Fine-Grained Entity Typing with a Type Taxonomy: a Systematic Review

Ruili Wang, Feng Hou, Steven F. Cahan, Li Chen, Xiaoyun Jia, and Wanting Ji

Abstract—Fine-grained entity typing (FGET) is an important natural language processing (NLP) task. It is to assign fine-grained semantic types of a type taxonomy (e.g., *Person*/*artist*/*actor*) to entity mentions. Fine-grained entity semantic types have been successfully applied in many natural language processing applications, such as relation extraction, entity linking and question answering. The key challenge for FGET is how to deal with label noises that disperse in corpora since the corpora are normally automatically annotated. Various type taxonomies, typing methods and representation learning approaches for FGET have been proposed and developed in the past two decades. This paper systematically categorizes and reviews these various typing methods and representation learning approaches to provide a reference for future studies on FGET. We also present a comprehensive review of type taxonomies, resources, applications for FGET and methods for automatically generating FGET training corpora. Furthermore, we identify the current trends in FGET research and discuss future research directions for FGET. To the best of our knowledge, this is the first comprehensive review of FGET.

Index Terms—entity analysis, fine-grained entity typing, semantic types, type taxonomy, knowledge base

1 Introduction

It is essential to recognize entities such as names (e.g., names of person, organization and location) and numeric expressions (e.g., time, date, money and percent expressions) in natural language text. Identifying references or mentions of these entities in text was acknowledged as an important natural language processing (NLP) task in the Sixth Message Understanding Conference (MUC-6) [1]. Named Entity Recognition (NER) [2], [3], [4], [5] is to recognize and categorize named entities into coarse-grained classes (typically 4 classes: *Person*, *Location*, *Organization* and *Miscellaneous* [6]).

However, this kind of coarse-grained type information is insufficient for many applications, while fine-grained entity types can lead to substantial improvements on many NLP tasks [7], [8], [9], [10]. Thus, there has been a booming research stream on Fine-grained Entity Typing (FGET) [11], which is to classify entity mentions into more fine-grained semantic types. For example, *Leonardo DiCaprio* is typed as *person/artist/actor*.

The mentions of entities are complicated. A large proportion of entity mentions are coreferences (alias, nickname, pronoun, abbreviation, etc.) and ambiguous names. Furthermore, quite a number of entity mentions in textual documents are new entities that are not included in a knowledge base (KB). Thus, several tasks of entity analysis were developed to tackle these problems. For example, Entity Mention Detection (EMD) [12], [13] is to identify all the mentions of entities, including names, noun phrases and pronouns that refer to entities; Entity Coreference Resolution [14] is to identify and cluster entity mentions that refer to the same entity; Entity Linking [15] is to resolve the ambiguity of an entity mention by linking it to a specific entity in a KB. All these tasks, along with FGET, are clustered as entity analysis [16], [17].

These entity analysis tasks are highly interdependent. Thus, an approach for FGET may tackle FGET with other tasks simutaneously [18], [19], [20], [21], [22], [23], [16], [24]. For example, Durrett and Klein [16] proposed a model that tackles multiple tasks jointly by taking advantage of cross-task interactions between FGET, entity coreference resolution and entity linking using a structured conditional random field (CRF).

Motivations for this review: Fine-grained type information of entities (mentions) have been successfully applied in many NLP tasks. FGET has attracted significant attention, and a considerable number of approaches with different type taxonomies have been proposed. However, there are no reviews on FGET approaches and related issues. To the best of our knowledge, this is the first comprehensive review of FGET.

Contributions of this review: We systematically categorize, compare and analyze various typing methods, representation learning approaches. The other issues, such as KB resources, FGET applications, training corpora generation and type taxonomies, are also reviewed to provide useful resources for the FGET research community. Further, we identify the current trends in FGET research: (i) Learn embedded feature representations to address the challenges posed by label noises, tail types and new entities; (ii) Tackling FGET and other entity analysis tasks jointly is also a promising direction.

Outline: The rest of this paper is organized as follows. The background, applications and the KB resources for FGET are reviewed in Section 2. In Section 3, we review the methods for curating type taxonomies and automat-

• *Ruili Wang, Feng Hou, Xiaoyun Jia and Wanting Ji are with the School of Natural and Computational Sciences, Massey University, Auckland 0632, New Zealand. (e-mail: ruili.wang@massey.ac.nz; f.hou@massey.ac.nz; sophiajianz@gmail.com; wantingji@outlook.com)*
• *Steven Cahan and Li Chen are with the University of Auckland. (s.cahan@auckland.ac.nz, li.chen@auckland.ac.nz)*

Manuscript received xx, 2021; revised xx, 202x.

ically creating training corpora. In Section 4, we review the techniques for feature representations. In Section 5, we present an inspection on the typing methods for FGET. In Section 6, we present a review of the special considerations for tail types and new entities. In Section 7, we review the evaluation metrics for FGET. Finally, we conclude this review and discuss future directions in Section 8.

2 Background

In this section, we first give a formal definition of the FGET task, and then review the applications of FGET as well as the KB resources for FGET.

2.1 Task Definition

Given a document d or a collection of documents D that contains a set of entity mentions M, and a predefined taxonomy of fine-grained semantic types T, the task of FGET is to assign each entity mention $m \in M$ an appropriate type $t \in T$ on the type taxonomy. An entity may have multiple types in different contexts, e.g., *Donald Trump* is a *political figure*, a *businessman* and an *actor*. FGET can be divided into two kinds of tasks: mention-level FGET and entity-level FGET. The formal definitions are given as follows:

Entity Mention: Entity mention is a continuous span of tokens in the text which refers to a real world entity. Entity mention can be a named entity mention, a nominal mention or pronoun coreference.

Type Taxonomy: Type taxonomy or type ontology is a tree or a directed acyclic graph (DAG) $O = (T, R)$, where T is the set of semantic types and R is the edge set. $R = \{(t_i, t_j) \mid t_i, t_j \in T,\ i \neq j\}$ is also called the relation set, in which (t_i, t_j) means that t_j is a sub-type of t_i.

Mention-level FGET: Mention-level FGET can be defined as $f : M \times C \mapsto T$(C is the set of the corresponding context of each mention in M), which is to find a semantic type with the appropriate degree of granularity for an entity mention within a specific context. Mention-level FGET is also called context-dependent FGET.

Entity-level FGET: Entity-level FGET can be defined as $f : M \times D \mapsto T$, which is to find all possible semantic types for an entity. The type set of an entity should be the union of mention-level FGET in different contexts. Entity-level FGET is also named context-independent FGET or corpus-level FGET. Entity-level FGET is mainly motivated for KB construction and completion.

An example of mention-level FGET is presented in Fig. 1. *Donald Trump* has multiple types, including *political figure*, *business* and TV show *actor*. But from the context given by the sentence, *Person/political_figure* is the appropriate type path, so the type label could be *Person* or *political_figure*.

For entity mentions, some approaches [25], [26], [27], [28], especially those entity-level FGET methods, assume that the gold entity mentions are segmented by NER or EMD tools. Some methods view FGET as fine-grained NER, treat the mention detection as an integrated part of FGET, and organize the mention detection and entity typing in a pipeline way [8], [29]. Some methods for automatically generating training data also work in this way. A detailed summary of mention detection in automatically creating training corpora is given in Section 3.2.1.

Entity-level FGET is related to word clustering [30] (or many other names, e.g., semantic class identification, hyponym acquisition [31], semantic lexicon induction, semantic class learning [32]), which is to assign words to classes based on the statistics from large corpus. This task originates from the class-based n-gram language model [33], which reduces the problematic effects of sparsity by grouping similar words into classes. Mention-level FGET is similar to word sense disambiguation (WSD) [34]. WSD is a task of determining the fine-grained semantic class of words in contexts. Both mention-level FGET and WSD need to make decisions based on the contexts of mentions or words.

Compiling semantic type taxonomy is related to the automatic ontology construction, which aims to create lexical hierarchies based on semantic classes [35]. However, type taxonomies used for FGET were manually curated or semi-automatically mapped from type taxonomies of a KB.

2.2 Applications

Fine-grained semantic types of entities have been applied in the following NLP tasks:

2.2.1 Relation Extraction

Relation has constraints (so called type signature) on types of entities as arguments. For example, the coarse-grained type signature of relation *"TeamPlaysInLeague"* is traditionally *(ORG, ORG)*. FGET can provide a fine-grained type signature *(Sports_team, Sports_league)* [8], which can provide cues for the relation extractor. Yaghoobzadeh *et al.* [36] improved the performance of relation extraction by incorporating fine-grained entity types. They showed that joint training of entity typing and relation extraction was better than a pipeline model [37], [23].

2.2.2 Entity Linking

The constraints of semantic types can drastically reduce the number of candidate entities during entity linking [38], [39]. Entity type information can alleviate the sparsity and noisy problems of entity linking in short text, especially for linking of tail entities [10]. Encoding entity types and other information into an unified representation of entities [24] can also boost the performance of entity linking. Finer-grained type information is also helpful for multilingual entity linking [40].

2.2.3 Question Answering

Questions can be classified into 18 broad classes [41] according to their answer semantic types. Li *et al.* [27] used the compatibility between the answer type and candidate type, to rank the answer candidates. Lee *et al.* [42], [43] defined 147 fine-grained named entity types in consideration of users' asking points for finding answer candidates. Other works on FGET for question answering include [44], [45], [46], [9], [27].

2.2.4 Coreference Resolution

Entity linking, coreference resolution and entity typing are heavily interdependent with each other [16]. A joint model for these three tasks [16] achieved significant improvements. Even coarse-grained entity types (18 types of OntoNotes) have been shown to be helpful for coreference resolution [47].

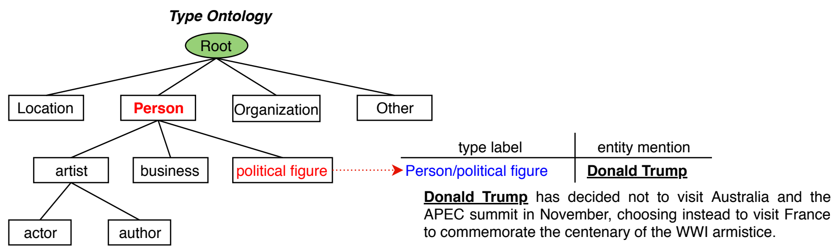
Type Ontology
Root
Location
Person
Organization
Other
artist
business
political figure
actor
author
type label
entity mention
Person/political figure
Donald Trump
Donald Trump has decided not to visit Australia and the APEC summit in November, choosing instead to visit France to commemorate the centenary of the WWI armistice.

Fig. 1. An example of mention-level FGET. The left part is a snippet of a type ontology tree; the right part is a sentence with one mention being labelled with a fine-grained type.

2.2.5 *Knowledge Base Construction and Completion*

When building a KB, semantic types of entities can be automatically recorded [48]. As existing KBs do not record all possible types of all entities, KB completion [49] can be implemented by assigning types learned from a huge corpus to those entities that have no such information [28].

2.2.6 *Query Analysis and Entity Search*

A significant proportion of web searches is to find names or entities. Three scenarios of searching can benefit from fine-grained semantic types of entities [50]: (i) Search for a list of items of a category; (ii) Retrieve similar names or sibling names of familiar names; (iii) Query refinement suggestions. Entity-oriented queries that characterize a specific relationship between entities (e.g., "*Airlines* that currently use Boeing 747 planes") also benefit from fine-grained types [51]. Query feature expansion with entity types results in significant improvements in retrieval effectiveness [52]. FGET can also significantly improve the performance of entity search [53], [54], which is to retrieve a ranked list of entities in response to a query.

2.2.7 *Other Applications*

Fine-grained entity types have also been applied in Text Generation [55], [56] and Entity Recommendation [57].

2.3 Knowledge Base Resources for FGET

In this subsection, we introduce those KB resources that have been widely used in FGET by reviewing how they are employed.

WordNet [58] is an electronic lexical database for English. In WordNet, words are grouped into sets of synonyms called synsets, and all synsets are organized by means of semantic relations, such as hypernyms (e.g., canine is a hypernym of dog) and hyponyms (e.g., dog is a hyponym of canine). Thus WordNet is a combination of dictionary and thesaurus. Some approaches [25], [59] induced a type hierarchy by exploiting the semantic relations between WordNet synsets and words.

Wikipedia is based on a model of openly editable and view-able content, a wiki. Each entry of Wikipedia is linked to an article, which provides a description of the entity or topic specified by the corresponding entry item. These articles are linked to each other by a reference or anchor link. Articles of Wikipedia have been used to create a training corpus for NER [60] and FGET [61]. Hou *et al.* [62] extracted manually curated fine-grained type words from the first paragraph of Wikipedia articles.

Freebase [63] is an online collection of structured data harvested from many sources, including individual, user-submitted wiki contributions. Freebase has a type ontology consisting of over 1,000 types. Each Freebase object is an entity with assigned types. Although Freebase was officially shut down in May 2016 and was replaced by Wikidata, many works on FGET have been based on Freebase. The Freebase type set was used as typing ontology [64], or used to generate type labels for training corpus [8], [29].

DBpedia [65], [66] is constructed by extracting structured information, e.g., the infobox data and category information, from Wikipedia. Currently, most of the entities in DBpedia have been classified according to a consistent fine-grained type ontology. The ontology is manually created based on the most commonly used infoboxes within Wikipedia, and currently covers 685 types. Li *et al.* [27] used the anchor link of an entity mention to find its corresponding entity. Then, the type of this entity was queried from DBpedia. DBpedia Spotlight [67] was also used to annotate mentions of DBpedia entities in text, hence the corresponding types could be labelled.

YAGO (Yet Another Great Ontology) [68], [69], [70] is a large semantic KB built on information extracted from Wikipedia, WordNet, WikiData, GeoNames, etc. One uniqueness about YAGO is that it assigns the entities to more than 350,000 fine-grained semantic types of a taxonomy which combines the manual semantic taxonomy of WordNet with the Wikipedia category data. HYENA [38] uses Wikipedia and the entity types of YAGO2 [71] to automatically generate training corpus.

3 Type Taxonomy and Training Corpus

In this section, we present a review of how the type taxonomies are curated and how the training corpora are automatically generated.

3.1 Type Taxonomies

FGET uses far more complex type taxonomies as opposed to traditional NER. There are two categories of type taxonomies: manually curated and semi-automatically curated.

