OntoType: Ontology-Guided and Pre-Trained Language Model Assisted Fine-Grained Entity Typing

Tanay Komarlu
University of Illinois Urbana-Champaign
Urbana, Illinois, USA
komarlu2@illinois.edu

Minhao Jiang
University of Illinois Urbana-Champaign
Urbana, IL, USA
minhaoj2@illinois.edu

Xuan Wang
Virginia Tech
Blacksburg, Virginia, USA
xuanw@vt.edu

Jiawei Han
University of Illinois Urbana-Champaign
Urbana, Illinois, USA
hanj@illinois.edu

ABSTRACT

Fine-grained entity typing (FET), which assigns entities in text with context-sensitive, fine-grained semantic types, is a basic but important task for knowledge extraction from unstructured text. FET has been studied extensively in natural language processing and typically relies on human-annotated corpora for training, which is costly and difficult to scale. Recent studies explore the utilization of pre-trained language models (PLMs) as a knowledge base to generate rich and context-aware weak supervision for FET. However, a PLM still requires direction and guidance to serve as a knowledge base as they often generate a mixture of rough and fine-grained types, or tokens unsuitable for typing. In this study, we vision that an ontology provides a semantics-rich, hierarchical structure, which will help select the best results generated by multiple PLM models and head words. Specifically, we propose a novel *annotation-free, ontology-guided* FET method, OntoType, which follows a type ontological structure, from coarse to fine, ensembles multiple PLM prompting results to generate a set of type candidates, and refines its type resolution, under the local context with a natural language inference model. Our experiments on the Ontonotes, FIGER, and NYT datasets using their associated ontological structures demonstrate that our method outperforms the state-of-the-art zero-shot fine-grained entity typing methods as well as a typical LLM method, ChatGPT. Our error analysis shows that refinement of the existing ontology structures will further improve fine-grained entity typing.

CCS CONCEPTS

• **Computing methodologies** → **Information extraction**; **Natural language processing**; *Language resources.*

KEYWORDS

Fine-Grained Entity Typing, Zero-Shot Entity Typing, Masked Language Model Prompting, Natural Language Understanding

KDD '24, August 25–29, 2024, Barcelona, Spain

ACM ISBN 979-8-4007-0490-1/24/08.
https://doi.org/10.1145/3637528.3671745

ACM Reference Format:
Tanay Komarlu, Minhao Jiang, Xuan Wang, and Jiawei Han. 2024. OntoType: Ontology-Guided and Pre-Trained Language Model Assisted Fine-Grained Entity Typing. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD '24), August 25–29, 2024, Barcelona, Spain.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3637528.3671745

1 INTRODUCTION

Fine-grained entity typing (FET) is a basic but important task for knowledge extraction and text content understanding/analysis. Assigning fine-grained semantic types to parsed entity mention spans based on the local context enables effective and structured analysis of unstructured text data, such as entity linking [15, 21], relation extraction [12], and coreference resolution [21].

Example 1. Given a sentence S_1: "*Sammy Sosa got a standing ovation at Wrigley Field.*" and a parsed entity mention span "*Sammy Sosa*" in the sentence, an FET method aims to assign it not only the coarse-grained type "Person" but also the fine-grained types "Athlete" or "Player".

FET on large text corpora is a challenging task due to (1) the high cost of obtaining a large amount of human-annotated training data, especially in dynamic and emerging domains, and (2) inaccurate annotations due to (i) different annotators marking concepts at different granularity (e.g., person vs. politician vs. president), and (ii) contextual subtlety on fine-grained types (e.g., Boston vs. Detroit could be two sports teams, instead of two cities). Most existing methods utilize weak or distant supervision to automatically generate training data for the FET tasks. There are three major approaches to obtaining weakly-labeled training data to tackle these challenges. The first is to automatically match the mentions in text with the concepts in some existing *knowledge bases* (e.g., Wikipedia) [15]. The typical workflow is to first detect entity mentions from a corpus, map these mentions to knowledge base (KB) entities of target types, and then leverage the confidently mapped types as pseudo-labeled data to infer the final type. The second is to directly obtain the *head words* of nominal mentions as its fine-grained type [3]. This approach leverages the head words of the entity mention to consolidate context-aware types derived from the KB matching. However, both approaches suffer from label sparsity and context-agnostic problems, resulting in the inability to generate high-quality training data for FET.

The third approach is to *probe the pre-trained language models* through the use of masked patterns and entailment templates. This method enables the use of PLM as a knowledge base for weak supervision. Leveraging masked language model (MLM) prompting to generate rich and context-aware weak supervisions for fine-grained entity typing is a recent trend, aiming to leverage a PLM's contextual and "common-sense" knowledge learned at training to reduce expensive human labor [4, 13].

Given a sentence that contains a mention, one can use a "[MASK]" token to replace the entity mention span to generate candidate entity types. However, such a method is unguided which may result in the generation of tokens that are inadequate for entity typing (e.g., {This, That, Him, Me, It} for "Wrigley Field" in S_1).

Alternatively, we can seek to extract hypernyms of the entity mention span of interest by inserting a short phrase that contains a "[MASK]" token (e.g., exploring the idea of Hearst Patterns [8]). This method conducts labeling in a more context-aware manner and greatly enriches the fine-grained types labeled for each mention.

This process, however, still has the potential to generate tokens unsuitable for typing (e.g., {Team, Thing} for "Wrigley Field" in S_1) or a mixture of rough and fine-grained types (e.g., {Location, Building, Stadium}). The difficulty cannot be resolved automatically due to the lack of hierarchical knowledge of the generated tokens/types.

With the emergence of large language model (LLM) (e.g., ChatGPT [22]), it is appealing to directly apply it to tackle these challenges. However, due to the lack of knowledge structures, an LLM may generate entity types at a too coarse or too fine level, or fail to commit to the right one from numerous fine-grained candidates.

In this study, we envision that an ontology structure, which provides a semantics-rich, hierarchical structure, may help select the best results generated by multiple PLM models. We propose a annotation-free, ontology-guided, fine-grained entity typing (FET) method, OntoType, that leverages an input ontology structure and the power of MLM prompting and Natural Language Inference (NLI). We first ensemble multiple Hearst patterns to perform MLM prompting, reducing the noise in the initial candidate type generation. Since the generated candidate labels for a given mention are likely a mixture of fine and coarse-grained labels, or tokens unsuitable for typing, we propose to automatically match the generated candidate labels to a coarse-grained type in our type ontology and then rank and select a coarse-grained type with a pre-trained entailment model under the local context. Based on the same principle of entailment score-based type selection, this type resolution process progresses deeper to finer levels, following the type ontology, until the finest possible label can be consolidated.

Example 2. For sentence S_1 in Ex. 1, candidate type generation (Step 1) ensembles prompting results of multiple Hearst patterns and generates a set of candidate labels: {Stadium, Venues, Location, Game} for "Wrigley Field" (Fig. 2). Using a given ontology structure (Fig. 1), this set of types is first resolved to the course-grained type "Location" via the assistance of a pre-trained entailment model and the local contextual information (Fig. 3). Note without the type structure, "Stadium" and "Venue" are rivals to "Location"; but with the structure, they become its supporters since both are fine-grained "Location". With the same principle, the type resolution proceeds deeper to finer-grained levels, along the type ontology, from "Location" to "Building" and further down to "Stadium" for "Wrigley Field", leading to the most accurate fine-grained type (Fig. 4).

Our contributions are summarized as follows.

(1) A fully annotation-free, ontology-guided, fine-grained typing method, OntoType, is proposed
(2) OntoType improves fine-grained entity typing (FET) by leveraging candidate labels generated and refined with three information sources: (i) pre-trained language models, (ii) a fine-grained type ontology, and (iii) head words
(3) Experiments on the Ontonotes, FIGER, and NYT datasets [6] using their associated ontological structures show that OntoType clearly outperforms existing zero-shot named entity typing methods as well as ChatGPT, and even rivals supervised methods. Our error analysis shows that refinement of ontology structures will further improve fine-grained entity typing.

