A Survey on Deep Learning for Named Entity Recognition

Jing Li, Aixin Sun, Jianglei Han, and Chenliang Li

Abstract—Named entity recognition (NER) is the task to identify mentions of rigid designators from text belonging to predefined semantic types such as person, location, organization etc. NER always serves as the foundation for many natural language applications such as question answering, text summarization, and machine translation. Early NER systems got a huge success in achieving good performance with the cost of human engineering in designing domain-specific features and rules. In recent years, deep learning, empowered by continuous real-valued vector representations and semantic composition through nonlinear processing, has been employed in NER systems, yielding stat-of-the-art performance. In this paper, we provide a comprehensive review on existing deep learning techniques for NER. We first introduce NER resources, including tagged NER corpora and off-the-shelf NER tools. Then, we systematically categorize existing works based on a taxonomy along three axes: distributed representations for input, context encoder, and tag decoder. Next, we survey the most representative methods for recent applied techniques of deep learning in new NER problem settings and applications. Finally, we present readers with the challenges faced by NER systems and outline future directions in this area.

Index Terms—Natural language processing, named entity recognition, deep learning, survey

1 Introduction

Named Entity Recognition (NER) aims to recognize mentions of rigid designators from text belonging to predefined semantic types such as person, location, organization etc [1]. NER not only acts as a standalone tool for information extraction (IE), but also plays an essential role in a variety of natural language processing (NLP) applications such as text understanding [2], [3], information retrieval [4], [5], automatic text summarization [6], question answering [7], machine translation [8], and knowledge base construction [9] etc.

Evolution of NER. The term "*Named Entity*" (NE) was first used at the sixth Message Understanding Conference (MUC-6) [10], as the task of identifying names of organizations, people and geographic locations in text, as well as currency, time and percentage expressions. Since MUC-6 there has been increasing interest in NER, and various scientific events (e.g., CoNLL03 [11], ACE [12], IREX [13], and TREC Entity Track [14]) devote much effort to this topic.

Regarding the problem definition, Petasis et al. [15] restricted the definition of named entities: "A NE is a proper noun, serving as a name for something or someone". This restriction is justified by the significant percentage of proper nouns present in a corpus. Nadeau and Sekine [1] claimed that the word "*Named*" restricted the task to only those entities for which one or many *rigid designators* stands for the referent. Rigid designator, defined in [16], include proper names and natural kind terms like biological species and substances. Despite the various definitions of NEs, researchers have reached common consensus on the types of NEs to recognize. We generally divide NEs into two categories: generic NEs (e.g., person and location) and domain-specific NEs (e.g., proteins, enzymes, and genes). In this paper, we mainly focus on generic NEs in English language. We do not claim this article to be exhaustive or representative of all NER works on all languages.

As to the techniques applied in NER, there are four main streams: 1) Rule-based approaches, which do not need annotated data as they rely on hand-crafted rules; 2) Unsupervised learning approaches, which rely on unsupervised algorithms without hand-labeled training examples; 3) Feature-based supervised learning approaches, which rely on supervised learning algorithms with careful feature engineering; 4) Deep-learning based approaches, which automatically discover representations needed for the classification and/or detection from raw input in an end-to-end manner. We brief 1), 2) and 3), and review 4) in detail.

Motivations for conducting this survey. In recent years, deep learning (DL, also named deep neural network) has attracted significant attention due to its success in various domains. Starting with Collobert et al. [17], DL-based NER systems with minimal feature engineering have been flourishing. Over the past few years, a considerable number of studies have applied deep learning to NER and successively advanced the state-of-the-art performance [17]–[21]. This trend motivates us to conduct a survey to report the current status of deep learning techniques in NER research. By comparing the choices of DL architectures, we aim to identify factors affecting NER performance as well as issues

- *J. Li is with the Inception Institute of Artificial Intelligence, United Arab Emirates. This work was done when the author was with Nanyang Technological University, Singapore. E-mail: jli030@e.ntu.edu.sg.*
- *A. Sun is with School of Computer Science and Engineering, Nanyang Technological University, Singapore. E-mail: axsun@ntu.edu.sg.*
- *J. Han is with SAP, Singapore. E-mail: ray.han@sap.com.*
- *C. Li is with School of Cyber Science and Engineering, Wuhan University, China. E-mail:cllee@whu.edu.cn.*

Accepted in IEEE TKDE.

and challenges.

On the other hand, although NER studies have been thriving for a few decades, to the best of our knowledge, there are few reviews in this field so far. Arguably the most established one was published by Nadeau and Sekine [1] in 2007. This survey presents an overview of the technique trend from hand-crafted rules towards machine learning. Marrero et al. [22] summarized NER works from the perspectives of fallacies, challenges and opportunities in 2013. Then Patawar and Potey [23] provided a short review in 2015. The two recent short surveys are on new domains [24] and complex entity mentions [25], respectively. In summary, existing surveys mainly cover feature-based machine learning models, but not the modern DL-based NER systems. More germane to this work are the two recent surveys [26], [27] in 2018. Goyal et al. [27] surveyed developments and progresses made in NER. However, they did not include recent advances of deep learning techniques. Yadav and Bethard [26] presented a short survey of recent advances in NER based on representations of words in sentence. This survey focuses more on the distributed representations for input (e.g., char- and word-level embeddings) and do not review the context encoders and tag decoders. The recent trend of applied deep learning on NER tasks (e.g., multi-task learning, transfer learning, reinforcement leanring and adversarial learning) are not in their servery as well.

Contributions of this survey. We intensely review applications of deep learning techniques in NER, to enlighten and guide researchers and practitioners in this area. Specifically, we consolidate NER corpora, off-the-shelf NER systems (from both academia and industry) in a tabular form, to provide useful resources for NER research community. We then present a comprehensive survey on deep learning techniques for NER. To this end, we propose a new taxonomy, which systematically organizes DL-based NER approaches along three axes: distributed representations for input, context encoder (for capturing contextual dependencies for tag decoder), and tag decoder (for predicting labels of words in the given sequence). In addition, we also survey the most representative methods for recent applied deep learning techniques in new NER problem settings and applications. Finally, we present readers with the challenges faced by NER systems and outline future directions in this area.

2 Background

We first give a formal formulation of the NER problem. We then introduce the widely-used NER datasets and tools. Next, we detail the evaluation metrics and summarize the traditional approaches to NER.

2.1 What is NER?

A named entity is a word or a phrase that clearly identifies one item from a set of other items that have similar attributes [28]. Examples of named entities are organization, person, and location names in general domain; gene, protein, drug and disease names in biomedical domain. NER is the process of locating and classifying named entities in text into predefined entity categories.

Formally, given a sequence of tokens $s = \langle w_1, w_2, ..., w_N \rangle$, NER is to output a list of tuples $\langle I_s, I_e, t \rangle$,

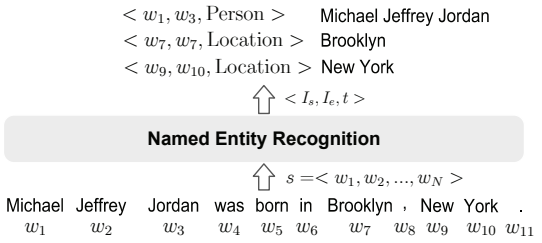


Fig. 1. An illustration of the named entity recognition task.

each of which is a named entity mentioned in s. Here, $I_s \in [1, N]$ and $I_e \in [1, N]$ are the start and the end indexes of a named entity mention; t is the entity type from a predefined category set. Figure 1 shows an example where a NER system recognizes three named entities from the given sentence. When NER was first defined in MUC-6 [10], the task is to recognize names of people, organizations, locations, and time, currency, percentage expressions in text. Note that the task focuses on a small set of coarse entity types and one type per named entity. We call this kind of NER tasks as coarse-grained NER [10], [11]. Recently, some fine-grained NER tasks [29]–[33] focus on a much larger set of entity types where a mention may be assigned multiple fine-grained types.

NER acts as an important pre-processing step for a variety of downstream applications such as information retrieval, question answering, machine translation, etc. Here, we use semantic search as an example to illustrate the importance of NER in supporting various applications. Semantic search refers to a collection of techniques, which enable search engines to understand the concepts, meaning, and intent behind the queries from users [34]. According to [4], about 71% of search queries contain at least one named entity. Recognizing named entities in search queries would help us to better understand user intents, hence to provide better search results. To incorporate named entities in search, entity-based language models [34], which consider individual terms as well as term sequences that have been annotated as entities (both in documents and in queries), have been proposed by Raviv et al. [35]. There are also studies utilizing named entities for an enhanced user experience, such as query recommendation [36], query auto-completion [37], [38] and entity cards [39], [40].

2.2 NER Resources: Datasets and Tools

High quality annotations are critical for both model learning and evaluation. In the following, we summarize widely-used datasets and off-the-shelf tools for English NER.

A tagged corpus is a collection of documents that contain annotations of one or more entity types. Table 1 lists some widely-used datasets with their data sources and number of entity types (also known as tag types). Summarized in Table 1, before 2005, datasets were mainly developed by annotating news articles with a small number of entity types, suitable for coarse-grained NER tasks. After that, more datasets were developed on various kinds of text sources including Wikipedia articles, conversation, and user-generated text (e.g., tweets and YouTube comments and StackExchange posts in W-NUT). The number of tag

TABLE 1
List of annotated datasets for English NER. "#Tags" refers to the number of entity types.

Corpus	Year	Text Source	#Tags	URL
MUC-6	1995	Wall Street Journal	7	https://catalog.ldc.upenn.edu/LDC2003T13
MUC-6 Plus	1995	Additional news to MUC-6	7	https://catalog.ldc.upenn.edu/LDC96T10
MUC-7	1997	New York Times news	7	https://catalog.ldc.upenn.edu/LDC2001T02
CoNLL03	2003	Reuters news	4	https://www.clips.uantwerpen.be/conll2003/ner/
ACE	2000 - 2008	Transcripts, news	7	https://www.ldc.upenn.edu/collaborations/past-projects/ace
OntoNotes	2007 - 2012	Magazine, news, web, etc.	18	https://catalog.ldc.upenn.edu/LDC2013T19
W-NUT	2015 - 2018	User-generated text	6/10	http://noisy-text.github.io
BBN	2005	Wall Street Journal	64	https://catalog.ldc.upenn.edu/LDC2005T33
WikiGold	2009	Wikipedia	4	https://figshare.com/articles/Learning_multilingual_named_entity_recognition_from_Wikipedia/5462500
WiNER	2012	Wikipedia	4	http://rali.iro.umontreal.ca/rali/en/winer-wikipedia-for-ner
WikiFiger	2012	Wikipedia	112	https://github.com/xiaoling/figer
HYENA	2012	Wikipedia	505	https://www.mpi-inf.mpg.de/departments/databases-and-information-systems/research/yago-naga/hyena
N^3	2014	News	3	http://aksw.org/Projects/N3NERNEDNIF.html
Gillick	2016	Magazine, news, web, etc.	89	https://arxiv.org/e-print/1412.1820v2
FG-NER	2018	Various	200	https://fgner.alt.ai/
NNE	2019	Newswire	114	https://github.com/nickyringland/nested_named_entities
GENIA	2004	Biology and clinical text	36	http://www.geniaproject.org/home
GENETAG	2005	MEDLINE	2	https://sourceforge.net/projects/bioc/files/
FSU-PRGE	2010	PubMed and MEDLINE	5	https://julielab.de/Resources/FSU_PRGE.html
NCBI-Disease	2014	PubMed	1	https://www.ncbi.nlm.nih.gov/CBBresearch/Dogan/DISEASE/
BC5CDR	2015	PubMed	3	http://bioc.sourceforge.net/
DFKI	2018	Business news and social media	7	https://dfki-lt-re-group.bitbucket.io/product-corpus/

types becomes significantly larger, e.g., 505 in HYENA. We also list a number of domain specific datasets, particularly developed on PubMed and MEDLINE texts. The number of entity types ranges from 1 in NCBI-Disease to 36 in GENIA.

We note that many recent NER works report their performance on CoNLL03 and OntoNotes datasets (see Table 3). CoNLL03 contains annotations for Reuters news in two languages: English and German. The English dataset has a large portion of sports news with annotations in four entity types (Person, Location, Organization, and Miscellaneous) [11]. The goal of the OntoNotes project was to annotate a large corpus, comprising of various genres (weblogs, news, talk shows, broadcast, usenet newsgroups, and conversational telephone speech) with structural information (syntax and predicate argument structure) and shallow semantics (word sense linked to an ontology and coreference). There are 5 versions, from Release 1.0 to Release 5.0. The texts are annotated with 18 entity types. We also note two Github repositores[1] which host some NER corpora.

There are many NER tools available online with pre-trained models. Table 2 summarizes popular ones for English NER by academia (top) and industry (bottom).

TABLE 2
Off-the-shelf NER tools offered by academia and industry projects.

NER System	URL
StanfordCoreNLP	https://stanfordnlp.github.io/CoreNLP/
OSU Twitter NLP	https://github.com/aritter/twitter_nlp
Illinois NLP	http://cogcomp.org/page/software/
NeuroNER	http://neuroner.com/
NERsuite	http://nersuite.nlplab.org/
Polyglot	https://polyglot.readthedocs.io
Gimli	http://bioinformatics.ua.pt/gimli
spaCy	https://spacy.io/api/entityrecognizer
NLTK	https://www.nltk.org
OpenNLP	https://opennlp.apache.org/
LingPipe	http://alias-i.com/lingpipe-3.9.3/
AllenNLP	https://demo.allennlp.org/
IBM Watson	https://natural-language-understanding-demo.ng.bluemix.net
FG-NER	https://fgner.alt.ai/extractor/
Intellexer	http://demo.intellexer.com/
Repustate	https://repustate.com/named-entity-recognition-api-demo
AYLIEN	https://developer.aylien.com/text-api-demo
Dandelion API	https://dandelion.eu/semantic-text/entity-extraction-demo
displaCy	https://explosion.ai/demos/displacy-ent
ParallelDots	https://www.paralleldots.com/named-entity-recognition
TextRazor	https://www.textrazor.com/named_entity_recognition

2.3 NER Evaluation Metrics

NER systems are usually evaluated by comparing their outputs against human annotations. The comparison can be quantified by either exact-match or relaxed match.

2.3.1 Exact-match Evaluation

NER essentially involves two subtasks: boundary detection and type identification. In "exact-match evaluation" [11], [41], [42], a correctly recognized instance requires a system to correctly identify its boundary and type, simultaneously. More specifically, the numbers of False positives (FP), False negatives (FN) and True positives (TP) are used to compute Precision, Recall, and F-score.

- False Positive (FP): entity that is returned by a NER system but does not appear in the ground truth.
- False Negative (FN): entity that is not returned by a NER system but appears in the ground truth.

1. https://github.com/juand-r/entity-recognition-datasets and https://github.com/cambridgeltl/MTL-Bioinformatics-2016/tree/master/data

- True Positive (TP): entity that is returned by a NER system and also appears in the ground truth.