Manually Curated Type Taxonomies Initially, *Locations* were manually classified into 8 fine-grained subtypes (i.e., *Country, City, Street, Territory, Region, Water, Mountain*, and *Artifact*) [84] and *Persons* were classified into 8 subcategories [85] (i.e., *Athlete, Politician/Government, Clergy, Businessperson, Entertainer/Artist, Lawyer, Doctor/Scientist, Police*).

TABLE 1
A summary of type taxonomies widely used in NER and FGET

Type Taxonomy	Number of Types	Hierarchical Depth	Gold KB Links	Deriving Manner
Sekine [72], [73], [74]	200	3	No	Manually
FIGER [8]	112	2	Yes	Freebase types/Manually merging
GFT [29]	88	3	Yes	Freebase types/Manually merging
HYENA [38], [75]	505	9	No	YAGO types
TypeNet [76]	1941	14	Yes	Mapping Freebase type to the WordNet
FINET [25]	16k	-	No	WordNet
PEARL [59]	200	1	No	WordNet
Ultra-Fine Types [77]	9/121/10201	-	No	Mapping FIGER/GFT
BBN [78], [11]	93	2	No	Manually
ACE [79]	52	2	No	Manually
MUC [1]	7	1	No	Manually
CoNLL-YAGO [6], [80]	4	1	Yes	Manually
OntoNotes 5.0 [81]	19	1	No	Manually
Freebase [63], [82]	2k	2	Yes	WordNet Mapping
DBpedia [65]	685	-	Yes	infoboxes within Wikipedia
YAGO3 [70]	More than 350k	-	Yes	Combining taxonomy of WordNet and the Wikipedia category
WordNet [58], [83]	16k	14	No	Manually

Sekine *et al.* [72] compiled a taxonomy of 150 types (extended to 200 types [73] later) with maximum depth 5 to make it broad enough for general applications. They used three methods (i.e., based on corpus, based on previous systems and tasks, and based on thesaurus) to design three initial taxonomies. Then they merged the three taxonomies into one and refine it. Lee *et al.* [42] introduced a set of 147 fine-grained types with two levels (the 15 top-level types include *Person, Study_Field, Theory, Artifacts, Organization, Location, Civilization*.) for question answering.

Semi-automatically Created Type Taxonomies KBs have been used for semi-automatically generating type taxonomies. For example, Wikipedia categories [1] have been used to derive a taxonomy [86], [87]. Those categories whose syntactic heads[2] cannot be found in WordNet [58] are removed, and types are obtained by mapping categories to their syntactic heads. Compared with the Sekine type taxonomy [73], Wikipedia category based type taxonomy is more like a semantic tag than a well-defined type taxonomy, while Freebase semantic types provide broader coverage of entities and types, and allow for multiple types being assigned to one entity. But the Freebase types are noisy since they are labelled by non-expert volunteers. Thus, Ling and Weld [8] took two steps to get a well-defined type taxonomy FIGER from Freebase types: (i) Irrelevant types are filtered out, and only those types with more than 5 ground instances in Freebase were kept; (ii) Those too specific types were manually merged. Similarly, the GFT type taxonomy [29] was obtained by first organizing the non-hierarchical types (the level 2 types) of FIGER [8] into a hierarchy and then refining them by discarding rare or ambiguous types. New types were inserted if there were enough instances to corroborate the insertion. Although the coverage and specificity limitations of Freebase types may be exacerbated by such manually merging, FIGER and GFT are two of the most widely used taxonomies.

The semantic types of other KBs are leveraged to derive type taxonomy as well. For example, Hyena [38] types were derived from 505 YAGO [70] types. The 22 top-level types of [27] were derived from the ontology of DBpedia [65]. PEARL [59] used the YAGO2 [71] type taxonomy, which was derived from 200 WordNet classes.

The ultra-fine grained type taxonomies normally have thousands of types. For example, FINET [25] consists of more than 16K types with top categories *Person, Location* and *Organization*. Choi *et al.* [77] organized types into three disjoint bins: 9 general types, 121 fine-grained types and 10,201 ultra-fine grained types. Murty *et al.* [76], [88] manually mapped Freebase types to the WordNet hierarchy, and got a type taxonomy of 1941 types with average depth 7.8. The purpose of introducing ultra-fine-grained types is to improve the coverage and diversity of type taxonomies, and to alleviate the skewed label distributions of corpus which is heavily skewed toward coarse-grained types. However, the taxonomy with thousands of types also poses a great challenge to typing models.

There are also domain-specific type ontologies. For example, Unified Medical Language System (UMLS) [89], a biomedical hierarchical concept ontology with average depth 14.4, contains over 3.5 million concepts.

A summary of FGET type taxonomies is listed in Table 1. As we can see, extensive work has been done to curate type taxonomies for FGET. However, no agreement has been reached by the research community, as the granularity of semantic types is quite subjective and dependent on task needs.

1. https://en.wikipedia.org/wiki/Wikipedia:Contents/Categories
2. The word that determines the syntactic category of a phrase or semantic category of a compound, e.g. the head of the noun phrase *boiling hot water* is the noun *water*.

3.2 Training Corpora Generation

Manually annotating data with multiple fine-grained types is a daunting task. Thus, almost all the FGET training corpora are generated automatically, i.e., mentions are linked to a specific entity of a KB using an entity linking tool (such as Dbpedia Spotlight [90] or AIDA [91]) or anchor links in the Wikipedia, and they label the mentions with all types from the KB. As for the mentions, some mention-level FGET methods [26], [27], [28] and all methods of entity-level FGET treat mention boundaries as given ground truth. There exist publicly available labelled corpora, such as Wikilinks [92] and ClueWeb [93], in which entity mentions are identified and linked to a KB.

In the following subsections, we review the issues involved in automatically generating training corpora.

3.2.1 *Entity Mention Detection*

Some works on FGET, especially those on mention-level FGET, treat mention boundaries detection as an integrated part of automatically generating training examples. Nominal, pronominal, and named entity mentions are all considered as targets.

FINET [25] directly uses Stanford CoreNLP tool to extract named entities with coarse-grained types. Thus, only the named entities are extracted, and the nominal phrases and pronouns are ignored. Nakashole *et al.* [59] proposed a method to detect and type noun phrases that refer to new entities. However, this method cannot be used to generate training examples directly, because it can only generate type labels in open vocabulary words and these labels need to be converted to type labels of a hierarchical taxonomy.

To extract both named entity mentions and noun phrase, Gillick *et al.* [29] applied a POS[3] tagger, a dependency parser and a noun phrase (NP) extractor in a pipeline way, where the noun phrase extractor relies on POS tags and dependency edges to detect entity mentions. But the pronominal mentions are ignored as there is no coreference resolution processing. Ling and Weld [8] treated the segmentation of mentions as a sequence labelling task and trained a linear-chain CRF model on a heuristically-labeled Wikipedia data to detect entity mentions. Ren *et al.* [94] used quality mention examples from a KB as distant supervision and a random forest classifier to score segmentation quality.

Many approaches [8], [27], [87] use the anchor links in Wikipedia to detect entity mentions, but only less than 3% of text tokens in Wikipedia are anchored strings. Ghaddar and Langlais [61] proposed finding more non-anchored entity mentions by leveraging: (i) The out-link structure of Wikipedia, i.e., search the titles of the out-links in the current article; (ii) The coreferent mentions that refer to the main concept of a Wikipedia article extracted by WikiCoref [95], [96]. This can raise the coverage of mentions from less than 3% to about 30%.

3.2.2 *Corpora Automatic Labelling*

One approach for obtaining semantic types of a mention is using Wikipedia anchor links [60]. For each linked mention in a sentence, Ling and Weld [8] and Dong *et al.* [27] found the corresponding Wikipedia entry via the anchor link; the semantic types were then mapped from Freebase or DBpedia to their own type taxonomy. WiFiNE [61], a large scale FGET corpus generated from Wikipedia, automatically maps the corresponding Freebase *object_type* of each entity to GFT type taxonomy [29] or FIGER taxonomy [8]. The Wikipedia's categories are not used since they are more like tags than a well-defined type taxonomy. The disadvantage of using anchor links is that the number of training examples is limited by the Wikipedia anchor links.

Another method of obtaining semantic types of a mention is using entity linking tools. Gillick *et al.* [29], Yogatam *et al.* [26] and Ren *et al.* [94] used an entity linking system to retrieve Freebase types of resolved entity mentions, and then mapped Freebase types to their taxonomy. They used heuristics to reduce label noises. However, the linking errors can introduce label errors in the training examples. Moreover, both the Wikipedia link method and the entity linking tool method fail to label out-of-knowledge-base entities.

The appositional[4] structures are used to generate training examples of out-of-knowledge-base entities. Ekbal *et al.* [97] leveraged appositional patterns to extract person name-type pairs from ukWaC[5] corpus. They used pattern frequency and WordNet information to reduce ambiguous and invalid semantic types. This method can achieve a 96.58% accuracy. But the number of examples is limited by the sparseness of apposition and the type is limited to *Person*.

For ultra-fine labeling, head words were directly used as type labels [77]. However, head words labels are scarce and not context-sensitive. Thus, Dai *et al*. [98] proposed a method that combines hypernym extraction patterns [31] and a pre-trained language model to generate ultra-fine labels .

3.2.3 *Label Noise Reduction*

The automatically generated type labels have two kinds of label noises: out-of-context noises and overly-specific noises [99]. To explain this, consider the mention *Hugh Laurie* in such two sentences: (i) *Hugh Laurie and his wife Jo Green were on the verge of divorce*. (ii) *Hugh Laurie wins Best Supporting Actor in a miniseries*. *Hugh Laurie* has multiple labels (e.g., *actor, director, musician, comedian*, and *author*) in a KB, and all these labels will be assigned to both mentions. In Sentence (i), the type label should be *Person*, all other types are **overly-specific** label noises (they are not out-of-context labels, since they are coherent with the context). In Sentence (ii), the type label should be *Person/artist/actor*, and all the other types are **out-of-context** label noises.

Using training examples with such label noises, some approaches ignore the noises and treat training corpus as normal [8], [26], [100], [101]. The performance of these approaches is decreased by such label noises.

Another approach is to devise mechanisms to deal with the noises, to either directly tweak the noisy labels, such as [29], [77], [61]; or enhance the typing model with the ability of tackling label noises, such as [102], [103], [104], [99]. These mechanisms are reviewed as follows:

3. Part of Speech, English POS include noun, verb, adjective, adverb etc.

4. Two noun phrases are placed next to each other to identify the same entity in a different way. For example, Hugh Laurie, an award winning *actor*.

5. http://wacky.sslmit.unibo.it/doku.php?id=corpora

Out-of-Context Label Noise Reduction Choi *et al.* [77] used head words as distant supervision to avoid out-of-context noisy labels. For example, for mention "the 44th president of US", a KB may provide multiple types such as *author, lawyer*, and *president*. Head words can reduce the type set to *president*. But such distant supervision is not always available, thus most of the noisy labels will not be reduced.

Gillick *et al.* [29] used the following two heuristics to reduce out-of-context noisy labels: (i) coarse type pruning: deleting type labels that are inconsistent with the prediction of a standard NER classifier trained on ACE [79] data; (ii) minimum count pruning: removing type labels that appear only once in a document. However, such heuristics may cause some mentions to have 'none' labels.

AFET [102] separates the loss functions for clean (with single type label) and noisy (with multiple type labels) examples, and uses clean examples to de-noise those examples with noisy labels. But the performance is sensitive to parameters, which are corpus dependent. To obviate the need for tuning parameters, Abhishek *et al.* [104] proposed a neural network model with a variant of hinge loss function for noisy examples. The model makes the maximum score for the set of positive but noisy types greater than one. But the possible semantic correlations among types were ignored during the training of type embeddings.

Overly-Specific Label Noise Reduction Gillick *et al.* [29] used the so called sibling pruning to heuristically remove overly-specific labels, i.e., sibling labels would be removed if they share a single parent type. For example, a mention labelled with */person/artist/director* and */person/artist/actor* would be reduced to */person/artist*. The performance of their typing model has noticeable improvement after reducing the noisy labels using this heuristic. But this also removes training examples for non-prominent types.

General Label Noise Reduction WiFiNE [61] uses the following two heuristics to de-noise the labels: (i) the type arguments of Freebase relations, e.g., the *place_of_birth* relation indicates that the type of the first argument is *person*; (ii) the sharing of common attributes, e.g., if two entities appear in the same sentence, and the non-noisy mention has a type set which is a subset of the noisy mention, then the noisy label is replaced with the non-noisy label. Such heuristics may keep noisy labels while removing true labels.

Ren *et al.* [103] used unambiguous mentions (only one type label) as distant supervision to de-noise those mentions with multiple labels that appear in a similar context. The mentions, contextual words and type labels, were embedded into a common dimensional space by optimizing a margin-based rank loss function. Because the noise reduction step was separated from the typing model, the errors engendered by noise reduction will be propagated into the typing model.

To alleviate this phenomenon, Xu and Barbosa [99] proposed a simple variant of cross-entropy loss to handle data with out-of-context noises, and introduced a hierarchical loss normalization process to let the model understand the type hierarchy and mitigate the negative effects of overly-specific noises. But the hierarchical loss normalization parameters need to be tuned on a different corpus.

Label Noise Reduction for Entity-level FGET Entity-level FGET methods [36], [88], [76] use multi-instance multi-label (MIML) learning [105] to reduce the negative effects of label noises. MIML operates over a bag of mentions (collection of all sentences/contexts containing the same entity) rather than over a single mention. For MIML, at least one mention of a bag justifies the types given in the KB, and one bag can have multiple types. But this may amplify the dominance of prominent entities with prominent types, and make the prediction on tail types more difficult.

4 Feature Representations

In this section, we present a comprehensive review of feature representations for FGET.

4.1 Hand-crafted Features

The following categories of hand-crafted features have been exploited for FGET: Mention-level, Context-level, Document-level, Corpus-level, Gazetteers, External and Type-based.

Mention-level features include: syntactic head, non-head token, POS tags, word shape, characters, and prefix/suffix. Brown cluster id [33], [30] of all words [8] or head word [29], WordNet synset of words [25] are used as well.

Contextual features used in FGET include: n-gram of words around the mention phrase, POS tags of contextual words. The syntactic dependency features, such as the role of the mention head on the dependency tree [8], [29], the lexical parent of the mention head [29], the verb governing an entity mention [11], are also used in FGET.