2 RELATED WORK

Fine-grained entity typing benefits various downstream tasks and has received extensive attention in natural language research. Recent studies focus on different contexts from the phrase level [35] to considering specific entity mentions in the sentence or document level [3, 7, 15]. Entity typing has been generally studied under supervised learning settings with significant human annotated data [3, 4, 15, 36].

Minimally Annotated Fine-Grained Entity Typing Methods. Some recent studies (e.g., [4, 9, 13, 14, 16, 33, 39, 40]) leverage cross-encoders, pre-trained language models and prompting templates to obtain knowledge for entity mentions in given sentences. [4] improves ultra-fine entity typing with a BERT Masked Language Model (MLM). [13] instead improves ultra-fine entity typing by treating the task of predicting an entity type as an NLI task. Similar to OntoType, ChemNER [30] leverages a type ontology structure to guide fine-grained Chemistry entity recognition. NFETC-SSL [33] proposes a semi-supervised learning method with mixed label smoothing and pseudo labeling. MCCE [9] proposes to first prune the large type set and generate K candidates. Then, MCCE's novel model concurrently encodes and scores these K candidates as final type labels. DenoiseFET [14] utilizes pre-trained label embeddings to group the given set of type labels into semantic domains. A fine-tuned UFET model is utilized to predict initial labels. Finally, the semantic domains as guidance to infer missing labels and remove conflicting labels. These recent methods leverage human annotations and seek to augment SOTA models with prior knowledge acquired through head words, ontology structures and pre-trained embeddings. OntoType seeks to perform FET without the use of any human annotations by amalgamating multiple MLM prompts and NLI results to reduce noises in candidate type generation. Finally, we also utilize the fine-grained type ontology structure as guidance to progressively resolve candidate labels from coarse to fine under the local context.

Distant Supervision via Knowledge Bases. To handle difficulties to acquire human annotation, the zero-shot learning setting has been introduced for named entity typing [32]. Several studies [37, 41] address the problems by grounding the mentions with Knowledge Base (i.e. Wikipedia) entries from the assembled related

pages. These methods achieve good performance but still require significant human resources to construct effective knowledge bases which can be difficult to obtain for emerging domains.

Transfer Learning Methods. Other studies explore learning pre-trained semantic word-level embeddings from Knowledge Bases and seen types [18, 23] or extracting raw embeddings without auxiliary information and utilize end-to-end neural networks [38]. However, these methods still suffer from low accuracy and inefficiency in zero-shot settings. As a result, OntoType turns to the weak supervision by exploiting pre-trained language models (i.e. BERT [5]) as a knowledge base due to their substantial knowledge in language understanding learned by training on massive text collections.

3 METHODOLOGY

We propose OntoType, an annotation-free, ontology-guided, fine-grained entity typing method using pre-trained language models and a fine-grained ontology structure. Given an input sentence and a set of pre-identified mentions in the sentence, OntoType consists of the following steps: (1) generating a set of candidate labels for each input mention with both head word parsing and an ensemble of MLM prompting (Fig. 2); (2) resolving the coarse-grained types by matching and ranking the generated labels to the type ontology using an entailment model (Fig. 3); and (3) progressively refining the fine-grained types along the type ontology following the principle of entailment score-based type selection (Fig. 4). We utilize the inherent structure of the fine-grained type ontology and a pre-trained entailment model (RoBERTa model pre-trained on the MNLI dataset [17]) to guide our fine-grained entity typing.

3.1 Problem Definition

The input to our proposed OntoType framework is a text corpus D and a fine-grained type ontology O. In this study, we assume our input text corpus D includes a set of pre-identified entity mentions. An entity mention, e, is a token span in the text document that refers to a real-world entity. Given a sentence S and a parsed entity mention $e \in S$, the **fine-grained entity typing** (FET) task is to identify one or more types t from the label space T (provided in a structured ontology G) for the entity mention e.

As an example of our FET task, in the sentence S_1 of Ex. 1, the entity mention to be typed is e_1: "*Wrigley Field*". It should be labeled progressively deeper as "Location→Building→Stadium" as opposed to other labels like "Organization", "Person", or "School".

3.1.1 Fine-Grained Type Ontology Structure. The structure of the type ontology is fundamental to the OntoType algorithm. In this study, we utilize zero human effort to construct our type ontologies. We leverage the fine-grained type ontologies provided in the OntoNotes and FIGER datasets. Extending these ontologies to new text corpora can be achieved with minimal human efforts through taxonomy completion and refinement methods[1, 10, 11, 25]. Furthermore, existing automatic ontology constructions methods [28, 29, 31] build ontologies also based on "is-A" relations (e.g., a "Canada" is a "Country") which minimize the need for human-annotated ontologies even in emerging domains.

Definition 3.1 (Fine-Grained Type Ontology). OntoType's Fine-Grained Type Ontologies are structured as a strict tree imposing an "is-a" type hierarchy stemming from a "root" concept. The "root" concept's children consist of a handful of coarse-grained types including but not limited to: "Organization", "Person", or "Location". In addition, the ontology follows the following structural constraints: (1) Each type has a singular parent type; (2) each type (except for the leaf node) can have a number of children; and (3) each type is connected by a directional edge indicating a hypernym-hyponym type relationship between the parent and child nodes.

It is important to recognize that a hypernym-hyponym or "is-a" relationship is critical for guiding OntoType's final fine-grained type selection. Note that in the given OntoNotes type ontology (Fig. 1), City and Country are sibling types since they share the same parent type Location. Although "City" can be connected to "Country" by a "is-in" type relationship, OntoType's input ontologies organize these entity types as siblings due to their hypernym-hyponym relationship to "Location". By consolidating hypernyms of the entity mention, we can identify the high-level type in the ontology that accurately provides a partial label for the entity mention of interest. Thus, we approach our entity typing process in a hierarchical manner and refine from a coarse-grained level down to the most accurate fine-grained level for the mention of interest.

Please note that an ontology structure can be extended to include "is-in", "is-part-of" and "is-property-of" relationships. Fine-grained entity typing using such an extended structure is left to future work.

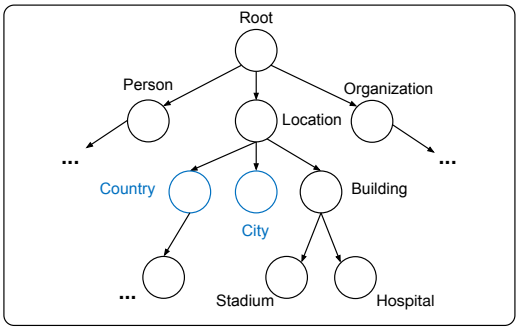

Figure 1: OntoNotes Type Ontology

3.2 Candidate Type Generation

To generate a set of candidate entity types for each input mention, we leverage two techniques: (1) head word parsing, and (2) ensemble of hypernyms derived from MLM prompting.

Head words and hypernyms can serve as powerful context-aware type indicators that can be leveraged as weak supervision sources for entity typing [3, 4]. We use them to select a set of candidate types to guide our fine-grained type refinement. We first generate and parse the head words for a given entity mention, and then generate candidate types with the use of ensembled MLM prompting[1]. The generated candidates are used as input to the following steps of ontology-guided type resolution.

[1]If there is no head word, our algorithm generates candidate labels using only MLM prompting.

3.2.1 Head Word Parsing. As discussed in [3], the input text often contains cues that explicitly match a mention to its type. These cues are often in the form of the mention's head word. Thus, given the pre-identified entity mentions in the input sentence, we first identify the head word of the input mention. We utilize the Stanford Dependency Parser [2] to extract the head word of the entity that we are interested in typing. Considering the sentence "*Governor Arnold Schwarzenegger gives a speech at Mission Serve's service project on Veterans Day 2010*", the entity mention "*Governor Arnold Schwarzenegger*" can be associated with a few types (e.g., actor, father, and governor). However, given the head word, one can easily consolidate it to the right one: "governor". Thus, OntoType leverages the head word of the input entity, if any, to select an initial context-sensitive type, which may guide the selection of the subsequent fine-grained type.

