|---|---|---|---|
| Transformation | # of inst. | +/- | Token | Span | Span-Class | Nested | Discontinuous |
| Sequential Labeling | $O(1)$ | 4/4 | ✓ | | | | |
| Token Prototype | $O(1)$ | 4/4 | ✓ | | | | |
| Token-pair Classification | $O(1)$ | 4/60 | ✓ | | | ✓ | ✓ |
| Span Classification | $O(1)$ | 3/27 | | ✓ | ✓ | ✓ | |
| Span Locating | $O(K)$ | 6/0 | ✓ | | ✓ | ✓ | |
| Span Generation | $O(\ell)$ | 6/8 | ✓ | | ✓ | | |

```
   TASK: NER + Span Locating

  INPUT: PER, LOC
  INDEX:     0        1       2       3
  INPUT:   Apple     CEO     Tim     Cook
            ↑↑                ↑       ↑
 TRANS: Start/End           Start    End

  INDEX:     4        5       6       7
  INPUT:    sold     his    Texas   house
                              ↑↑
 TRANS:                   Start/End

OUTPUT: [(0, 0, ORG), (2, 3, PER), (6, 6, LOC)]
```

kens that begin and end the span that represents the class labels within the sentence.

### 5.1.6 *Span Generation*

Text generation models have become popular especially with the rise in LLMs. These generation models provide another possibility in span-oriented transformations. The core idea of span generation is to transform the original token sequence into an expanded token sequence with span-tokens [46,79], similar to machine translation and other natural language generation tasks. In order to represent span labels in the token sequence, span generation typically inserts distinct characteristics to indicate span labels including both span positions and span classes [12,26,58].

```
Task: NER + Span Generation
   INDEX: 0      1      2      3
   INPUT: Apple  CEO    Tim    Cook
   INDEX: 4      5      6      7
   INPUT: sold   his    Texas  house

   TRANS: ORG-L  Apple  ORG-R  CEO
          PER-L  Tim    Cook   PER-R sold
          his    LOC-L  Texas  LOC-R house
OUTPUT: [(0, 0, ORG), (2, 3, PER), (6, 6, LOC)]
```

Continuing the running example illustrated above, span generation might transform the input token sequence into a token sequence having spans represented by special span tokens like `[PER-L]` indicating the start of a person span or `LOC-R` indicating the end of a location. Span generation models are commonly designed as an autoregressive token generation task [12,26,58]. They take a token sequence and a generated token as input, and predict the subsequent token. These predictions essentially represent classifications from a predefined dictionary. During inference, the generation process continues iteratively until all the desired spans are generated.

The span generation approach is widely utilized in many information extraction tasks. For example, GENRE employs an auto-regressive model to transform the entity disambiguation and entity linking tasks into a joint text/entity-name generation task [12]. For the entity disambiguation task, target entities are selected with a conditional generation method based on the provided token sequence. As for the entity linking task, the span indices and their labels are together transformed into an augmentation of the original sentence. Specific to the NER task, the BartNER model transforms the token sequence into uniform index pointers [111]. And more recently, Universal Information Extraction (UIE) architectures have also been developed to transform different information extraction tasks (*e.g.* NER, EL, ED) into the same format through generative language modeling [26,58]. UIE systems can extract shared features and joint correlations from training labels of different information extraction tasks. Furthermore, different structural signals across different information extraction tasks can be encoded into similar text allowing efficient and effective knowledge transfer from pre-trained models [5,12].

## 5.2 Transformations Trade Offs

As we alluded to in the previous section, different transformations have distinct trade-offs. Following Tab. 3, we present these considerations along four dimensions: (1) computational complexity, (2) the number of positive and negative class labels, (3) the features considered, and (4) applicability to nested and discontinuous spans. It is important to note that our discussion focuses on the general setup of these transformations without considering any specific design modifications. Although some models and methods may have specific designs tailored to address certain special cases, it is not our intention to delve into specific design strategies in the following discussion.

### 5.2.1 *Complexity*

Given an input document with $\ell$ tokens and $K$ different target span classes, different transformation approaches have different computational complexities. The main difference in complexity is the number of actual instances (*i.e.* # of instances in Tab. 3). This count directly corresponds to the number of times the feed-forward process needs to be executed in order to generate an inference output. Most transformations require a single input, namely, the token sequence, and makes several span predictions. The span locating transformation needs to consider each provided span class as an individual instance and is therefore in $O(K)$. Like machine translation, the span generation transformation considers each token in the input sequence as an individual instance yielding $O(\ell)$.

### 5.2.2 *Positive and Negative Span Distributions*

Different transformation approaches produce a different number of targets, *i.e.* positive and negative spans, and therefore the choice of transformation has a significant impact on the label distribution and, as a result, the performance metrics.

Again consider the running example illustrated above, which includes two named entities. In the sequence labelling transformation, four tokens including `Apple`, `Tim`, `Cook`, and `Texas` are positive instances representing span tokens; the other four tokens are negative instances representing non-span tokens. In token-pair classification, there are $\ell^2 = 64$ token pairs in total; of which two are positive samples. In span classification, with a window-length of five, there are a total of 30 possible spans with length less or equal to 5 from which only three represent positive spans. Finally, the span generation transformation produces three positive spans using six special tokens to annotate positive spans. Therefore the six special tokens are the positive instances and the original eight tokens are considered negative tokens. Clearly, different transformation approaches yield substantially different class distributions, which by definition has a large impact on performance metrics.

Noisy or incomplete labels are also differently impacted by the choice of transformation. Understanding these differences is important because many information extraction datasets have noisy training labels of 5% or more [103, 124] even for the well-known CONLL03 NER dataset [96]. In an interesting empirical study on the missing labels for information extraction tasks, Li *et al.* considered as a special case where noise is only present in negative samples [52]. They show that, during training, ignoring positive examples has small impact, but incorrectly labelling positive spans as being negative samples has significant impact on the final results. The same idea also applies for different transformations with different positive and negative span distributions.