Precision refers to the percentage of your system results which are correctly recognized. Recall refers to the percentage of total entities correctly recognized by your system.

$$\text{Precision} = \frac{\#TP}{\#(TP+FP)} \qquad \text{Recall} = \frac{\#TP}{\#(TP+FN)}$$

A measure that combines precision and recall is the harmonic mean of precision and recall, the traditional F-measure or balanced F-score:

$$\text{F-score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

In addition, the macro-averaged F-score and micro-averaged F-score both consider the performance across multiple entity types. Macro-averaged F-score independently calculates the F-score on different entity types, then takes the average of the F-scores. Micro-averaged F-score sums up the individual false negatives, false positives and true positives across all entity types then applies them to get the statistics. The latter can be heavily affected by the quality of recognizing entities in large classes in the corpus.

2.3.2 Relaxed-match Evaluation

MUC-6 [10] defines a relaxed-match evaluation: a correct type is credited if an entity is assigned its correct type regardless its boundaries as long as there is an overlap with ground truth boundaries; a correct boundary is credited regardless an entity's type assignment. Then ACE [12] proposes a more complex evaluation procedure. It resolves a few issues like partial match and wrong type, and considers subtypes of named entities. However, it is problematic because the final scores are comparable only when parameters are fixed [1], [22], [23]. Complex evaluation methods are not intuitive and make error analysis difficult. Thus, complex evaluation methods are not widely used in recent studies.

2.4 Traditional Approaches to NER

Traditional approaches to NER are broadly classified into three main streams: rule-based, unsupervised learning, and feature-based supervised learning approaches [1], [26].

2.4.1 Rule-based Approaches

Rule-based NER systems rely on hand-crafted rules. Rules can be designed based on domain-specific gazetteers [9], [43] and syntactic-lexical patterns [44]. Kim [45] proposed to use Brill rule inference approach for speech input. This system generates rules automatically based on Brill's part-of-speech tagger. In biomedical domain, Hanisch et al. [46] proposed ProMiner, which leverages a pre-processed synonym dictionary to identify protein mentions and potential gene in biomedical text. Quimbaya et al. [47] proposed a dictionary-based approach for NER in electronic health records. Experimental results show the approach improves recall while having limited impact on precision.

Some other well-known rule-based NER systems include LaSIE-II [48], NetOwl [49], Facile [50], SAR [51], FASTUS [52], and LTG [53] systems. These systems are mainly based on hand-crafted semantic and syntactic rules to recognize entities. Rule-based systems work very well when lexicon is exhaustive. Due to domain-specific rules and incomplete dictionaries, high precision and low recall are often observed from such systems, and the systems cannot be transferred to other domains.

2.4.2 Unsupervised Learning Approaches

A typical approach of unsupervised learning is clustering [1]. Clustering-based NER systems extract named entities from the clustered groups based on context similarity. The key idea is that lexical resources, lexical patterns, and statistics computed on a large corpus can be used to infer mentions of named entities. Collins et al. [54] observed that use of unlabeled data reduces the requirements for supervision to just 7 simple "seed" rules. The authors then presented two unsupervised algorithms for named entity classification. Similarly, KNOWITALL [9] leveraged a set of predicate names as input and bootstraps its recognition process from a small set of generic extraction patterns.

Nadeau et al. [55] proposed an unsupervised system for gazetteer building and named entity ambiguity resolution. This system combines entity extraction and disambiguation based on simple yet highly effective heuristics. In addition, Zhang and Elhadad [44] proposed an unsupervised approach to extracting named entities from biomedical text. Instead of supervision, their model resorts to terminologies, corpus statistics (e.g., inverse document frequency and context vectors) and shallow syntactic knowledge (e.g., noun phrase chunking). Experiments on two mainstream biomedical datasets demonstrate the effectiveness and generalizability of their unsupervised approach.

2.4.3 Feature-based Supervised Learning Approaches

Applying supervised learning, NER is cast to a multi-class classification or sequence labeling task. Given annotated data samples, features are carefully designed to represent each training example. Machine learning algorithms are then utilized to learn a model to recognize similar patterns from unseen data.

Feature engineering is critical in supervised NER systems. Feature vector representation is an abstraction over text where a word is represented by one or many Boolean, numeric, or nominal values [1], [56]. Word-level features (e.g., case, morphology, and part-of-speech tag) [57]–[59], list lookup features (e.g., Wikipedia gazetteer and DBpedia gazetteer) [60]–[63], and document and corpus features (e.g., local syntax and multiple occurrences) [64]–[67] have been widely used in various supervised NER systems. More feature designs are discussed in [1], [28], [68]

Based on these features, many machine learning algorithms have been applied in supervised NER, including Hidden Markov Models (HMM) [69], Decision Trees [70], Maximum Entropy Models [71], Support Vector Machines (SVM) [72], and Conditional Random Fields (CRF) [73].

Bikel et al. [74], [75] proposed the first HMM-based NER system, named IdentiFinder, to identify and classify names, dates, time expressions, and numerical quantities. In addition, Szarvas et al. [76] developed a multilingual NER system by using C4.5 decision tree and AdaBoostM1 learning algorithm. A major merit is that it provides an opportunity

to train several independent decision tree classifiers through different subsets of features then combine their decisions through a majority voting scheme. Borthwick et al. [77] proposed "maximum entropy named entity" (MENE) by applying the maximum entropy theory. MENE is able to make use of an extraordinarily diverse range of knowledge sources in making its tagging decisions. Other systems using maximum entropy can be found in [78]–[80].

McNamee and Mayfield [81] used 1000 language-related and 258 orthography and punctuation features to train SVM classifiers. Each classifier makes binary decision whether the current token belongs to one of the eight classes, i.e., B- (Beginning), I- (Inside) for *PERSON*, *ORGANIZATION*, *LOCATION*, and *MIS* tags. SVM does not consider "neighboring" words when predicting an entity label. CRFs takes context into account. McCallum and Li [82] proposed a feature induction method for CRFs in NER. Experiments were performed on CoNLL03, and achieved F-score of 84.04% for English. Krishnan and Manning [67] proposed a two-stage approach based on two coupled CRF classifiers. The second CRF makes use of the latent representations derived from the output of the first CRF. We note that CRF-based NER has been widely applied to texts in various domains, including biomedical text [58], [83], tweets [84], [85] and chemical text [86].

3 Deep Learning Techniques for NER

In recent years, DL-based NER models become dominant and achieve state-of-the-art results. Compared to feature-based approaches, deep learning is beneficial in discovering hidden features automatically. Next, we first briefly introduce what deep learning is, and why deep learning for NER. We then survey DL-based NER approaches.

3.1 Why Deep Learning for NER?

Deep learning is a field of machine learning that is composed of multiple processing layers to learn representations of data with multiple levels of abstraction [87]. The typical layers are artificial neural networks which consists of the forward pass and backward pass. The forward pass computes a weighted sum of their inputs from the previous layer and pass the result through a non-linear function. The backward pass is to compute the gradient of an objective function with respect to the weights of a multilayer stack of modules via the chain rule of derivatives. The key advantage of deep learning is the capability of representation learning and the semantic composition empowered by both the vector representation and neural processing. This allows a machine to be fed with raw data and to automatically discover latent representations and processing needed for classification or detection [87].

There are three core strengths of applying deep learning techniques to NER. First, NER benefits from the non-linear transformation, which generates non-linear mappings from input to output. Compared with linear models (e.g., log-linear HMM and linear chain CRF), DL-based models are able to learn complex and intricate features from data via non-linear activation functions. Second, deep learning saves significant effort on designing NER features. The traditional

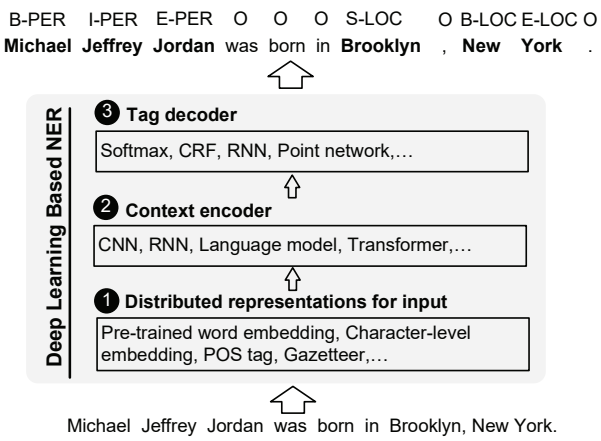


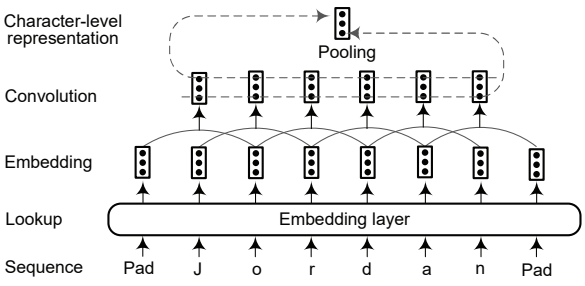
Fig. 2. The taxonomy of DL-based NER. From input sequence to predicted tags, a DL-based NER model consists of distributed representations for input, context encoder, and tag decoder.

feature-based approaches require considerable amount of engineering skill and domain expertise. DL-based models, on the other hand, are effective in automatically learning useful representations and underlying factors from raw data. Third, deep neural NER models can be trained in an end-to-end paradigm, by gradient descent. This property enables us to design possibly complex NER systems.

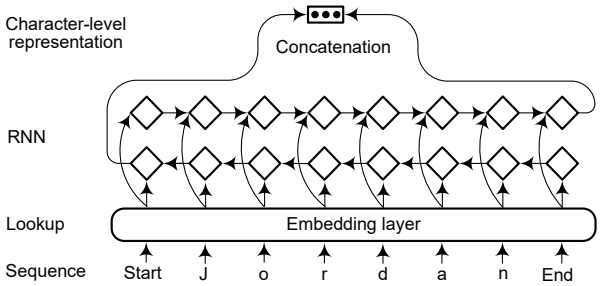
Why we use a new taxonomy in this survey? Existing taxonomy [26], [88] is based on character-level encoder, word-level encoder, and tag decoder. We argue that the description of "word-level encoder" is inaccurate because word-level information is used twice in a typical DL-based NER model: 1) word-level representations are used as raw features, and 2) word-level representations (together with character-level representations) are used to capture context dependence for tag decoding. In this survey, we summarize recent advances in NER with the general architecture presented in Figure 2. *Distributed representations for input* consider word- and character-level embeddings as well as incorporation of additional features like POS tag and gazetteer that have been effective in feature-based based approaches. *Context encoder* is to capture the context dependencies using CNN, RNN, or other networks. *Tag decoder* predict tags for tokens in the input sequence. For instance, in Figure 2 each token is predicted with a tag indicated by B-(begin), I-(inside), E-(end), S-(singleton) of a named entity with its type, or O-(outside) of named entities. Note that there are other tag schemes or tag notations, e.g., BIO. Tag decoder may also be trained to detect entity boundaries and then the detected text spans are classified to the entity types.

3.2 Distributed Representations for Input

A straightforward option of representing a word is *one-hot* vector representation. In one-hot vector space, two distinct words have completely different representations and are orthogonal. *Distributed representation* represents words in low dimensional real-valued dense vectors where each dimension represents a latent feature. Automatically learned from text, distributed representation captures semantic and syntactic properties of word, which do not explicitly present in



(a) CNN-based character-level representation.



(b) RNN-based character-level representation.

Fig. 3. CNN-based and RNN-based models for extracting character-level representation for a word.

the input to NER. Next, we review three types of distributed representations that have been used in NER models: word-level, character-level, and hybrid representations.

3.2.1 Word-level Representation

Some studies [89]–[91] employed word-level representation, which is typically pre-trained over large collections of text through unsupervised algorithms such as continuous bag-of-words (CBOW) and continuous skip-gram models [92]. Recent studies [88], [93] have shown the importance of such pre-trained word embeddings. Using as the input, the pre-trained word embeddings can be either fixed or further fine-tuned during NER model training. Commonly used word embeddings include Google Word2Vec, Stanford GloVe, Facebook fastText and SENNA.

Yao et al. [94] proposed Bio-NER, a biomedical NER model based on deep neural network architecture. The word representation in Bio-NER is trained on PubMed database using skip-gram model. The dictionary contains 205,924 words in 600 dimensional vectors. Nguyen et al. [89] used word2vec toolkit to learn word embeddings for English from the Gigaword corpus augmented with newsgroups data from BOLT (Broad Operational Language Technologies). Zhai et al. [95] designed a neural model for sequence chunking, which consists of two sub-tasks: segmentation and labeling. The neural model can be fed with SENNA embeddings or randomly initialized embeddings.

Zheng et al. [90] jointly extracted entities and relations using a single model. This end-to-end model uses word embeddings learned on NYT corpus by word2vec tookit. Strubell et al. [91] proposed a tagging scheme based on Iterated Dilated Convolutional Neural Networks (ID-CNNs). The lookup table in their model are initialized by 100-dimensional embeddings trained on SENNA corpus by skip-n-gram. In their proposed neural model for extracting entities and their relations, Zhou et al. [96] used the pre-trained 300-dimensional word vectors from Google. In addition, GloVe [97], [98] and fastText [99] are also widely used in NER tasks.

3.2.2 Character-level Representation

Instead of only considering word-level representations as the basic input, several studies [100], [101] incorporated character-based word representations learned from an end-to-end neural model. Character-level representation has been found useful for exploiting explicit sub-word-level information such as prefix and suffix. Another advantage of character-level representation is that it naturally handles out-of-vocabulary. Thus character-based model is able to infer representations for unseen words and share information of morpheme-level regularities. There are two widely-used architectures for extracting character-level representation: *CNN-based* and *RNN-based* models. Figures 3(a) and 3(b) illustrate the two architectures.

Ma et al. [97] utilized a CNN for extracting character-level representations of words. Then the character representation vector is concatenated with the word embedding before feeding into a RNN context encoder. Likewise, Li et al. [98] applied a series of convolutional and highway layers to generate character-level representations for words. The final embeddings of words are fed into a bidirectional recursive network. Yang et al. [102] proposed a neural reranking model for NER, where a convolutional layer with a fixed window-size is used on top of a character embedding layer. Recently, Peters et al. [103] proposed ELMo word representation, which are computed on top of two-layer bidirectional language models with character convolutions.

