1 Introduction

Natural Language Processing (NLP) is a field of computer science, artificial intelligence, and computational linguistics concerned with fruitfully processing large natural language corpora. There are many commonly researched tasks in NLP that work on syntax, semantics, discourse, and speech. Relation extraction is one of the tasks for semantics, and specifically, relation classification is one of its major subtasks. The task of relation classification is to predict semantic relations between pairs of nominal entities in a sentence. In other words, given a pair of nominal entities, we aim to fill the relation type (or called “relation slot”) between them (Zeng et al., 2014). In 1999, National Institute of Standards and Technology (NIST) initiated the Automatic Content Extraction (ACE) program in order to develop technologies for extracting and characterizing meanings of human languages in the form of text. The ACE corpora, especially ACE04 and ACE05, have been widely used for end-to-end relation extraction tasks. In the International Workshop on Semantic Evaluation (SemEval) 2010, SemEval, an ongoing series of evaluations of computational semantic analysis systems, released a benchmark (often referred to as “SemEval-2010 Task 8”) (Hendrickx et al., 2009). Since then, it has become the most popular dataset for evaluating the performance on relation classification. Table 1 shows an example of classifying the relation between two nominal entities in the SemEval dataset. The task of “relation-classification slot filling (RC-SF)” is defined as below: Given a sentence and two nominal entities in the sentence, identify the most probable semantic relation between the entities (Zeng et al., 2014). Table 2 shows most popular datasets used in relation classification evaluation.

Bach et al. gave a nice overview through relation classification methods before 2007 (Bach and Badaskar, 2007). They began with supervised approaches which formulate the relation extraction task as a binary classification problem. Further, they discussed feature-based methods (Kambhatla, 2004; Zhao and Grishman, 2005) and kernel-based methods (Zelenko et al., 2003; Culotta and Sorensen, 2004; Bunescu and Mooney, 2005; Mooney and Bunescu, 2006) for supervised relation classification. Pattern mining, semi-supervised bootstrapping, and their combination have been another important line of relation classification approaches. DIPRE (Brin, 1998) and Snowball (Agichtein and Gravano, 2000) started from a bag of high-quality tagged seeds to extract and expand relational patterns. KnowItAll (Etzioni et al., 2005) and TextRunner (Banko et al., 2007) proposed large scale relation extraction systems based on a self-trained binary relation classifier.

In 2009, Mintz et al. proposed a distant supervised method to fetch large-scale training data without heavy manual annotation (Mintz et al., 2009). The basic assumption is “if two entities participate in a relation, all sentences that mention these two entities express that relation.” (Riedel et al., 2010) Following this assumption, they align text data with knowledge bases (KBs) like Freebase to gain labeled data for training and testing. This provides a large amount of labeled data that were hardly able to obtain from human annotations on the specific text data with little cost. Considerable amount of works have taken this setting (Lin et al., 2016; Zeng et al., 2014, 2015; Riedel et al., 2010). Some following works try to alleviate noise generated in the alignment by multi-instance learning (Riedel et al., 2010; Hoffmann et al., 2011; Surdeanu et al., 2012) and embedding techniques (Ren et al., 2016).

The main weakness of these conventional methods is that most features are explicitly derived from NLP tools such as Part-of-Speech (POS) tagging and errors generated by the NLP tools will propagate through the pipeline (Lin et al., 2016). Since the year of 2014, we have seen a move towards deep architectures that are capable of learning relevant representations and features without extensive manual feature engineering or use of external resources. A number of models based on convolutional neural networks (Liu et al., 2013; Zeng...
Table 1: An example of the input and output of RC-SF

<table>
<thead>
<tr>
<th>Sentences</th>
<th>“Smoking causes cancer.” “The farmer grows apple.”</th>
</tr>
</thead>
<tbody>
<tr>
<td>Slots</td>
<td>☐(smoking, cancer); ☐(apple, farmer)</td>
</tr>
<tr>
<td>Output</td>
<td>Cause-Effect(smoking, cancer); Product-Producer(apple, farmer)</td>
</tr>
</tbody>
</table>