### 5.2.3 *Features*

Different transformations use various models to produce spans, resulting in different abilities to encode different types of features. We focus on three main types of features: (1) token embeddings, (2) span embeddings, and (3) the span-class representation.

Token-wise transformations such as sequential labeling, token prototype, and token-pair classification decompose span labels into token-wise classes, making token features easy to encode but precluding the consideration of span features. In span classification, the embeddings of span candidates can be obtained and then span classes can be assigned to the entire span. In contrast, span locating encodes the span class along with the original sentence as input, and results are obtained by locating the span boundaries using the span indices, meaning that the span embedding cannot be considered. Likewise, span generation cannot use span embeddings, but instead represents span classes as a sequence of tokens to generate.

### 5.2.4 *Nested and Discontinuous Spans*

Overlapping spans is an important complication for training, inferences, and evaluation. This occurs when a single token is made to belong to two or more different spans. Different transformations handle these cases in different ways. We categorize them into two distinct cases: nested and discontinuous.

Nested spans are situations where a token can belong to multiple spans simultaneously. Transformations which allow for nested spans include token-pair classification, span classification, and span locating. Token-pair considers nested cases into different start-end token pairs; span classification considers all the possible span candidates, which includes overlapping spans; and span locating has no restriction on where the beginning and ending indices of different classes may be placed. In contrast, sequential labeling and token prototype transformations decompose span labels into token-wise labels; as a result, a token can not belong to two classes simultaneously. Likewise, span generation requires the injection of special tokens to indicate span classes, and therefore can not provide nested spans either.

Discontinuous cases, on the other hand, involve the formation of spans using non-adjacent tokens. In these cases, not-contiguous tokens in the text can be grouped together to form a single span. Because the token-pair matrix provides the flexibility to link non-contiguous tokens, it is the only method that can effectively handle discontinuous cases.

## 5.3 Training Strategies

After a span transformation approach is chosen, the systems needs to be properly employed to be effective. In other words, different training strategies have to be considered for a working framework. From our perspective, we categorize the various standard training strategies into four broad classes: (*5.3.1*) Feature tuning, which selects and engineers the most relevant features for a particular information extraction task; (*5.3.2*) Model tuning, which refers to the process of optimizing the parameters of a machine learning model; (*5.3.3*) Prompt tuning, which is the relatively new task of fine-tuning the prompts fed to LLMs to achieve more accurate results; and (*5.3.4*) In-Context learning, which involved training models in specific context information, for example, on specific datasets or niche tasks.

### 5.3.1 *Feature Tuning*

One of the outputs of pre-trained (large) language models are informative and well-trained embeddings. These embeddings almost always represent a span—sometimes as short as a word and sometimes as long as a whole sentence or paragraph—and are effective features that can be used for information extraction tasks. Prior to the rise of LLMs, most features used in information extraction tasks came from linguistic cues such as part-of-speech (POS) tags [84], word stems and lemmas [8], and syntactic parsers [75]; as well as statistical learning approaches like word frequency counts [2], word co-occurrence analysis [65], and semantic analysis [42]. Although these methods have their own strengths and weaknesses, they are generally less effective than LLM-based embeddings in capturing the complex relationships between words in a language [11, 19, 46, 65, 79]. Feature turning, therefore, refers to the numerous strategies that have been developed to learn ever-more creative and interesting features for spans. Equipped with these pretrained features, spans can be clustered or classified or labeled to solve any number of information extraction tasks.

### 5.3.2 *Model Tuning*

Sometimes, the pre-trained features from an LLM do not align well with the task that is being asked of the system. This misalignment will degrade the system's performance. In these cases it is common for LLMs to undergo a fine-tuning process, which adapts the pre-trained model parameters, including span features, to the specific task. Previous studies have shown that fine-tuning the model outperforms feature engineering with similar settings [19,46], but can be prone to catastrophic forgetting [54] and other maladies; see the survey by Li *et al.* [49] for details. One major problem is that model tuning requires that the model be loaded into memory and trained, which, for even medium-sized LLMs, is a non-trivial task.

### 5.3.3 *Prompt Tuning*

The rise of ChatGPT and other proprietary LLMs has spawned an entirely new kind of NLP task called prompt-tuning. In this case, instead of using or training span embeddings, prompt-tuning is the task of devising clever ways to query the LLM. The advantages of prompt tuning are clear. Because there is no need to extract, tune, or train any model or features, it is relatively easy to use the system. The task instead becomes finding the best prompts to feed to the LLM so that it returns the answers you seek. Another often overlooked advantage of prompt tuning is that by simply querying the system, it does not change. As a result any prompts or other rules that are learned can be maintained.

| | |
|---|---|
| TASK: | NER |
| INPUT: | Apple CEO Tim Cook sold his Texas house. |
| Prompt Input: | Apple CEO Tim Cook sold his Texas house. Tim Cook is a [MASK] |
| Prompt Output: | [MASK] $\rightarrow$ technology executive |
| OUTPUT: | $[(2, 3,$ PER$)]$ |

There are two types of prompt tuning: hard prompt tuning and soft prompt tuning. In hard prompt tuning, a hand-crafted prompt is used to glean results from the system. Conversely, in soft prompt tuning, the prompt itself can be trained. This means that an additional NLP model is trained to predict an adaptable prompt based on some input and labels. During the inference process, the prompt model initially generates a prompt, which is then concatenated with a test sample and fed into the LLM to obtain the final prediction; *i.e.* soft prompt tuning is a model generating input to feed to another model.

### 5.3.4 *In-context Learning*

Finally, In-context learning uses language models directly without any extra training process. This is accomplished by also injecting a few training examples along with corresponding labels into the prompts. The idea is that language models are able to see the mapping function between example input data and their corresponding labels, and then they can subsequently infer that same correspondence on unseen input data for label prediction.