For *RNN-based* models, Long Short-Term Memory (LSTM) and Gated Recurrent Unit (GRU) are two typical choices of the basic units. Kuru et al. [100] proposed CharNER, a character-level tagger for language-independent NER. CharNER considers a sentence as a sequence of characters and utilizes LSTMs to extract character-level representations. It outputs a tag distribution for each character instead of each word. Then word-level tags are obtained from the character-level tags. Their results show that taking characters as the primary representation is superior to words as the basic input unit. Lample et al. [19] utilized a bidirectional LSTM to extract character-level representations of words. Similar to [97], character-level representation is concatenated with pre-trained word-level embedding from a word lookup table. Gridach [104] investigated word embeddings and character-level representation in identifying biomedical named entities. Rei et al. [105] combined character-level representations with word embeddings using a gating mechanism. In this way, Rei's model dynamically decides how much information to use from a character- or word-level component. Tran et al. [101] introduced a neural NER model with stack residual LSTM and trainable bias decoding, where word features are extracted from word embeddings and character-level RNN. Yang et al. [106] developed a model to handle both cross-lingual and multi-task joint training in a unified manner. They employed a deep bidirectional GRU to learn informative morphological

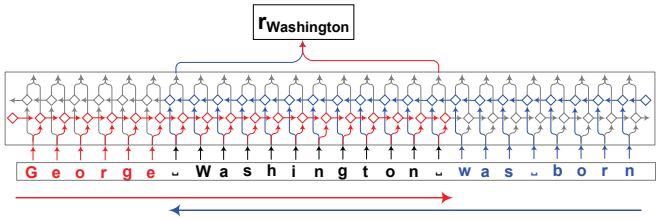


Fig. 4. Extraction of a contextual string embedding for word "Washington" in a sentential context [107]. From the forward language model (shown in red), the model extracts the output hidden state after the last character in the word. From the backward language model (shown in blue), the model extracts the output hidden state before the first character in the word. Both output hidden states are concatenated to form the final embedding of a word.

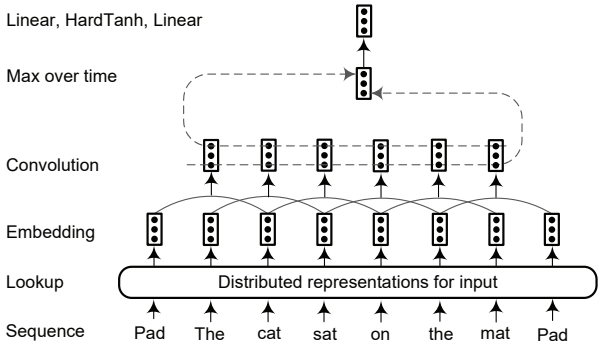


Fig. 5. Sentence approach network based on CNN [17]. The convolution layer extracts features from the whole sentence, treating it as a *sequence* with global structure.

representation from the character sequence of a word. Then character-level representation and word embedding are concatenated to produce the final representation for a word.

Recent advances in language modeling using recurrent neural networks made it viable to model language as distributions over characters. The contextual string embeddings by Akbik et al. [107], uses character-level neural language model to generate a contextualized embedding for a string of characters in a sentential context. An important property is that the embeddings are contextualized by their surrounding text, meaning that the same word has different embeddings depending on its contextual use. Figure 4 illustrates the architecture of extracting a contextual string embedding for word "Washington" in a sentential context.

3.2.3 Hybrid Representation

Besides word-level and character-level representations, some studies also incorporate additional information (e.g., gazetteers [18], [108], lexical similarity [109], linguistic dependency [110] and visual features [111]) into the final representations of words, before feeding into context encoding layers. In other words, the DL-based representation is combined with feature-based approach in a hybrid manner. Adding additional information may lead to improvements in NER performance, with the price of hurting generality of these systems.

The use of neural models for NER was pioneered by [17], where an architecture based on temporal convolutional neural networks over word sequence was proposed. When incorporating common priori knowledge (e.g., gazetteers and POS), the resulting system outperforms the baseline using only word-level representations. In the BiLSTM-CRF model by Huang et al. [18], four types of features are used for the NER task: spelling features, context features, word embeddings, and gazetteer features. Their experimental results show that the extra features (i.e., gazetteers) boost tagging accuracy. The BiLSTM-CNN model by Chiu and Nichols [20] incorporates a bidirectional LSTM and a character-level CNN. Besides word embeddings, the model uses additional word-level features (capitalization, lexicons) and character-level features (4-dimensional vector representing the type of a character: upper case, lower case, punctuation, other).

Wei et al. [112] presented a CRF-based neural system for recognizing and normalizing disease names. This system employs rich features in addition to word embeddings, including words, POS tags, chunking, and word shape features (e.g., dictionary and morphological features). Strubell et al. [91] concatenated 100-dimensional embeddings with a 5-dimensional word shape vector (e.g., all capitalized, not capitalized, first-letter capitalized or contains a capital letter). Lin et al. [113] concatenated character-level representation, word-level representation, and syntactical word representation (i.e., POS tags, dependency roles, word positions, head positions) to form a comprehensive word representation. A multi-task approach for NER was proposed by Aguilar et al. [114]. This approach utilizes a CNN to capture orthographic features and word shapes at character level. For syntactical and contextual information at word level, e.g., POS and word embeddings, the model implements a LSTM architecture. Jansson and Liu [115] proposed to combine Latent Dirichlet Allocation (LDA) with deep learning on character-level and word-level embeddings.

Xu et al. [116] proposed a local detection approach for NER based on fixed-size ordinally forgetting encoding (FOFE) [117], FOFE explores both character-level and word-level representations for each fragment and its contexts. In the multi-modal NER system by Moon et al. [118], for noisy user-generated data like tweets and Snapchat captions, word embeddings, character embeddings, and visual features are merged with modality attention. Ghaddar and Langlais [109] found that it was unfair that lexical features had been mostly discarded in neural NER systems. They proposed an alternative lexical representation which is trained offline and can be added to any neural NER system. The lexical representation is computed for each word with a 120-dimensional vector, where each element encodes the similarity of the word with an entity type. Recently, Devlin et al. [119] proposed a new language representation model called BERT, bidirectional encoder representations from transformers. BERT uses masked language models to enable pre-trained deep bidirectional representations. For a given token, its input representation is comprised by summing the corresponding position, segment and token embeddings. Note that pre-trained language model embeddings often require large-scale corpora for training, and intrinsically incorporate auxiliary embeddings (e.g., position and segment embeddings). For this reason, we category these contextualized language-model embeddings as hybrid representations in this survey.

3.3 Context Encoder Architectures

Here, we now review widely-used context encoder architectures: convolutional neural networks, recurrent neural networks, recursive neural networks, and deep transformer.

3.3.1 Convolutional Neural Networks

Collobert et al. [17] proposed a sentence approach network where a word is tagged with the consideration of *whole* sentence, shown in Figure 5. Each word in the input sequence is embedded to an N-dimensional vector after the stage of input representation. Then a convolutional layer is used to produce local features around each word, and the size of the output of the convolutional layers depends on the number of words in the sentence. The global feature vector is constructed by combining local feature vectors extracted by the convolutional layers. The dimension of the global feature vector is fixed, independent of the sentence length, in order to apply subsequent standard affine layers. Two approaches are widely used to extract global features: a max or an averaging operation over the position (i.e., "time" step) in the sentence. Finally, these fixed-size global features are fed into tag decoder to compute distribution scores for all possible tags for the words in the network input. Following Collobert's work, Yao et al. [94] proposed Bio-NER for biomedical NER. Wu et al. [120] utilized a convolutional layer to generate global features represented by a number of global hidden nodes. Both local features and global features are then fed into a standard affine network to recognize named entities in clinical text.

Zhou et al. [96] observed that with RNN latter words influence the final sentence representation more than former words. However, important words may appear anywhere in a sentence. In their proposed model, named BLSTM-RE, BLSTM is used to capture long-term dependencies and obtain the whole representation of an input sequence. CNN is then utilized to learn a high-level representation, which is then fed into a sigmoid classifier. Finally, the whole sentence representation (generated by BLSTM) and the relation presentation (generated by the sigmoid classifier) are fed into another LSTM to predict entities.

Traditionally, the time complexity of LSTMs for a sequence of length N is $\mathcal{O}(\mathcal{N})$ in a parallelism manner. Strubell et al. [91] proposed ID-CNNs, referred to Iterated Dilated Convolutional Neural Networks, which is more computationally efficient due to the capacity of handling larger context and structured prediction. Figure 6 shows the architecture of a dilated CNN block, where four stacked dilated convolutions of width 3 produce token representations. Experimental results show that ID-CNNs achieves 14-20x test-time speedups compared to Bi-LSTM-CRF while retaining comparable accuracy.

3.3.2 Recurrent Neural Networks

Recurrent neural networks, together with its variants such as gated recurrent unit (GRU) and long-short term memory (LSTM), have demonstrated remarkable achievements in modeling sequential data. In particular, bidirectional RNNs efficiently make use of past information (via forward states) and future information (via backward states) for a specific time frame [18]. Thus, a token encoded by a bidirectional

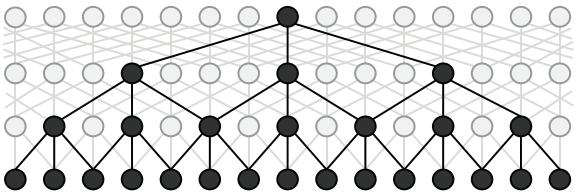


Fig. 6. The architecture of ID-CNNs with filter width 3 and maximum dilation width 4. [91].

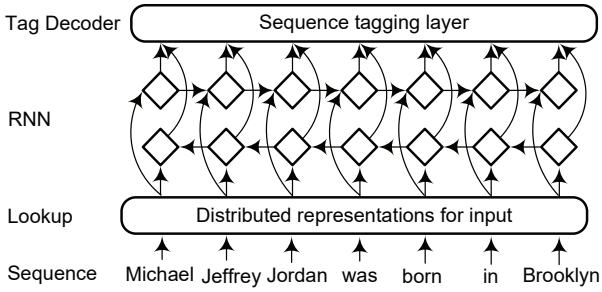


Fig. 7. The architecture of RNN-based context encoder.

RNN will contain evidence from the whole input sentence. Bidirectional RNNs therefore become de facto standard for composing deep context-dependent representations of text [91], [97]. A typical architecture of RNN-based context encoder is shown in Figure 7.

The work by Huang et al. [18] is among the first to utilize a bidirectional LSTM CRF architecture to sequence tagging tasks (POS, chunking and NER). Following [18], a body of works [19], [20], [89], [90], [95]–[97], [101], [105], [112], [113] applied BiLSTM as the basic architecture to encode sequence context information. Yang et al. [106] employed deep GRUs on both character and word levels to encode morphology and context information. They further extended their model to cross-lingual and multi-task joint trained by sharing the architecture and parameters.

Gregoric et al. [121] employed multiple independent bidirectional LSTM units across the same input. Their model promotes diversity among the LSTM units by employing an inter-model regularization term. By distributing computation across multiple smaller LSTMs, they found a reduction in total number of parameters. Recently, some studies [122], [123] designed LSTM-based neural networks for nested named entity recognition. Katiyar and Cardie [122] presented a modification to standard LSTM-based sequence labeling model to handle nested named entity recognition. Ju et al. [123] proposed a neural model to identify nested entities by dynamically stacking flat NER layers until no outer entities are extracted. Each flat NER layer employs bidirectional LSTM to capture sequential context. The model merges the outputs of the LSTM layer in the current flat NER layer to construct new representations for the detected entities and then feeds them into the next flat NER layer.

3.3.3 Recursive Neural Networks

Recursive neural networks are non-linear adaptive models that are able to learn deep structured information, by traversing a given structure in topological order. Named

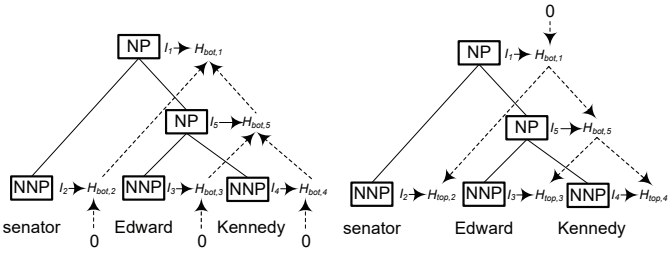


Fig. 8. Bidirectional recursive neural networks for NER [98]. The computations are done recursively in two directions. The bottom-up direction computes the semantic composition of the subtree of each node, and the top-down counterpart propagates to that node the linguistic structures which contain the subtree.

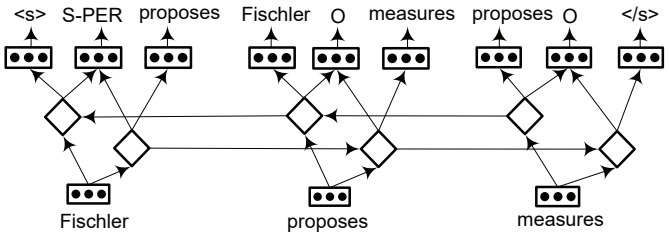


Fig. 9. A sequence labeling model with an additional language modeling objective [124], performing NER on the sentence "*Fischler proposes measures*". At each token position (e.g., "proposes"), the network is optimised to predict the previous word ("Fischler"), the current label ("O"), and the next word ("measures") in the sequence.

entities are highly related to linguistic constituents, e.g., noun phrases [98]. However, typical sequential labeling approaches take little into consideration about phrase structures of sentences. To this end, Li et al. [98] proposed to classify every node in a constituency structure for NER. This model recursively calculates hidden state vectors of every node and classifies each node by these hidden vectors. Figure 8 shows how to recursively compute two hidden state features for every node. The bottom-up direction calculates the semantic composition of the subtree of each node, and the top-down counterpart propagates to that node the linguistic structures which contain the subtree. Given hidden vectors for every node, the network calculates a probability distribution of entity types plus a special non-entity type.

3.3.4 Neural Language Models

Language model is a family of models describing the generation of sequences. Given a token sequence, $(t_1, t_2, \ldots, t_N)$, a forward language model computes the probability of the sequence by modeling the probability of token t_k given its history $(t_1, \ldots, t_{k-1})$ [21]:

$$p(t_1, t_2, \ldots, t_N) = \prod_{k=1}^{N} p(t_k | t_1, t_2, \ldots, t_{k-1}) \quad (1)$$

A backward language model is similar to a forward language model, except it runs over the sequence in reverse order, predicting the previous token given its future context:

$$p(t_1, t_2, \ldots, t_N) = \prod_{k=1}^{N} p(t_k | t_{k+1}, t_{k+2}, ..., t_N) \quad (2)$$

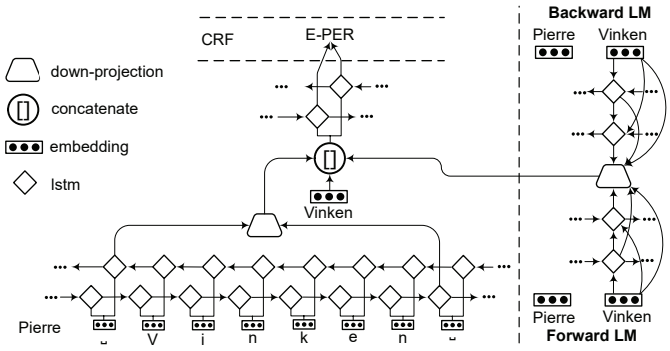


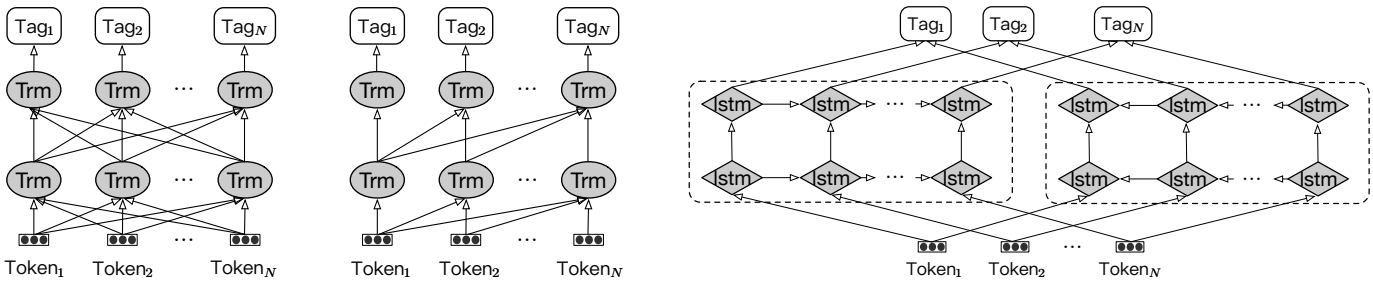
Fig. 10. Sequence labeling architecture with contextualized representations [125]. Character-level representation, pre-trained word embedding and contextualized representation from bidirectional language models are concatenated and further fed into context encoder.