3.2.2 Ensembled MLM Prompting. While head words can provide strong type indicators, they do not always provide sufficient information to consolidate a high-level type. In some cases, head words (e.g., "Red Sox star") cannot directly provide accurate type information as they are not directly present in the input type ontology. Furthermore, head words are not always available in the input sentences. Thus, with the parsed entity mention span as input, we propose to leverage context-aware hypernyms as initial type candidates for the target mention. With an ensembled cloze prompting method, OntoType generates candidate types of the mention with masked language models and performs an initial high-level typing on the input mention. Specifically, we leverage the BERT model [5] and artificial Hearst patterns [8] to generate context-aware hypernyms that serve as candidate types for the mentions. We first modify the input sentence by inserting a Hearst pattern and [MASK] token into the sentence. Then we use the BERT to generate candidate types for the target mention under the local context. For example, in Fig. 2, we first insert Hearst patterns such as "and the other [MASK]" in the input sentence and then use the BERT MLM model to generate candidate types such as "Venue", "Team", and "Stadium".

We evaluated the quality of hypernyms generated with direct masked prompting and the 44 patterns proposed in [24] on the Ontonotes Development Dataset. When generating hypernyms, we expect high-quality candidates to be semantically equivalent to concepts contained in our input type ontology.

Unfortunately, we found that direct masked prompting resulted in tokens that were indefinite and unsuitable to serve as candidate types. Based on our observation, the four Hearst patterns in Table 1 provide the highest quality hypernyms under a simple direct matching to types in the OntoNotes ontology.

Nevertheless, based on the syntax and grammar of the sentence, hearst patterns can generate tokens that are unsuitable to serve as fine-grained entity types. For example, with a single prompt, the MLM can generate "Famous", "Actor", "Celebrity" and "Person" as the most probable words, where "Famous" is unsuitable to serve as a fine-grained entity type. To reduce the noises caused by the use of a single Hearst pattern, we ensemble n Hearst Patterns to consolidate the most commonly generated candidate types. For each pattern in the pattern list P, we collect the top k most probable tokens from the probability distribution predicted by the BERT MLM. Then, we aggregate the tokens and identify the set of candidates that have the largest overlap. We perform the voting ensemble as follows:

Masked Prompt Pattern	Prec	Rec	F1
[MASK]	11.8	5.2	7.2
[MASK] such as	53.3	72.4	61.4
such [MASK] as	47.9	68.7	56.5
and some other [MASK]	48.8	66.6	56.4
and the other [MASK]	47.6	68.3	56.1

Table 1: Performance of direct masked token prompting & Hearst patterns with simple type mapping on Ontonotes Dataset.

$$y = count_{(m)}\{H_1(x), H_2(x), ..., H_n(x)\} \quad (1)$$

where $H_n(x)$ is the candidate type generated by the n^{th} hearst pattern and $count_{(m)}$ is the function that selects all candidates generated at least m times. In our experiments, we take $m = \lfloor \frac{n}{2} \rfloor + 1$. We observe that the number of Hearst patterns employed is not sensitive as ensembling ensures the most confident candidates retain.

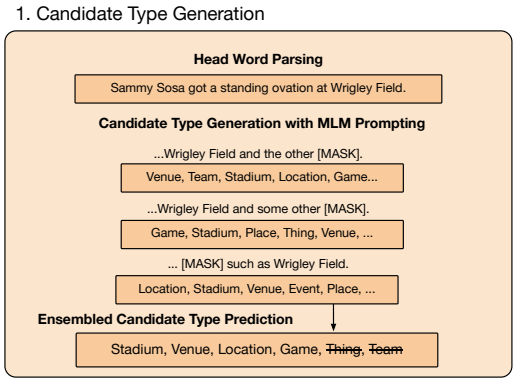

Figure 2: Candidate Type Generation

Example 3: As shown in Fig. 2, by ensembling the results from prompting with several Hearst patterns, the quality types for e_1 "Stadium, Venue, Location, Game" retain but the noisy types "Thing" and "Team" are removed.

3.3 High-Level Type Resolution

Given the generated candidate types and head words for each mention in the sentence, we seek to resolve the concrete type for this candidate type set at the high levels of the type ontology. Specifically, we first align the generated candidates to several high-level types in the type ontology, and then select the most accurate high-level types with a pre-trained entailment language model.

3.3.1 Candidate Type Alignment. Following the previous step of candidate type generation, we combine the generated candidates from both the parsed head words and the ensembled cloze prompting to form a candidate type set. These candidates are generally noisy and may not exist directly in our type ontology. However, we observe that the generated candidates will usually cluster closely around a high-level concept in the ontology. Thus, we perform our high-level type alignment with a cosine-similarity-based matching.

We use Word2Vec[2] [19] embeddings to construct our type embeddings for the cosine-similarity-based type alignment. We construct a verbalizer by selecting at least l semantically related words for each coarse type. For a high-level type of "Organization", we might include seed types such as "Corporation", "University", "Firm", "Business", and "Government" in its verbalizer. This verbalizer is then utilized to systematically map the MLM hypernym prediction to the most relevant type. In our experiments, we provide at least five seed types S for each type node c in the first level of the type ontology. Increasing the number of seed types increases the coverage and confidence of the verbalizer. Since the most commonly generated hypernyms for each concept are in the input ontologies, OntoType is not sensitive to the number of seed types collected. With the seed types, we construct a node embedding N by taking the average of word embeddings from both the first-level type and its corresponding seed types,

$$N = \frac{\sum_{s_i \in S} emb(s_i) + emb(c)}{|S| + 1}. \tag{2}$$

where $emb(\cdot)$ indicates the Word2Vec embeddings.

Then we rank each generated candidate type to a first-level type on our type ontology by calculating the cosine similarity between the embeddings of the generated candidate labels b and that of the first-level types T,

$$score(b, T) \overset{\text{rank}}{=} cosine(emb(b), emb(T)). \tag{3}$$

Finally, the first-level type that has the highest similarity is selected as the aligned high-level type.

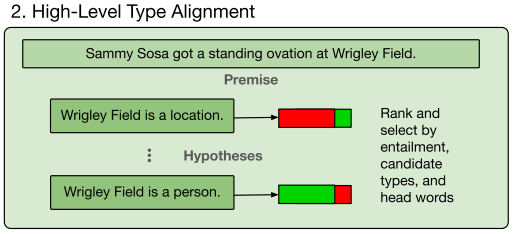

Figure 3: Candidate Type Generation

Example 4: In Figures 2 and 3, the consolidated candidate labels "Location", "Stadium", "Venue", and "Game" are closely related to the high-level type "Location" in our ontology. By performing the cosine-similarity-based matching, "Location" is identified as e_1's high-level type over "Organization" or "Person". Thus, we select the most similar high-level type given by the head word, generated candidate labels, and the entailment model.

3.3.2 High-Level Type Selection. After the previous step of candidate type alignment, we obtain several high-level types for each entity mention in the sentence. Given these types, we seek to select the most accurate high-level type for each entity mention under the local context. We observe that the task of selecting the most appropriate entity type can be viewed as a Natural Language Inference (NLI) task. Thus, we treat the input sentence as the premise in NLI and generate the hypothesis using a pre-defined declarative template. To resolve the type of the input mention, we use the template: "`In this sentence, [MENTION] is a [TYPE].`" We then rank each type in the first level of the ontology with the entailment score from a RoBERTa NLI model [17].

Example 5: In Fig. 3, we align a majority of the generated candidate type set to the Location seed types. The NLI model further ranks Location over Organization or Person. By utilizing the information in conjunction, e_1 is solidly aligned to the high-level type "Location".