Topic features are from a simple bag-of-words topic model with eight topics (e.g., *arts, business, entertainment* etc.) [29], or unigrams, bigrams, and trigrams extracted from a paragraph window [75], [38].

Gazetteer features signify whether the mention contains a word in a type's gazetteer or not. Gazetteers are from WordNet hyponyms [11], YAGO [38], [75] or Web data [109], [111].

External features include: WordNet being used to expand context features [85], Wikipedia articles [108], [112].

Type-based features [87] consider the compatibility between the mention and the types in the predicted type set, and the relation and compatibility among the types in the predicted type set.

Table 2 lists the hand-crafted features used in FGET. However, there is paucity comparison of the effectiveness of these features. Rahman and Ng [11] analyzed the effects of their seven types of features. Their results show that the best accuracy is achieved when only features of mention tokens, WordNet synsets and POS tags of mention tokens are used. This indicates that their gazetteers, morphological features, capitalization, and the semantic role of governing verb are redundant. They also found that the most useful features are the mention tokens. While Yosef *et al.* [38] found the performance drops significantly when the gazetteer features are disabled.

The hand-crafted features represent the human knowledge on FGET, but the feature independence assumption is too loose. These features are often represented as dimensional one-hot vectors; therefore the typing model will inevitably suffer from the curse of dimensionality. Moreover, most of these features are generated by NLP tools, the errors of NLP tools will definitely propagate into typing models.

TABLE 2
Lists of hand-crafted features used for FGET

Category	Feature Name	Description
Mention	Head [8], [29], [26], [106], [87]	Syntactic head of mention phrase
	Non-head [29], [26], [87]	Each non-head token in mention phrase, [8], [106] consider all tokens
	Shape [8], [26], [106], [87]	Word shape of tokens in mention phrase
	POS tags [8], [106], [11]	POS tags of words in mention phrase
	Length [8]	Number of tokens in mention phrase
	Characters [29], [26], [106]	Character trigram in the mention head
	Prefix/Suffix [97], [11]	Prefixes and suffixes of tokens
	Digit/Symbol [97]	Whether mention phrase contains digit strings, non-characters(e.g., slashes) or number expressions
	Word length [97]	Whether token is smaller than a pre-defined threshold (i.e., less than 5 characters)
	Brown clusters [8], [30] [29], [16]	Word cluster id for each token in mention phrase
	WordNet synset [25], [11]	WordNet synset of tokens
	NP or NE [11]	Whether the mention phrase is a nominal phrase or a named entity
	Semantic [11]	Coarse-grained type of NE, or WordNet semantic class of NP
Context	Unigrams [8], [38], [106], [87]	Token unigram before and after the mention phrase
	Bigrams [8], [38], [106], [87]	Token bigrams in context window (including the mention phrase [8])
	Trigrams [38], [75]	Token trigrams in context window
	POS tags [97], [38]	POS tags of context
	Parent [29], [26], [87]	Lexical parent of the mention head on dependency tree
	Role [8], [29], [26], [106], [11]	Dependency label of the mention head on dependency tree
	ReVerb Patterns [8]	Meaningful relation phrases in ReVerb [107]
Document	Topic [29], [26], [108], [85]	The most likely topic label for the paragraph/document
Corpus	Textual Relations [45]	The phrases that frequently occur with entity mentions
	Relation Patterns [59], [64], [102]	The phrases that connect two entities in a sentence
Gazetteer	Type gazetteer [11], [38], [75], [109], [110]	Whether the mention contains a word in corresponding type's gazetteer or not
External	WordNet Score [85]	A score for each word on WordNet hypernym tree obtained by expanding each context word
	Wikipedia Entity Vector [108]	A vector trained on Wikipedia articles using Word2Vec, each anchored entity mention is treated as word
Type-based	Mention-type Conjunction [87]	Conjunction of mention features with indicators on types in the predicted type set
	Type pairs [87]	Indicators on pair of types in the predicted type set
	Type Relations [87]	Indicators on whether types in the predicted type set have some relations, e.g., sibling

4.2 Embedded Features

Since the successful applications of neural network in natural language processing [113], [114], [115], feature embeddings have been an effective way of learning and representing features. Most FGET approaches use the pre-trained word embeddings as input, e.g., GloVe [116] is used in [99], [117], [118], [119], [101], [120], Word2Vec [121] and *fastText* [122] are used in [123], [106], [28].

4.2.1 Mention Embeddings

Mention embeddings encode the compositional, morphological and orthographic information about entity mentions. For convenience, we represent the words in a mention as $m_1, m_2, \ldots$ ($m_{|m|}$, $|m|$ is the length of mention span), and $\mathbf{u}$ is a mapping function from tokens to embeddings.

Averaging Encoder Compute the mention embeddings by averaging embeddings of words that comprise the mention phrase [100], [101], [99], [28], [117], [119], [118], [76], [88], [124], [125], [126].

$$\mathbf{v}_m = \frac{1}{|m|} \sum_{i=1}^{|m|} \mathbf{u}(m_i) \tag{1}$$

Variant forms include: a learnable n-grams encoder [28], and a non-linear function on the averaged mention embeddings [123]. This relatively simple method for mention embedding is less prone to overfitting [101]. But this method cannot capture the internal structure and compositional nature of the mention phrase, and the word orders are ignored.

Attentive Encoder Some words in an entity mention may provide more useful information for typing, e.g., the head words. Lin and Ji [127] used the self-attention mechanism to compute the weight of each word.

Word-level RNN Dong *et al.* [27] used the recurrent neural network (RNN) [128] to encode the internal structure and compositional nature of entity mentions. Xu and Barbosa [99] applied **Word-level LSTM** [129] to the mention sequence from left to right, and the last output $\mathbf{h}_{|m|+1}$ is used as mention embeddings. The single directional RNN can capture the word order information, but it cannot capture the character-level patterns of words, especially the out of vocabulary words.

Character-level CNN Yaghoobzadeh and Schutze [28] applied a convolutional neural network (CNN) [130] on the sequence of mention characters. Each character has an embedding in a lookup table. Then max pooling is performed on the output of the CNN filters. This is to capture the character patterns of entities of the same type.

Character-level LSTM Abhishek *et al.* [104] decomposed an entity mention phrase into character sequence and the spaces between tokens are conserved. A vanilla LSTM was then applied to the character sequence from left to right and the final output was used as the mention embedding.

Multi-level Representations Character-level and word-level information can give complementary information about mentions. Choi *et al.* [77], López *et al.* [131] and Xiong *et al.* [120] used the concatenation of character-level CNN and weighted sum of word embedding as the mention embedding. Yaghoobzadeh and Schutze [28] explored the combination of word-level averaging vector and character-level CNN, LSTM, *Bi*LSTM, and experiments show that character-level CNN is the most effective.

4.2.2 Context Embeddings

Context representations capture information about the context surrounding an entity mention. The context can be the whole sentence where the mention phrase appears [104], [132], or a fixed sized context window [27], [117].

Some approaches [99], [77], [131], incorporate the position embeddings to reflect the relative distance between context words and mention phrase, i.e., $\mathbf{c}_i = \left[\mathbf{c}_i^d, \mathbf{c}_i^p\right]$, where $\mathbf{c}_i^d$, $\mathbf{c}_i^p$ are word embedding and position embedding of ith context word, respectively.

MLP Encoder Dong *et al.* [27] encoded the left context and right context separately using two Multi-Layer Perceptrons (MLPs) with single hidden layer, then the outputs of the two MLPs were concatenated as context embedding. But this context encoder can only process fixed-size context.

Averaging Encoder Shimaoka *et al.* [101], [100] used the averaging encoder that is similar to the averaging encoder for mention embeddings. This encoder treats the context as a bag of words, and the word orders are ignored.

Word-level CNN Jia *et al.* [123] inserted $\langle$*et_begin*$\rangle$, $\langle$*et_end*$\rangle$ before and after a mention respectively, and used CNN to encode the context. Murty *et al.* [76], [88] performed max-pooling operation on the CNN outputs. CNN is good at detecting key phrase patterns, but performs poorly in capturing the semantic relations between mention and context words.

Word-level LSTM/GRU Encoder Shimaoka *et al.* [100] applied *Uni*-directional LSTM to the left and the right context separately, in reverse direction. The final hidden states of the two LSTMs were concatenated as context embedding. Abhishek *et al.* [104] applied *Bi*-directional LSTM to the left and right context separately. The final outputs of both directions were concatenated as feature representation of each context. Karn *et al.* [117] applied a *Bi*-directional GRU [133] on the left and right context separately. While Dai *et al.* [124] used a special token w_m to replace the mention in the context and applied two layers of BiLSTMs to the whole context.

The RNN based encoders use the final hidden state or output as context embeddings. This may lose key information for typing, especially when the key word lies at the middle of the context. Thus, the attention mechanism [134], [135] is incorporated to discriminate key information for typing.

Attentive Encoder Shimaoka *et al.* [101] and Obeidat *et al.* [125] explored the self-attention mechanism on their *Bi*-LSTM encoder, and treated the left and right context separately.

$$\mathbf{v}_c = \sum_i \alpha_i^l \mathbf{h}_i^l + \sum_j \alpha_j^r \mathbf{h}_j^r \tag{2}$$

where $\mathbf{h}^l$ and $\mathbf{h}^r$ are the *Bi*-LSTM output of the left and right context, respectively, α are the attention weights. Some similar methods do not separate the left and right context [77], [99], [131], [127]. Karn *et al.* [117] applied encoder-decoder attention mechanism [134] on their *Bi*-GRU context encoder. Zhang *et al.* [119] used an L-layer stacked LSTM with mention attention, a dot-product attention [135] computed based on the alignment between the entity and its context. Xin *et al.* [118] explored three kinds of attention on their *Bi*-LSTM: self-attention, mention attention and knowledge attention, a dot-product attention computed using entity embedding learned from an external KB. Zhang *et al.* [136] proposed path-based attention model. The path-based attention for each type is computed on its path to its parent type on the type taxonomy. Such attention mechanisms improved the discriminativeness of attention weights, but still focus on local context.

Document-level Context Encoder Zhang *et al.* [119] proposed a document-level context encoder based on distributed memory model [137] and MLP.

$$\mathbf{v}_d = ReLu\left(\mathbf{W}_{d1} tanh\left(\mathbf{W}_{d2} DM(x_d)\right)\right) \tag{3}$$

where DM is a pre-trained distributed memory model which converts a sequence of words of a document into a vector. $\mathbf{W}$ are the weight matrices.

Pre-trained Contextualized Encoders Using pre-trained contextualized encoders [138] has achieved significant improvements on almost all NLP tasks. Most of the pre-trained contextualized encoders are based on the Transformer [139] architecture. Notably, ELMo [138], [127], BERT [140], [141], [142], [143], [144], SpanBERT [145], [146], RoBERTa [147], [142], XLNet [148], [142] have been employed to encode contexts for FGET.

4.2.3 Entity Embeddings

Entity-level FGET makes predictions either based on a bag of mentions [88], entity graphs [151], or based solely on learned entity embedding.

Corpus-level Entity Encoder Yaghoobzadeh *et al.* [28] treated the mentions of the same entity as a special word, and learned a corpus-level representation of that entity through a Structured SkipGram [121] (SSKIP [152]) which incorporates order of words into learning objectives. The entity embeddings learned in this way can only capture a short context of each mention.

TABLE 3
Summary of methods for encoding (learning) feature embeddings

Category	Method (Neural Networks)
Mention	Averaging Encoder [100], [101], [99], [28], [117], [119], [118], [76], [88], [124], [125], [126], [149]
	Attentive Encoder [127]
	Word-level RNN [27], [99]
	Character-level CNN [28]
	Character-level LSTM [104]
	Multi-level Representations [77], [131], [120], [28]
Context	MLP Encoder [27]
	Averaging Encoder [101], [100]
	Word-level CNN [123], [76], [88]
	Word-level LSTM/GRU [100], [104], [117], [149], [124]
	Attentive Encoder [101], [125], [77], [99], [131], [127], [117], [119], [118]
	Document-level Context Encoder [119]
	Pre-trained Contextualized Encoder [127], [141], [142], [143], [146], [144]
Entity	Corpus-level Encoder [28]
	Attributed and Predictive [150]
Type (Label)	WSABIE based [26], [106], [102], [104], [101]
	Word Embeddings based [149]
	Box Embeddings [144]
	Euclidean space [150], [125], [141]
	Hyperbolic space [131], [127]
Strategy	Hierarchical Loss Function [88], [76], [150], [99]
	Multi-task Objective [36], [150], [77]

Attributed and Predictive Entity Embeddings Jin *et al.* [150] employed a standard feed forward neural network to learn entity embeddings. The input is the adjacency vector (a binary vector represents whether two entities in a KB have relations) and the attribute vector of the entity. The neural network was trained with multi-tasks: (i) predict the neighbouring entities; (ii) predict the semantic types of typed entities. The learned entity embedding can be used to predict types of entities without type tags. But this model can only learn the embeddings of in-knowledge-base entities.

4.2.4 *Type (Label) Embeddings*

Embedding feature representations and type labels into a common dimensional space [26], [153], [102], [106], [104], [101], [132], [150], [127], [125], [131], [141] allows for information sharing between related types. Such label embeddings can make typing models robust to label noises and types that are not present in the training data.

WSABIE [154] based approaches [26], [106], [102], [104], [101] learn two mapping matrices to project the hand-crafted feature vectors and one-hot binary type vectors into a common low-dimensional space respectively. The matrices are learned using the weighted approximate pairwise (WARP) loss [154], a ranking loss that encourage placing positive types above negative types.

Such approaches still rely on NLP tools to extract hand-crafted features, and a large number of parameters in the two matrices need to be learned. The average of the embeddings of the words comprising the type name was used as type embeddings [149]. Directly learning entity embeddings and type embeddings in a common Euclidean space [150], [125], [141] can obviate the reliance on NLP tools and reduce the number of parameters. But the Euclidean space is not good at capturing type correlations or hierarchies. Thus, some approaches [131], [127] propose to project feature and type vectors onto a hyperbolic space to capture the underlying type hierarchical information. Box embeddings, which represent types and mentions as boxes [144], can capture hierarchies of types even when these relationships are not defined explicitly in the taxonomy.

4.2.5 *Strategies for Learning Feature Embeddings*

The following two strategies have been widely employed to learn effective feature embeddings for FGET.