An example of in-context learning on the NER task might resemble something like this:

| | |
|---|---|
| TASK: | NER |
| INPUT: | Apple CEO Tim Cook sold his Texas house. |
| Prompt Input: | Satya Nadella says that Microsoft products will soon connect to OpenAI.<br>Satya Nadella is a PER<br>Apple CEO Tim Cook sold his Texas house.<br>Tim Cook is a [MASK] |
| Prompt Output: | [MASK] is a PER |
| OUTPUT: | $[(2, 3,$ PER$)]$ |

With in-context learning a system can achieve robust capabilities with little cost. Another instance of this kind of learning is the chain-of-source approach, wherein a question and its corresponding answer are broken down into a series of sub-problems [54]. By addressing these sub-questions in a sequential manner, the system is able to arrive at more-comprehensive and nuanced solutions.

## 6. DISCUSSION

Over the past millennia, text has been made *by* humans *for* humans. The recent and broad digitization of human-generated text has served to propel AI systems and tasks like IE. When humans are tasked with performing information extraction, they do so—with relative ease—by first understanding the concepts and definitions of the labels in their context, even on unseen classes or in unknown languages. In contrast, we often find that AI systems in few-shot and zero-shot scenarios still perform much worse than humans [21,80]. This gap in performance is due to the AI system's inability to reason about the relationships between the context of the input and the context of the class label. Ongoing work in this area aims to properly encode these contexts. For example, previous work in encoding entity descriptions as search query targets has shown some ability to retrieve relevant entity candidates [55,105]. However, these relationships are nuanced; this research gap has not been fully explored, and a wide gap remains.

### 6.1 Challenges of Language Models in Information Extraction

While LLMs such as GPT-4 and its successors have demonstrated impressive performance on various natural language understanding tasks, they still face significant challenges in the domain of information extraction. One primary challenge lies in their lack of explicit understanding of span relationships, as discussed in earlier sections. LLMs excel in general language generation and can provide surprisingly coherent and contextually relevant outputs, but they struggle to extract and organize specific spans of text that correspond to labeled entities or relationships. The high variance in input phrasing, ambiguous contexts, and lack of consistent structuring of target outputs complicates the task further.

Moreover, although LLMs have been fine-tuned on a variety of tasks, the models' behavior in few-shot or zero-shot scenarios remains below human performance. These models often exhibit difficulties in identifying the correct spans for unseen classes or when confronted with ambiguous or contradictory inputs. Their limited ability to apply prior knowledge in a way that maps seamlessly to class-specific outputs is a key barrier.

Additionally, LLMs tend to "hallucinate" information, providing outputs that sound plausible but are factually incorrect or inconsistent with the context. While recent advancements in prompt engineering and in-context learning offer some solutions, these techniques are not always robust across domains. For example, a model may perform well on certain tasks but falter when transferring to another context without careful retraining or adaptation. This ongoing inconsistency in performance is a significant hurdle in making LLMs reliable for tasks that require structured information extraction.

## 6.2 Future Work

Looking ahead, there are several promising directions for advancing both information extraction tasks and the integration of language models with these tasks.

#### *6.2.0.1 Improved Span Representations.*

A major avenue for future work involves developing more sophisticated methods for span representation. While much progress has been made in learning to extract spans via attention mechanisms, there is still a need to refine how these spans are represented and related to their associated labels. More nuanced encoding methods that can capture the duality of surface forms and meaning will be essential for addressing tasks in few-shot and zero-shot settings. In particular, a more explicit handling of nested or overlapping spans could open up new avenues for tasks like coreference resolution and event argument extraction.

#### *6.2.0.2 Contextual and Transfer Learning.*

The future of information extraction lies in leveraging the vast amounts of contextual data that LLMs have access to. Research should focus on enhancing transfer learning capabilities, specifically focusing on how pretrained language models can be adapted to more specialized IE tasks through minimal labeled data. This includes advancing methods like in-context learning, prompt-based techniques, and few-shot learning paradigms. An exciting direction would be integrating task-specific knowledge into pre-trained LMs to enable more efficient extraction and minimize the need for retraining on every new domain.

#### *6.2.0.3 Span-Oriented Models for Cross-Task Generalization.*

As discussed throughout the paper, viewing IE tasks as span-extraction problems allows for a unified approach to these tasks. Future work could involve building models that are not only generalizable across related tasks but also capable of handling multiple forms of information extraction simultaneously. For example, a span-oriented model could be designed to extract both named entities and relations in a single pass, drastically reducing the complexity of current systems that require separate pipelines for each task. This could also facilitate more holistic systems for multi-task learning.

#### *6.2.0.4 Integrated Systems for Fact Verification and Evidence Retrieval.*

Another promising area for future work lies in integrating information extraction with downstream tasks like fact verification. Given the growing reliance on LLMs in generating information, it is crucial to ensure that these models are grounded in reliable, evidence-based facts. This involves developing systems where IE tasks are seamlessly integrated with external fact-checking mechanisms. In particular, span-extraction systems can play a pivotal role in identifying verifiable facts that LLMs can then use to ground their generative responses. Thus, future work should explore tighter integration between these components to improve the trustworthiness and reliability of LLM outputs.

#### *6.2.0.5 Evaluation and Benchmarking Frameworks.*

As the information extraction field continues to evolve, so too must our evaluation strategies. Currently, benchmarks for IE tasks often differ widely across tasks, making it difficult to draw meaningful comparisons between systems. Future work should aim to develop unified evaluation protocols that are applicable across a wide range of IE tasks. Metrics like span-level precision, recall, and F1 score should become the standard, allowing researchers to measure performance more consistently. Furthermore, benchmarks should evolve to account for the nuanced challenges posed by emerging techniques like generative IE.