For neural language models, probability of token t_k can be computed by the output of recurrent neural networks. At each position k, we can obtain two context-dependent representations (forward and backward) and then combine them as the final language model embedding for token t_k. Such language-model-augmented knowledge has been empirically verified to be helpful in numerous sequence labeling tasks [21], [103], [124]–[127].

Rei [124] proposed a framework with a secondary objective – learning to predict surrounding words for each word in the dataset. Figure 9 illustrates the architecture with a short sentence on the NER task. At each time step (i.e., token position), the network is optimised to predict the previous token, the current tag, and the next token in the sequence. The added language modeling objective encourages the system to learn richer feature representations which are then reused for sequence labeling.

Peters et al. [21] proposed TagLM, a language model augmented sequence tagger. This tagger considers both pre-trained word embeddings and bidirectional language model embeddings for every token in the input sequence for sequence labeling task. Figure 10 shows the architecture of LM-LSTM-CRF model [125], [126]. The language model and sequence tagging model share the same character-level layer in a multi-task learning manner. The vectors from character-level embeddings, pre-trained word embeddings, and language model representations, are concatenated and fed into the word-level LSTMs. Experimental results demonstrate that multi-task learning is an effective approach to guide the language model to learn task-specific knowledge.

Figure 4 shows the contextual string embedding using neural character-level language modeling by Akbik et al. [107]. They utilized the hidden states of a forward-backward recurrent neural network to create contextualized word embeddings. A major merit of this model is that character-level language model is independent of tokenization and a fixed vocabulary. Peters et al. [103] proposed ELMo representations, which are computed on top of two-layer bidirectional language models with character convolutions. This new type of deep contextualized word representation is capable of modeling both complex characteristics of word usage (e.g., semantics and syntax), and usage variations across linguistic contexts (e.g., polysemy).



(a) Google BERT (b) OpenAI GPT (c) AllenNLP ELMo

Fig. 11. Differences in pre-training model architectures [119]. Google BERT uses a bidirectional Transformer (abbreviated as "Trm"). OpenAI GPT uses a left-to-right Transformer. AllenNLP ELMo uses the concatenation of independently trained left-to-right and right-to-left LSTM to generate features for downstream tasks.

3.3.5 Deep Transformer

Neural sequence labeling models are typically based on complex convolutional or recurrent networks which consists of encoders and decoders. *Transformer*, proposed by Vaswani et al. [128], dispenses with recurrence and convolutions entirely. Transformer utilizes stacked self-attention and point-wise, fully connected layers to build basic blocks for encoder and decoder. Experiments on various tasks [128]–[130] show Transformers to be superior in quality while requiring significantly less time to train.

Based on transformer, Radford et al. [131] proposed Generative Pre-trained Transformer (GPT) for language understanding tasks. GPT has a two-stage training procedure. First, they use a language modeling objective with Transformers on unlabeled data to learn the initial parameters. Then they adapt these parameters to a target task using the supervised objective, resulting in minimal changes to the pre-trained model. Unlike GPT (a left-to-right architecture), Bidirectional Encoder Representations from Transformers (BERT) is proposed to pre-train deep bidirectional Transformer by jointly conditioning on both left and right context in all layers [119]. Figure 11 summarizes BERT [119], GPT [131] and ELMo [103]. In addition, Baevski et al. [132] proposed a novel cloze-driven pre-training regime based on a bi-directional Transformer, which is trained with a cloze-style objective and predicts the center word given all left and right context.

These language model embeddings pre-trained using Transformer are becoming a new paradigm of NER. First, these embeddings are contextualized and can be used to replace the traditional embeddings, such as Google Word2vec and Stanford GloVe. Some studies [108], [110], [133]–[136] have achieved promising performance via leveraging the combination of traditional embeddings and language model embeddings. Second, these language model embeddings can be further fine-tuned with one additional output layer for a wide range of tasks including NER and chunking. Especially, Li et al. [137], [138] framed the NER task as a machine reading comprehension (MRC) problem, which can be solved by fine-tuning the BERT model.

3.4 Tag Decoder Architectures

Tag decoder is the final stage in a NER model. It takes context-dependent representations as input and produce a sequence of tags corresponding to the input sequence. Figure 12 summarizes four architectures of tag decoders: MLP + softmax layer, conditional random fields (CRFs), recurrent neural networks, and pointer networks.

3.4.1 Multi-Layer Perceptron + Softmax

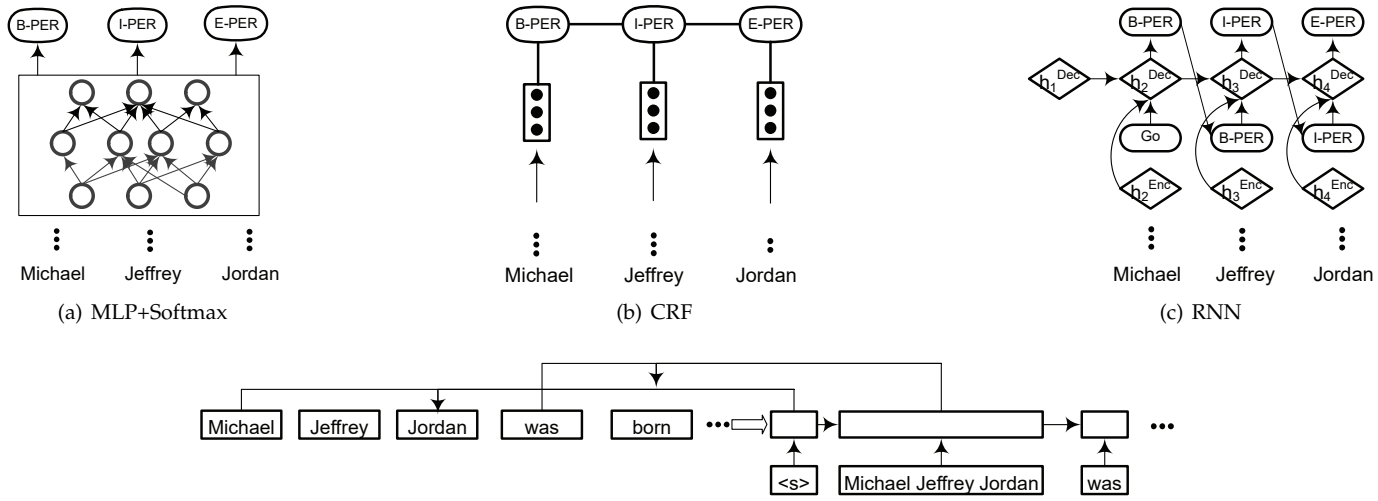
NER is in general formulated as a sequence labeling problem. With a multi-layer Perceptron + Softmax layer as the tag decoder layer, the sequence labeling task is cast as a multi-class classification problem. Tag for each word is independently predicted based on the context-dependent representations without taking into account its neighbors.

A number of NER models [91], [98], [116], [119], [139] that have been introduced earlier use MLP + Softmax as the tag decoder. As a domain-specific NER task, Tomori et al. [140] used softmax as tag decoder to predict game states in Japanese chess game. Their model takes both input from text and input from chess board (9×9 squares with 40 pieces of 14 different types) and predict 21 named entities specific to this game. Text representations and game state embeddings are both fed to a softmax layer for prediction of named entities using BIO tag scheme.

3.4.2 Conditional Random Fields

A conditional random field (CRF) is a random field globally conditioned on the observation sequence [73]. CRFs have been widely used in feature-based supervised learning approaches (see Section 2.4.3). Many deep learning based NER models use a CRF layer as the tag decoder, e.g., on top of an bidirectional LSTM layer [18], [90], [103], [141], and on top of a CNN layer [17], [91], [94]. Listed in Table 3, CRF is the most common choice for tag decoder, and the state-of-the-art performance on CoNLL03 and OntoNotes5.0 is achieved by [107] with a CRF tag decoder.

CRFs, however, cannot make full use of segment-level information because the inner properties of segments cannot be fully encoded with word-level representations. Zhuo et al. [142] then proposed gated recursive semi-markov CRFs, which directly model segments instead of words, and automatically extract segment-level features through a gated recursive convolutional neural network. Recently, Ye and Ling [143] proposed hybrid semi-Markov CRFs for neural sequence labeling. This approach adopts segments instead of words as the basic units for feature extraction and transition modeling. Word-level labels are utilized in



(a) MLP+Softmax (b) CRF (c) RNN

(d) Pointer Network

Fig. 12. Differences in four tag decoders: MLP+Softmax, CRF, RNN, and Pointer network.

deriving segment scores. Thus, this approach is able to leverage both word- and segment-level information for segment score calculation.

3.4.3 Recurrent Neural Networks

A few studies [88]–[90], [96], [144] have explored RNN to decode tags. Shen et al. [88] reported that RNN tag decoders outperform CRF and are faster to train when the number of entity types is large. Figure 12(c) illustrates the workflow of RNN-based tag decoders, which serve as a language model to greedily produce a tag sequence. The [GO]-symbol at the first step is provided as y_1 to the RNN decoder. Subsequently, at each time step i, the RNN decoder computes current decoder hidden state h_{i+1}^{Dec} in terms of previous step tag y_i, previous step decoder hidden state h_i^{Dec} and current step encoder hidden state h_{i+1}^{Enc}; the current output tag y_{i+1} is decoded by using a softmax loss function and is further fed as an input to the next time step. Finally, we obtain a tag sequence over all time steps.

3.4.4 Pointer Networks

Pointer networks apply RNNs to learn the conditional probability of an output sequence with elements that are discrete tokens corresponding to the positions in an input sequence [145], [146]. It represents variable length dictionaries by using a softmax probability distribution as a "pointer". Zhai et al. [95] first applied pointer networks to produce sequence tags. Illustrated in Figure 12(d), pointer networks first identify a chunk (or a segment), and then label it. This operation is repeated until all the words in input sequence are processed. In Figure 12(d), given the start token "<s>", the segment "Michael Jeffery Jordan" is first identified and then labeled as "PERSON". The segmentation and labeling can be done by two separate neural networks in pointer networks. Next, "Michael Jeffery Jordan" is taken as input and fed into pointer networks. As a result, the segment "was" is identified and labeled as "O".

3.5 Summary of DL-based NER

Architecture Summary. Table 3 summarizes recent works on neural NER by their architecture choices. BiLSTM-CRF is the most common architecture for NER using deep learning. The method [132] which pre-trains a bi-directional Transformer model in a cloze-style manner, achieves the state-of-the-art performance (93.5%) on CoNLL03. The work with BERT and dice loss [138] achieves the state-of-the-art performance (92.07%) on OntoNotes5.0.

The success of a NER system heavily relies on its input representation. Integrating or fine-tuning pre-trained language model embeddings is becoming a new paradigm for neural NER. When leveraging these language model embeddings, there are significant performance improvements [103], [107], [108], [132]–[138]. The last column in Table 3 lists the reported performance in F-score on a few benchmark datasets. While high F-scores have been reported on formal documents (e.g., CoNLL03 and OntoNotes5.0), NER on noisy data (e.g., W-NUT17) remains challenging.

Architecture Comparison. We discuss pros and cons from three perspectives: input, encoder, and decoder. First, no consensus has been reached about whether external knowledge should be or how to integrate into DL-based NER models. Some studies [108], [110], [133], [142] shows that NER performance can be boosted with external knowledge. However, the disadvantages are also apparent: 1) acquiring external knowledge is labor-intensive (e.g., gazetteers) or computationally expensive (e.g., dependency); 2) integrating external knowledge adversely affects end-to-end learning and hurts the generality of DL-based systems.

Second, Transformer encoder is more effective than LSTM when Transformer is pre-trained on huge corpora. Transformers fail on NER task if they are not pre-trained and when the training data is limited [147], [148]. On the other hand, Transformer encoder is faster than recursive layers when the length of the sequence n is smaller than the dimensionality of the representation d (complexities: self-attention $\mathcal{O}(n^2 \cdot d)$ and recurrent $\mathcal{O}(n \cdot d^2)$) [128].

TABLE 3
Summary of recent works on neural NER. LSTM: long short-term memory, CNN: convolutional neural network, GRU: gated recurrent unit, LM: language model, ID-CNN: iterated dilated convolutional neural network, BRNN: bidirectional recursive neural network, MLP: multi-layer perceptron, CRF: conditional random field, Semi-CRF: Semi-markov conditional random field, FOFE: fixed-size ordinally forgetting encoding.