Loss Functions Hinge loss function is used to maximize the scoring margin between positive examples and negative examples [153]. But the definition of negative examples is not trivial. Moreover, both the hinge loss and the commonly used (binary) cross-entropy loss function [117], [28], [36], [77], [118] assume the types to be independent. Thus, type hierarchy loss functions are proposed to incorporate the hierarchical correlations among types, such as (i) the binary cross-entropy loss based on a score of a hypernym link between types [88], [76]; (ii) the type order loss function for making the correct type get a higher score than its ancestor types and sibling types [150]; (iii) the hierarchical loss normalization process for penalizing less in the case where types are related [99].

Multi-task Objective As we noted in Section 1, FGET is highly related with other entity analysis tasks. Multi-task learning for FGET [36], [150], [77] can encourage the model to embed more information. For example, Yaghoobzadeh *et al.* [36] proposed a joint model for FGET and relation extraction. Jin *et al.* [150] used the same neural networks to jointly predict the neighbouring entities and semantic types of an entity in a KB, Choi *et al.* [77] organized type labels into three bins (coarse-grained, fine-grained and ultra-fine), with the training objective set to minimize the logistic loss at three granularity.

4.3 Hybrid Feature Representations

Most works on FGET have used either the hand-crafted sparse features or learned dense features. While Shimaoka *et al.* [101] considered the combination of hand-crafted features and embedded features with an attentive neural model. The sparse features are projected into a low-dimensional vector by a parametric matrix, then the projected vectors are used with mention embeddings and context embeddings. The experiment results show that the hybrid feature representation outperforms embedding-only features.

5 Typing Methods

In this section, we review the main techniques used to predict the types of entities (mentions), based on how the

TABLE 4
Comparison of different semi-supervised approaches

Approach	Need a Corpus	On-site Typing	Entity-level FGET	Need a KB
Corpus-based Entity Set Expansion	✓			
KB-based Candidate Types Ranking		✓		✓
Type Propagation on Graphs	✓		✓	

problem is modelled. In Section 5.1, we review the unsupervised methods. In Section 5.2, we inspect the semi-supervised typing methods. In Section 5.3 - 5.5, we review the supervised methods of three categories: the independent model, the collective model and the joint model.

5.1 Unsupervised Approaches

Unsupervised approaches [50], [31], [2] heavily rely on the lexicon-syntactic patterns to extract lists of entities of some category, e.g., list of cities, films.

Paşca [50] used manually selected lexicon-syntactic patterns (e.g., ⟨[Start_Of_Sentence] X [such as|including] N [and|,|.]⟩) to extract pairs of type information and entity (X, N). For example, the underlined parts of "*That is because software firewalls, including Zone Alarm, offer some semblance of this feature*" will be extracted as X and N, respectively. The fine-grained types were obtained by selecting the rightmost non-recursive noun phrase whose last component is a plural-form noun, e.g., *software firewalls* is selected as the type of *Zone Alarm*. Considering the low recall of such patterns, this method relies on millions of web documents to extract a comparably small list of entities.

Etzioni *et al.* [2] proposed improving the recall of the syntactic patterns by learning high-quality patterns. They extracted candidate patterns from web search results of seed names. Each pattern was assessed with precision and recall, computed based on the statistics from search results. But this method relies on a search engine and is limited to simple text patterns.

Elsner *et al.* [155] proposed an unsupervised generative model for clustering named entities that incorporate named entity internal structure, its syntactic context and coreference information from an unsupervised pronoun resolver. A non-parametric Bayesian inference over context-free grammars is employed to find the types. However, this model is extremely prone to fall into local minima during inference because of the complexity of the model and needs to tune hyperparameters.

5.2 Semi-supervised Approaches

5.2.1 Corpus-based Entity Set Expansion

This approach iteratively extracts new typed entities from a corpus by exploiting the similarities between candidate mentions and the typed seed entities. This approach can only extract entities similar to seed entities from a corpus. Thus, it is not applicable for on-site typing.

Using Contextual Similarity

Cimiano and Völker [156] represented types and entities as context vectors using vector space model (VSM). The similarities in vector space were used to find the types of entities. The ontology population method [157] consists of five steps: (i) Collect N snippets containing mention i; (ii) Derive a list of hypothesis phrases by replacing i with each seed entity j; (iii) Calculate the plausibility score s_j using a scoring function; (iv) Obtain an overall score s_c for type c by summing the scores of all hypothesis phrases of type c; (v) Type the mention i with the type having maximum score.

Pantel *et al.* [158] applied a matrix of terms (words and entity mentions) to ESE using the set expansion algorithm of [159]. Each term is represented by a SVM vector of numerical features $\mathbf{v}_i$ learnt using MapReduce on 200 billion words corpus. For a given set S of seed entities of a type, the centroid of S is a weighted average of its element entities. Then the similarity between seed set centroid and candidate mention was used to find the type.

The performance of these methods is heavily influenced by the size and quality of seed entities and corpus. The similarity is computed based on the sparse feature vectors using a vector space model. This makes these methods perform badly in discriminating some similar fine-grained types, such as *actor* and *director*. Yan *et al.* [160] proposed a novel Bootstrapping method combining the Monte Carlo Tree Search (MCTS) algorithm with a deep similarity network to address the sparse feature.

Using Similarity of Syntactic Patterns

Paşca [161] extracted typed entities from the Web search queries using seed entities. Each query that contains a seed entity generates a query (syntactic) template. These template queries were used to extract other candidate entities. Each extracted candidate was represented as a binary search-signature vector. The search-signature of a type was generated by combining all seed entities vector. The similarity score between a candidate entity vector and a type vector was used to determine the type. But the performance of this method is hindered by the noises in the search queries.

Carlson *et al.* [23] proposed a method that starts with a type taxonomy, a set of relations, a handful of seed entities, and a set of coupling constraints. During each iteration of Bootstrapping, four steps are taken: (i) Extract new candidate mentions and contextual patterns using recently promoted patterns; (ii) Remove candidate patterns that violate constraints; (iii) Rank those candidate patterns; (iv) Promote top patterns. But this method suffers from the sparsity of patterns, and only considers the *Company* and *Sport* domain.

To alleviate the context sparsity issue, ClustType [64] proposed the soft clustering of relation phrases that signifies a unary or binary relation between two entity mentions. They used a phrase mining [162] algorithm on POS-tagged corpus to get candidate mentions and relation phrases. But their seed entities were generated by an entity linking tool [6]. Thus, the errors of the entity linking tool were inevitably propagated into the typing phase.

5.2.2 KB-based Candidate Types Ranking

This approach can perform on-site typing. For each untyped entity mention, this approach first retrieves a collection of

6. http://spotlight.dbpedia.org

related entities from a KB and generates some candidate types from the context and related entities. Then the candidate types are ranked according to the context.

Vallet and Zaragoza [163] and Balog and Neumayer [51] used a collection of Wikipedia entities (with types) as supervision for ranking the types of entities. The former method [163] consists of four steps: (i) Execute the query using an information retrieval (IR) module to retrieve the 500 most relevant passages; (ii) Collect all entities in the retrieved passages; (iii) Rank these entities based on Kullback-Leibler distance; (iv) Rank the types for the query based on the ranking of those entities and their types. The latter method [51] represents each type as a pseudo-document, which contains the descriptions of all entities of that type. The types are ranked by an IR algorithm that execute the entity query on the corresponding documents. Both methods rely on an IR module, and the performance on entities having a large number of candidate types is poor. Instead of using an IR module to retrieve related entities, Zhou *et al.* [164] used a pre-computed word-concept(entity) map and mention-entity prior to retrieve the most relevant concepts(entities) for an entity mention. The word-concept map, learned from the Wikilinks [92] corpus, is a collection of word representations, where each word is represented with a vector of Wikipedia concepts (entities).

To reduce the number of candidate types, FINET [25] and SANE [165], [166] decomposed the ranking into two steps: candidate types extraction and type selection. They extracted candidate types (restricted to hyponyms of coarse-grained type) from context using Hearst patterns [31], Appositional, Copular patterns, etc. To select the most appropriate types that best fit the context, FINET [25] trained a Naive Bayes classifier per coarse-grained type using automatically generated data from WordNet. This classifier works like WSD using WordNet. However, the performance of this model is hindered by the NER tool, which may generate false coarse-grained types.

5.2.3 *Type Propagation on Graphs*

This approach first extracts information from corpora and build graphs to represent the relations between typed (or linked) entities, untyped entities and type labels. Then different algorithms are used to propagate type labels from typed entities to untyped mentions.

Talukdar *et al.* [167] used graph model to represent type-instance relations, extracted from unstructured text or structured HTML tables. Kozareva *et al.* [168] found that the instance-instance graph, whose edges represent the neighbouring appearance in the doubly-anchored patterns (DAP, e.g., $\langle$*semantic-type*$\rangle$ such as $\langle$seed$\rangle$ and $*$), was more effective than type-instance graph for type propagation. Lin *et al.* [45] exploited the textual relations. For example, the typed *Microsoft* and *Google* occur with textual relations such as "has already announced" and "has released updates for", and the untyped *Sun Microsystems* also appear with similar textual relations. The most frequent Freebase types of similar entities were extracted as the type for *Sun Microsystems*. Such type propagation methods inevitably suffer from sparsity as they rely on the textual phrases.

The above methods do not consider assigning multiple compatible types to an entity. PEARL [59] finds the most appropriate set of types for an entity using an integer linear program (ILP) under type disjointness or type correlation constraints. To overcome the sparsity, it allows for fuzzy matches of patterns by introducing the type-pattern likelihood, which is the likelihood of relational patterns p (i.e., the PATTY collection [169] of 300,000 typed relational paraphrases) occurring with entities of types t_1, t_2. But this method performs poorly on the infrequent entities, since it does not have sufficient evidence for reliable type assignment.

Another method that can propagate multiple compatible types to entity mentions is the so-called universal schema for FGET [170]. It embeds entities and types by maximizing the log likelihood of the observed units of an entity-type matrix. The missing units, i.e., the missing types of an entity, are predicted by matrix completion. However, it can only extend the type set of an entity by using semantic implications between types learned from the observed units of the matrix. The ambiguities in the 16k flat types are ignored.

The above methods can be viewed as entity-level FGET, and they ignore the label noises in the seed entities. AFET [102], a mention-level FGET method, reduces the label noises by separating the loss functions for *clean* and *noisy* seed examples. It embeds hand-crafted features and type correlation in a common dimensional space. In addition to the text features and type labels, CoType [94] also embeds the relation phrases and relation types into the common space. It selects types by computing the cosine similarity between entity (relation) mention embeddings and type label embeddings. But the loss function parameters for both methods need to be tuned on a different corpus.

5.3 Supervised Approaches: Independent Models

Most of the supervised typing models apply a classifier on each mention independently. Such independent models do not consider the type relatedness between neighbouring mentions in the same document. Most methods for mention-level FGET are of this category. Entity-level FGET methods apply three variants [82], [28], [36], [171], [88]: (i) Use the same classifier with entity-level feature representations of the entity as input; (ii) Treat each resolved mention as mention-level FGET, then combine the types of the same entity in different contexts; (iii) Combine the predictions of (i) and (ii).

FGET is a hierarchical classification problem. According to the categorization of hierarchical classifiers by Silla and Freitas [172], there are mainly four categories of hierarchical classifiers used for FGET.

- **Flat**: using a single multi-class classifier for all types.
- **Local**: using a binary classifier for each type, enforcing label consistency at inference time.
- **Local per Parent Node**: using a multi-class classifier for each parent type node; the target classes are the children types.
- **Global**: using a single multi-class classifier trained with a loss function that considers label similarity.

The characteristics of these classifiers are summarized in Table 5.

TABLE 5
Summary of characteristics of different supervised approaches

Model	Approach	Advantages	Disadvantages
Independent	Flat	Simplicity;	Ignores the type hierarchy; Not applicable to ultra-fine typing;
	Local	Simplicity; Higher precision;	Label inconsistency; Uses a large number of classifiers;
	Local per Parent Node	Consistent labels;	Error Propagation;
	Global	Considers label hierarchy;	Computationally more expensive;
Collective	-	Using document-level relatedness;	Data scarcity;
Joint	-	Using knowledge from a KB;	Data scarcity;

5.3.1 *Flat Classifiers*

Flat classifiers ignore the type hierarchy and treat FGET as a multi-class classification problem. For example, Fleischman and Hovy [85] employed a decision tree model trained with C4.5 algorithm [173] to classify person names into 8 types.

Ling and Weld [8] treated the mention-level FGET as a multi-class multi-label classification, and employed a classic linear classifier, Perceptron [174] as their flat classifier.

$$\hat{t} = \arg\max_{t} w^T \cdot f(x, t) \tag{4}$$

where $\hat{t}$ is a predicted type, $f(x, t)$ is the sparse feature vector of a mention m with a type $t \in T$, and w is the weight vectors of the feature functions. Apparently, such a classifier does not consider the compatibility between multiple types assigned to the same entity. The skewness of the corpus (217 of the 562 entities in the test data are labelled with only *Person*) makes this model sensitive to label noises in training examples.

Similar linear classifiers for FGET [153], [26], [106], [104] use the dot product between feature embeddings and label embeddings. The inference strategy is different, they either select the top-k scored types and greedily remove inconsistent types (i.e., not on the same type path) [26], [106] or recursively select the type with highest score among siblings till a leaf type is encountered [104]. It should be noted that mapping hand-crafted feature vectors into low-dimensional space may cause the training to converge only to a local optimum solution because of the non-convexity of the loss functions [153], [26], [106].

Another approach uses softmax classifier or multi-class maximum entropy model [175] as flat classifier for FGET [29], [27], [150]. This approach usually transforms a multi-label training example into examples for multi-class classification, i.e., convert a training example with multiple labels into multiple examples with a single type label. For example, mention "Canada" with coarse-grained types *location* and *organization* are transformed into two training examples, one with fine-grained type of *location* and the other with type of *organization*. The inference strategy is selecting all types whose probability exceeds a threshold [29].

However, the experimental results show that the predictions on top-level types are more reliable than that on lower-level types [29], [26]. This indicates the performance of flat classifiers is sensitive to the number of types.

5.3.2 *Local Classifiers*

Local classifiers make binary local predictions for each type independently. The local classifiers for FGET include: binary Support Vector Machines (SVM) [38], [75], maximum entropy classifier [175] or logistic regression classifier [97], [29], [108], bi-linear score [125], sigmoid Multi-Layer Perceptrons (MLP) [82], [100], [123], [101], [76], [28], [136], [119], [77], [88], [118], [171], [149], [124], [120].