#### *6.2.0.6 Ethical Considerations in Information Extraction.*

Finally, as with all AI systems, ethical considerations will play a pivotal role in shaping the future of information extraction. The potential for biases in training data and model predictions, particularly when models are deployed in sensitive or high-stakes applications, requires careful attention. Future work must explore methods for ensuring fairness, transparency, and accountability in IE systems. Moreover, we must be vigilant in preventing misuse, such as the extraction of private information or the reinforcement of harmful stereotypes. Developing ethical guidelines and establishing frameworks for responsible deployment will be crucial for the widespread adoption of these systems.

In conclusion, information extraction remains a challenging and dynamic field, but the advancements discussed here signal a promising future. The integration of language models with structured IE tasks offers a new paradigm for creating systems that are not only more powerful but also more aligned with human reasoning and understanding. Through continued research and collaboration across disciplines, we can bridge the gap between human-level understanding and AI's ability to perform complex information extraction tasks.

## Acknowledgements

This project was funded by DARPA under contracts HR001121C0168 and HR00112290106.

## 7. REFERENCES

[1] Dhruv Agarwal, Rico Angell, Nicholas Monath, and Andrew McCallum. Entity linking via explicit

mention-mention coreference modeling. In *NAACL-HLT*, pages 4644–4658, Seattle, United States, July 2022. Association for Computational Linguistics.

[2] Akiko Aizawa. An information-theoretic perspective of tf–idf measures. *Information Processing & Management*, 39(1):45–65, 2003.

[3] Rico Angell, Nicholas Monath, Sunil Mohan, Nishant Yadav, and Andrew McCallum. Clustering-based inference for biomedical entity linking. In *NAACL-HLT*, pages 2598–2608, Online, June 2021. Association for Computational Linguistics.

[4] Tom Ayoola, Joseph Fisher, and Andrea Pierleoni. Improving entity disambiguation by reasoning over a knowledge base. In *NAACL-HLT*, pages 2899–2912, Seattle, United States, July 2022. Association for Computational Linguistics.

[5] Tom Ayoola, Shubhi Tyagi, Joseph Fisher, Christos Christodoulopoulos, and Andrea Pierleoni. ReFinED: An efficient zero-shot-capable approach to end-to-end entity linking. In *NAACL-HLT*, pages 209–220, Hybrid: Seattle, Washington + Online, July 2022. Association for Computational Linguistics.

[6] Seohyun Back, Sai Chetan Chinthakindi, Akhil Kedia, Haejun Lee, and Jaegul Choo. Neurquri: Neural question requirement inspector for answerability prediction in machine reading comprehension. In *ICLR*, 2020.

[7] Edoardo Barba, Luigi Procopio, and Roberto Navigli. ExtEnD: Extractive entity disambiguation. In *ACL*, pages 2478–2488, Dublin, Ireland, May 2022. Association for Computational Linguistics.

[8] Steven Bird and Edward Loper. NLTK: The natural language toolkit. In *ACL*, pages 214–217, Barcelona, Spain, July 2004. Association for Computational Linguistics.

[9] Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. Enriching Word Vectors with Subword Information. *TACL*, 5:135–146, 06 2017.

[10] Samuel Broscheit. Investigating entity knowledge in BERT with simple neural end-to-end entity linking. In *CoNLL*, pages 677–685, Hong Kong, China, November 2019. Association for Computational Linguistics.

[11] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *NeurIPS*, 33:1877–1901, 2020.

[12] Nicola De Cao, Gautier Izacard, Sebastian Riedel, and Fabio Petroni. Autoregressive entity retrieval. In *ICLR*, 2021.

[13] Ilias Chalkidis, Manos Fergadiotis, Prodromos Malakasiotis, Nikolaos Aletras, and Ion Androutsopoulos. LEGAL-BERT: The muppets straight out of law school. In *Findings of EMNLP*, pages 2898–2904, Online, November 2020. Association for Computational Linguistics.

[14] Danqi Chen and Christopher Manning. A fast and accurate dependency parser using neural networks. In *EMNLP*, pages 740–750, Doha, Qatar, October 2014. Association for Computational Linguistics.

[15] Haotian Chen, Xi Li, Andrej Zukov Gregoric, and Sahil Wadhwa. Contextualized end-to-end neural entity linking. In *AACL*, pages 637–642, Suzhou, China, December 2020. Association for Computational Linguistics.

[16] Shuang Chen, Jinpeng Wang, Feng Jiang, and Chin-Yew Lin. Improving entity linking by modeling latent entity type information. In *AAAI*, volume 34, pages 7529–7537, Apr. 2020.

[17] Eunsol Choi, Omer Levy, Yejin Choi, and Luke Zettlemoyer. Ultra-fine entity typing. In *ACL*, pages 87–96, Melbourne, Australia, July 2018. Association for Computational Linguistics.

[18] Luciano Del Corro, Abdalghani Abujabal, Rainer Gemulla, and Gerhard Weikum. FINET: Context-aware fine-grained named entity typing. In *EMNLP*, pages 868–878, Lisbon, Portugal, September 2015. Association for Computational Linguistics.

[19] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In *NAACL-HLT*, pages 4171–4186, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics.

[20] Bhuwan Dhingra, Hanxiao Liu, Zhilin Yang, William Cohen, and Ruslan Salakhutdinov. Gated-attention readers for text comprehension. In *ACL*, pages 1832–1846, Vancouver, Canada, July 2017. Association for Computational Linguistics.

[21] Ning Ding, Yulin Chen, Xu Han, Guangwei Xu, Pengjun Xie, Hai-Tao Zheng, Zhiyuan Liu, Juanzi Li, and Hong-Gee Kim. Prompt-learning for fine-grained entity typing. *arXiv preprint arXiv:2108.10604*, 2021.

[22] Yifan Ding, Nicholas Botzer, and Tim Weninger. Posthoc verification and the fallibility of the ground truth. In *Proceedings of the First Workshop on Dynamic Adversarial Data Collection*, pages 23–29, Seattle, WA, July 2022. Association for Computational Linguistics.