Work	Input representation			Context encoder	Tag decoder	Performance (F-score)
	Character	Word	Hybrid			
[94]	-	Trained on PubMed	POS	CNN	CRF	GENIA: 71.01%
[89]	-	Trained on Gigaword	-	GRU	GRU	ACE 2005: 80.00%
[95]	-	Random	-	LSTM	Pointer Network	ATIS: 96.86%
[90]	-	Trained on NYT	-	LSTM	LSTM	NYT: 49.50%
[91]	-	SENNA	Word shape	ID-CNN	CRF	CoNLL03: 90.65%; OntoNotes5.0: 86.84%
[96]	-	Google word2vec	-	LSTM	LSTM	CoNLL04: 75.0%
[100]	LSTM	-	-	LSTM	CRF	CoNLL03: 84.52%
[97]	CNN	GloVe	-	LSTM	CRF	CoNLL03: 91.21%
[105]	LSTM	Google word2vec	-	LSTM	CRF	CoNLL03: 84.09%
[19]	LSTM	SENNA	-	LSTM	CRF	CoNLL03: 90.94%
[106]	GRU	SENNA	-	GRU	CRF	CoNLL03: 90.94%
[98]	CNN	GloVe	POS	BRNN	Softmax	OntoNotes5.0: 87.21%
[107]	LSTM-LM	-	-	LSTM	CRF	CoNLL03: 93.09%; OntoNotes5.0: 89.71%
[103]	CNN-LSTM-LM	-	-	LSTM	CRF	CoNLL03: 92.22%
[17]	-	Random	POS	CNN	CRF	CoNLL03: 89.86%
[18]	-	SENNA	Spelling, n-gram, gazetteer	LSTM	CRF	CoNLL03: 90.10%
[20]	CNN	SENNA	capitalization, lexicons	LSTM	CRF	CoNLL03: 91.62%; OntoNotes5.0: 86.34%
[116]	-	-	FOFE	MLP	CRF	CoNLL03: 91.17%
[101]	LSTM	GloVe	-	LSTM	CRF	CoNLL03: 91.07%
[113]	LSTM	GloVe	Syntactic	LSTM	CRF	W-NUT17: 40.42%
[102]	CNN	SENNA	-	LSTM	Reranker	CoNLL03: 91.62%
[114]	CNN	Twitter Word2vec	POS	LSTM	CRF	W-NUT17: 41.86%
[115]	LSTM	GloVe	POS, topics	LSTM	CRF	W-NUT17: 41.81%
[118]	LSTM	GloVe	Images	LSTM	CRF	SnapCaptions: 52.4%
[109]	LSTM	SSKIP	Lexical	LSTM	CRF	CoNLL03: 91.73%; OntoNotes5.0: 87.95%
[119]	-	WordPiece	Segment, position	Transformer	Softmax	CoNLL03: 92.8%
[121]	LSTM	SENNA	-	LSTM	Softmax	CoNLL03: 91.48%
[124]	LSTM	Google Word2vec	-	LSTM	CRF	CoNLL03: 86.26%
[21]	GRU	SENNA	LM	GRU	CRF	CoNLL03: 91.93%
[126]	LSTM	GloVe	-	LSTM	CRF	CoNLL03: 91.71%
[142]	-	SENNA	POS, gazetteers	CNN	Semi-CRF	CoNLL03: 90.87%
[143]	LSTM	GloVe	-	LSTM	Semi-CRF	CoNLL03: 91.38%
[88]	CNN	Trained on Gigaword	-	LSTM	LSTM	CoNLL03: 90.69%; OntoNotes5.0: 86.15%
[110]	-	GloVe	ELMo, dependency	LSTM	CRF	CoNLL03: 92.4%; OntoNotes5.0: 89.88%
[108]	CNN	GloVe	ELMo, gazetteers	LSTM	Semi-CRF	CoNLL03: 92.75%; OntoNotes5.0: 89.94%
[133]	LSTM	GloVe	ELMo, POS	LSTM	Softmax	CoNLL03: 92.28%
[137]	-	-	BERT	-	Softmax	CoNLL03: 93.04%; OntoNotes5.0: 91.11%
[138]	-	-	BERT	-	Softmax +Dice Loss	CoNLL03: 93.33%; **OntoNotes5.0: 92.07**%
[134]	LSTM	GloVe	BERT, document-level embeddings	LSTM	CRF	CoNLL03: 93.37%; OntoNotes5.0: 90.3%
[135]	CNN	GloVe	BERT, global embeddings	GRU	GRU	CoNLL03: 93.47%
[132]	CNN	-	Cloze-style LM embeddings	LSTM	CRF	**CoNLL03: 93.5%**
[136]	-	GloVe	Plooled contextual embeddings	RNN	CRF	CoNLL03: 93.47%

Third, a major disadvantage of RNN and Pointer Network decoders lies in greedily decoding, which means that the input of current step needs the output of previous step. This mechanism may have a significant impact on the speed and is an obstacle to parallelization. CRF is the most common choice for tag decoder. CRF is powerful to capture label transition dependencies when adopting non-language-model (i.e., non-contextualized) embeddings such as Word2vec and GloVe. However, CRF could be computationally expensive when the number of entity types is large. More importantly, CRF does not always lead to better performance compared with softmax classification when adopting contextualized language model embeddings such as BERT and ELMo [137], [139].

For end users, what architecture to choose is data and domain task dependent. If data is abundant, training models with RNNs from scratch and fine-tuning contextualized language models could be considered. If data is scarce, adopting transfer strategies might be a better choice. For newswires domain, there are many pre-trained off-the-shelf models available. For specific domains (e.g., medical and social media), fine-tuning general-purpose contextualized language models with domain-specific data is often an effective way.

NER for Different Languages. In this survey, we mainly focus on NER in English and in general domain. Apart from English language, there are many studies on other languages or cross-lingual settings. Wu et al. [120] and Wang et al. [149] investigated NER in Chinese clinical text. Zhang and Yang [150] proposed a lattice-structured LSTM model for Chinese NER, which encodes a sequence of input characters as well as all potential words that match a lexicon. Other than Chinese, many studies are conducted on other languages. Examples include Mongolian [151], Czech [152], Arabic [153], Urdu [154], Vietnamese [155], Indonesian [156], and Japanese [157]. Each language has its own characteristics for understanding the fundamentals of NER task on that language. There are also a number of studies [106], [158]–[160] aiming to solve the NER problem in a cross-lingual setting by transferring knowledge from a source language to a target language with few or no labels.

4 Applied Deep Learning for NER

Sections 3.2, 3.3, and 3.4 outline typical network architectures for NER. In this section, we survey recent applied deep learning techniques that are being explored for NER.

4.1 Deep Multi-task Learning for NER

Multi-task learning [161] is an approach that learns a group of related tasks together. By considering the relation between different tasks, multi-task learning algorithms are expected to achieve better results than the ones that learn each task individually.

Collobert et al. [17] trained a window/sentence approach network to jointly perform POS, Chunk, NER, and SRL tasks. This multi-task mechanism lets the training algorithm to discover internal representations that are useful for all the tasks of interest. Yang et al. [106] proposed a multi-task joint model, to learn language-specific regularities, jointly trained for POS, Chunk, and NER tasks. Rei [124] found that by including an unsupervised language modeling objective in the training process, the sequence labeling model achieves consistent performance improvement. Lin et al. [160] proposed a multi-lingual multi-task architecture for low-resource settings, which can effectively transfer different types of knowledge to improve the main model.

Other than considering NER together with other sequence labeling tasks, multi-task learning framework can be applied for joint extraction of entities and relations [90], [96], or to model NER as two related subtasks: entity segmentation and entity category prediction [114], [162]. In biomedical domain, because of the differences in different datasets, NER on each dataset is considered as a task in a multi-task setting [163], [164]. A main assumption here is that the different datasets share the same character- and word-level information. Then multi-task learning is applied to make more efficient use of the data and to encourage the models to learn more generalized representations.

4.2 Deep Transfer Learning for NER

Transfer learning aims to perform a machine learning task on a target domain by taking advantage of knowledge learned from a source domain [165]. In NLP, transfer learning is also known as domain adaptation. On NER tasks, the traditional approach is through bootstrapping algorithms [166]–[168]. Recently, a few approaches [127], [169]–[173] have been proposed for low-resource and across-domain NER using deep neural networks.

Pan et al. [169] proposed a transfer joint embedding (TJE) approach for cross-domain NER. TJE employs label embedding techniques to transform multi-class classification to regression in a low-dimensional latent space. Qu et al. [174] observed that related named entity types often share lexical and context features. Their approach learns the correlation between source and target named entity types using a two-layer neural network. Their approach is applicable to the setting that the source domain has similar (but not identical) named entity types with the target domain. Peng and Dredze [162] explored transfer learning in a multi-task learning setting, where they considered two domains: news and social media, for two tasks: word segmentation and NER.

In the setting of transfer learning, different neural models commonly share different parts of model parameters between source task and target task. Yang et al. [175] first investigated the transferability of different layers of representations. Then they presented three different parameter-sharing architectures for cross-domain, cross-lingual, and cross-application scenarios. If two tasks have mappable label sets, there is a shared CRF layer, otherwise, each task learns a separate CRF layer. Experimental results show significant improvements on various datasets under low-resource conditions (i.e., fewer available annotations). Pius and Mark [176] extended Yang's approach to allow joint training on informal corpus (e.g., WNUT 2017), and to incorporate sentence level feature representation. Their approach achieved the 2nd place at the WNUT 2017 shared task for NER, obtaining an F1-score of 40.78%. Zhao et al. [177] proposed a multi-task model with domain adaption, where the fully connection ayer are adapted to different datasets, and the CRF features are computed separately. A major merit of Zhao's model is that the instances with different distribution and misaligned annotation guideline are filtered out in data selection procedure. Different from these parameter-sharing architectures, Lee et al. [170] applied transfer learning in NER by training a model on source task and using the trained model on target task for fine-tuning. Recently, Lin and Lu [171] also proposed a fine-tuning approach for NER by introducing three neural adaptation layers: word adaptation layer, sentence adaptation layer, and output adaptation layer. Beryozkin et al. [178] proposed a tag-hierarchy model for heterogeneous tag-sets NER setting, where the hierarchy is used during inference to map fine-grained tags onto a target tag-set. In addition, some studies [164], [179], [180] explored transfer learning in biomedical NER to reduce the amount of required labeled data.

4.3 Deep Active Learning for NER

The key idea behind active learning is that a machine learning algorithm can perform better with substantially less data from training, if it is allowed to choose the data from which it learns [181]. Deep learning typically requires a large amount of training data which is costly to obtain. Thus,

combining deep learning with active learning is expected to reduce data annotation effort.

Training with active learning proceeds in multiple rounds. However, traditional active learning schemes are expensive for deep learning because after each round they require complete retraining of the classifier with newly annotated samples. Because retraining from scratch is not practical for deep learning, Shen et al. [88] proposed to carry out incremental training for NER with each batch of new labels. They mix newly annotated samples with the existing ones, and update neural network weights for a small number of epochs, before querying for labels in a new round. Specifically, at the beginning of each round, the active learning algorithm chooses sentences to be annotated, to the predefined budget. The model parameters are updated by training on the augmented dataset, after receiving chose annotations. The active learning algorithm adopts uncertainty sampling strategy [182] in selecting sentences to be annotated. Experimental results show that active learning algorithms achieve 99% performance of the best deep learning model trained on full data using only 24.9% of the training data on the English dataset and 30.1% on Chinese dataset. Moreover, 12.0% and 16.9% of training data were enough for deep active learning model to outperform shallow models learned on full training data [183].

4.4 Deep Reinforcement Learning for NER

Reinforcement learning (RL) is a branch of machine learning inspired by behaviorist psychology, which is concerned with how software agents take actions in an environment so as to maximize some cumulative rewards [184], [185]. The idea is that an agent will learn from the environment by interacting with it and receiving rewards for performing actions. Specifically, the RL problem can be formulated as follows [186]: the environment is modeled as a stochastic finite state machine with inputs (actions from agent) and outputs (observations and rewards to the agent). It consists of three key components: (i) state transition function, (ii) observation (i.e., output) function, and (iii) reward function. The agent is also modeled as a stochastic finite state machine with inputs (observations/rewards from the environment) and outputs (actions to the environment). It consists of two components: (i) state transition function, and (ii) policy/output function. The ultimate goal of an agent is to learn a good state-update function and policy by attempting to maximize the cumulative rewards.

Narasimhan et al. [187] modeled the task of information extraction as a Markov decision process (MDP), which dynamically incorporates entity predictions and provides flexibility to choose the next search query from a set of automatically generated alternatives. The process is comprised of issuing search queries, extraction from new sources, and reconciliation of extracted values, and the process repeats until sufficient evidence is obtained. In order to learn a good policy for an agent, they utilize a deep Q-network [188] as a function approximator, in which the state-action value function (i.e., Q-function) is approximated by using a deep neural network. Recently, Yang et al. [189] utilized the data generated by distant supervision to perform new type named entity recognition in new domains. The instance selector is based on reinforcement learning and obtains the feedback reward from the NE tagger, aiming at choosing positive sentences to reduce the effect of noisy annotation.

4.5 Deep Adversarial Learning for NER

Adversarial learning [190] is the process of explicitly training a model on adversarial examples. The purpose is to make the model more robust to attack or to reduce its test error on clean inputs. Adversarial networks learn to generate from a training distribution through a 2-player game: one network generates candidates (generative network) and the other evaluates them (discriminative network). Typically, the generative network learns to map from a latent space to a particular data distribution of interest, while the discriminative network discriminates between candidates generated by the generator and instances from the real-world data distribution [191].

For NER, adversarial examples are often produced in two ways. Some studies [192]–[194] considered the instances in a source domain as adversarial examples for a target domain, and vice versa. For example, Li et al. [193] and Cao et al. [194] both incorporated adversarial examples from other domains to encourage domain-invariant features for cross-domain NER. Another option is to prepare an adversarial sample by adding an original sample with a perturbation. For example, dual adversarial transfer network (DATNet), proposed in [195], aims to deal with the problem of low-resource NER. An adversarial sample is produced by adding original sample with a perturbation bounded by a small norm ϵ to maximize the loss function as follows: $\eta_x = \arg \max_{\eta:\|\eta\|_2 \leq \epsilon} l(\Theta; x+\eta)$, where Θ is the current model parameters set, ϵ can be determined on validation set. An adversarial example is constructed by $x_{adv} = x + \eta_x$. The classifier is trained on the mixture of original and adversarial examples to improve generalization.

4.6 Neural Attention for NER

The attention mechanism is loosely based on the visual attention mechanism found in human [196]. For example, people usually focus on a certain region of an image with "high resolution" while perceiving the surrounding region with "low resolution". Neural attention mechanism allows neural networks have the ability to focus on a subset of its inputs. By applying attention mechanism, a NER model could capture the most informative elements in the inputs. In particular, the Transformer architecture reviewed in Section 3.3.5 relies entirely on attention mechanism to draw global dependencies between input and output.

There are many other ways of applying attention mechanism in NER tasks. Rei et al. [105] applied an attention mechanism to dynamically decide how much information to use from a character- or word-level component in an end-to-end NER model. Zukov-Gregoric et al. [197] explored the self-attention mechanism in NER, where the weights are dependent on a single sequence (rather than on the relation between two sequences). Xu et al. [198] proposed an attention-based neural NER architecture to leverage document-level global information. In particular, the document-level information is obtained from document represented by pre-trained bidirectional language model with neural attention.

Zhang et al. [199] used an adaptive co-attention network for NER in tweets. This adaptive co-attention network is a multi-modal model using co-attention process. Co-attention includes visual attention and textual attention to capture the semantic interaction between different modalities.

5 Challenges and Future Directions

Discussed in Section 3.5, the choices tag decoders do not vary as much as the choices of input representations and context encoders. From Google Word2vec to the more recent BERT model, DL-based NER benefits significantly from the advances made in pre-trained embeddings in modeling languages. Without the need of complicated feature-engineering, we now have the opportunity to re-look the NER task for its challenges and potential future directions.

5.1 Challenges

Data Annotation. Supervised NER systems, including DL-based NER, require big annotated data in training. However, data annotation remains time consuming and expensive. It is a big challenge for many resource-poor languages and specific domains as domain experts are needed to perform annotation tasks.

Quality and consistency of the annotation are both major concerns because of the language ambiguity. For instance, a same named entity may be annotated with different types. As an example, "*Baltimore*" in the sentence "*Baltimore defeated the Yankees*", is labeled as Location in MUC-7 and Organization in CoNLL03. Both "*Empire State*" and "*Empire State Building*", is labeled as Location in CoNLL03 and ACE datasets, causing confusion in entity boundaries. Because of the inconsistency in data annotation, model trained on one dataset may not work well on another even if the documents in the two datasets are from the same domain.