According to the survey of Silla and Freitas [172], the local classifiers usually perform better than flat classifiers. Gillick *et al.* [29] showed that local classifiers outperformed flat classifiers based on their empirical settings. However, defining positive and negative training examples for local binary classifiers is not trivial. Moreover, inference strategies need to be devised to enforce label consistency (e.g., *person* and *artist* are consistent, while *location* and *artist* are not consistent) as the local classifiers make predictions independently.

The positive examples for type t are generally from the examples labelled with t or its descendants in the taxonomy [29], [38], [97]. For example, a mention labelled *person/political_figure* is considered a positive example for *person*.

Negative examples are defined in three ways: (i) Examples of all the other sibling types; (ii) Examples of all the other types at the same level; (iii) Examples of all the other types. HYENA [38], [75] used negative examples (i) to train their binary SVM classifier. A child type *Others* was added to each of the non-leaf types, since the child types of a parent type do not necessarily cover all possible subtypes. Positive samples for *Others* are instances of type t that do not belong to any of its sibling types. When Gillick *et al.* [29] experimented with negative examples (i)-(iii), they found that negative examples (ii) was the most effective way for training local classifiers.

Different inference strategies are devised to enforce the label consistency between local classifiers. Gillick *et al.* [29] considered three inference strategies: (i) Independently assign all types whose probability exceeds the threshold, which may lead to multiple type paths or no path on a taxonomy; (ii) Multiply the probabilities on a type path, which ensures the parent type is also assigned with child type; (iii) Compute the marginalization of a type using the sum-product algorithm on all possible type paths, and then employ strategy (i). Their experimental results showed that strategy (iii) is the best. Considering the fact that the prediction threshold for each SVM can be highly type-dependent, HYENA [75] trained a meta-classifier to select the optimal number of top-n types for each mention during inference. Ekbal *et al.* [97] experimented with level-wise strategy and global strategy. The level-wise strategy selected the type with the highest probability among types of the same level.

The global strategy was similar to the strategy (ii) of [29]. Their experimental results preferred the level-wise strategy.

5.3.3 Local per Parent Node

Local per Parent Node (LPPN) approach makes top-down predictions. Starts from the virtual root of a type taxonomy tree, LPPN implements a multi-class classifier for a parent type to determine which child type should be assigned to the mention. The assigned child type is then treated as a new parent type and a multi-class classifier is implemented for it. This process continues until a leaf type is assigned to the mention. Thus, LLPN always generates consistent type labels. Compared with binary local classifiers, LPPN only implements multi-class classifiers for parent types and a virtual root type.

Karn *et al.* [117] proposed an encoder-decoder architecture for the mention-level FGET. The averaged mention embedding was used to initialize the state s_0 of the decoder, a standard RNN with attentive mechanism whose output at each step represents a node on the predicted type path. At each step, the attentive weight of each context vector was updated based on the state vector of the last step, and the output was computed using a MLP with element-wise sigmoid: $\mathbf{t}_i = \sigma(\mathbf{s}_{i-1}, \mathbf{c}_i)$. The next state was computed as $\mathbf{s}_i = f(\mathbf{s}_{i-1}, \mathbf{t}_i, \mathbf{c}_i)$. For example, the output at each step could be t_0=<SOL> (Start of Label), t_1=*Person*, t_2 =*political_figure*.

The advantage of this classifier is that it always predicts consistent types (types on a type path). The disadvantage is that the errors at the first step may cause all the types to be wrong.

5.3.4 Global Classifiers

Using global classifiers, the dependencies and hierarchical relations between different types (e.g., any example belonging to *person/artist* automatically belongs to *person*) can be taken into account [172].

The intuition behind a global model is that a parent type of the true type is more preferable to the other unrelated types. For example, if a mention's true type is *person/artist/actor*, it is better to predict its type as *person/artist* than *location, organization* or their child types. To encourage this preference, the loss function should be modified to penalise the scenario less when predicted types and true types are related. Xu and Barbosa [99] introduced a process called hierarchical loss normalization, which computes the estimated probability as follows.

$$p^*(\hat{t} \mid m, c) = \hat{p}(\hat{t} \mid m, c) + \beta * \sum_{t \in \Gamma} \hat{p}(t \mid m, c) \tag{5}$$

where Γ is the set of ancestor types along the *type-path* of $\hat{t}$, β is a hyperparameter. This value then is re-normalized into a real probability distribution for maximizing the likelihood of a corpus.

Given a featurizer φ that takes as input a sentence x and entity e, Rabinovich and Klein [87] found a set of types T^* using a linear model. The feature functions consider the compatibility of types as follows:

$$\begin{aligned} f(x, e) &= \arg\max_{T'} w^\intercal \varphi(x, e, T') \\ \varphi(x, e, T') &= \sum_{t \in T'} \varphi(x, e, t) + \sum_{t, t' \in T'} \varphi(t, t') \end{aligned} \tag{6}$$

The disadvantage of global models is that they are computationally more expensive.

5.4 Supervised Methods: Collective Models

One weakness of independent models is being unable to take into account correlations between entity mentions in a document. Collective models for FGET type the mentions in a document simultaneously through incorporating relational information.

Rahman and Ng [11] proposed a collective model that exploited the relationship between two mentions by introducing a pairwise factor node in a factor graph [176], where each variable node represented a mention in a document. The pairwise relationship between two mentions was determined by a heuristic coreference resolution [14] process. If two variable nodes are determined to be coreferent, a factor node will connect both nodes, and the compatibility function of the pairwise factor encourages the assignment of the same labels to the two variable nodes. They incorporated hierarchical information of type taxonomy into the definition of compatibility function as follows:

$$\begin{aligned} & f_{pair}(m_i, m_j) \\ =& P(m_i = t_p, m_j = t_q), \text{where } t_p, t_q \in T \\ =& \begin{cases} P_{sup}(sup(t_p) \mid m_i) P_{sup}(sup(t_q) \mid m_j) & \text{if } t_p = t_q \\ 0 & \text{otherwise} \end{cases} \end{aligned} \tag{7}$$

where $sup(t_p)$ is the parent type of t_p in the hierarchical taxonomy, and $P_{sup}(sup(t_p) \mid m_i)$ is the probability that mention m_i is type $sup(t_p)$ according to the classification model P_{sup}, a Maximum Entropy model determining the coarse-grained types of entity mentions.

However, this collective model only considered the coreference relations between entity mentions of a document.

5.5 Supervised Methods: Joint Models

As we mentioned, the tasks of entity analysis are highly interdependent. Joint models tackle FGET and other tasks jointly to capture more information for making globally optimized decisions [20], [18], [16], [19]. These models are mainly based on a factor graph model [16], [19], and their experimental results show that joint models improve performance on all the tasks incorporated.

Pantel *et al.* [177] proposed a graph based generative model to jointly model user intent and query entity types. The model was trained by maximizing the probability of observing a large collection of real-world queries and their clicked hosts. This method could only type the entities that appear in their web queries. Singh *et al.* [19] used the factor graph model to represent the dependencies between entity typing, relation extraction and coreference resolution. Instead of training all the factors jointly, they used a *piece-wise training* approach to estimate the parameters of the model. Parameters for each factor were learned independently by maximizing the piece-wise likelihood.

Durrett and Klein [16] tackled coreference resolution, entity typing and entity linking simultaneously using a conditional random fields (CRF) model. Unary factors in the CRF model defined the features for solving each task independently. The binary and ternary factors defined the features that captured the interactions or constraints between tasks. The model was trained by maximizing the joint probability of three labels for all mentions in the corpus. However, for both learning and decoding, exact inference would be intractable because of the loops in the factor graph. Although belief propagation can perform efficient inference, it would still be computationally exorbitant due to the ternary factor. Thus, they used a pre-trained coarse model to prune 90% of the possible coreference arcs. However, this joint model requires a training corpus that has labels of all three tasks, and this is not readily available.

6 Tail Types and New Entities

In this section, we review the special considerations for tail types and new entities.

Tail types, which form the *long tail* of distributions of types in a corpus or KB, are those semantic types that have few or even no training examples in a corpus. It was noted that the performance of FGET models on a type is highly dependent on the number of training instances for that type [153], [168]. The performance of HYENA [38] on the 5 top-level types is higher than on the tail types. Choi *et al.* [77] observed that their typing model often preferred coarse-grained types instead of ultra-fine-grained types, because many ultra-fine-grained types have few instances. Based on the WSABIE [154] method, a joint representation of features and type labels can improve performance on tail types [26], [104]. Using Glove embeddings directly as type embeddings [149] is simple but effective for tail types. Pre-trained label embeddings that encode semantic relations and dependency between types [106], [120], [143], label embeddings in box space [144] and label embeddings from Wikipedia [125] or label components embeddings [141] are proven to be effective for tail types.

New entities are those neither included in nor linked to a KB. It is noted that entity linking systems perform poor on those new entities [8], thus the entity-level FGET systems are unlikely to handle those new entities competently. One category of methods is using linked entities as distant supervision for typing the new entities [45], [169]. They both leverage the relation phrase patterns, extracted from linked entities with types, to predict the types of new entities. Another category of methods utilizes the compositional nature of entity mentions to improve performance on uncommon (unseen) entity mentions [27], [28]. It has been also found that fuzzy linking an entity mention to a collection of Wikipedia entities is effective for new entities and tail types [164].

The performance of FIGMENT [82] on tail types and new entities is poor, but the authors proposed the following suggestions for improvement: (i) Training on larger corpora; (ii) Refining type taxonomy to cover more entities; (iii) Exploiting hierarchical relations of types; (iv) Exploring more effective feature representations. Multi-level representations [28] include entity-level, word-level, subword-level, character-level representations, etc. The performance of multi-level representations on tail types is more than 6% higher than standard entity-level representations. The performance on tail entities is improved as well.

7 Evaluation and Datasets

In this section, we summarise some issues related to evaluation metrics and datasets for FGET.

7.1 Evaluation Metrics

FGET predicts a type path on the hierarchical taxonomy for each mention, and all the subtypes on the path can be viewed as one correct label. Thus, the target labels for each mention are a set of types. Let t_m denote the golden true type set for mention m, $\hat{t}_m$denote the type set predicted by FGET system, P denote the mentions detected, and G denote the mentions of golden truth. The metrics with different granularity are defined as follows [8]:

- *Strict*: The predicted type is considered correct only if the type set is exactly the same as the golden truth.

$$precision = \frac{|\{mentions\ whose\ \hat{t}_m = t_m\}|}{|\{mentions\ detected\}|} \tag{8}$$

$$recall = \frac{|\{mentions\ whose\ \hat{t}_m = t_m\}|}{|\{mentions\ golden\ truth\}|} \tag{9}$$

- *Loose Macro*: The precision and recall scores are computed independently for each mention, and then the average scores over all mentions is denoted as overall metrics.

$$precision = \frac{1}{|P|}\sum_{m \in P}\frac{|\hat{t}_m \cap t_m|}{\hat{t}_m} \tag{10}$$

$$recall = \frac{1}{|G|}\sum_{m \in G}\frac{|\hat{t}_m \cap t_m|}{t_m} \tag{11}$$

- *Loose Micro*: The precision and recall scores are measured globally across all mentions.

$$precision = \frac{\sum_{m \in P}|\hat{t}_m \cap t_m|}{\sum_{m \in P}|\hat{t}_m|} \tag{12}$$

$$recall = \frac{\sum_{m \in G}|\hat{t}_m \cap t_m|}{\sum_{m \in G}|t_m|} \tag{13}$$

These three methods for computing precision and recall have been widely used in evaluation of FGET systems [123], [8], [119], [99].

7.2 Datasets

7.2.1 Datasets in English

Widely used datasets for FGET include: BBN [78], OntoNotes [81], [29] and ACE 2005 [178]. The Wikilinks [92] and CoNLL-YAGO [6], [80] can also be transformed for FGET since they have gold entity links.

The FIGER dataset [8] from Wikipedia includes 2 million sentences and 18 manually annotated news reports. The WiFiNE [61] corpus annotated more than the anchored mentions in Wikipedia. The ultra-fine entity typing dataset [77] has 10,331 labels and most of them are defined as free form text phrases.

MedMentions [88] has 246k concepts annotated with links to the UMLS [179] ontology, which includes over 3.5 million medical concepts. ScienceExamCER [180] is a corpus of 133k mentions in the science exam domain with 601 types.

7.2.2 Datasets in Other Languages

Studies of FGET for other languages are scarce. Mai *et al.* [110] applied the LSTM+CNN+CRF model on a FGET dataset in Japanese (19,594 sentences labeled using Sekine [73] taxonomy). Lee *et al.* [181] used BERT-base-Chinese as context encoder and performed experiments on a FGET dataset in Chinese (4,800 mentions labeled with free-form types). Ruppenhofer *et al.* [182] experimented with a BERT-base-German-cased based sequence tagger on a FGET dataset in German (30 types).

8 Summary and Future Directions

In this paper, we present a comprehensive and extensive review of fine-grained entity typing in the following aspects: the type taxonomy, the methods for automatically labeling corpus, the representation of features, typing models, methods for tail types and evaluation metrics. How to efficiently embed features and reduce the negative impact of label noises and skewed training examples is challenging. Embedded features and deep neural networks have been proven to be an effective solution for discerning those label noises. There is a paucity of research on the semantic type relatedness between entity mentions in a document. The document-level entity relatedness has been exploited in entity linking and proved to be effective [183], [184].

Although there has been plenty of research on FGET as presented in this review, we believe that more work still needs to be done to substantially improve the performance of FGET for real world applications. We propose the following research directions for FGET.

Firstly, we can employ more effective feature encoders to learn feature embeddings. Most neural network based FGET systems employ shallow neural networks (such as CNN or *Bi*LSTM) to learn the representations of mentions and contexts. We believe the Wikipedia pre-trained language model [185] that encode knowledge into Transformer model parameters can be more effective for feature embedding.

Secondly, we can incorporate the document-level relatedness between semantic types into a typing model to make collective decisions, i.e., a collective model for FGET [11]. The global model for entity linking [186], [183], [184], which links the entity mentions collectively by considering the coherence between the referenced entities in a document, has been proven to be effective for improving performance. A deep learning based collective model for FGET can be a promising direction.