[23] Yifan Ding, Yan Liang, Nasser Zalmout, Xian Li, Christan Grant, and Tim Weninger. Ask-and-verify: Span candidate generation and verification for attribute value extraction. In *EMNLP*, pages 110–110, Abu Dhabi, UAE, December 2022. Association for Computational Linguistics.

[24] Yifan Ding, Daheng Wang, Tim Weninger, and Meng Jiang. Preserving composition and crystal structures of chemical compounds in atomic embedding. In *Big Data*, pages 6037–6039, 2019.

[25] George Doddington, Alexis Mitchell, Mark Przybocki, Lance Ramshaw, Stephanie Strassel, and Ralph Weischedel. The automatic content extraction (ACE) program – tasks, data, and evaluation. In *LREC*, Lisbon, Portugal, May 2004. European Language Resources Association (ELRA).

[26] Hao Fei, Shengqiong Wu, Jingye Li, Bobo Li, Fei Li, Libo Qin, Meishan Zhang, Min Zhang, and Tat-Seng Chua. LasUIE: Unifying information extraction with latent adaptive structure-aware generative language

model. In Alice H. Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho, editors, *NeurIPS*, 2022.

[27] Paolo Ferragina and Ugo Scaiella. Tagme: On-the-fly annotation of short text fragments (by wikipedia entities). In *CIKM*, page 1625–1628, New York, NY, USA, 2010. Association for Computing Machinery.

[28] Octavian-Eugen Ganea and Thomas Hofmann. Deep joint entity disambiguation with local neural attention. In *EMNLP*, pages 2619–2629, Copenhagen, Denmark, September 2017. Association for Computational Linguistics.

[29] Ralph Grishman and Beth Sundheim. Message Understanding Conference- 6: A brief history. In *COLING*, 1996.

[30] Yingjie Gu, Xiaoye Qu, Zhefeng Wang, Baoxing Huai, Nicholas Jing Yuan, and Xiaolin Gui. Read, retrospect, select: An mrc framework to short text entity linking. In *AAAI*, volume 35, pages 12920–12928, 2021.

[31] Imane Guellil, Antonio Garcia-Dominguez, Peter R Lewis, Shakeel Hussain, and Geoffrey Smith. Entity linking for english and other languages: a survey. *KAIS*, 66(7):3773–3824, 2024.

[32] Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, et al. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025.

[33] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[34] Johannes Hoffart, Mohamed Amir Yosef, Ilaria Bordino, Hagen Fürstenau, Manfred Pinkal, Marc Spaniol, Bilyana Taneva, Stefan Thater, and Gerhard Weikum. Robust disambiguation of named entities in text. In *EMNLP*, pages 782–792, Edinburgh, Scotland, UK., July 2011. Association for Computational Linguistics.

[35] Matthew Honnibal and Ines Montani. spacy 2: Natural language understanding with bloom embeddings, convolutional neural networks and incremental parsing. 7(1):411–420, 2017.

[36] Jiaxin Huang, Chunyuan Li, Krishan Subudhi, Damien Jose, Shobana Balakrishnan, Weizhu Chen, Baolin Peng, Jianfeng Gao, and Jiawei Han. Few-shot named entity recognition: An empirical baseline study. In *EMNLP*, pages 10408–10423, Online and Punta Cana, Dominican Republic, November 2021. Association for Computational Linguistics.

[37] Zhiheng Huang, Wei Xu, and Kai Yu. Bidirectional lstm-crf models for sequence tagging. *arXiv:1508.01991*, 2015.

[38] Rudolf Kadlec, Martin Schmid, Ondrej Bajgar, and Jan Kleindienst. Text understanding with the attention sum reader network. In *ACL*, pages 908–918, Berlin, Germany, August 2016. Association for Computational Linguistics.

[39] Nikolaos Kolitsas, Octavian-Eugen Ganea, and Thomas Hofmann. End-to-end neural entity linking. In *CoNLL*, pages 519–529, Brussels, Belgium, October 2018. Association for Computational Linguistics.

[40] Paul R Kroeger. *Analyzing grammar: An introduction.* Cambridge University Press, 2005.

[41] John D. Lafferty, Andrew McCallum, and Fernando C. N. Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *ICML*, page 282–289, San Francisco, CA, USA, 2001. Morgan Kaufmann Publishers Inc.

[42] Thomas K Landauer, Peter W Foltz, and Darrell Laham. An introduction to latent semantic analysis. *Discourse processes*, 25(2-3):259–284, 1998.

[43] Phong Le and Ivan Titov. Improving entity linking by modeling latent relations between mentions. In *ACL*, pages 1595–1604, Melbourne, Australia, July 2018. Association for Computational Linguistics.

[44] Yann LeCun, Yoshua Bengio, et al. Convolutional networks for images, speech, and time series. *The handbook of brain theory and neural networks*, 3361(10):1995, 1995.

[45] Jinhyuk Lee, Wonjin Yoon, Sungdong Kim, Donghyeon Kim, Sunkyu Kim, Chan Ho So, and Jaewoo Kang. Biobert: a pre-trained biomedical language representation model for biomedical text mining. *Bioinformatics*, 36(4):1234–1240, 2020.

[46] Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *ACL*, pages 7871–7880, Online, July 2020. Association for Computational Linguistics.

[47] Jing Li, Aixin Sun, Jianglei Han, and Chenliang Li. A survey on deep learning for named entity recognition. *IEEE TKDE*, 34(1):50–70, 2022.

[48] Jingye Li, Hao Fei, Jiang Liu, Shengqiong Wu, Meishan Zhang, Chong Teng, Donghong Ji, and Fei Li. Unified named entity recognition as word-word relation classification. *AAAI*, 36(10):10965–10973, Jun. 2022.

[49] Junyi Li, Tianyi Tang, Wayne Xin Zhao, and Ji-Rong Wen. Pretrained language models for text generation: A survey. *arXiv preprint arXiv:2105.10311*, 2021.