To make data annotation even more complicated, Katiyar and Cardie [122] reported that nested entities are fairly common: 17% of the entities in the GENIA corpus are embedded within another entity; in the ACE corpora, 30% of sentences contain nested entities. There is a need to develop common annotation schemes to be applicable to both nested entities and fine-grained entities, where one named entity may be assigned multiple types.

Informal Text and Unseen Entities. Listed in Table 3, decent results are reported on datasets with formal documents (e.g., news articles). However, on user-generated text e.g., WUT-17 dataset, the best F-scores are slightly above 40%. NER on informal text (e.g., tweets, comments, user forums) is more challenging than on formal text due to the shortness and noisiness. Many user-generated texts are domain specific as well. In many application scenarios, a NER system has to deal with user-generated text such as customer support in e-commerce and banking.

Another interesting dimension to evaluate the robustness and effectiveness of NER system is its capability of identifying unusual, previously-unseen entities in the context of emerging discussions. There exists a shared task[2] for this direction of research on WUT-17 dataset [200].

2. https://noisy-text.github.io/2017/emerging-rare-entities.html

5.2 Future Directions

With the advances in modeling languages and demand in real-world applications, we expect NER to receive more attention from researchers. On the other hand, NER is in general considered as a pre-processing component to downstream applications. That means a particular NER task is defined by the requirement of downstream application, e.g., the types of named entities and whether there is a need to detect nested entities [201]. Based on the studies in this survey, we list the following directions for further exploration in NER research.

Fine-grained NER and Boundary Detection. While many existing studies [19], [97], [109] focused on coarse-grained NER in general domain, we expect more research on fine-grained NER in domain-specific areas to support various real word applications [202]. The challenges in fine-grained NER are the significant increase in NE types and the complication introduced by allowing a named entity to have multiple NE types. This calls for a re-visit of the common NER approaches where the entity boundaries and the types are detected simultaneously e.g., by using B- I- E- S-(entity type) and O as the decoding tags. It is worth considering to define *named entity boundary detection* as a dedicated task to detect NE boundaries while ignoring the NE types. The decoupling of boundary detection and NE type classification enables common and robust solutions for boundary detection that can be shared across different domains, and dedicated domain-specific approaches for NE type classification. Correct entity boundaries also effectively alleviate error propagation in entity linking to knowledge bases. There has been some studies [95], [203] which consider entity boundary detection as an intermediate step (i.e., a subtask) in NER. To the best of our knowledge, no existing work separately focuses on entity boundary detection to provide a robust recognizer. We expect a breakout in this research direction in the future.

Joint NER and Entity Linking. Entity linking (EL) [204], also referred to as named entity normalization or disambiguation, aims at assigning a unique identity to entities mentioned in text with reference to a knowledge base, e.g., Wikipedia in general domain and the Unified Medical Language System (UMLS) in biomedical domain. Most existing works individually solve NER and EL as two separate tasks in a pipeline setting. We consider that the semantics carried by the successfully linked entities (e.g., through the related entities in the knowledge base) are significantly enriched [66], [205]. That is, linked entities contributes to the successful detection of entity boundaries and correct classification of entity types. It is worth exploring approaches for jointly performing NER and EL, or even entity boundary detection, entity type classification, and entity linking, so that each subtask benefits from the partial output by other subtasks, and alleviate error propagations that are unavoidable in pipeline settings.

DL-based NER on Informal Text with Auxiliary Resource. As discussed in Section 5.1, performance of DL-based NER on informal text or user-generated content remains low. This calls for more research in this area. In particular, we note that the performance of NER benefits significantly from

the availability of auxiliary resources [206]–[208], e.g., a dictionary of location names in user language. While Table 3 does not provide strong evidence of involving gazetteer as additional features leads to performance increase to NER in general domain, we consider auxiliary resources are often necessary to better understand user-generated content. The question is how to obtain matching auxiliary resources for a NER task on user-generated content or domain-specific text, and how to effectively incorporate the auxiliary resources in DL-based NER.

Scalability of DL-based NER. Making neural NER models more scalable is still a challenge. Moreover, there is still a need for solutions on optimizing exponential growth of parameters when the size of data grows [209]. Some DL-based NER models have achieved good performance with the cost of massive computing power. For example, the ELMo representation represents each word with a 3×1024-dimensional vector, and the model was trained for 5 weeks on 32 GPUs [107]. Google BERT representations were trained on 64 cloud TPUs. However, end users are not able to fine-tune these models if they have no access to powerful computing resources. Developing approaches to balancing model complexity and scalability will be a promising direction. On the other hand, model compression and pruning techniques are also options to reduce the space and computation time required for model learning.

Deep Transfer Learning for NER. Many entity-focused applications resort to off-the-shelf NER systems to recognize named entities. However, model trained on one dataset may not work well on other texts due to the differences in characteristics of languages as well as the differences in annotations. Although there are some studies of applying deep transfer learning to NER (see Section 4.2), this problem has not been fully explored. More future efforts should be dedicated on how to effectively transfer knowledge from one domain to another by exploring the following research problems: (a) developing a robust recognizer, which is able to work well across different domains; (b) exploring zero-shot, one-shot and few-shot learning in NER tasks; (c) providing solutions to address domain mismatch, and label mismatch in cross-domain settings.

An Easy-to-use Toolkit for DL-based NER. Recently, Röder et al. [210] developed GERBIL, which provides researchers, end users and developers with easy-to-use interfaces for benchmarking entity annotation tools with the aim of ensuring repeatable and archiveable experiments. However, it does not involve recent DL-based techniques. Ott [211] presented FAIRSEQ, a fast, extensible toolkit for sequence modeling, especially for machine translation and text stigmatization. Dernoncourt et al. [212] implemented a framework, named NeuroNER, which only relies on a variant of recurrent neural network. In recent years, many deep learning frameworks (e.g., TensorFlow, PyTorch, and Keras) have been designed to offer building blocks for designing, training and validating deep neural networks, through a high level programming interface.[3] In order to re-implement the architectures in Table 3, developers may write codes from scratch with existing deep learning frameworks. We envision that an easy-to-use NER toolkit can guide developers to complete it with some standardized modules: data-processing, input representation, context encoder, tag decoder, and effectiveness measure. We believe that experts and non-experts can both benefit from such toolkits.

3. https://developer.nvidia.com/deep-learning-frameworks

6 Conclusion

This survey aims to review recent studies on deep learning-based NER solutions to help new researchers building a comprehensive understanding of this field. We include in this survey the background of the NER research, a brief of traditional approaches, current state-of-the-arts, and challenges and future research directions. First, we consolidate available NER resources, including tagged NER corpora and off-the-shelf NER systems, with focus on NER in general domain and NER in English. We present these resources in a tabular form and provide links to them for easy access. Second, we introduce preliminaries such as definition of NER task, evaluation metrics, traditional approaches to NER, and basic concepts in deep learning. Third, we review the literature based on varying models of deep learning and map these studies according to a new taxonomy. We further survey the most representative methods for recent applied deep learning techniques in new problem settings and applications. Finally, we summarize the applications of NER and present readers with challenges in NER and future directions. We hope that this survey can provide a good reference when designing DL-based NER models.

References

[1] D. Nadeau and S. Sekine, "A survey of named entity recognition and classification," *Lingvist. Investig.*, vol. 30, no. 1, pp. 3–26, 2007.

[2] Z. Zhang, X. Han, Z. Liu, X. Jiang, M. Sun, and Q. Liu, "ERNIE: enhanced language representation with informative entities," in *ACL*, 2019, pp. 1441–1451.

[3] P. Cheng and K. Erk, "Attending to entities for better text understanding," *arXiv preprint arXiv:1911.04361*, 2019.

[4] J. Guo, G. Xu, X. Cheng, and H. Li, "Named entity recognition in query," in *SIGIR*, 2009, pp. 267–274.

[5] D. Petkova and W. B. Croft, "Proximity-based document representation for named entity retrieval," in *CIKM*, 2007, pp. 731–740.

[6] C. Aone, M. E. Okurowski, and J. Gorlinsky, "A trainable summarizer with knowledge acquired from robust nlp techniques," *Adv. Autom. Text Summ.*, vol. 71, 1999.

[7] D. M. Aliod, M. van Zaanen, and D. Smith, "Named entity recognition for question answering," in *ALTA*, 2006, pp. 51–58.

[8] B. Babych and A. Hartley, "Improving machine translation quality with automatic named entity recognition," in *EAMT*, 2003, pp. 1–8.

[9] O. Etzioni, M. Cafarella, D. Downey, A.-M. Popescu, T. Shaked, S. Soderland, D. S. Weld, and A. Yates, "Unsupervised named-entity extraction from the web: An experimental study," *Artif. Intell.*, vol. 165, no. 1, pp. 91–134, 2005.

[10] R. Grishman and B. Sundheim, "Message understanding conference-6: A brief history," in *COLING*, vol. 1, 1996.

[11] E. F. Tjong Kim Sang and F. De Meulder, "Introduction to the conll-2003 shared task: Language-independent named entity recognition," in *NAACL-HLT*, 2003, pp. 142–147.

[12] G. R. Doddington, A. Mitchell, M. A. Przybocki, L. A. Ramshaw, S. Strassel, and R. M. Weischedel, "The automatic content extraction (ace) program-tasks, data, and evaluation." in *LREC*, vol. 2, 2004, p. 1.

[13] G. Demartini, T. Iofciu, and A. P. De Vries, "Overview of the inex 2009 entity ranking track," in *INEX*, 2009, pp. 254–264.

[14] K. Balog, P. Serdyukov, and A. P. De Vries, "Overview of the trec 2010 entity track," in *TREC*, 2010.

[15] G. Petasis, A. Cucchiarelli, P. Velardi, G. Paliouras, V. Karkaletsis, and C. D. Spyropoulos, "Automatic adaptation of proper noun dictionaries through cooperation of machine learning and probabilistic methods," in *SIGIR*, 2000, pp. 128–135.

[16] S. A. Kripke, "Naming and necessity," in *Semantics of natural language*. Springer, 1972, pp. 253–355.

[17] R. Collobert, J. Weston, L. Bottou, M. Karlen, K. Kavukcuoglu, and P. Kuksa, "Natural language processing (almost) from scratch," *J. Mach. Learn. Res.*, vol. 12, no. Aug, pp. 2493–2537, 2011.

[18] Z. Huang, W. Xu, and K. Yu, "Bidirectional lstm-crf models for sequence tagging," *arXiv preprint arXiv:1508.01991*, 2015.

[19] G. Lample, M. Ballesteros, S. Subramanian, K. Kawakami, and C. Dyer, "Neural architectures for named entity recognition," in *NAACL*, 2016, pp. 260–270.

[20] J. P. Chiu and E. Nichols, "Named entity recognition with bidirectional lstm-cnns," *Trans. Assoc. Comput. Linguist.*, pp. 357–370, 2016.

[21] M. E. Peters, W. Ammar, C. Bhagavatula, and R. Power, "Semi-supervised sequence tagging with bidirectional language models," in *ACL*, 2017, pp. 1756–1765.

[22] M. Marrero, J. Urbano, S. Sánchez-Cuadrado, J. Morato, and J. M. Gómez-Berbís, "Named entity recognition: fallacies, challenges and opportunities," *Comput. Stand. Interfaces*, vol. 35, no. 5, pp. 482–489, 2013.

[23] M. L. Patawar and M. Potey, "Approaches to named entity recognition: a survey," *Int. J. Innov. Res. Comput. Commun. Eng.*, vol. 3, no. 12, pp. 12 201–12 208, 2015.

[24] C. J. Saju and A. Shaja, "A survey on efficient extraction of named entities from new domains using big data analytics," in *ICRTCCM*, 2017, pp. 170–175.

[25] X. Dai, "Recognizing complex entity mentions: A review and future directions," in *ACL*, 2018, pp. 37–44.

[26] V. Yadav and S. Bethard, "A survey on recent advances in named entity recognition from deep learning models," in *COLING*, 2018, pp. 2145–2158.

[27] A. Goyal, V. Gupta, and M. Kumar, "Recent named entity recognition and classification techniques: A systematic review," *Comput. Sci. Rev.*, vol. 29, pp. 21–43, 2018.

[28] R. Sharnagat, "Named entity recognition: A literature survey," *Center For Indian Language Technology*, 2014.

[29] X. Ling and D. S. Weld, "Fine-grained entity recognition." in *AAAI*, vol. 12, 2012, pp. 94–100.

[30] X. Ren, W. He, M. Qu, L. Huang, H. Ji, and J. Han, "Afet: Automatic fine-grained entity typing by hierarchical partial-label embedding," in *EMNLP*, 2016, pp. 1369–1378.

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

[32] A. Lal, A. Tomer, and C. R. Chowdary, "Sane: System for fine grained named entity typing on textual data," in *WWW*, 2017, pp. 227–230.

[33] L. d. Corro, A. Abujabal, R. Gemulla, and G. Weikum, "Finet: Context-aware fine-grained named entity typing," in *EMNLP*, 2015, pp. 868–878.

[34] K. Balog, *Entity-Oriented Search*. Springer, 2018.

[35] H. Raviv, O. Kurland, and D. Carmel, "Document retrieval using entity-based language models," in *SIGIR*, 2016, pp. 65–74.

[36] P. Boldi, F. Bonchi, C. Castillo, D. Donato, A. Gionis, and S. Vigna, "The query-flow graph: model and applications," in *CIKM*, 2008, pp. 609–618.

[37] F. Cai, M. De Rijke *et al.*, "A survey of query auto completion in information retrieval," *Found. Trends® in Inf. Retr.*, vol. 10, no. 4, pp. 273–363, 2016.

[38] Z. Bar-Yossef and N. Kraus, "Context-sensitive query auto-completion," in *WWW*, 2011, pp. 107–116.

[39] G. Saldanha, O. Biran, K. McKeown, and A. Gliozzo, "An entity-focused approach to generating company descriptions," in *ACL*, vol. 2, 2016, pp. 243–248.

[40] F. Hasibi, K. Balog, and S. E. Bratsberg, "Dynamic factual summaries for entity cards," in *SIGIR*, 2017, pp. 773–782.

[41] S. Pradhan, A. Moschitti, N. Xue, O. Uryupina, and Y. Zhang, "Conll-2012 shared task: Modeling multilingual unrestricted coreference in ontonotes," in *EMNLP*, 2012, pp. 1–40.

[42] C. Dogan, A. Dutra, A. Gara, A. Gemma, L. Shi, M. Sigamani, and E. Walters, "Fine-grained named entity recognition using elmo and wikidata," *CoRR*, vol. abs/1904.10503, 2019.

[43] S. Sekine and C. Nobata, "Definition, dictionaries and tagger for extended named entity hierarchy." in *LREC*, 2004, pp. 1977–1980.

[44] S. Zhang and N. Elhadad, "Unsupervised biomedical named entity recognition: Experiments with clinical and biological texts," *J. Biomed. Inform.*, vol. 46, no. 6, pp. 1088–1098, 2013.

[45] J.-H. Kim and P. C. Woodland, "A rule-based named entity recognition system for speech input," in *ICSLP*, 2000.