3.4 Fine-Grained Type Resolution

Given the high-level types of the entity mentions, OntoType further leverages the ontology structure to progressively resolve the fine-grained label. Following the same principle of entailment-based type selection for the high-level types, we utilize the entailment model [17] to compute the entailment score, σ_{entail}. Then, OntoType can automatically select the most accurate fine-grained entity types along our type ontology. Specifically, we first examine the child types of the previously determined higher-level types and then select the child type with the highest ranked score as the fine-grained type.

In addition to the entailment model, we also utilize the candidate type set to refine our fine-grained type selection. If the parsed head word is present in our type ontology, the entailment scores of that parent and its child types are weighted higher through σ_{cand}. Similarly, if the generated candidate types are in our type ontology, the entailment scores of their parents and children are also weighted higher through σ_{cand}. Finally, we select the fine-grained type for each mention with the highest-ranked score.

We calculate the ranking score for the entities at the current level of the ontology as follows:

$$rank(type) = \sigma_{entail} + \sigma_{cand} \tag{4}$$

We first leverage our NLI pre-trained model [17] to find the entailment score σ_{entail} for each entity type. Then if a type in the candidate type set is a descendent to the entity type, we add a weight σ_{cand}. We repeat this entailment-based selection process recursively along the type ontology to select the best fine-grained type.

Definition (Entity Type Granularity Parameter θ): We assume there is a scalar of θ indicating how granular or specific the final selected entity type should be. The smaller θ is, the more granular entity types are consolidated as the final one.

Thus, if the child types do not have a sufficient gain of at least θ in ranking score over the parent type at a certain level, we stop the recursion and select the parent type as the final fine-grained type. We conduct a parameter study to explore the sensitivity of OntoType to the parameter θ in Section 4.4.1.

Example 6: In Fig. 4, we consider all descendent types of "Location" as potential fine-grained entity types for e_1. To begin, OntoType generates the hypotheses and ranks all child types of "Location". With the resultant entailment score rankings of these sibling types, we consolidate and select "Building" as the highest-ranked type. Then, a similar process is done at a deeper level of the ontology to select the final type "Stadium".

[2]https://code.google.com/archive/p/word2vec/

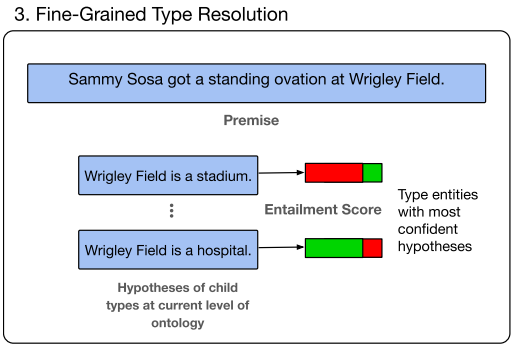

Figure 4: Fine-Grained Type Refinement

Datasets	Ontonotes	FIGER	NYT
# of Types	89	113	125
# of Documents	300k	3.1M	295k
# of Entity Mentions	242K	2.7M	1.18M
# of Train Mentions	223K	2.69M	701K
# of Test Mentions	8,963	563	1,010

Table 2: Dataset Statistics

4 EXPERIMENTS

4.1 Datasets

We compare the performance of OntoType and several baseline methods on three benchmark FET datasets: NYT, Ontonotes [6] and FIGER [15]. The basic statistics of the datasets are shown in Table 2. For the OntoNotes and FIGER datasets, we leverage the included type ontologies while the NYT dataset leverages the input FIGER ontology. All NER test sets are annotated using the ontologies as a set of type labels. Thus, each entity mention is labeled with a fine-grained label represented as a path within the ontology. The ontologies have a maximum depth of three and contain four to six high-level types (e.g., Location, Person, and Organization).

4.2 Baselines

OntoType is a FET method that does not require human annotation as supervision. We compare OntoType with four weakly supervised and nine zero-shot FET methods utilizing the evaluation metrics detailed in Appendix A.

For each baseline method, we utilize the default parameters as detailed in their studies. Furthermore, we conduct all experiments on a single NVIDIA RTX A6000 GPU. Finally, we evaluate generative LLMs on FET using Gemma 2b model[26], Llama 2 7b model[27] and OpenAI "gpt-3.5-turbo"[22]. OntoType leverages a pre-trained BERT [5] (BERT-base, uncased) and pre-trained RoBERTa fine-tuned on the MNLI dataset [17] available in the HuggingFace Library. In addition, OntoType utilizes Word2Vec[3] [19] embeddings to construct our type embeddings. Finally, we conduct parameter and ablation studies listed in Sections 4.4.1 and 4.4.2 respectively.

Weak Supervision with Human Annotations.

UFET [3]: This weakly supervised baseline is a model that predicts open types and is trained using a multitask objective that pools head-word supervision with supervision from entity linking.

[3]https://code.google.com/archive/p/word2vec/

BERT MLMET [4]: This weakly supervised baseline fine-tunes a BERT-based model using human annotations, supervision from head words and mention hypernyms generated from Hearst patterns.

LITE [13]: This weakly supervised baseline is a fine-tuned MNLI model leveraged to rank ultra-fine entity types.

NFETC-SSL [34]: This weakly supervised baseline proposes a semi-supervised learning method with mixed label smoothing and pseudo labeling for fine-grained entity typing.

Distant Supervision from Knowledge Bases.

ZOE [41]: This zero-shot baseline leverages a type taxonomy defined as Boolean functions of Freebase types and grounds a given entity mention to the type-compatible Wikipedia entries to infer the target mention's types.

DZET [20]: This zero-shot baseline utilizes the type description available from Wikipedia to build a distributed semantic representation of the types and aligns the target entity mention type representations onto the known types.

Transfer Learning Methods.

OTyper [37]: This zero-shot baseline is a neural network trained on a limited training dataset. The model is evaluated on the Open Named Entity Typing task, which is FET where the set of target types is not unknown.

MZET [38]: This zero-shot baseline leverages the semantic meaning and the hierarchical structure into the type representation. The method leverages the knowledge from seen types to label the zero-shot types through the use of a memory component.

ChatGPT [22]: This zero-shot, annotation-free baseline leverages a generative large language model as a knowledge source for entity typing. We evaluate ChatGPT on three different prompts to mitigate possible prompt construction-based errors.

Prompt1: *"Return the fine-grained entity types of the given entity mentioned in the sentences below. Be concise and ONLY utilize the types from this list of possible entity types. [entity types] {entity types} [sentence] {sentence} [entity mention] {entity mention}".*

Prompt2: *"Select a single label from the following list that best serves as a hyponym for the entity mention. [labels] {entity types} [sentence] {sentence} [entity mention] {entity mention}"*

Prompt3: *"Select the single most fine-grained entity type from the following list for the given entity mention. [entity types] {entity types} [sentence] {sentence} [entity mention] {entity mention}"*

We discuss the limitations of leveraging ChatGPT as a fine-grained entity typing method in Section 5.3.

Gemma [26]: This zero-shot, annotation-free baseline leverages a generative large language model as a knowledge source for entity typing. We utilize Prompt1 for evaluation on the FET task.

LLaMA 2 [27]: This zero-shot, annotation-free baseline leverages a generative large language model as a knowledge source for entity typing. We utilize Prompt1 for evaluation on the FET task.