[50] Peng-Hsuan Li, Ruo-Ping Dong, Yu-Siang Wang, Ju-Chieh Chou, and Wei-Yun Ma. Leveraging linguistic structures for named entity recognition with bidirectional recursive neural networks. In *EMNLP*, pages 2664–2669, Copenhagen, Denmark, September 2017. Association for Computational Linguistics.

[51] Xiaoya Li, Jingrong Feng, Yuxian Meng, Qinghong Han, Fei Wu, and Jiwei Li. A unified MRC framework for named entity recognition. In *ACL*, pages 5849–5859, Online, July 2020. Association for Computational Linguistics.

[52] Yangming Li, lemao liu, and Shuming Shi. Empirical analysis of unlabeled entity problem in named entity recognition. In *ICLR*, 2021.

[53] Aixin Liu, Bei Feng, Bing Xue, Bingxuan Wang, Bochao Wu, Chengda Lu, Chenggang Zhao, Chengqi

Deng, Chenyu Zhang, Chong Ruan, et al. Deepseek-v3 technical report. *arXiv preprint arXiv:2412.19437*, 2024.

[54] Pengfei Liu, Weizhe Yuan, Jinlan Fu, Zhengbao Jiang, Hiroaki Hayashi, and Graham Neubig. Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing. *ACM Computing Surveys*, 55(9):1–35, 2023.

[55] Lajanugen Logeswaran, Ming-Wei Chang, Kenton Lee, Kristina Toutanova, Jacob Devlin, and Honglak Lee. Zero-shot entity linking by reading entity descriptions. In *ACL*, pages 3449–3460, Florence, Italy, July 2019. Association for Computational Linguistics.

[56] Jie Lou, Yaojie Lu, Dai Dai, Wei Jia, Hongyu Lin, Xianpei Han, Le Sun, and Hua Wu. Universal information extraction as unified semantic matching. *AAAI*, 37(11):13318–13326, Jun. 2023.

[57] Yaojie Lu, Qing Liu, Dai Dai, Xinyan Xiao, Hongyu Lin, Xianpei Han, Le Sun, and Hua Wu. Unified structure generation for universal information extraction. In Smaranda Muresan, Preslav Nakov, and Aline Villavicencio, editors, *ACL*, pages 5755–5772, Dublin, Ireland, May 2022. Association for Computational Linguistics.

[58] Yaojie Lu, Qing Liu, Dai Dai, Xinyan Xiao, Hongyu Lin, Xianpei Han, Le Sun, and Hua Wu. Unified structure generation for universal information extraction. In *ACL*, pages 5755–5772, Dublin, Ireland, May 2022. Association for Computational Linguistics.

[59] Xuezhe Ma and Eduard Hovy. End-to-end sequence labeling via bi-directional LSTM-CNNs-CRF. In *ACL*, pages 1064–1074, Berlin, Germany, August 2016. Association for Computational Linguistics.

[60] Yukun Ma, Erik Cambria, and Sa Gao. Label embedding for zero-shot fine-grained named entity typing. In *COLING*, pages 171–180, Osaka, Japan, December 2016.

[61] Christopher Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven Bethard, and David McClosky. The Stanford CoreNLP natural language processing toolkit. In *ACL*, pages 55–60, Baltimore, Maryland, June 2014. Association for Computational Linguistics.

[62] Mitchell Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. Building a large annotated corpus of english: The penn treebank. 1993.

[63] Andrew McCallum, Dayne Freitag, and Fernando CN Pereira. Maximum entropy markov models for information extraction and segmentation. In *ICML*, volume 17, pages 591–598, 2000.

[64] Yuxian Meng, Wei Wu, Fei Wang, Xiaoya Li, Ping Nie, Fan Yin, Muyu Li, Qinghong Han, Xiaofei Sun, and Jiwei Li. Glyce: Glyph-vectors for chinese character representations. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *NeurIPS*, volume 32. Curran Associates, Inc., 2019.

[65] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In C.J. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K.Q. Weinberger, editors, *NeurIPS*, volume 26. Curran Associates, Inc., 2013.

[66] Sewon Min, Victor Zhong, Luke Zettlemoyer, and Hannaneh Hajishirzi. Multi-hop reading comprehension through question decomposition and rescoring. In *ACL*, pages 6097–6109, Florence, Italy, July 2019. Association for Computational Linguistics.

[67] Sudha Morwal, Nusrat Jahan, and Deepti Chopra. Named entity recognition using hidden markov model (hmm). *IJNLC*, 1, 2012.

[68] Shikhar Murty, Patrick Verga, Luke Vilnis, Irena Radovanovic, and Andrew McCallum. Hierarchical losses and new resources for fine-grained entity typing and linking. In *ACL* , pages 97–109, Melbourne, Australia, July 2018. Association for Computational Linguistics.

[69] Zara Nasar, Syed Waqar Jaffry, and Muhammad Kamran Malik. Named entity recognition and relation extraction: State-of-the-art. *ACM Computing Surveys*, 54(1):1–39, 2021.

[70] Iulian Neamtiu, Jeffrey S Foster, and Michael Hicks. Understanding source code evolution using abstract syntax tree matching. In *Proceedings of the 2005 international workshop on Mining software repositories*, pages 1–5, 2005.

[71] Yasumasa Onoe and Greg Durrett. Fine-grained entity typing for domain independent entity linking. In *AAAI*, volume 34, pages 8576–8583, 2020.

[72] Jeffrey Pennington, Richard Socher, and Christopher Manning. GloVe: Global vectors for word representation. In *EMNLP*, pages 1532–1543, Doha, Qatar, October 2014. Association for Computational Linguistics.

[73] Minh C. Phan, Aixin Sun, Yi Tay, Jialong Han, and Chenliang Li. Neupl: Attention-based semantic matching and pair-linking for entity disambiguation. In *CIKM*, page 1667–1676, New York, NY, USA, 2017. Association for Computing Machinery.

[74] Minh C Phan, Aixin Sun, Yi Tay, Jialong Han, and Chenliang Li. Pair-linking for collective entity disambiguation: Two could be better than all. *IEEE TKDE*, 31(7):1383–1396, 2018.