[46] D. Hanisch, K. Fundel, H.-T. Mevissen, R. Zimmer, and J. Fluck, "Prominer: rule-based protein and gene entity recognition," *BMC Bioinform.*, vol. 6, no. 1, p. S14, 2005.

[47] A. P. Quimbaya, A. S. Múnera, R. A. G. Rivera, J. C. D. Rodríguez, O. M. M. Velandia, A. A. G. Peña, and C. Labbé, "Named entity recognition over electronic health records through a combined dictionary-based approach," *Procedia Comput. Sci.*, vol. 100, pp. 55–61, 2016.

[48] K. Humphreys, R. Gaizauskas, S. Azzam, C. Huyck, B. Mitchell, H. Cunningham, and Y. Wilks, "University of sheffield: Description of the lasie-ii system as used for muc-7," in *MUC-7*, 1998.

[49] G. Krupka and K. IsoQuest, "Description of the nerowl extractor system as used for muc-7," in *MUC-7*, 2005, pp. 21–28.

[50] W. J. Black, F. Rinaldi, and D. Mowatt, "Facile: Description of the ne system used for muc-7," in *MUC-7*, 1998.

[51] C. Aone, L. Halverson, T. Hampton, and M. Ramos-Santacruz, "Sra: Description of the ie2 system used for muc-7," in *MUC-7*, 1998.

[52] D. E. Appelt, J. R. Hobbs, J. Bear, D. Israel, M. Kameyama, D. Martin, K. Myers, and M. Tyson, "Sri international fastus system: Muc-6 test results and analysis," in *MUC-6*, 1995, pp. 237–248.

[53] A. Mikheev, M. Moens, and C. Grover, "Named entity recognition without gazetteers," in *EACL*, 1999, pp. 1–8.

[54] M. Collins and Y. Singer, "Unsupervised models for named entity classification," in *EMNLP*, 1999, pp. 100–110.

[55] D. Nadeau, P. D. Turney, and S. Matwin, "Unsupervised named-entity recognition: Generating gazetteers and resolving ambiguity," in *CSCSI*, 2006, pp. 266–277.

[56] S. Sekine and E. Ranchhod, *Named entities: recognition, classification and use*. John Benjamins Publishing, 2009, vol. 19.

[57] G. Zhou and J. Su, "Named entity recognition using an hmm-based chunk tagger," in *ACL*, 2002, pp. 473–480.

[58] B. Settles, "Biomedical named entity recognition using conditional random fields and rich feature sets," in *ACL*, 2004, pp. 104–107.

[59] W. Liao and S. Veeramachaneni, "A simple semi-supervised algorithm for named entity recognition," in *NAACL-HLT*, 2009, pp. 58–65.

[60] A. Mikheev, "A knowledge-free method for capitalized word disambiguation," in *ACL*, 1999, pp. 159–166.

[61] J. Kazama and K. Torisawa, "Exploiting wikipedia as external knowledge for named entity recognition," in *EMNLP-CoNLL*, 2007.

[62] A. Toral and R. Munoz, "A proposal to automatically build and maintain gazetteers for named entity recognition by using wikipedia," in *Workshop on NEW TEXT Wikis and blogs and other dynamic text sources*, 2006.

[63] 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 *EMNLP*, 2011, pp. 782–792.

[64] Y. Ravin and N. Wacholder, *Extracting names from natural-language text*. IBM Research Report RC 2033, 1997.

[65] J. Zhu, V. Uren, and E. Motta, "Espotter: Adaptive named entity recognition for web browsing," in *WM*. Springer, 2005, pp. 518–529.

[66] Z. Ji, A. Sun, G. Cong, and J. Han, "Joint recognition and linking of fine-grained locations from tweets," in *WWW*, 2016, pp. 1271–1281.

[67] V. Krishnan and C. D. Manning, "An effective two-stage model for exploiting non-local dependencies in named entity recognition," in *ACL*, 2006, pp. 1121–1128.

[68] D. Campos, S. Matos, and J. L. Oliveira, "Biomedical named entity recognition: a survey of machine-learning tools," in *Theory Appl. Adv. Text Min.*, 2012.

[69] S. R. Eddy, "Hidden markov models," *Curr. Opin. Struct. Biol.*, vol. 6, no. 3, pp. 361–365, 1996.

[70] J. R. Quinlan, "Induction of decision trees," *Mach. Learn.*, vol. 1, no. 1, pp. 81–106, 1986.

[71] J. N. Kapur, *Maximum-entropy models in science and engineering*. John Wiley & Sons, 1989.

[72] M. A. Hearst, S. T. Dumais, E. Osuna, J. Platt, and B. Scholkopf, "Support vector machines," *IEEE Intell. Syst. Their Appl.*, vol. 13, no. 4, pp. 18–28, 1998.

[73] J. D. Lafferty, A. McCallum, and F. C. N. Pereira, "Conditional random fields: Probabilistic models for segmenting and labeling sequence data," pp. 282–289, 2001.

[74] D. M. Bikel, S. Miller, R. Schwartz, and R. Weischedel, "Nymble: a high-performance learning name-finder," in *ANLC*, 1997, pp. 194–201.

[75] D. M. Bikel, R. Schwartz, and R. M. Weischedel, "An algorithm that learns what's in a name," *Mach. Learn.*, vol. 34, no. 1-3, pp. 211–231, 1999.

[76] G. Szarvas, R. Farkas, and A. Kocsor, "A multilingual named entity recognition system using boosting and c4. 5 decision tree learning algorithms," in *DS*. Springer, 2006, pp. 267–278.

[77] A. Borthwick, J. Sterling, E. Agichtein, and R. Grishman, "Nyu: Description of the mene named entity system as used in muc-7," in *MUC-7*, 1998.

[78] O. Bender, F. J. Och, and H. Ney, "Maximum entropy models for named entity recognition," in *HLT-NAACL*, 2003, pp. 148–151.

[79] H. L. Chieu and H. T. Ng, "Named entity recognition: a maximum entropy approach using global information," in *CoNLL*, 2002, pp. 1–7.

[80] J. R. Curran and S. Clark, "Language independent ner using a maximum entropy tagger," in *HLT-NAACL*, 2003, pp. 164–167.

[81] P. McNamee and J. Mayfield, "Entity extraction without language-specific resources," in *CoNLL*, 2002, pp. 1–4.

[82] A. McCallum and W. Li, "Early results for named entity recognition with conditional random fields, feature induction and web-enhanced lexicons," in *HLT-NAACL*, 2003, pp. 188–191.

[83] S. Liu, Y. Sun, B. Li, W. Wang, and X. Zhao, "Hamner: Headword amplified multi-span distantly supervised method for domain specific named entity recognition," *arXiv preprint arXiv:1912.01731*, 2019.

[84] A. Ritter, S. Clark, O. Etzioni *et al.*, "Named entity recognition in tweets: an experimental study," in *EMNLP*, 2011, pp. 1524–1534.

[85] X. Liu, S. Zhang, F. Wei, and M. Zhou, "Recognizing named entities in tweets," in *ACL*, 2011, pp. 359–367.

[86] T. Rocktäschel, M. Weidlich, and U. Leser, "Chemspot: a hybrid system for chemical named entity recognition," *Bioinformatics*, vol. 28, no. 12, pp. 1633–1640, 2012.

[87] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, p. 436, 2015.

[88] Y. Shen, H. Yun, Z. C. Lipton, Y. Kronrod, and A. Anandkumar, "Deep active learning for named entity recognition," in *ICLR*, 2017.

[89] T. H. Nguyen, A. Sil, G. Dinu, and R. Florian, "Toward mention detection robustness with recurrent neural networks," *arXiv preprint arXiv:1602.07749*, 2016.

[90] S. Zheng, F. Wang, H. Bao, Y. Hao, P. Zhou, and B. Xu, "Joint extraction of entities and relations based on a novel tagging scheme," in *ACL*, 2017, pp. 1227–1236.

[91] E. Strubell, P. Verga, D. Belanger, and A. McCallum, "Fast and accurate entity recognition with iterated dilated convolutions," in *ACL*, 2017, pp. 2670–2680.

[92] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," in *ICLR*, 2013.

[93] J. Yang, S. Liang, and Y. Zhang, "Design challenges and misconceptions in neural sequence labeling," in *COLING*, 2018, pp. 3879–3889.

[94] L. Yao, H. Liu, Y. Liu, X. Li, and M. W. Anwar, "Biomedical named entity recognition based on deep neutral network," *Int. J. Hybrid Inf. Technol.*, vol. 8, no. 8, pp. 279–288, 2015.

[95] F. Zhai, S. Potdar, B. Xiang, and B. Zhou, "Neural models for sequence chunking." in *AAAI*, 2017, pp. 3365–3371.

[96] P. Zhou, S. Zheng, J. Xu, Z. Qi, H. Bao, and B. Xu, "Joint extraction of multiple relations and entities by using a hybrid neural network," in *CCL-NLP-NABD*. Springer, 2017, pp. 135–146.

[97] X. Ma and E. Hovy, "End-to-end sequence labeling via bi-directional lstm-cnns-crf," in *ACL*, 2016, pp. 1064–1074.

[98] P.-H. Li, R.-P. Dong, Y.-S. Wang, J.-C. Chou, and W.-Y. Ma, "Leveraging linguistic structures for named entity recognition with bidirectional recursive neural networks," in *EMNLP*, 2017, pp. 2664–2669.

[99] C. Wang, K. Cho, and D. Kiela, "Code-switched named entity recognition with embedding attention," in *CALCS*, 2018, pp. 154–158.

[100] O. Kuru, O. A. Can, and D. Yuret, "Charner: Character-level named entity recognition," in *COLING*, 2016, pp. 911–921.

[101] Q. Tran, A. MacKinlay, and A. J. Yepes, "Named entity recognition with stack residual lstm and trainable bias decoding," in *IJCNLP*, 2017, pp. 566–575.

[102] J. Yang, Y. Zhang, and F. Dong, "Neural reranking for named entity recognition," in *RANLP*, 2017, pp. 784–792.

[103] M. E. Peters, M. Neumann, M. Iyyer, M. Gardner, C. Clark, K. Lee, and L. Zettlemoyer, "Deep contextualized word representations," in *NAACL-HLT*, 2018, pp. 2227–2237.

[104] M. Gridach, "Character-level neural network for biomedical named entity recognition," *J. Biomed. Inform.*, vol. 70, pp. 85–91, 2017.

[105] M. Rei, G. K. Crichton, and S. Pyysalo, "Attending to characters in neural sequence labeling models," in *COLING*, 2016, pp. 309–318.

[106] Z. Yang, R. Salakhutdinov, and W. Cohen, "Multi-task cross-lingual sequence tagging from scratch," *arXiv preprint arXiv:1603.06270*, 2016.

[107] A. Akbik, D. Blythe, and R. Vollgraf, "Contextual string embeddings for sequence labeling," in *COLING*, 2018, pp. 1638–1649.

[108] T. Liu, J. Yao, and C. Lin, "Towards improving neural named entity recognition with gazetteers," in *ACL*, 2019, pp. 5301–5307.

[109] A. Ghaddar and P. Langlais, "Robust lexical features for improved neural network named-entity recognition," in *COLING*, 2018, pp. 1896–1907.

[110] Z. Jie and W. Lu, "Dependency-guided lstm-crf for named entity recognition," in *EMNLP*, 2018, pp. 3860–3870.

[111] D. Lu, L. Neves, V. Carvalho, N. Zhang, and H. Ji, "Visual attention model for name tagging in multimodal social media," in *ACL*, 2018, pp. 1990–1999.

[112] Q. Wei, T. Chen, R. Xu, Y. He, and L. Gui, "Disease named entity recognition by combining conditional random fields and bidirectional recurrent neural networks," *Database*, vol. 2016, 2016.

[113] B. Y. Lin, F. Xu, Z. Luo, and K. Zhu, "Multi-channel bilstm-crf model for emerging named entity recognition in social media," in *W-NUT*, 2017, pp. 160–165.

[114] G. Aguilar, S. Maharjan, A. P. L. Monroy, and T. Solorio, "A multi-task approach for named entity recognition in social media data," in *W-NUT*, 2017, pp. 148–153.

[115] P. Jansson and S. Liu, "Distributed representation, lda topic modelling and deep learning for emerging named entity recognition from social media," in *W-NUT*, 2017, pp. 154–159.

[116] M. Xu, H. Jiang, and S. Watcharawittayakul, "A local detection approach for named entity recognition and mention detection," in *ACL*, vol. 1, 2017, pp. 1237–1247.

[117] S. Zhang, H. Jiang, M. Xu, J. Hou, and L. Dai, "A fixed-size encoding method for variable-length sequences with its application to neural network language models," *arXiv preprint arXiv:1505.01504*, 2015.

[118] S. Moon, L. Neves, and V. Carvalho, "Multimodal named entity recognition for short social media posts," in *NAACL*, 2018, pp. 852–860.

[119] J. Devlin, M. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," in *NAACL-HLT*, 2019, pp. 4171–4186.

[120] Y. Wu, M. Jiang, J. Lei, and H. Xu, "Named entity recognition in chinese clinical text using deep neural network," *MEDINFO*, pp. 624–628, 2015.

[121] A. Z. Gregoric, Y. Bachrach, and S. Coope, "Named entity recognition with parallel recurrent neural networks," in *ACL*, vol. 2, 2018, pp. 69–74.

[122] A. Katiyar and C. Cardie, "Nested named entity recognition revisited," in *ACL*, vol. 1, 2018, pp. 861–871.

[123] M. Ju, M. Miwa, and S. Ananiadou, "A neural layered model for nested named entity recognition," in *NAACL-HLT*, vol. 1, 2018, pp. 1446–1459.

[124] M. Rei, "Semi-supervised multitask learning for sequence labeling," in *ACL*, 2017, pp. 2121–2130.

[125] L. Liu, X. Ren, J. Shang, J. Peng, and J. Han, "Efficient contextualized representation: Language model pruning for sequence labeling," in *EMNLP*, 2018, pp. 1215–1225.

[126] L. Liu, J. Shang, F. Xu, X. Ren, H. Gui, J. Peng, and J. Han, "Empower sequence labeling with task-aware neural language model," in *AAAI*, 2017, pp. 5253–5260.

[127] C. Jia, L. Xiao, and Y. Zhang, "Cross-domain NER using cross-domain language modeling," in *ACL*, 2019, pp. 2464–2474.

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

[129] P. J. Liu, M. Saleh, E. Pot, B. Goodrich, R. Sepassi, L. Kaiser, and N. Shazeer, "Generating wikipedia by summarizing long sequences," *arXiv preprint arXiv:1801.10198*, 2018.

[130] N. Kitaev and D. Klein, "Constituency parsing with a self-attentive encoder," in *ACL*, 2018, pp. 2675–2685.

[131] A. Radford, K. Narasimhan, T. Salimans, and I. Sutskever, "Improving language understanding by generative pre-training," *Technical Report, OpenAI*, 2018.

[132] A. Baevski, S. Edunov, Y. Liu, L. Zettlemoyer, and M. Auli, "Cloze-driven pretraining of self-attention networks," *CoRR*, vol. abs/1903.07785, 2019.