Table 2: Datasets for Relation Classification

<table>
<thead>
<tr>
<th></th>
<th>SemEval</th>
<th>NYTC</th>
<th>Others</th>
</tr>
</thead>
<tbody>
<tr>
<td>(Riedel et al., 2010)</td>
<td>✔</td>
<td></td>
<td></td>
</tr>
<tr>
<td>(Hearst et al., 2013)</td>
<td>✔</td>
<td></td>
<td></td>
</tr>
<tr>
<td>(Zeng et al., 2014)</td>
<td>✔</td>
<td></td>
<td></td>
</tr>
<tr>
<td>(Zeng et al., 2015)</td>
<td>✔</td>
<td></td>
<td></td>
</tr>
<tr>
<td>(Cai et al., 2016)</td>
<td>✔</td>
<td></td>
<td></td>
</tr>
<tr>
<td>(Lin et al., 2016)</td>
<td>✔</td>
<td></td>
<td></td>
</tr>
<tr>
<td>(Yang et al., 2016)</td>
<td>✔</td>
<td></td>
<td></td>
</tr>
<tr>
<td>(Xiao and Liu, 2016)</td>
<td>✔</td>
<td></td>
<td></td>
</tr>
<tr>
<td>(Wang et al., 2016)</td>
<td>✔</td>
<td></td>
<td></td>
</tr>
<tr>
<td>(Miwa and Bansal, 2016)</td>
<td>✔</td>
<td>✓ACE04, 05</td>
<td>✓</td>
</tr>
<tr>
<td>(Takase and Inui, 2016)</td>
<td>✔</td>
<td>✓ukWac</td>
<td></td>
</tr>
<tr>
<td>(Li et al., 2017)</td>
<td>✔</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Figure 1: The main components of Snowball (Agichtein and Gravano, 2000). The bootstrapping process is similar to DIPRE (Brin, 1998).

2 Pattern-based Methods for RC-SF

Pattern-based methods can be traced back to 20 years ago, when Brin et al. (Brin, 1998) proposed the Dual Iterative Pattern Relation Expansion (DIPRE) system to extract (author, book) pairs from the Web. DIPRE defines a pattern as a five-tuple: [order, url, prefix, middle, suffix], where order is a boolean value indicating the order in which the entities appear, and other attributes are strings. The prefix, middle and suffix are m characters before, between and after the target entities respectively. url indicates the URL of the document they occurred on. They start with a group of carefully selected “seed” book-author pairs, then search through Web pages for their occurrences where the book title and the author name appear in the same sentence. The pattern is generated by the longest common substrings of prefix/middle/suffix/url from these occurrences with a wild card expression, and used to search through the Web pages to find matches. Entities in a matched sentence are considered to have a book-author relation. Then, DIPRE follows a semi-supervised bootstrapping procedure to expand the pattern set. The extracted pairs are added to the “seed” set as labeled data for the next iteration of pattern generation.

Recently, CNNs, especially Attention CNNs, are the most popular models for the RC-SF task. On the SemEval-2010 Task 8 dataset, the best F1 score reported so far is 88.0%, which is achieved by a multi-level attention CNN model (Wang et al., 2016). We will review methods for RC-SF in the following sections.
Table 3: Popular relation-classification slot filling methods.