[75] Martin J Pickering and Roger PG Van Gompel. Syntactic parsing. In *Handbook of psycholinguistics*, pages 455–503. Elsevier, 2006.

[76] Weizhen Qi, Yu Yan, Yeyun Gong, Dayiheng Liu, Nan Duan, Jiusheng Chen, Ruofei Zhang, and Ming Zhou. ProphetNet: Predicting future n-gram for sequence-to-SequencePre-training. In *Findings of EMNLP*, pages 2401–2410, Online, November 2020. Association for Computational Linguistics.

[77] Alec Radford, Karthik Narasimhan, Tim Salimans, Ilya Sutskever, et al. Improving language understanding by generative pre-training. 2018.

[78] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019.

[79] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *JMLR*, 21:1–67, 2020.

[80] Jonathan Raiman. Deeptype 2: Superhuman entity linking, all you need is type interactions. In *AAAI*, volume 36, pages 8028–8035, 2022.

[81] Jonathan Raiman and Olivier Raiman. Deeptype: multilingual entity linking by neural type system evolution. In *AAAI*, volume 32, 2018.

[82] Pranav Rajpurkar, Robin Jia, and Percy Liang. Know what you don't know: Unanswerable questions for SQuAD. In *ACL*, pages 784–789, Melbourne, Australia, July 2018. Association for Computational Linguistics.

[83] Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. SQuAD: 100,000+ questions for machine comprehension of text. In *EMNLP*, pages 2383–2392, Austin, Texas, November 2016. Association for Computational Linguistics.

[84] Jan Rijkhoff. Word classes. *Language and Linguistics Compass*, 1(6):709–726, 2007.

[85] Minjoon Seo, Aniruddha Kembhavi, Ali Farhadi, and Hannaneh Hajishirzi. Bidirectional attention flow for machine comprehension. In *ICLR*, 2017.

[86] Özge Sevgili, Artem Shelmanov, Mikhail Arkhipov, Alexander Panchenko, and Chris Biemann. Neural entity linking: A survey of models based on deep learning. *Semant. Web*, 13(3):527–570, jan 2022.

[87] Jingbo Shang, Jialu Liu, Meng Jiang, Xiang Ren, Clare R. Voss, and Jiawei Han. Automated phrase mining from massive text corpora. *IEEE TKDE*, 30(10):1825–1837, 2018.

[88] Wei Shen, Yuhan Li, Yinan Liu, Jiawei Han, Jianyong Wang, and Xiaojie Yuan. Entity linking meets deep learning: Techniques and solutions. *IEEE TKDE*, pages 1–1, 2021.

[89] Wei Shen, Jianyong Wang, and Jiawei Han. Entity linking with a knowledge base: Issues, techniques, and solutions. *IEEE TKDE*, 27(2):443–460, 2014.

[90] Yongliang Shen, Xinyin Ma, Zeqi Tan, Shuai Zhang, Wen Wang, and Weiming Lu. Locate and label: A two-stage identifier for nested named entity recognition. In *ACL*, pages 2782–2794, Online, August 2021. Association for Computational Linguistics.

[91] Mohammad Golam Sohrab and Makoto Miwa. Deep exhaustive model for nested named entity recognition. In *EMNLP*, pages 2843–2849, Brussels, Belgium, October-November 2018. Association for Computational Linguistics.

[92] Alessandro Sordoni, Philip Bachman, Adam Trischler, and Yoshua Bengio. Iterative alternating neural attention for machine reading. *arXiv preprint arXiv:1606.02245*, 2016.

[93] Emma Strubell, Patrick Verga, David Belanger, and Andrew McCallum. Fast and accurate entity recognition with iterated dilated convolutions. In *EMNLP*, pages 2670–2680, Copenhagen, Denmark, September 2017. Association for Computational Linguistics.

[94] Zijun Sun, Xiaoya Li, Xiaofei Sun, Yuxian Meng, Xiang Ao, Qing He, Fei Wu, and Jiwei Li. ChineseBERT: Chinese pretraining enhanced by glyph and Pinyin information. In *ACL*, pages 2065–2075, Online, August 2021. Association for Computational Linguistics.

[95] Chuanqi Tan, Wei Qiu, Mosha Chen, Rui Wang, and Fei Huang. Boundary enhanced neural span classification for nested named entity recognition. *AAAI*, 34(05):9016–9023, Apr. 2020.

[96] Erik F. Tjong Kim Sang and Fien De Meulder. Introduction to the CoNLL-2003 shared task: Language-independent named entity recognition. In *NAACL*, pages 142–147, 2003.

[97] Johannes M Van Hulst, Faegheh Hasibi, Koen Dercksen, Krisztian Balog, and Arjen P de Vries. Rel: An entity linker standing on the shoulders of giants. In *SIGIR*, pages 2197–2200, 2020.

[98] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *NeurIPS*, 30, 2017.

[99] Oriol Vinyals, Meire Fortunato, and Navdeep Jaitly. Pointer networks. In C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, and R. Garnett, editors, *NeurIPS*, volume 28. Curran Associates, Inc., 2015.

[100] Qifan Wang, Li Yang, Bhargav Kanagal, Sumit Sanghai, D. Sivakumar, Bin Shu, Zac Yu, and Jon Elsas. Learning to extract attribute value from product via question answering: A multi-task approach. In *SIGKDD*, page 47–55, New York, NY, USA, 2020. Association for Computing Machinery.

[101] Shuohang Wang and Jing Jiang. Machine comprehension using match-lstm and answer pointer. In *arXiv preprint arXiv:1608.07905*, 2016.

[102] Xuan Wang, Vivian Hu, Xiangchen Song, Shweta Garg, Jinfeng Xiao, and Jiawei Han. ChemNER: Fine-grained chemistry named entity recognition with ontology-guided distant supervision. In *EMNLP*, pages 5227–5240, Online and Punta Cana, Dominican Republic, November 2021. Association for Computational Linguistics.