Thirdly, a joint model for FGET and other entity analysis tasks is also a promising research direction, e.g., jointly tackle the entity linking and mention typing tasks with a unified model. Actually, plenty of research on joint models for entity analysis tasks has been conducted [16], [23], [24], [94], but most of current models are solely based on factor graph models with hand-crafted features. A multi-task neural network for FGET and other entity analysis tasks may learn more sufficient feature representations and capture the relatedness between these tasks.

Acknowledgement

This work was partially supported by the 2020 Catalyst: Strategic NZ-Singapore Data Science Research Programme Fund, MBIE, New Zealand.

References

[1] Ralph Grishman and Beth Sundheim, “Message under- standing conference - 6: A brief history,” in *Proceedngs of COLING-96*, 1996.

[2] Oren Etzioni, Michael J. Cafarella, Doug Downey, Ana-Maria Popescu, Tal Shaked, Stephen Soderland, Daniel S. Weld, and Alexander Yates, “Unsupervised named-entity extraction from the web: An experimental study,” *Artificial Intelligence*, vol. 165, no. 1, pp. 91–134, 2005.

[3] David Nadeau and Satoshi Sekine, “A survey of named entity recognition and classification,” *Lingvisticae Investigationes*, vol. 30, no. 1, pp. 3–26, 2007.

[4] Zornitsa Kozareva, “Bootstrapping named entity recognition with automatically generated gazetteer lists,” in *Proceedings of EACL*, 2006.

[5] J. Li, A. Sun, J. Han, and C. Li, “A survey on deep learning for named entity recognition,” *IEEE Transactions on Knowledge and Data Engineering*, 2020.

[6] Erik F. Tjong Kim Sang and Fien De Meulder, “Introduction to the CoNLL-2003 shared task: Language-independent named entity recognition,” in *Proceedings of HLT-NAACL*, 2003.

[7] Mike Mintz, Steven Bills, Rion Snow, and Dan Jurafsky, “Distant supervision for relation extraction without labeled data,” in *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, 2009, pp. 1003–1011.

[8] Xiao Ling and Daniel S Weld, “Fine-grained entity recognition,” in *Proceedngs of Association for the Advancement of Artificial Intelligence*, 2012.

[9] Rajarshi Das, Manzil Zaheer, Siva Reddy, and Andrew McCallum, “Question answering on knowledge bases and text using universal schema and memory networks,” in *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*, 2017.

[10] Lihan Chen, Jiaqing Liang, Chenhao Xie, and Yanghua Xiao, “Short text entity linking with fine-grained topics,” in *CIKM'18*, 2018-10-22.

[11] Altaf Rahman and Vincent Ng, “Inducing finegrained semantic classes via hierarchical and collective classification,” in *Proceedings of COLING*, 2010.

[12] Radu Florian, Hongyan Jing, Nanda Kambhatla, and Imed Zitouni, “Factorizing complex models: a case study in mention detection,” in *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics*, 2006-07-17.

[13] Thien Huu Nguyen, Avirup Sil, Georgiana Dinu, and Radu Florian, “Toward mention detection robustness with recurrent neural networks,” in *Proceedings of IJCAI Workshop on Deep Learning for Artificial Intelligence*, 2016.

[14] Vincent Ng, “Machine learning for enity coreference resolution: A retrospective look at two decades of research,” in *Proceedings of the 31st AAAI conference on Artificial Intelligence*, 2017.

[15] Wei Shen, Jianyong Wang, and Jiawei Han, “Entity linking with a knowledge base: Issues, techniques, and solutions,” *IEEE Transactions on Knowledge and Data Engineering*, vol. 27, no. 2, pp. 443–460, 2015.

[16] Greg Durrett and Dan Klein, “A joint model for entity analysis: Coreference, typing, and linking,” in *In Transactions of the Association for Computational Linguistics*, 2014.

[17] David Nadeau, “Semi-supervised named entity recognition: Learning to recognize 100 entity types with little supervision,” 2007.

[18] Gang Luo, Xiaojiang Huang, Chin-Yew Lin, and Zaiqing Nie, “Joint named entity recognition and disambiguation,” in *Proceedings ofthe 2015 Conference on Empirical Methods in Natural Language Processing*, 2015, pp. 879–888.

[19] Sameer Singh, Sebastian Riedel, Brian Martin, Jiaping Zheng, and Andrew McCallum, “Joint inference of entities, relations, and coreference,” in *Proceedings of the 2013 Workshop on Automated Knowledge Base Construction*, 2013.

[20] Dat Ba Nguyen, Martin Theobald, and Gerhard Weikum, "J-NERD: Joint named entity recognition and disambiguation with rich linguistic features," in *Transactions of the Association for Computational Linguistics*, Hwee Tou Ng, Ed., vol. 4, 2016, pp. 215–229.

[21] Qi Li and Heng Ji, "Incremental joint extraction of entity mentions and relations," in *Proceedings ofthe 52nd Annual Meeting ofthe Association for Computational Linguistics*, 2014.

[22] Rosa Stern, Benoît Sagot, and Frédéric Béchet, "A joint named entity recognition and entity linking system," in *EACL 2012 Workshop on Innovative hybrid approaches to the processing of textual data*, 2012.

[23] Andrew Carlson, Justin Betteridge, Richard C. Wang, Estevam R. Hruschka Jr, and Tom M. Mitchell, "Coupled semi-supervised learning for information extraction," in *In Proc. of WSDM*, 2010.

[24] Nitish Gupta, Sameer Singh, and Dan Roth, "Entity linking via joint encoding of types, descriptions, and context," in *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, 2017-09-07, pp. 2671–2680.

[25] Luciano Del Corro, Abdalghani Abujabal, Rainer Gemulla, and Gerhard Weikum, "Finet: Context-aware fine-grained named entity typing," in *Proceedngs of the conference on Empirical Methods in Natural Language Processing*, 2015.

[26] Dani Yogatama, Daniel Gillick, and Nevena Lazic, "Embedding methods for fine grained entity type classification," in *Proceedngs of Association for Computational Linguistics (ACL)*, 2015.

[27] Li Dong, Furu Wei, Hong Sun, Ming Zhou, and Ke Xu, "A hybrid neural model for type classification of entity mentions," in *Proceedngs of International Joint Conference on Artificial Intelligence*, 2015.

[28] Yadollah Yaghoobzadeh and Hinrich Schutze, "Multi-level representations for fine-grained typing of knowledge base entities," in *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics*, 2017, pp. 578–589.

[29] Daniel Gillick, Nevena Lazic, Kuzman Ganchev, Jesse Kirchner, and David Huynh, "Contextdependent fine-grained entity type tagging," *arXiv preprint arXiv:1412.1820*, 2014.

[30] Jakob Uszkoreit and Thorsten Brants, "Distributed word clustering for large scale class-based language modeling in machine translation," in *Proceedings of ACL-08: HLT*, 2008.

[31] M. A. Hearst, "Automatic acquisition of hyponyms from large text corpora," in *Proceedings of the 14th conference on Computational linguistics-Volume 2*. Association for Computational Linguistics, 1992, pp. 539–545.

[32] Z. Kozareva, E. Riloff, and E. Hovy, "Semantic class learning from the web with hyponym pattern linkage graphs," *Proceedings of ACL-08: HLT*, pp. 1048–1056, 2008.

[33] Peter F. Brown, Peter V. deSouza, Robert L. Mercer, Vincent J. Della Pietra, and Jenifer C. Lai, "Class-based n-gram models of natural language," *Computational Linguistics*, vol. 18, no. 4, 1992.

[34] R. Navigli, "Word sense disambiguation: A survey," *ACM computing surveys (CSUR)*, vol. 41, no. 2, p. 10, 2009.

[35] S. A. Caraballo, "Automatic construction of a hypernym-labeled noun hierarchy from text," in *Proceedings of the 37th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, 1999-06, pp. 120–126.

[36] Yadollah Yaghoobzadeh, Heike Adel, and Hinrich Schutze, "Noise mitigation for neural entity typing and relation extraction," in *Proceedngs of European Chapter of Association for Computational Linguistics*, 2017.

[37] Limin Yao, Sebastian Riedel, and Andrew McCallum, "Collective cross-document relation extraction without labelled data," in *In Proc. of EMNLP*, 2010.

[38] Mohamed Amir Yosef, Sandro Bauer, Johannes Hoffart, Marc Spaniol, and Gerhard Weikum, "HYENA: Hierarchical type classification for entity names," in *Proceedngs of COLING 2012*, 2012, pp. 1361–1370.

[39] W. Shen, J. Han, J. Wang, X. Yuan, and Z. Yang, "Shine+: A general framework for domain-specific entity linking with heterogeneous information networks," *IEEE Transactions on Knowledge and Data Engineering*, vol. 30, no. 2, pp. 353–366, 2017.

[40] Jonathan Raiman and Olivier Raiman, "Deeptype: Multilingual entity linking by neural type system evolution," in *In Association for the Advancement of Artificial Intelligence*, 2018.

[41] Zhiheng Huang, Marcus Thint, and Zengchang Qin, "Question classification using head words and their hypernyms," in *EMNLP 2008*, 2008.

[42] Changki Lee, Yi-Gyu Hwang, Hyo-Jung Oh, Soojong Lim, Jeong Heo, Chung-Hee Lee, Hyeon-Jin Kim, Ji-Hyun Wang, and Myung-Gil Jang, "Fine-grained named entity recognition using conditional random fields for question answering," in *Asia Information Retrieval Symposium AIRS 2006: Information Retrieval Technology pp*, 2006.

[43] Changki Lee, Yi-Gyu Hwang, and Myung-Gil Jang, "Fine-grained named entity recognition and relation extraction for question answering," in *SIGIR 2007 Proceedings*, 2007.

[44] Semih Yavuz, Izzeddin Gur, Yu Su, Mudhakar Srivatsa, and Xifeng Yan, "Improving semantic parsing via answer type inference," in *Proceedngs of Empirical Methods in Natural Language Processing*, 2016.

[45] Thomas Lin and Oren Etzioni, "No noun phrase left behind: detecting and typing unlinkable entities," in *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, 2012, pp. 893–903.

[46] Johannes Welbl, Pontus Stenetorp, and Sebastian Riedel, "Constructing datasets for multi-hop reading comprehension across documents," 2017.

[47] Marta Recasens, Marie-Catherine de Marneffe, and Christopher Potts, "The life and death of discourse entities: Identifying singleton mentions," in *In Proc. of NAACL.*, 2013.

[48] Benjamin Roth, Nicholas Monath, David Belanger, Emma Strubell, Patrick Verga, and Andrew McCallum, "Building knowledge bases with universal schema: Cold start and slot-filling approaches," in *TAC Workshop*, 2015.

[49] Heng Ji and Ralph Grishman, "Knowledge base population: Successful approaches and challenges," in *Proceedings ofthe 49th Annual Meeting ofthe Association for Computational Linguistics*, 2011, pp. 1148–1158.

[50] Marius Paşca, "Acquisition of categorized named entities for web search," in *Proceedings of the ACM International Conference on Information and Knowledge Management (CIKM)*, 2004.

[51] Krisztian Balog and Robert Neumayer, "Hierarchical target type identification for entity-oriented queries," in *Proceedngs of the Conference on Information and Knowledge Management*, 2012.

[52] Jeffrey Dalton, Laura Dietz, and James Allan, "Entity query feature expansion using knowledge base links," in *SIGIR'14*, 2014-07-06.

[53] T. Cheng, H. W. Lauw, and S. Paparizos, "Entity synonyms for structured web search," *IEEE Transactions on Knowledge and Data Engineering*, vol. 24, no. 10, pp. 1862–1875, 2011.

[54] Denghao Ma, Yueguo Chen, Kevin Chen-Chuan Chang, Xiaoyong Du, Chuanfei Xu, and Yi Chang, "Leveraging fine-grained wikipedia categories for entity search," in *WWW 2018*, 2018-04-23.

[55] Rajarshi Bhowmik and Gerard de Melo, "Generating fine-grained open vocabulary entity type descriptions," in *Proceedings ofthe 56th Annual Meeting ofthe Association for Computational Linguistics*, 2018-07-15.

[56] Parvez, Md Rizwan, Chakraborty, Saikat, Ray, Baishakhi, and Chang, Kai-Wei, "Building language models for text with named entities," in *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, 2018, pp. 2373–2383.

[57] Xiao Yu, Xiang Ren, Yizhou Sun, Quanquan Gu, Bradley Sturt, Urvashi Khandelwal, Brandon Norick, and Jiawei Han, "Personalized entity recommendation: A heterogeneous information network approach," in *Proceedings of the 7th ACM international conference on Web search and data mining*, 2014.

[58] George A. Miller, "WordNet: A lexical database for english," *Communications of the ACM*, vol. 38, no. 11, pp. 39–41, 1995.

[59] Ndapandula Nakashole, Tomasz Tylenda, and Gerhard Weikum, "Fine-grained semantic typing of emerging entities," in *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics*, 2013.

[60] Joel Nothman, James R. Curran, and Tara Murphy, "Transforming wikipedia into named entity training data," in *Proceedngs of the Australasian Language Technology Association Workshop*, 2008.

[61] Abbas Ghaddar and Philippe Langlais, "Transforming wikipedia into a large-scale fine-grained entity type corpus," in *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*, 2018.

[62] F. Hou, R. Wang, J. He, and Y. Zhou, "Improving entity linking through semantic reinforced entity embeddings," in *Proceedings*

of the 58th Annual Meeting of the Association for Computational Linguistics. Online: Association for Computational Linguistics, Jul. 2020, pp. 6843–6848.

[63] Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor, "Freebase: a collaboratively created graph database for structuring human knowledge," in *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*, 2008-06-09.

[64] Xiang Ren, Ahmed El-Kishky, Chi Wang, Fangbo Tao, Clare R. Voss, Heng Ji, and Jiawei Han, "ClusType: Effective entity recognition and typing by relation phrase-based clustering," in *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2015.

[65] Sören Auer, Christian Bizer, Georgi Kobilarov, Jens Lehmann, Richard Cyganiak, and Zachary Ives, "DBpedia: A nucleus for a web of open data," in *International Semantic Web Conference Asian Semantic Web Conference ISWC 2007, ASWC 2007*, 2007.