[133] C. Xia, C. Zhang, T. Yang, Y. Li, N. Du, X. Wu, W. Fan, F. Ma, and P. S. Yu, "Multi-grained named entity recognition," in *ACL*, 2019, pp. 1430–1440.

[134] Y. Luo, F. Xiao, and H. Zhao, "Hierarchical contextualized representation for named entity recognition," *CoRR*, vol. abs/1911.02257, 2019.

[135] Y. Liu, F. Meng, J. Zhang, J. Xu, Y. Chen, and J. Zhou, "GCDT: A global context enhanced deep transition architecture for sequence labeling," in *ACL*, 2019, pp. 2431–2441.

[136] Y. Jiang, C. Hu, T. Xiao, C. Zhang, and J. Zhu, "Improved differentiable architecture search for language modeling and named entity recognition," in *EMNLP*, 2019, pp. 3576–3581.

[137] X. Li, J. Feng, Y. Meng, Q. Han, F. Wu, and J. Li, "A unified MRC framework for named entity recognition," *CoRR*, vol. abs/1910.11476, 2019.

[138] X. Li, X. Sun, Y. Meng, J. Liang, F. Wu, and J. Li, "Dice loss for data-imbalanced NLP tasks," *CoRR*, vol. abs/1911.02855, 2019.

[139] L. Cui and Y. Zhang, "Hierarchically-refined label attention network for sequence labeling," in *EMNLP*, 2019, pp. 4113–4126.

[140] S. Tomori, T. Ninomiya, and S. Mori, "Domain specific named entity recognition referring to the real world by deep neural networks," in *ACL*, vol. 2, 2016, pp. 236–242.

[141] Y. Lin, L. Liu, H. Ji, D. Yu, and J. Han, "Reliability-aware dynamic feature composition for name tagging," in *ACL*, 2019, pp. 165–174.

[142] J. Zhuo, Y. Cao, J. Zhu, B. Zhang, and Z. Nie, "Segment-level sequence modeling using gated recursive semi-markov conditional random fields," in *ACL*, vol. 1, 2016, pp. 1413–1423.

[143] Z.-X. Ye and Z.-H. Ling, "Hybrid semi-markov crf for neural sequence labeling," in *ACL*, 2018, pp. 235–240.

[144] A. Vaswani, Y. Bisk, K. Sagae, and R. Musa, "Supertagging with lstms," in *NAACL-HLT*, 2016, pp. 232–237.

[145] O. Vinyals, M. Fortunato, and N. Jaitly, "Pointer networks," in *NIPS*, 2015, pp. 2692–2700.

[146] J. Li, A. Sun, and S. Joty, "Segbot: A generic neural text segmentation model with pointer network," in *IJCAI*, 2018, pp. 4166–4172.

[147] Q. Guo, X. Qiu, P. Liu, Y. Shao, X. Xue, and Z. Zhang, "Star-transformer," in *NAACL-HLT*, 2019, pp. 1315–1325.

[148] H. Yan, B. Deng, X. Li, and X. Qiu, "Tener: Adapting transformer encoder for name entity recognition," *arXiv preprint arXiv:1911.04474*, 2019.

[149] Q. Wang, Y. Zhou, T. Ruan, D. Gao, Y. Xia, and P. He, "Incorporating dictionaries into deep neural networks for the chinese clinical named entity recognition," *J. Biomed. Inform.*, vol. 92, 2019.

[150] Y. Zhang and J. Yang, "Chinese ner using lattice lstm," in *ACL*, 2018, pp. 1554–1564.

[151] W. Wang, F. Bao, and G. Gao, "Mongolian named entity recognition with bidirectional recurrent neural networks," in *ICTAI*, 2016, pp. 495–500.

[152] J. Straková, M. Straka, and J. Hajič, "Neural networks for featureless named entity recognition in czech," in *TSD*, 2016, pp. 173–181.

[153] M. Gridach, "Character-aware neural networks for arabic named entity recognition for social media," in *WSSANLP*, 2016, pp. 23–32.

[154] M. K. Malik, "Urdu named entity recognition and classification system using artificial neural network," *ACM Trans. Asian Low-Resour. Lang. Inf. Process.*, vol. 17, no. 1, p. 2, 2017.

[155] T.-H. Pham and P. Le-Hong, "End-to-end recurrent neural network models for vietnamese named entity recognition: Word-level vs. character-level," in *PACLING*, 2017, pp. 219–232.

[156] K. Kurniawan and S. Louvan, "Empirical evaluation of character-based model on neural named-entity recognition in indonesian conversational texts," *arXiv preprint arXiv:1805.12291*, 2018.

[157] K. Yano, "Neural disease named entity extraction with character-based bilstm+ crf in japanese medical text," *arXiv preprint arXiv:1806.03648*, 2018.

[158] A. Bharadwaj, D. Mortensen, C. Dyer, and J. Carbonell, "Phonologically aware neural model for named entity recognition in low resource transfer settings," in *EMNLP*, 2016, pp. 1462–1472.

[159] J. Xie, Z. Yang, G. Neubig, N. A. Smith, and J. Carbonell, "Neural cross-lingual named entity recognition with minimal resources," in *EMNLP*, 2018, pp. 369–379.

[160] Y. Lin, S. Yang, V. Stoyanov, and H. Ji, "A multi-lingual multi-task architecture for low-resource sequence labeling," in *ACL*, 2018, pp. 799–809.

[161] R. Caruana, "Multitask learning," *Mach. learn.*, vol. 28, no. 1, pp. 41–75, 1997.

[162] N. Peng and M. Dredze, "Multi-task domain adaptation for sequence tagging," in *RepL4NLP*, 2017, pp. 91–100.

[163] G. Crichton, S. Pyysalo, B. Chiu, and A. Korhonen, "A neural network multi-task learning approach to biomedical named entity recognition," *BMC Bioinform.*, vol. 18, no. 1, p. 368, 2017.

[164] X. Wang, Y. Zhang, X. Ren, Y. Zhang, M. Zitnik, J. Shang, C. Langlotz, and J. Han, "Cross-type biomedical named entity recognition with deep multi-task learning," *arXiv preprint arXiv:1801.09851*, 2018.

[165] S. J. Pan, Q. Yang *et al.*, "A survey on transfer learning," *IEEE Trans. Knowl. Data Eng.*, vol. 22, no. 10, pp. 1345–1359, 2010.

[166] J. Jiang and C. Zhai, "Instance weighting for domain adaptation in nlp," in *ACL*, 2007, pp. 264–271.

[167] D. Wu, W. S. Lee, N. Ye, and H. L. Chieu, "Domain adaptive bootstrapping for named entity recognition," in *EMNLP*, 2009, pp. 1523–1532.

[168] A. Chaudhary, J. Xie, Z. Sheikh, G. Neubig, and J. G. Carbonell, "A little annotation does a lot of good: A study in bootstrapping low-resource named entity recognizers," pp. 5163–5173, 2019.

[169] S. J. Pan, Z. Toh, and J. Su, "Transfer joint embedding for cross-domain named entity recognition," *ACM Trans. Inf. Syst.*, vol. 31, no. 2, p. 7, 2013.

[170] J. Y. Lee, F. Dernoncourt, and P. Szolovits, "Transfer learning for named-entity recognition with neural networks," *arXiv preprint arXiv:1705.06273*, 2017.

[171] B. Y. Lin and W. Lu, "Neural adaptation layers for cross-domain named entity recognition," in *EMNLP*, 2018, pp. 2012–2022.

[172] Y. Cao, Z. Hu, T. Chua, Z. Liu, and H. Ji, "Low-resource name tagging learned with weakly labeled data," in *EMNLP*, 2019, pp. 261–270.

[173] X. Huang, L. Dong, E. Boschee, and N. Peng, "Learning A unified named entity tagger from multiple partially annotated corpora for efficient adaptation," in *CoNLL*, 2019, pp. 515–527.

[174] L. Qu, G. Ferraro, L. Zhou, W. Hou, and T. Baldwin, "Named entity recognition for novel types by transfer learning," in *EMNLP*, 2016, pp. 899–905.

[175] Z. Yang, R. Salakhutdinov, and W. W. Cohen, "Transfer learning for sequence tagging with hierarchical recurrent networks," in *ICLR*, 2017.

[176] P. von Däniken and M. Cieliebak, "Transfer learning and sentence level features for named entity recognition on tweets," in *W-NUT*, 2017, pp. 166–171.

[177] H. Zhao, Y. Yang, Q. Zhang, and L. Si, "Improve neural entity recognition via multi-task data selection and constrained decoding," in *NAACL-HLT*, vol. 2, 2018, pp. 346–351.

[178] G. Beryozkin, Y. Drori, O. Gilon, T. Hartman, and I. Szpektor, "A joint named-entity recognizer for heterogeneous tag-sets using a tag hierarchy," in *ACL*, 2019, pp. 140–150.

[179] J. M. Giorgi and G. D. Bader, "Transfer learning for biomedical named entity recognition with neural networks," *Bioinformatics*, 2018.

[180] Z. Wang, Y. Qu, L. Chen, J. Shen, W. Zhang, S. Zhang, Y. Gao, G. Gu, K. Chen, and Y. Yu, "Label-aware double transfer learning for cross-specialty medical named entity recognition," in *NAACL-HLT*, 2018, pp. 1–15.

[181] B. Settles, "Active learning," *Synth. Lect. Artif. Intell. Mach. Learn.*, vol. 6, no. 1, pp. 1–114, 2012.

[182] D. D. Lewis and W. A. Gale, "A sequential algorithm for training text classifiers," in *SIGIR*, 1994, pp. 3–12.
[183] S. Pradhan, A. Moschitti, N. Xue, H. T. Ng, A. Björkelund, O. Uryupina, Y. Zhang, and Z. Zhong, "Towards robust linguistic analysis using ontonotes," in *CoNLL*, 2013, pp. 143–152.
[184] L. P. Kaelbling, M. L. Littman, and A. W. Moore, "Reinforcement learning: A survey," *J. Artif. Intell. Res.*, vol. 4, pp. 237–285, 1996.
[185] R. S. Sutton and A. G. Barto, *Introduction to reinforcement learning*. MIT press Cambridge, 1998, vol. 135.
[186] S. C. Hoi, D. Sahoo, J. Lu, and P. Zhao, "Online learning: A comprehensive survey," *arXiv preprint arXiv:1802.02871*, 2018.
[187] K. Narasimhan, A. Yala, and R. Barzilay, "Improving information extraction by acquiring external evidence with reinforcement learning," in *EMNLP*, 2016, pp. 2355–2365.
[188] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski *et al.*, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, p. 529, 2015.
[189] Y. Yang, W. Chen, Z. Li, Z. He, and M. Zhang, "Distantly supervised NER with partial annotation learning and reinforcement learning," in *COLING*, 2018, pp. 2159–2169.
[190] D. Lowd and C. Meek, "Adversarial learning," in *SIGKDD*, 2005, pp. 641–647.
[191] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *NIPS*, 2014, pp. 2672–2680.
[192] L. Huang, H. Ji, and J. May, "Cross-lingual multi-level adversarial transfer to enhance low-resource name tagging," in *NAACL-HLT*, 2019, pp. 3823–3833.
[193] J. Li, D. Ye, and S. Shang, "Adversarial transfer for named entity boundary detection with pointer networks," in *IJCAI*, 2019, pp. 5053–5059.
[194] P. Cao, Y. Chen, K. Liu, J. Zhao, and S. Liu, "Adversarial transfer learning for chinese named entity recognition with self-attention mechanism," in *EMNLP*, 2018, pp. 182–192.
[195] J. T. Zhou, H. Zhang, D. Jin, H. Zhu, M. Fang, R. S. M. Goh, and K. Kwok, "Dual adversarial neural transfer for low-resource named entity recognition," in *ACL*, 2019, pp. 3461–3471.
[196] D. Britz, "Attention and memory in deep learning and nlp," *Online: http://www. wildml. com/2016/01/attention-and-memory-in-deeplearning-and-nlp*, 2016.
[197] A. Zukov-Gregoric, Y. Bachrach, P. Minkovsky, S. Coope, and B. Maksak, "Neural named entity recognition using a self-attention mechanism," in *ICTAI*, 2017, pp. 652–656.
[198] G. Xu, C. Wang, and X. He, "Improving clinical named entity recognition with global neural attention," in *APWeb-WAIM*, 2018, pp. 264–279.
[199] Q. Zhang, J. Fu, X. Liu, and X. Huang, "Adaptive co-attention network for named entity recognition in tweets," in *AAAI*, 2018.
[200] L. Derczynski, E. Nichols, M. van Erp, and N. Limsopatham, "Results of the wnut2017 shared task on novel and emerging entity recognition," in *W-NUT*, 2017, pp. 140–147.
[201] J. Fisher and A. Vlachos, "Merge and label: A novel neural network architecture for nested NER," in *ACL*, 2019, pp. 5840–5850.
[202] D. Ye, Z. Xing, C. Y. Foo, Z. Q. Ang, J. Li, and N. Kapre, "Software-specific named entity recognition in software engineering social content," in *SANER*, 2016, pp. 90–101.
[203] I. Partalas, C. Lopez, N. Derbas, and R. Kalitvianski, "Learning to search for recognizing named entities in twitter," in *W-NUT*, 2016, pp. 171–177.
[204] 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 Trans. Knowl. Data Eng.*, vol. 30, no. 2, pp. 353–366, 2018.
[205] M. C. Phan, A. Sun, Y. Tay, J. Han, and C. Li, "Pair-linking for collective entity disambiguation: Two could be better than all," *arXiv preprint arXiv:1802.01074*, 2018.
[206] C. Li and A. Sun, "Extracting fine-grained location with temporal awareness in tweets: A two-stage approach," *J. Assoc. Inf. Sci. Technol.*, vol. 68, no. 7, pp. 1652–1670, 2017.
[207] J. Han, A. Sun, G. Cong, W. X. Zhao, Z. Ji, and M. C. Phan, "Linking fine-grained locations in user comments," *IEEE Trans. Knowl. Data Eng.*, vol. 30, no. 1, pp. 59–72, 2018.
[208] M. C. Phan and A. Sun, "Collective named entity recognition in user comments via parameterized label propagation," *J. Assoc. Inf. Sci. Technol.*, 2019.
[209] Z. Batmaz, A. Yurekli, A. Bilge, and C. Kaleli, "A review on deep learning for recommender systems: challenges and remedies," *Artif. Intell. Rev.*, pp. 1–37, 2018.
[210] M. Röder, R. Usbeck, and A. N. Ngomo, "GERBIL - benchmarking named entity recognition and linking consistently," *Semantic Web*, vol. 9, no. 5, pp. 605–625, 2018.
[211] M. Ott, S. Edunov, A. Baevski, A. Fan, S. Gross, N. Ng, D. Grangier, and M. Auli, "fairseq: A fast, extensible toolkit for sequence modeling," in *NAACL-HLT*, 2019, pp. 48–53.
[212] F. Dernoncourt, J. Y. Lee, and P. Szolovits, "NeuroNER: an easy-to-use program for named-entity recognition based on neural networks," in *EMNLP*, 2017, pp. 97–102.