[103] Zihan Wang, Jingbo Shang, Liyuan Liu, Lihao Lu, Jiacheng Liu, and Jiawei Han. CrossWeigh: Training named entity tagger from imperfect annotations. In *EMNLP-IJCNLP*, pages 5154–5163, Hong Kong, China, November 2019. Association for Computational Linguistics.

[104] Zihan Wang, Kewen Zhao, Zilong Wang, and Jingbo Shang. Formulating few-shot fine-tuning towards language model pre-training: A pilot study on named entity recognition. *arXiv preprint arXiv:2205.11799*, 2022.

[105] Ledell Wu, Fabio Petroni, Martin Josifoski, Sebastian Riedel, and Luke Zettlemoyer. Scalable zero-shot entity linking with dense entity retrieval. In *EMNLP*, pages 6397–6407, Online, November 2020. Association for Computational Linguistics.

[106] Huimin Xu, Wenting Wang, Xin Mao, Xinyu Jiang, and Man Lan. Scaling up open tagging from tens to thousands: Comprehension empowered attribute value extraction from product title. In *ACL*, pages 5214–5223, Florence, Italy, July 2019. Association for Computational Linguistics.

[107] Ikuya Yamada, Akari Asai, Jin Sakuma, Hiroyuki Shindo, Hideaki Takeda, Yoshiyasu Takefuji, and Yuji Matsumoto. Wikipedia2Vec: An efficient toolkit for learning and visualizing the embeddings of words and entities from Wikipedia. In *EMNLP*, pages 23–30, Online, October 2020. Association for Computational Linguistics.

[108] Ikuya Yamada, Akari Asai, Hiroyuki Shindo, Hideaki Takeda, and Yuji Matsumoto. LUKE: Deep contextualized entity representations with entity-aware self-attention. In *EMNLP*, pages 6442–6454, Online, nov 2020. Association for Computational Linguistics.

[109] Ikuya Yamada, Hiroyuki Shindo, Hideaki Takeda, and Yoshiyasu Takefuji. Joint learning of the embedding of words and entities for named entity disambiguation. In *CoNLL*, pages 250–259, Berlin, Germany, August 2016. Association for Computational Linguistics.

[110] Ikuya Yamada, Koki Washio, Hiroyuki Shindo, and Yuji Matsumoto. Global entity disambiguation with BERT. In *NAACL-HLT*, pages 3264–3271, Seattle, United States, July 2022. Association for Computational Linguistics.

[111] Hang Yan, Tao Gui, Junqi Dai, Qipeng Guo, Zheng Zhang, and Xipeng Qiu. A unified generative framework for various NER subtasks. In *ACL*, pages 5808–5822, Online, August 2021. Association for Computational Linguistics.

[112] Jun Yan, Nasser Zalmout, Yan Liang, Christan Grant, Xiang Ren, and Xin Luna Dong. AdaTag: Multi-attribute value extraction from product profiles with adaptive decoding. In *ACL*, pages 4694–4705, Online, August 2021. Association for Computational Linguistics.

[113] Yang Yu, Wei Zhang, Kazi Hasan, Mo Yu, Bing Xiang, and Bowen Zhou. End-to-end answer chunk extraction and ranking for reading comprehension. *arXiv preprint arXiv:1610.09996*, 2016.

[114] Nasser Zalmout and Xian Li. Prototype-representations for training data filtering in weakly-supervised information extraction. In *EMNLP*, pages 467–474, Abu Dhabi, UAE, December 2022. Association for Computational Linguistics.

[115] Qingkai Zeng, Wenhao Yu, Mengxia Yu, Tianwen Jiang, Tim Weninger, and Meng Jiang. Tri-train: Automatic pre-fine tuning between pre-training and fine-tuning for SciNER. In *Findings of EMNLP*, pages 4778–4787, Online, November 2020. Association for Computational Linguistics.

[116] Feifei Zhai, Saloni Potdar, Bing Xiang, and Bowen Zhou. Neural models for sequence chunking. *AAAI*, 31(1), Feb. 2017.

[117] Wenzheng Zhang, Wenyue Hua, and Karl Stratos. EntQA: Entity linking as question answering. In *ICLR*, 2022.

[118] Zhen Zhang, Yuhua Zhao, Hang Gao, and Mengting Hu. Linkner: linking local named entity recognition models to large language models using uncertainty. In *WebConf*, pages 4047–4058, 2024.

[119] Zhengyan Zhang, Xu Han, Zhiyuan Liu, Xin Jiang, Maosong Sun, and Qun Liu. ERNIE: Enhanced language representation with informative entities. In *ACL*, pages 1441–1451, Florence, Italy, July 2019. Association for Computational Linguistics.

[120] Zhuosheng Zhang, Junjie Yang, and Hai Zhao. Retrospective reader for machine reading comprehension. *AAAI*, 35(16):14506–14514, May 2021.

[121] Changmeng Zheng, Yi Cai, Jingyun Xu, Ho-fung Leung, and Guandong Xu. A boundary-aware neural model for nested named entity recognition. In *EMNLP-IJCNLP*, pages 357–366, Hong Kong, China, November 2019. Association for Computational Linguistics.

[122] Guineng Zheng, Subhabrata Mukherjee, Xin Luna Dong, and Feifei Li. Opentag: Open attribute value extraction from product profiles. In *SIGKDD*, page 1049–1058, New York, NY, USA, 2018. Association for Computing Machinery.

[123] Zexuan Zhong and Danqi Chen. A frustratingly easy approach for entity and relation extraction. In *NAACL*, pages 50–61, Online, June 2021. Association for Computational Linguistics.

[124] Wenxuan Zhou and Muhao Chen. Learning from noisy labels for entity-centric information extraction. In *EMNLP*, pages 5381–5392, Online and Punta Cana, Dominican Republic, November 2021. Association for Computational Linguistics.