[66] J. Lehmann, R. Isele, M. Jakob, A. Jentzsch, D. Kontokostas, P. N. Mendes, S. Hellmann, M. Morsey, P. Van Kleef, and S. Auer, "DBpedia–a large-scale, multilingual knowledge base extracted from wikipedia," *Semantic Web*, vol. 6, no. 2, pp. 167–195, 2015.

[67] Joachim Daiber, Max Jakob, Chris Hokamp, and Pablo N. Mendes, "Improving efficiency and accuracy in multilingual entity extraction," in *Proceedings of the 9th International Conference on Semantic Systems (I-Semantics)*, 2013.

[68] Fabian M. Suchanek, Gjergji Kasneci, and Gerhard Weikum, "YAGO: A core of semantic knowledge unifying WordNet and wikipedia," in *WWW 2007*, 2007.

[69] T. Rebele, F. Suchanek, J. Hoffart, J. Biega, E. Kuzey, and G. Weikum, "YAGO: A multilingual knowledge base from wikipedia, wordnet, and geonames," in *International Semantic Web Conference*. Springer, 2016, pp. 177–185.

[70] Mahdisoltani, Farzaneh, Joanna Biega, and Fabian Suchanek, "YAGO3: A knowledge base from multilingual wikipedias," in *In 7th Biennial Conference on Innovative Data Systems Research(CIDR 2015)*, 2015.

[71] J. Hoffart, F. M. Suchanek, K. Berberich, and G. Weikum, "Yago2: A spatially and temporally enhanced knowledge base from wikipedia," *Artificial Intelligence*, vol. 194, pp. 28–61, 2013.

[72] Satoshi Sekine, Kiyoshi Sudo, and Chikashi Noba, "Extended named entity hierarchy," in *LREC 2002*, 2002.

[73] Satoshi Sekine and Chikashi Nobata, "Definition, dictionaries and tagger for extended named entity hierarchy," in *Proceedings of the 4th International Conference on Language Resources and Evaluation (LREC)*, 2004.

[74] Satoshi Sekine, "Extended named entity ontology with attribute information," in *Proceeding of LREC*, 2008.

[75] Mohamed Amir Yosef, Sandro Bauer, Johannes Hoffart, Marc Spaniol, and Gerhard Weikum, "HYENA-live: Fine-grained online entity type classification from natural-language text," in *Proceedings of the 51st Annual Meeting of ACL*, 2013.

[76] Shikhar Murty, Patrick Verga, Luke Vilnis, and Andrew McCallum, "Finer grained entity typing with typenet," in *31st Conference on Neural Information Processing Systems (NIPS 2017)*, 2017.

[77] Eunsol Choi, Omer Levy, Yejin Choi, and Luke Zettlemoyer, "Ultra-fine entity typing," in *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics*, 2018-07-15, pp. 87–96.

[78] Ralph M Weischedel, Ada Brunstein, and Linguistic Data Consortium, "BBN pronoun coreference and entity type corpus," *Linguistic Data Consortium©2005*, 2005.

[79] George R Doddington, Alexis Mitchell, Mark Przybocki, Lance Ramshaw, Stephanie Strassel, and Ralph Weischedel, "The automatic content extraction (ace) program: tasks, data, and evaluation," in *LREC*, 2004.

[80] J. Hoffart, M. A. Yosef, I. Bordino, H. Fürstenau, M. Pinkal, M. Spaniol, B. Taneva, S. Thater, and G. Weikum, "Robust disambiguation of named entities in text," in *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, 2011-07, pp. 782–792.

[81] Ralph Weischedel, Martha Palmer, Mitchell Marcus, Eduard Hovy, Sameer Pradhan, Lance Ramshaw, Nianwen Xue, Ann Taylor, Jeff Kaufman, and Michelle Franchini, "Ontonotes release 5.0 with OntoNotes DB tool v0.999 beta," in *Linguistic Data Consortium*, 2013.

[82] Yadollah Yaghoobzadeh and Hinrich Schutze, "Corpus-level fine-grained entity typing using contextual information," in *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, 2015.

[83] J. Chang, R. T.-H. Tsai, and J. S. Chang, "WikiSense: Supersense tagging of wikipedia named entities based WordNet," in *Proceedings of the 23rd Pacific Asia Conference on Language, Information and Computation*. City University of Hong Kong, 2009-12, pp. 72–81.

[84] Michael Fleischman, "Automated subcategorization of named entities," in *Proc. of the ACL Student Workshop*, 2001.

[85] Michael Fleischman and Eduard Hovy, "Fine grained classification of named entities," in *The 19th International Conference on Computational Linguistics*, 2002.

[86] Simone Paolo Ponzetto and Michael Strube, "Deriving a large scale taxonomy from wikipedia," in *AAAI'07 Proceedings of the 22nd national conference on Artificial intelligence*, 2007-07-22.

[87] Maxim Rabinovich and Dan Klein, "Fine-grained entity typing with high-multiplicity assignments," in *Proceedngs of the 55th Annual Meeting of ACL*, 2017.

[88] Shikhar Murty, Patrick Verga, Luke Vilnis, Irena Radovanovic, and Andrew McCallum, "Hierarchical losses and new resources for fine-grained entity typing and linking," in *Proceedings of the 56th Annual Meeting of ACL*, 2018, pp. 97–109.

[89] O. Bodenreider, "The unified medical language system (UMLS): integrating biomedical terminology," *Nucleic acids research*, vol. 32, pp. D267–D270, 2004.

[90] P. N. Mendes, M. Jakob, A. García-Silva, and C. Bizer, "DBpedia spotlight: shedding light on the web of documents," in *Proceedings of the 7th international conference on semantic systems*. ACM, 2011, pp. 1–8.

[91] M. A. Yosef, J. Hoffart, I. Bordino, M. Spaniol, and G. Weikum, "Aida: An online tool for accurate disambiguation of named entities in text and tables," *Proceedings of the VLDB Endowment*, vol. 4, no. 12, pp. 1450–1453, 2011.

[92] S. Singh, A. Subramanya, F. Pereira, and A. McCallum, "Wikilinks: A large-scale cross-document coreference corpus labeled via links to wikipedia," *University of Massachusetts, Amherst, Tech. Rep. UM-CS-2012*, vol. 15, 2012.

[93] Evgeniy Gabrilovich, Michael Ringgaard, and Amarnag Subramanya, "FACC1: Freebase annotation of ClueWeb corpora, version 1 (release date 2013-06-26, format version 1, correction level 0)," 2013.

[94] Xiang Ren, Zeqiu Wu, Wenqi He, Meng Qu, Clare R. Voss, Heng Ji, Tarek F. Abdelzaher, and Jiawei Han, "Cotype: Joint extraction of typed entities and relations with knowledge bases," in *Proceedngs of World Wide Web Conference.*, 2017.

[95] A. Ghaddar and P. Langlais, "Coreference in wikipedia: Main concept resolution," in *Proceedings of The 20th SIGNLL Conference on Computational Natural Language Learning*, 2016, pp. 229–238.

[96] ——, "WikiCoref: An English coreference-annotated corpus of Wikipedia articles," in *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC'16)*. Portorož, Slovenia: European Language Resources Association (ELRA), May 2016, pp. 136–142.

[97] Asif Ekbal, Eva Sourjikova, Anette Frank, and Simone Paolo Ponzetto, "Assessing the challenge of fine-grained named entity recognition and classification," in *Proceedings ofthe 2010 Named Entities Workshop, ACL 2010*, 2010.

[98] H. Dai, Y. Song, and H. Wang, "Ultra-fine entity typing with weak supervision from a masked language model," in *Proceedings of the 59th Annual Meeting of ACL*. Online: Association for Computational Linguistics, Aug. 2021, pp. 1790–1799.

[99] Peng Xu and Denilson Barbosa, "Neural fine-grained entity type classification with hierarchy-aware loss," in *Proceedings of NAACL-HLT*, 2018-06-01, pp. 16–25.

[100] Sonse Shimaoka, Pontus Stenetorp, Kentaro Inui, and Sebastian Riedel, "An attentive neural architecture for fine-grained entity type classification," in *Proceedings of AKBC*, 2016.

[101] ——, "Neural architectures for fine-grained entity type classification," in *Proceedngs of the 15th Conference of the European Chapter of the Association for Computational Linguistics (EACL) .*, 2017.

[102] Xiang Ren, Wenqi He, Meng Qu, Lifu Huang, Heng Ji, and Jiawei Han, "Afet: Automatic finegrained entity typing by hierarchical partial-label embedding," in *Proceedngs Empirical Methods in Natural Language Processing*, 2016.

[103] Xiang Ren, Wenqi He, Meng Qu, Clare R Voss, Heng Ji, and Jiawei Han, "Label noise reduction in entity typing by heterogeneous partial-label embedding," in *KDD*, 2016.

[104] Abhishek, Ashish Anand, and Amit Awekar, "Fine-grained entity type classification by jointly learning representations and label embeddings," in *Proceedngs of EACL*, 2017, pp. 797–807.

[105] Mihai Surdeanu, Julie Tibshirani, Ramesh Nallapati, and Christopher D. Manning, "Multi-instance multi-label learning for relation extraction," in *Proceedings of the Joint Conference of EMNLP-CoNLL*, 2012.

[106] Yukun Ma, Erik Cambria, and Sa Gao, "Label embedding for zero-shot fine-grained named entity typing." in *Proceedngs of the 26th International Conference on Computational Linguistics: Technical Papers*, 2016, pp. 171–180.

[107] A. Fader, S. Soderland, and O. Etzioni, "Identifying relations for open information extraction," in *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, 2011, pp. 1535–1545.

[108] Masatoshi Suzuki, Koji Matsuda, Satoshi Sekine, Naoaki Okazaki, and Kentaro Inui, "Fine-grained named entity classification with wikipedia article vectors," in *IEEE/WIC/ACM International Conference on Web Intelligence*, 2016.

[109] Maksim Tkatchenko, Alexander Ulanov, and Andrey Simanovsky, "Classifying wikipedia entities into fine-grained classes," in *ICDE Workshops*, 2011.

[110] K. Mai, T.-H. Pham, M. T. Nguyen, T. D. Nguyen, D. Bollegala, R. Sasano, and S. Sekine, "An empirical study on fine-grained named entity recognition," in *Proceedings of the 27th International Conference on Computational Linguistics*. Santa Fe, New Mexico, USA: ACL, Aug. 2018, pp. 711–722.

[111] D. Nadeau, P. D. Turney, and S. Matwin, "Unsupervised named-entity recognition: Generating gazetteers and resolving ambiguity," in *Conference of the Canadian Society for Computational Studies of Intelligence*. Springer, 2006, pp. 266–277.

[112] R. Higashinaka, K. Sadamitsu, K. Saito, T. Makino, and Y. Matsuo, "Creating an extended named entity dictionary from wikipedia," in *Proceedings of COLING 2012*. The COLING 2012 Organizing Committee, 2012-12, pp. 1163–1178.

[113] Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Jauvin, "A neural probabilistic language model," *Journal ofMachine Learning Research*, vol. 3, pp. 1137–1155, 2003.

[114] Ronan Collobert, Jason Weston, L ́eon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa, "Natural language processing (almost) from scratch," *Journal ofMachine Learning Research*, vol. 12, pp. 2493–2537, 2011.

[115] Tomas Mikolov, Greg Corrado, Kai Chen, and Jeffrey Dean, "Efficient estimation of word representations in vector space," in *ICLR Worshop*, 2013.

[116] Jeffrey Pennington, Richard Socher, and Christopher D. Manning, "GloVe: Global vectors for word representation," in *Empirical Methods in Natural Language Processing (EMNLP)*, 2014, pp. 1532–1543.

[117] Sanjeev Karn, Ulli Waltinger, and Hinrich Schutze, "End-to-end trainable attentive decoder for hierarchical entity classification," in *Proceedngs of European Chapter of Association for Computational Linguistics*, 2017, pp. 752–758.

[118] Ji Xin, Yankai Lin, Zhiyuan Liu, and Maosong Sun, "Improving neural fine-grained entity typing with knowledge attention," in *Association for the Advancement of Artificial Intelligence*, 2018.

[119] Sheng Zhang, Kevin Duh, and Benjamin Van Durme, "Fine-grained entity typing through increased discourse context and adaptive classification thresholds," in *Proceedings of the Seventh Joint Conference on Lexical and Computational Semantics*, 2018.

[120] W. Xiong, J. Wu, D. Lei, M. Yu, S. Chang, X. Guo, and W. Y. Wang, "Imposing label-relational inductive bias for extremely fine-grained entity typing," in *Proceedings of the 2019 Conference of the NAACL: Human Language Technologies*. Minneapolis, Minnesota: ACL, Jun. 2019, pp. 773–784.

[121] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean, "Distributed representations of words and phrases and their compositionality," in *Advances in Neural Information Processing Systems*, 2013, pp. 3111–3119.

[122] P. Bojanowski, E. Grave, A. Joulin, and T. Mikolov, "Enriching word vectors with subword information," *Transactions of the Association for Computational Linguistics*, vol. 5, pp. 135–146, 2017.

[123] Yidong Jia, Weiran Xu, Pengda Qin, and Zuyi Bao, "Fine-grained entity typing for knowledge base completion," in *2016 IEEE International Conference on Network Infrastructure and Digital Content (IC-NIDC)*, 2016.

[124] H. Dai, D. Du, X. Li, and Y. Song, "Improving fine-grained entity typing with entity linking," in *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*. Hong Kong, China: Association for Computational Linguistics, Nov. 2019, pp. 6210–6215.

[125] R. Obeidat, X. Fern, H. Shahbazi, and P. Tadepalli, "Description-based zero-shot fine-grained entity typing," in *Proceedings of the 2019 Conference of NAACL: Human Language Technologies, Volume 1 (Long and Short Papers)*. Minneapolis, Minnesota: ACL, Jun. 2019, pp. 807–814.

[126] C. X. Chu, S. Razniewski, and G. Weikum, "ENTYFI: A system for fine-grained entity typing in fictional texts," in *Proceedings of the 2020 Conference on EMNLP: System Demonstrations*. Online: ACL, Oct. 2020, pp. 100–106.

[127] Y. Lin and H. Ji, "An attentive fine-grained entity typing model with latent type representation," in *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*. Hong Kong, China: Association for Computational Linguistics, Nov. 2019, pp. 6197–6202.

[128] T. Mikolov, M. Karafiát, L. Burget, J. Černockỳ, and S. Khudanpur, "Recurrent neural network based language model," in *Eleventh Annual Conference of the International Speech Communication Association*, 2010.