4.3 Main Results

Table 3 shows our results on the test set of NYT, FIGER and Ontonotes, especially comparing with the notable exising method ZOE. On the NYT dataset, OntoType achieves the best zero-shot performance on this dataset. It achieves 5.9 absolute Ma-F1 improvement over the state-of-the-art zero-shot fine-grained entity typing method

Settings	Model	NYT			FIGER			Ontonotes		
		Acc	Mi-F1	Ma-F1	Acc	Mi-F1	Ma-F1	Acc	Mi-F1	Ma-F1
Weak Supervision with Human Annotations	UFET [3]	-	-	-	-	-	-	59.5	71.8	76.8
	BERT-MLMET [4]	-	-	-	-	-	-	67.44	80.35	85.44
	LITE [13]	-	-	-	66.2	74.7	80.1	68.2	81.4	86.6
	NFETC-SSL [34]	-	-	-	71.2	80.2	81.9	64.4	74.3	79.7
Distant Supervision via KBs	DZET [20]	27.3	53.1	51.6	28.5	56.0	55.1	23.1	28.1	27.6
	ZOE [41]	62.1	73.7	76.9	**58.8**	**71.3**	74.8	50.7	60.8	66.9
Transfer Learning	OTyper [37]	46.4	65.7	67.3	47.2	67.2	69.1	31.8	36.0	39.1
	MZET [38]	30.7	58.2	56.7	31.9	57.9	55.5	33.7	43.7	42.3
Annotation-Free	ChatGPT Prompt 1[22]	47.3	59.1	54.3	51.7	65.3	58.3	27.7	37.5	32.6
	ChatGPT Prompt 2 [22]	45.1	64.0	61.9	52.3	67.8	61.4	31.3	41.3	35.9
	ChatGPT Prompt 3 [22]	44.8	56.9	52.1	49.9	61.1	55.8	24.7	33.8	29.4
	Gemma [26]	44.8	56.9	52.1	49.9	61.1	55.8	24.7	33.8	29.4
	LLaMA 2 [27]	43.2	55.6	51.7	48.5	59.7	54.4	24.1	33.2	28.8
	OntoType + Original Ontology	**69.6**	**78.4**	**82.8**	49.1	67.4	**75.1**	**65.7**	**73.4**	**81.5**
	OntoType + Modified Ontology	-	-	-	51.1	68.9	**77.2**	-	-	-

Table 3: Results (%) on Three Test Sets (Some slots in the benchmarked methods marked "-" due to no fully annotated training data).

ZOE. While on the Ontonotes dataset, OntoType achieves the best zero-shot performance on this dataset while trailing the best supervised method by 3.94 Ma-F1 points. OntoType achieves 14.6 absolute Ma-F1 improvement over the state-of-the-art zero-shot fine-grained entity typing method ZOE.

A performance comparison between OntoType and ZOE demonstrates the benefit of leveraging the knowledge embedded in pre-trained language models as a form of minimal supervision to identify entity mention types. ZOE leverages a type ontology and maps a given mention to type-compatible Wikipedia Entries. As a result, ZOE relies on surface-level information from the mention string. OntoType ensembles contextual information from various PLMs to consolidate the final entity type. Given a sentence: "*The biggest cause for concern for McGuff is the bruised hamstring Regina Rogers suffered against (Utah) last Saturday*", ZOE incorrectly utilizes the surface string to label "Utah" as a location. With PLMs, OntoType recognizes "Team" or "Opponent" as candidates and finally consolidates to "Sports Team".

MZET	**US President Joe Biden \Person\Politician** was one of many foreign leaders to speak with President Zelensky, and he "pledged to continue providing **Ukraine \Location** with the support needed to defend itself, including advanced air defence systems", **the White House \Location\Building** said.
ZOE	**US President Joe Biden \Person\Politician** was one of many foreign leaders to speak with President Zelensky, and he "pledged to continue providing **Ukraine \Location\Country** with the support needed to defend itself, including advanced air defence systems", the **White House \Location\Building** said.
OntoType	**US President Joe Biden \Person\Politician\President** was one of many foreign leaders to speak with President Zelensky, and he "pledged to continue providing **Ukraine \Location\Country** with the support needed to defend itself, including advanced air defence systems", **the White House \Organization\Government** said.

Table 4: Type predictions (in color) on three entity mentions (in bold): ONTOTYPE vs. two other Zero-Shot FET methods.

On the FIGER dataset, OntoType achieves 0.3 absolute Macro-F1 improvement over state-of-the-art zero-shot fine-grained entity typing method ZOE [41]. However, our method trails ZOE in strict accuracy and Micro-F1. In the FIGER dataset, predictions are made based on both the surface-level information and the contextual information in the sentence. The major advantage of OntoType is to accurately capture the contextual information to provide a fine-grained entity type. However, OntoType does not involve a mechanism to capture the surface-level information of an entity mention. We discuss this issue further in our error analysis (Section 5.2).

4.4 Ablation and Parameter Studies

4.4.1 Study of Sensitivity to Parameters. A potential concern with the experimental setup can be overtly high sensitivity of OntoType to the Entity Type Specificity parameter θ. For all experiments in Table 3, we leverage the same θ value of 0.3. Additionally, from the plot in Figure 5, we clearly see that F1 scores are not drastically sensitive to theta with standard deviations of 0.3631 and 0.6262 on FIGER and OntoNotes respectively.

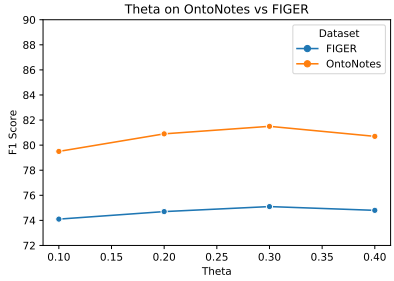

Figure 5: Parameter study: θ on OntoNotes and FIGER

4.4.2 OntoType Module Ablation Study. We also include the prediction results of our ablation models to demonstrate how the NLI module contributes to the final type assignment. We utilize a simple type mapping to evaluate our ensembled BERT Cloze Prompting module. Figure 6 shows the results of ablation studies on the test set of the Ontonotes dataset. We compared the OntoType full model with various ablations and variations. We find that direct prompting performs the worst whereas including parsed head words further improves the recall. Including the NLI module significantly improves the precision of the framework as we are able to leverage the type hierarchy to consolidate the fine-grained type that best represents the initial candidates. Furthermore, we include Table 6 to examine alternative embedding approaches for the high-level type assignment. When performing high-level type assignment, we aim

to leverage the surface-level information of the candidate type set to select the high-level type that is most conceptually similar. As a result, we find the one with word2vec embeddings performs better than BERT[5] and RoBERTa[17] which incorporate the semantic context of a word in the sentence.

4.4.3 Efficiency & Scalability Study. Finally, we conduct an efficiency and scalability study to gauge the impact of larger scale data sets on our method and include the results in Table 5.

Number of Mentions	Time to Predict Types
1000	1.75 min
25000	57.1 min
100000	4.3 hr

Table 5: Results of the efficiency & scalability study.

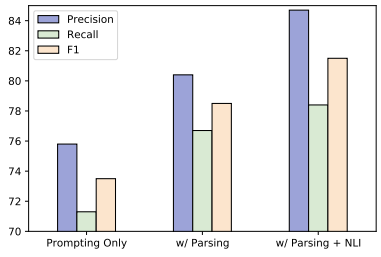

Figure 6: OntoType ablation study on OntoNotes: Results (%)

Model	Prec	Rec	Ma-F1
OntoType$_{BERT}$	82.3	77.1	79.6
OntoType$_{RoBERTa}$	81.9	76.9	79.4
OntoType$_{Word2Vec}$	**84.7**	**78.4**	**81.5**

Table 6: High-level type alignment ablation study: Results (%)

4.5 Comparative Case Study

Table 4 presents a sentence from a recent news article with tagged mentions and predicted entity types. We find that methods like MZET sometimes predict incompatible types due to incorrect or misleading surface information. For example, when typing "The White House", MZET and ZOE leverage the surface-level information from large KBs to identify the mention as a location. However, given the local context, the White House clearly refers to the U.S. government. When considering the mention "US President Joe Biden", our method utilizes the type information from the candidate type set (Official, Leader, Politician, Individual) to explore the "Person" branch within the ontology, and then it searches the "Politician" branch to select the best fine-grained context-aware type "President" for our given entity mention. Thus, with the assistance of the PLMs, we can incorporate contextual information to derive more context-aware type labels.

5 DISCUSSION

5.1 Modified Fine-Grained Type Ontology

The structure of the fine-grained type ontology is important to the performance of OntoType. The input ontology must be built using hypernym-type relationships where each parent type is a generalization of the child type. The provided FIGER ontology contains logical inconsistencies in how various types of relations are organized.

For example, the FIGER ontology considers the parent of the "Road" type to be "Transportation" rather than "Location". This logical inconsistency leads to erroneous typing from our method.