[129] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[130] N. Kalchbrenner, E. Grefenstette, and P. Blunsom, "A convolutional neural network for modelling sentences," in *Proceedings of the 52nd Annual Meeting of the ACL*, Baltimore, Maryland, June 2014.

[131] F. López, B. Heinzerling, and M. Strube, "Fine-grained entity typing in hyperbolic space," in *Proceedings of the 4th Workshop on Representation Learning for NLP (RepL4NLP-2019)*. Florence, Italy: ACL, Aug. 2019, pp. 169–180.

[132] Abhishek, "FgER: Fine-grained entity recognition," in *The Twenty-Third AAAI/SIGAI Doctoral Consortium*, 2018.

[133] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, "Empirical evaluation of gated recurrent neural networks on sequence modeling," *arXiv preprint arXiv:1412.3555*, 2014.

[134] Dzmitry Bahdanau, KyungHyun Cho, and Yoshua Bengio, "Neural machine translation by jointly learning to align and translate," in *ICLR 2015*, 2015.

[135] T. Luong, H. Pham, and C. D. Manning, "Effective approaches to attention-based neural machine translation," in *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. Lisbon, Portugal: Association for Computational Linguistics, September 2015, pp. 1412–1421.

[136] Denghui Zhang, Manling Li, Pengshan Cai, Yantao Jia, and YuanzhuoWang, "Path-based attention neural model for fine-grained entity typing," in *The Thirty-Second AAAI Conference on Artificial Intelligence (AAAI-18)*, 2018.

[137] Q. Le and T. Mikolov, "Distributed representations of sentences and documents," in *International Conference on Machine Learning*, 2014, pp. 1188–1196.

[138] Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer, "Deep contextualized word representations," in *Proceedings of NAACL-HLT 2018*, 2018.

[139] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in neural information processing systems*, 2017, pp. 5998–6008.

[140] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding," in *Proceedings of the 2019 Conference of NAACL*. Minneapolis, Minnesota: ACL, Jun. 2019, pp. 4171–4186.

[141] T. Kato, K. Abe, H. Ouchi, S. Miyawaki, J. Suzuki, and K. Inui, "Embeddings of label components for sequence labeling: A case study of fine-grained named entity recognition," in *Proceedings of the 58th Annual Meeting of ACL: Student Research Workshop*. Online: ACL, Jul. 2020, pp. 222–229.

[142] C. Lothritz, K. Allix, L. Veiber, T. F. Bissyandé, and J. Klein, "Evaluating pretrained transformer-based models on the task of fine-grained named entity recognition," in *Proceedings of the 28th International Conference on Computational Linguistics*. Barcelona,

Spain (Online): International Committee on Computational Linguistics, Dec. 2020, pp. 3750–3760.

[143] T. Zhang, C. Xia, C.-T. Lu, and P. Yu, "MZET: Memory augmented zero-shot fine-grained named entity typing," in *Proceedings of the 28th International Conference on Computational Linguistics*. Barcelona, Spain (Online): International Committee on Computational Linguistics, Dec. 2020, pp. 77–87.

[144] Y. Onoe, M. Boratko, A. McCallum, and G. Durrett, "Modeling fine-grained entity types with box embeddings," in *Proceedings of the 59th Annual Meeting of ACL*. Online: ACL, Aug. 2021, pp. 2051–2064.

[145] M. Joshi, D. Chen, Y. Liu, D. S. Weld, L. Zettlemoyer, and O. Levy, "Spanbert: Improving pre-training by representing and predicting spans," *Transactions of the Association for Computational Linguistics*, vol. 8, pp. 64–77, 2020.

[146] F. Hou, R. Wang, and Y. Zhou, "Transfer learning for fine-grained entity typing," *Knowledge and Information Systems*, vol. 63, no. 4, pp. 845–866, 2021.

[147] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, and V. Stoyanov, "Roberta: A robustly optimized bert pretraining approach," *arXiv preprint arXiv:1907.11692*, 2019.

[148] Z. Yang, Z. Dai, Y. Yang, J. Carbonell, R. R. Salakhutdinov, and Q. V. Le, "Xlnet: Generalized autoregressive pretraining for language understanding," in *Advances in neural information processing systems*, 2019, pp. 5753–5763.

[149] Z. Yuan and D. Downey, "Otyper: A neural architecture for open named entity typing," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 32, no. 1, 2018.

[150] Hailong Jin, Lei Hou, Juanzi Li, and Tiansi Dong, "Attributed and predictive entity embedding for fine-grained entity typing in knowledge bases," in *Proceedings of the 27th International Conference on Computational Linguistics*, 2018-08-20, pp. 282–292.

[151] H. Jin, L. Hou, J. Li, and T. Dong, "Fine-grained entity typing via hierarchical multi graph convolutional networks," in *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*. Hong Kong, China: Association for Computational Linguistics, Nov. 2019, pp. 4969–4978.

[152] W. Ling, C. Dyer, A. W. Black, and I. Trancoso, "Two/too simple adaptations of word2vec for syntax problems," in *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 2015, pp. 1299–1304.

[153] Arvind Neelakantan and Ming-Wei Chang, "Inferring missing entity type instances for knowledge base completion: New dataset and methods," in *The 2015 Annual Conference ofthe North American Chapter ofthe ACL*, 2015.

[154] J. Weston, S. Bengio, and N. Usunier, "Wsabie: Scaling up to large vocabulary image annotation," in *IJCAI*, vol. 11, 2011, pp. 2764–2770.

[155] M. Elsner, E. Charniak, and M. Johnson, "Structured generative models for unsupervised named-entity clustering," in *Proceedings of Human Language Technologies: The 2009 Annual Conference of NAACL*. ACL, 2009, pp. 164–172.

[156] Philipp Cimiano and Johanna Volker, "Towards large-scale, open-domain and ontology-based named entity classification," in *Proceedings of Recent Advances in Natural Language Processing*, 2005, pp. 166–172.

[157] Claudio Giuliano and Alfio Gliozzo, "Instance-based ontology population exploiting named-entity substitution," in *Proceedings of the 22nd International Conference on Computational Linguistics (Coling 2008)*, 2008-08.

[158] Patrick Pantel, Eric Crestan, Arkady Borkovsky, Ana-Maria Popescu, and Vishnu Vyas, "Web-scale distributional similarity and entity set expansion," in *Proceedngs of EMNLP*, 2009.

[159] L. Sarmento, V. Jijkuon, M. de Rijke, and E. Oliveira, ""more like these": Growing entity classes from seeds," in *Proceedings of the Sixteenth ACM Conference on Information and Knowledge Management*. New York, NY, USA: ACM, 2007, pp. 959–962.

[160] L. Yan, X. Han, L. Sun, and B. He, "Learning to bootstrap for entity set expansion," in *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*. Hong Kong, China: ACL, Nov. 2019, pp. 292–301.

[161] M. Paşca, "Weakly-supervised discovery of named entities using web search queries," in *Proceedings of the Sixteenth ACM Conference on Conference on Information and Knowledge Management*, ser. CIKM '07. New York, NY, USA: ACM, 2007, pp. 683–690.

[162] Jialu Liu, Jingbo Shang, Chi Wang, Xiang Ren, and Jiawei Han, "Mining quality phrases from massive text corpora," in *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data*, 2015-06-31.

[163] D. Vallet and H. Zaragoza, "Inferring the most important types of a query: a semantic approach," in *Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval*. ACM, 2008, pp. 857–858.

[164] B. Zhou, D. Khashabi, C.-T. Tsai, and D. Roth, "Zero-shot open entity typing as type-compatible grounding," in *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*. Brussels, Belgium: Association for Computational Linguistics, Oct.-Nov. 2018, pp. 2065–2076.

[165] Anurag Lal, Apoorve Tomer, and C. Ravindranath Chowdary, "SANE: System for fine grained named entity typing on textual data," in *2017 International World Wide Web Conference*, 2017.

[166] A. Lal *et al.*, "Sane 2.0: System for fine grained named entity typing on textual data," *Engineering Applications of Artificial Intelligence*, vol. 84, pp. 11–17, 2019.

[167] P. P. Talukdar, J. Reisinger, M. Pasca, D. Ravichandran, R. Bhagat, and F. Pereira, "Weakly-supervised acquisition of labeled class instances using graph random walks," in *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, 2008-10, pp. 582–590.

[168] Zornitsa Kozareva, Konstantin Voevodski, and Shang-Hua Teng, "Class label enhancement via related instances," in *Proceedings of EMNLP*, 2011.

[169] N. Nakashole, G. Weikum, and F. Suchanek, "Patty: A taxonomy of relational patterns with semantic types," in *Proceedings of the 2012 Joint Conference of EMNLP-CoNLL*. Jeju Island, Korea: ACL, July 2012, pp. 1135–1145.

[170] Limin Yao, Sebastian Riedel, and Andrew McCallum, "Universal schema for entity type prediction," in *Automatic KnowledgeBase Construction Workshop at the Conference on Information and Knowledge Management*, 2013.

[171] Yadollah Yaghoobzadeh, Heike Adel, and Hinrich Schutze, "Corpus-level fine-grained entity typing," *Journal of Artificial Intelligence Research*, vol. 61, pp. 835–862, 2018.

[172] Carlos N. Silla and Alex A. Freitas, "A survey of hierarchical classification across different application domains," *Data Mining and Knowledge Discovery*, vol. 22, no. 1, pp. 31–72, 2011.

[173] J. Quinlan, *C4. 5: programs for machine learning*. San Mateo, CA: Morgan Kaufmann Publishers, 1993.

[174] F. Rosenblatt, "The perceptron: a probabilistic model for information storage and organization in the brain." *Psychological review*, vol. 65, no. 6, p. 386, 1958.

[175] Adam L. Berger, Stephen A. Della Pietra, and Vincent J. Della Pietra, "A maximum entropy approach to natural language processing," *Computational Linguistics*, vol. 22, no. 1, pp. 39–71, 1996.

[176] Charles Sutton and Andrew McCallum, "An introduction to conditional random fields," *Foundations and Trends in Machine Learning*, vol. 4, no. 4, pp. 267–373, 2011.

[177] P. Pantel, T. Lin, and M. Gamon, "Mining entity types from query logs via user intent modeling," in *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics*. ACL, 2012, pp. 563–571.

[178] NIST, "The ACE 2005 (ACE05) evaluation plan," 2005.

[179] O. Bodenreider, "The unified medical language system integrating biomedical terminology," *Nucleic acids research*, vol. 32, no. suppl 1, pp. 267–270, 2005.

[180] H. Smith, Z. Zhang, J. Culnan, and P. Jansen, "ScienceExamCER: A high-density fine-grained science-domain corpus for common entity recognition," in *Proceedings of the 12th Language Resources and Evaluation Conference*. Marseille, France: European Language Resources Association, May 2020, pp. 4529–4546.

[181] C. Lee, H. Dai, Y. Song, and X. Li, "A Chinese corpus for fine-grained entity typing," in *Proceedings of the 12th Language Resources and Evaluation Conference*. Marseille, France: European Language Resources Association, May 2020, pp. 4451–4457.

[182] J. Ruppenhofer, I. Rehbein, and C. Flinz, "Fine-grained named entity annotations for German biographic interviews," in *Proceedings of the 12th Language Resources and Evaluation Conference*. Marseille, France: European Language Resources Association, May 2020, pp. 4605–4614.

[183] Octavian-Eugen Ganea and Thomas Hofmann, "Deep joint entity disambiguation with local neural attention," in *Proceedings of the 2017 EMNLP Conference*, 2017.
[184] P. Le and I. Titov, "Improving entity linking by modeling latent relations between mentions," in *Proceedings of the 56th Annual Meeting of the ACL*. Melbourne, Australia: ACL, Jul. 2018, pp. 1595–1604.
[185] W. Xiong, J. Du, W. Y. Wang, and V. Stoyanov, "Pretrained encyclopedia: Weakly supervised knowledge-pretrained language model," in *International Conference on Learning Representations (ICLR)*, 2020.
[186] Amir Globerson, Nevena Lazic, Soumen Chakrabarti, Amarnag Subramanya, Michael Ringgaard, and Fernando Pereira, "Collective entity resolution with multi-focal attention," in *Proceedings of the 54th Annual Meeting of ACL*, 2016.

Ruili Wang received the Ph.D. degree in computer science from Dublin City University, Dublin, Ireland. He is currently a Professor of Artificial Intelligence and Chair of Research in the School of Natural and Computational Sciences, Massey University, Auckland, New Zealand, where he is the Director of the Centre of Language and Speech Processing. His current research interests include speech processing, language processing, video processing, data mining, and intelligent systems. Dr. Wang serves as a member and an Associate Editor of the editorial boards for international journals, such as the journals of IEEE Transactions on Emerging Topics in Computational Intelligence, Knowledge and Information Systems and Applied Soft Computing.

Feng Hou Received the PhD degree in Computer Science from Massey University, New Zealand. He is now a Research Fellow at Massey University, New Zealand. His research interests include machine learning methods for natural language processing.

Steven Cahan received his PhD from the University of Colorado, Boulder. He is currently a Professor of Financial Accounting at the University of Auckland. He has published over 75 refereed publications in journals such as The Accounting Review, Accounting, Organizations, and Society, and Contemporary Accounting Research. He is currently a Co-Editor for the Journal of Contemporary Accounting and Economics. He is on the Editorial Board for 12 journals, and he regularly serves as a referee for leading journals as well as research funding organizations. He has been President of the Accounting and Finance Association of Australia and New Zealand and is a Life Member of that organisation. He is also a Fellow of Chartered Accountants Australia and New Zealand. He has held roles with international professional bodies such as the International Integrated Reporting Council and International Accounting Education Standards Board.

Li Chen received her PhD from the University of Auckland in 2012 and was placed on the Dean's List. Her research interests are in the areas of financial reporting and disclosure analysis, financial analysts, corporate governance, corporate social responsibility, and integrated reporting.

Xiaoyun Jia received her master degree at University of Auckland (New Zealand), and obtained her Ph.D at Massey University (New Zealand). With a diverse background, her current research interests focus on online behavior analysis, user studies, E-commerce, data mining, human-computer interactions, information systems etc.

Wanting Ji received the Ph.D. degree in computer science from Massey University, Auckland, New Zealand, in 2020. She is currently a lecturer with the School of Information, Liaoning University, Shenyang, China. Her research interests include deep learning and neural language processing.