Table 3 includes our results on the test set of FIGER with and without a modified type ontology.

The modification of an ontology is done by reorganizing the types to leverage hypernym-hyponym type relationships. That is, in the revised type ontology, each parent type is a generalization of its child(ren) type(s). The included FIGER type ontology considers the parent type of the "Road" type to be "Other\Transportation" and the parent type of "Building" to be "Other". While "Other" can be considered a valid generalization, it is extremely broad and the coarse-grained type of "Location" serves as a stronger parent for both fine-grained types. Thus, we modify the included ontology by reorganizing the fine-grained types under parent types that have stronger generalizations (e.g., Building & Road under Location rather than Other). OntoType achieves 2.1 absolute Ma-F1 improvement given this modified type ontology. Thus, we find that OntoType can be further improved with correct type ontologies to leverage the inherent hierarchical relationships between coarse and fine-grained types. We provide insights into specific reasons for the mistakes made by the OntoType framework. For our analysis, we collect some of the erroneous decisions in the Ontonotes and FIGER test sets. We highlight the Gold Type for each mention in blue.

5.2 Error Analysis: OntoType

OntoType, though having high performance, still generates non-trivial errors. We analyze the reasons behind the errors on the Ontonotes and FIGER test sets and categorize them into three types. Note the Gold Type for each mention is highlighted in blue (and *Italian font*).

Error Type 1: Lack of Knowledge Base

Sentence E1: He was a caseworker in **Minnesota** *[\Location\Region]* but left the job because he found himself perpetually sick from the environments in which he worked.

In E1, OntoType incorrectly types Minnesota as a Country rather than the gold type of Region. In context, we should consider State a more reasonable type to predict. With a simple KB-matching method, we would be able to capture the surface-level information to type it as a State/Province. OntoType relies on neither human annotations nor a knowledge base. Obviously, introducing a knowledge base and the KB-matching mechanism will further improve its performance.

Error Type 2: Incomplete Fine-Grained Type Ontology

Sentence E2: Valley Federal Savings & Loan Association said Imperial Corp. of America withdrew from regulators its application to

buy **five Valley Federal branches** *[\Location\Structure]*, leaving the transaction in limbo.

In E2, OntoType generates \Other. Even when OntoType is able to generate high-quality candidate labels, it can sometimes fail to align to the correct entity type due to an incomplete type set. In this example, our Candidate Type set generated by prompting consists of Asset, Property, Facility, Bank, and Branch. Clearly, the best fine-grained type should be Asset or Property (rather than the Gold Type: Location or Structure). Since the provided OntoNotes ontology does not include such fine-grained types, OntoType is unable to generate the correct answer. Clearly, a refined ontology will further improve its performance.

Error Type 3: Incapability to Type Nested Entities

Sentence E3: The 33-year-old **Billings** *[\Location\City]* native enlisted as a military veterinarian.

In E3, OntoType identifies Billings as \Person. This mistake can be caused by confusing the type of the whole entity "Billings native" with the type of the nested entity "Billings". PLM is good at generating candidate types for the whole entity based on its contextual structure, whereas the knowledge about a nested entity like "Billings" (in front of head word "native") can be more easily derived from a knowledge-base or from some nested entity type patterns. We will leave the issue of typing nested entities to future work.

5.3 Error Analysis: LLM on Fine-grained Typing

Recent developments of large language models (LLMs) (e.g., ChatGPT and GPT-4 [22]) lead to enhanced capability at generating high-quality responses to prompts based on the knowledge learned from their extensive training corpora. Clearly, LLMs can provide stronger *context-aware* knowledge for OntoType's candidate type generation. However, without an ontological structure as guidance, an LLM (e.g. ChaptGPT) may still generate many errors, leading to lower performance than our method, as shown in Table 3. Here we conduct error analysis for types generated by ChatGPT through the following prompt: *"Return the fine-grained entity types of the given entity mentioned in the sentences below. Be concise and ONLY utilize the types from this list of possible entity types. [entity types] {entity types} [sentence] {sentence} [entity mention] {entity mention}"*.

LLM Error Type 1: Incomplete Entity Types

LLM-E1: Given a sentence S_2: "*The ceremony will take place Feb. 16–20.*" and a parsed entity mention span "*Feb. 16–20*", we prompted ChatGPT to generate candidate types for the masked entity mention span, which generates: "Later", "There", "Outside" and "Indoors". While these tokens complete the sentence accurately, they cannot serve as precise hypernyms or fine-grained types for the entity mention span of interest. With Hearst patterns, we can elicit more accurate hypernyms like "Time", "Date" and "Event". However, we are still unable to derive the most accurate fine-grained label of "Period" or "Interval". In contrast, OntoType utilizes a top-down approach guided by a fine-grained type ontology to refine and finally select the most conclusive entity type.

LLM Error Type 2: Incorrect Entity Types

LLM-E2: Given a sentence S_3: "*It will be the first time the Falcon Heavy has conducted a launch for the U.S. military's secretive X-37B spaceplane project*" and a parsed entity mention span "*Falcon Heavy*", we can again prompt ChatGPT to generate candidate types for the masked entity mention span. With our prompt, ChatGPT generates: "Company", "Organization", "Team", "Government", and "Agency". With Hearst patterns, we can extract more accurate candidates like "Vehicle", "Carrier" and "Launcher". Furthermore, ChatGPT is unable to derive the most accurate fine-grained label of "Spacecraft", "Shuttle" or "Rocket". While some generated tokens accurately type the mention span, ChatGPT still requires a structured mechanism to mitigate hallucinations and select the most fine-grained entity type.

Overall, to achieve high quality fine-grained typing, we believe an LLM should be assisted with structured knowledge (e.g., a fine-grained ontological structure), which is an interesting direction for future research.

6 CONCLUSIONS

We propose OntoType, which leverages the weak supervision setting of pre-trained language model prompting. We use head words and MLM cloze prompting for fine-grained candidate label generation. Then we automatically match the generated fine-grained types to our type ontology with an inference method to select the most appropriate fine-grained types under the local context. Extensive experiments on real-world datasets show that OntoType is highly effective and substantially outperforms the state-of-the-art zero-shot FET methods. In the future, we plan to further refine and enrich a type ontology which will enable us to incorporate more type information for even better performance. Furthermore, OntoType lacks the capability to address nested entities (e.g., *Denver* Native). We plan to resolve this in future work by incorporating boundary knowledge to extract accurate and complete type information.

ACKNOWLEDGEMENT

Research was supported in part by the Molecule Maker Lab Institute: An AI Research Institutes program supported by NSF under Award No. 2019897, US DARPA KAIROS Program No. FA8750-19-2-1004, INCAS Program No. HR001121C0165, National Science Foundation IIS-19-56151, and the Institute for Geospatial Understanding through an Integrative Discovery Environment (I-GUIDE) by NSF under Award No. 2118329. Any opinions, findings, and conclusions or recommendations expressed herein are those of the authors and do not necessarily represent the views, either expressed or implied, of National Science Foundation, DARPA or the U.S. Government. The views and conclusions contained in this paper are those of the authors and should not be interpreted as representing any funding agencies.

REFERENCES

[1] Ines Arous, Ljiljana Dolamic, and Philippe Cudré-Mauroux. 2023. TaxoComplete: Self-Supervised Taxonomy Completion Leveraging Position-Enhanced Semantic Matching. *Proceedings of the ACM Web Conference 2023* (2023). https://api.semanticscholar.org/CorpusID:258333923

[2] Danqi Chen and Christopher Manning. 2014. A Fast and Accurate Dependency Parser using Neural Networks. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics, Doha, Qatar, 740–750. https://doi.org/10.3115/v1/D14-1082

[3] Eunsol Choi, Omer Levy, Yejin Choi, and Luke Zettlemoyer. 2018. Ultra-Fine Entity Typing. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, Melbourne, Australia, 87–96. https://doi.org/10.18653/v1/P18-1009

[4] Hongliang Dai, Yangqiu Song, and Haixun Wang. 2021. Ultra-Fine Entity Typing with Weak Supervision from a Masked Language Model. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*. Association for Computational Linguistics, Online, 1790–1799. https://doi.org/10.18653/v1/2021.acl-long.141

[5] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*. Association for Computational Linguistics, Minneapolis, Minnesota, 4171–4186. https://doi.org/10.18653/v1/N19-1423

[6] Daniel Gillick, Nevena Lazic, Kuzman Ganchev, Jesse Kirchner, and David Huynh. 2014. Context-Dependent Fine-Grained Entity Type Tagging. *ArXiv* abs/1412.1820 (2014).

[7] Nitish Gupta, Sameer Singh, and Dan Roth. 2017. Entity Linking via Joint Encoding of Types, Descriptions, and Context. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Copenhagen, Denmark, 2681–2690. https://doi.org/10.18653/v1/D17-1284

[8] Marti A. Hearst. 1992. Automatic Acquisition of Hyponyms from Large Text Corpora. In *COLING 1992 Volume 2: The 14th International Conference on Computational Linguistics*. https://aclanthology.org/C92-2082

[9] Chengyue Jiang, Wenyang Hui, Yong Jiang, Xiaobin Wang, Pengjun Xie, and Kewei Tu. 2023. Recall, Expand, and Multi-Candidate Cross-Encode: Fast and Accurate Ultra-Fine Entity Typing. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, Toronto, Canada, 11597–11609. https://doi.org/10.18653/v1/2023.acl-long.648

[10] Minhao Jiang, Xiangchen Song, Jieyu Zhang, and Jiawei Han. 2022. TaxoEnrich: Self-Supervised Taxonomy Completion via Structure-Semantic Representations. *Proceedings of the ACM Web Conference 2022* (2022). https://api.semanticscholar.org/CorpusID:246706087

[11] Song Jiang, Qiyue Yao, Qifan Wang, and Yizhou Sun. 2023. A Single Vector Is Not Enough: Taxonomy Expansion via Box Embeddings. *Proceedings of the ACM Web Conference 2023* (2023). https://api.semanticscholar.org/CorpusID:258333891

[12] Mitchell Koch, John Gilmer, Stephen Soderland, and Daniel S Weld. 2014. Type-aware distantly supervised relation extraction with linked arguments. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. 1891–1901.

[13] Bangzheng Li, Wenpeng Yin, and Muhao Chen. 2022. Ultra-fine Entity Typing with Indirect Supervision from Natural Language Inference. *Transactions of the Association for Computational Linguistics* 10 (2022), 607–622. https://doi.org/10.1162/tacl_a_00479

[14] N. Li, Zied Bouraoui, and Steven Schockaert. 2023. Ultra-Fine Entity Typing with Prior Knowledge about Labels: A Simple Clustering Based Strategy. *ArXiv* abs/2305.12802 (2023). https://api.semanticscholar.org/CorpusID:258832899

[15] Xiao Ling and Daniel S. Weld. 2012. Fine-Grained Entity Recognition. In *Proceedings of the Twenty-Sixth AAAI Conference on Artificial Intelligence* (Toronto, Ontario, Canada) *(AAAI'12)*. AAAI Press, 94–100.

[16] Pengfei Liu, Weizhe Yuan, Jinlan Fu, Zhengbao Jiang, Hiroaki Hayashi, and Graham Neubig. 2021. Pre-train, Prompt, and Predict: A Systematic Survey of Prompting Methods in Natural Language Processing. *CoRR* abs/2107.13586 (2021). arXiv:2107.13586 https://arxiv.org/abs/2107.13586

[17] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. RoBERTa: A Robustly Optimized BERT Pretraining Approach. https://doi.org/10.48550/ARXIV.1907.11692

[18] Yukun Ma, Erik Cambria, and Sa Gao. 2016. Label Embedding for Zero-shot Fine-grained Named Entity Typing. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*. The COLING 2016 Organizing Committee, Osaka, Japan, 171–180. https://aclanthology.org/C16-1017

[19] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed Representations of Words and Phrases and their Compositionality. In *Advances in Neural Information Processing Systems*, C.J. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K.Q. Weinberger (Eds.), Vol. 26. Curran Associates, Inc. https://proceedings.neurips.cc/paper/2013/file/9aa42b31882ec039965f3c4923ce901b-Paper.pdf

[20] Rasha Obeidat, Xiaoli Fern, Hamed Shahbazi, and Prasad Tadepalli. 2019. Description-Based Zero-shot Fine-Grained Entity Typing. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*. Association for Computational Linguistics, Minneapolis, Minnesota, 807–814. https://doi.org/10.18653/v1/N19-1087

[21] Yasumasa Onoe and Greg Durrett. 2020. Interpretable Entity Representations through Large-Scale Typing. In *Findings of the Association for Computational Linguistics: EMNLP 2020*. 612–624.

[22] OpenAI. 2023. GPT-4 Technical Report. arXiv:2303.08774 [cs.CL]

[23] Xiang Ren, Wenqi He, Meng Qu, Lifu Huang, Heng Ji, and Jiawei Han. 2016. AFET: Automatic Fine-Grained Entity Typing by Hierarchical Partial-Label Embedding. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Austin, Texas, 1369–1378. https://doi.org/10.18653/v1/D16-1144

[24] Julian Seitner, Christian Bizer, Kai Eckert, Stefano Faralli, Robert Meusel, Heiko Paulheim, and Simone Paolo Ponzetto. 2016. A Large DataBase of Hypernymy Relations Extracted from the Web.. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC'16)*. European Language Resources Association (ELRA), Portorož, Slovenia, 360–367. https://aclanthology.org/L16-1056

[25] Jiaming Shen, Zhihong Shen, Chenyan Xiong, Chi Wang, Kuansan Wang, and Jiawei Han. 2020. TaxoExpan: Self-supervised Taxonomy Expansion with Position-Enhanced Graph Neural Network. *Proceedings of The Web Conference 2020* (2020). https://api.semanticscholar.org/CorpusID:210921213

[26] Gemma Team, Thomas Mesnard, Cassidy Hardin, Robert Dadashi, Surya Bhupatiraju, Shreya Pathak, Laurent Sifre, Morgane Rivière, Mihir Sanjay Kale, Juliette Love, Pouya Tafti, Léonard Hussenot, Pier Giuseppe Sessa, Aakanksha Chowdhery, Adam Roberts, Aditya Barua, Alex Botev, Alex Castro-Ros, Ambrose Slone, Amélie Héliou, Andrea Tacchetti, Anna Bulanova, Antonia Paterson, Beth Tsai, Bobak Shahriari, Charline Le Lan, Christopher A. Choquette-Choo, Clément Crepy, Daniel Cer, Daphne Ippolito, David Reid, Elena Buchatskaya, Eric Ni, Eric Noland, Geng Yan, George Tucker, George-Christian Muraru, Grigory Rozhdestvenskiy, Henryk Michalewski, Ian Tenney, Ivan Grishchenko, Jacob Austin, James Keeling, Jane Labanowski, Jean-Baptiste Lespiau, Jeff Stanway, Jenny Brennan, Jeremy Chen, Johan Ferret, Justin Chiu, Justin Mao-Jones, Katherine Lee, Kathy Yu, Katie Millican, Lars Lowe Sjoesund, Lisa Lee, Lucas Dixon, Machel Reid, Maciej Mikuła, Mateo Wirth, Michael Sharman, Nikolai Chinaev, Nithum Thain, Olivier Bachem, Oscar Chang, Oscar Wahltinez, Paige Bailey, Paul Michel, Petko Yotov, Rahma Chaabouni, Ramona Comanescu, Reena Jana, Rohan Anil, Ross McIlroy, Ruibo Liu, Ryan Mullins, Samuel L Smith, Sebastian Borgeaud, Sertan Girgin, Sholto Douglas, Shree Pandya, Siamak Shakeri, Soham De, Ted Klimenko, Tom Hennigan, Vlad Feinberg, Wojciech Stokowiec, Yu hui Chen, Zafarali Ahmed, Zhitao Gong, Tris Warkentin, Ludovic Peran, Minh Giang, Clément Farabet, Oriol Vinyals, Jeff Dean, Koray Kavukcuoglu, Demis Hassabis, Zoubin Ghahramani, Douglas Eck, Joelle Barral, Fernando Pereira, Eli Collins, Armand Joulin, Noah Fiedel, Evan Senter, Alek Andreev, and Kathleen Kenealy. 2024. Gemma: Open Models Based on Gemini Research and Technology. arXiv:2403.08295 [cs.CL]

[27] Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Dan Bikel, Lukas Blecher, Cristian Canton Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, Brian Fuller, Cynthia Gao, Vedanuj Goswami, Naman Goyal, Anthony Hartshorn, Saghar Hosseini, Rui Hou, Hakan Inan, Marcin Kardas, Viktor Kerkez, Madian Khabsa, Isabel Kloumann, Artem Korenev, Punit Singh Koura, Marie-Anne Lachaux, Thibaut Lavril, Jenya Lee, Diana Liskovich, Yinghai Lu, Yuning Mao, Xavier Martinet, Todor Mihaylov, Pushkar Mishra, Igor Molybog, Yixin Nie, Andrew Poulton, Jeremy Reizenstein, Rashi Rungta, Kalyan Saladi, Alan Schelten, Ruan Silva, Eric Michael Smith, Ranjan Subramanian, Xiaoqing Ellen Tan, Binh Tang, Ross Taylor, Adina Williams, Jian Xiang Kuan, Puxin Xu, Zheng Yan, Iliyan Zarov, Yuchen Zhang, Angela Fan, Melanie Kambadur, Sharan Narang, Aurelien Rodriguez, Robert Stojnic, Sergey Edunov, and Thomas Scialom. 2023. Llama 2: Open Foundation and Fine-Tuned Chat Models. arXiv:2307.09288 [cs.CL]

[28] Paola Velardi, Stefano Faralli, and Roberto Navigli. 2013. Ontolearn reloaded: A graph-based algorithm for taxonomy induction. *Computational Linguistics* 39, 3 (2013), 665–707.

[29] Chengyu Wang, Xiaofeng He, and Aoying Zhou. 2017. A short survey on taxonomy learning from text corpora: Issues, resources and recent advances. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*. 1190–1203.

[30] Xuan Wang, Vivian Hu, Xiangchen Song, Shweta Garg, Jinfeng Xiao, and Jiawei Han. 2021. CHEMNER: Fine-Grained Chemistry Named Entity Recognition with Ontology-Guided Distant Supervision. In *Proceedings of the 2021 Conference on*

Empirical Methods in Natural Language Processing. 5227–5240.

[31] Wentao Wu, Hongsong Li, Haixun Wang, and Kenny Q Zhu. 2012. Probase: A probabilistic taxonomy for text understanding. In *Proceedings of the 2012 ACM SIGMOD international conference on management of data*. 481–492.

[32] Congying Xia, Chenwei Zhang, Xiaohui Yan, Yi Chang, and Philip Yu. 2018. Zero-shot User Intent Detection via Capsule Neural Networks. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Brussels, Belgium, 3090–3099. https://doi.org/10.18653/v1/D18-1348

[33] Bo Xu, Zhengqi Zhang, Ming Du, Hongya Wang, Hui Song, and Yanghua Xiao. 2023. Semi-supervised Learning for Fine-Grained Entity Typing with Mixed Label Smoothing and Pseudo Labeling. In *International Conference on Database Systems for Advanced Applications*. https://api.semanticscholar.org/CorpusID:258240743

[34] Bo Xu, Zhengqi Zhang, Ming Du, Hongya Wang, Hui Song, and Yanghua Xiao. 2023. Semi-Supervised Learning For Fine-Grained Entity Typing With Mixed Label Smoothing And Pseudo Labeling. In *Database Systems for Advanced Applications: 28th International Conference, DASFAA 2023, Tianjin, China, April 17–20, 2023, Proceedings, Part III* (Tianjin, China). Springer-Verlag, Berlin, Heidelberg, 727–736. https://doi.org/10.1007/978-3-031-30675-4_53

[35] Limin Yao, Sebastian Riedel, and Andrew McCallum. 2013. Universal Schema for Entity Type Prediction *(AKBC '13)*. Association for Computing Machinery, New York, NY, USA. https://doi.org/10.1145/2509558.2509572

[36] Mohamed Amir Yosef, Sandro Bauer, Johannes Hoffart, Marc Spaniol, and Gerhard Weikum. 2012. HYENA: Hierarchical Type Classification for Entity Names. In *Proceedings of COLING 2012: Posters*. The COLING 2012 Organizing Committee, Mumbai, India, 1361–1370. https://aclanthology.org/C12-2133

[37] Zheng Yuan and Doug Downey. 2018. OTyper: A Neural Architecture for Open Named Entity Typing *(AAAI'18/IAAI'18/EAAI'18)*. AAAI Press, Article 741, 8 pages.

[38] Tao Zhang, Congying Xia, Chun-Ta Lu, and Philip Yu. 2020. MZET: Memory Augmented Zero-Shot Fine-grained Named Entity Typing. In *Proceedings of the 28th International Conference on Computational Linguistics*. International Committee on Computational Linguistics, Barcelona, Spain (Online), 77–87. https://doi.org/10.18653/v1/2020.coling-main.7

[39] Yue Zhang, Hongliang Fei, and Ping Li. 2023. Denoising Enhanced Distantly Supervised Ultrafine Entity Typing. In *Findings of the Association for Computational Linguistics: ACL 2023*. Association for Computational Linguistics, Toronto, Canada, 9880–9892. https://doi.org/10.18653/v1/2023.findings-acl.626

[40] Yunyi Zhang, Jiaming Shen, Jingbo Shang, and Jiawei Han. 2020. Empower Entity Set Expansion via Language Model Probing. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, Online, 8151–8160. https://doi.org/10.18653/v1/2020.acl-main.725

[41] Ben Zhou, Daniel Khashabi, Chen-Tse Tsai, and Dan Roth. 2018. Zero-Shot Open Entity Typing as Type-Compatible Grounding. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Brussels, Belgium, 2065–2076. https://doi.org/10.18653/v1/D18-1231

A EVALUATION METRICS

Following the prior FET studies ([4], [15], [18]), we evaluate our methods and the baselines using three evaluation metrics: Strict Accuracy (Acc), Micro-F1 (Mi-F1), and Macro-F1 (Ma-F1).

Accuracy. Given a set of entity mentions M, we denote the set of ground truths and predicted types as t_M and $\hat{t_M}$ respectively. Given σ as an indicator function, strict accuracy is defined as

$$Acc = \frac{\Sigma_{m \in M} \sigma(t_m == \hat{t_m})}{M}$$

Macro-F1. Macro-F1 is calculated using Macro-Precision (P_{ma}) and Macro-Recall (R_{ma}) where

$$P_{ma} = \frac{1}{|M|} \Sigma_{m \in M} \frac{|t_m \cap \hat{t_m}|}{\hat{t_m}}$$

$$R_{ma} = \frac{1}{|M|} \Sigma_{m \in M} \frac{|t_m \cap \hat{t_m}|}{t_m}$$

Micro-F1. Micro-F1 is calculated using Micro-Precision (P_{mi}) and Micro-Recall (R_{mi}) where

$$P_{mi} = \frac{\Sigma_{m \in M} |t_m \cap \hat{t_m}|}{\Sigma_{m \in M} \hat{t_m}}$$

$$R_{mi} = \frac{\Sigma_{m \in M} |t_m \cap \hat{t_m}|}{\Sigma_{m \in M} t_m}$$

Macro-F1 and Micro-F1 are calculated using the F1 score formula with their respective granular precision and recall scores.

$$F_1 = \frac{2 * Precision * Recall}{Precision + Recall}$$