<table>
<thead>
<tr>
<th>Method</th>
<th>Pattern</th>
<th>Kernel</th>
<th>Feature</th>
<th>RecursiveNN</th>
<th>CNN</th>
<th>RNN</th>
<th>LSTM</th>
<th>Attention</th>
<th>Others</th>
</tr>
</thead>
<tbody>
<tr>
<td>(Brin, 1998)</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(Agichtein and Gravano, 2000)</td>
<td></td>
<td></td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>(Zelenko et al., 2003)</td>
<td></td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(Culotta and Sorensen, 2004)</td>
<td></td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(Kambhatla, 2004)</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(Bunescu and Mooney, 2005)</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(Zhao and Grishman, 2000)</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(GuoDong et al., 2005)</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(Mooney and Bunescu, 2006)</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(Mintz et al., 2009)</td>
<td></td>
<td></td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>(Socher et al., 2012)</td>
<td></td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(Hashimoto et al., 2013)</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(Riedel et al., 2013)</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>(Liu et al., 2014)</td>
<td>✓</td>
<td></td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(Zeng et al., 2014)</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(Zeng et al., 2015)</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(Zhang et al., 2015)</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(Luo et al., 2015)</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(Xu et al., 2015)</td>
<td>✓</td>
<td></td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(Hu et al., 2015)</td>
<td>✓</td>
<td></td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(Nguyen and Grishman, 2015)</td>
<td>✓</td>
<td></td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(Santos et al., 2015)</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(Jiang et al., 2016)</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(Cai et al., 2016)</td>
<td>✓</td>
<td></td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(Xiao and Liu, 2016)</td>
<td>✓</td>
<td></td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(Shen and Huang, 2016)</td>
<td>✓</td>
<td></td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(Miwa and Bansal, 2016)</td>
<td>✓</td>
<td></td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(Liu et al., 2016)</td>
<td>✓</td>
<td></td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(Ren et al., 2016)</td>
<td>✓</td>
<td></td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(Wang et al., 2016)</td>
<td>✓</td>
<td></td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(Ji et al., 2017)</td>
<td>✓</td>
<td></td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(Liu et al., 2017)</td>
<td>✓</td>
<td></td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Snowball defines the similarity score as the sum of local similarities.

After grouping tuples into classes, Snowball induces a single tuple pattern \( P \) for each class which is a centroid vector tuple. Each pattern \( P \) is assigned a confidence score which measure the quality of a newly-proposed pattern

\[
\text{Confidence}(P) = \frac{P_{\text{positive}}}{P_{\text{positive}} + P_{\text{negative}}},
\]

where \( P_{\text{positive}} \) is the number of times the new pattern recovers an (organization, location) pair seen at a previous iteration of training; and \( P_{\text{negative}} \) is number of times it recovers a pair where the organization has been seen with a different location at a previous iteration. To label new data, Snowball first runs a named-entity recognizer over the data to identify all location and organization entities. Within a sentence, for each (organization, location) relation pair the system forms a tuple. Therefore, a popular pair will have a set of tuples associated with it, and new patterns are induced. The system matches each relation candidate with all patterns and only keeps candidates that have similarity score greater than some threshold. Next, Snowball assigns a high confidence score for a relation candidate when the candidate is matched by many tuples with high similarity to pattern that the system confident in. Finally, the new relation is added to the seed set and the process is repeated iteratively.

Compared with DIPRE, the major novelty of Snowballs are (1) Contexts are represented as feature vectors, instead of plain strings. This allows a flexible similarity measurement between patterns that cannot be exactly matched. (2) Named-entity recognizers are introduced to improve precision. (3) Generated patterns and tuples are evaluated with a confidence score. Only reliable results are added to the seed set for the next iteration. This effectively alleviates error propagation in the bootstrapping process.

Zhu et al. (Zhu et al., 2009) further developed the Snowball system in 2009. As they point out, despite the significant technical contribution of Snowball, it has two obvious limitations. (1) The target of Snowball is to extract a specific type of relation, which makes it hard to scale up. (2) The evaluation measures and pattern selection criteria are not directly adaptable to general patterns, which can hurt the recall. Zhu et al. presented a system called Statistical Snowball (StatSnowball). StatSnowball adopted the bootstrapping architecture and applies feature selection methods using \( l_1 \)-norm (Tibshirani, 1996; Kabán, 2007) to select extraction patterns – both keyword matching and general patterns. Starting with a handful set of initial seeds, it iteratively generates new extraction
patterns; performs an $l_1$-norm regularized maximum likelihood estimation (MLE) to select good patterns; and extracts new relation tuples. StatSnowball is a general framework and the statistical model can be any probabilistic model. StatSnowball uses the general discriminative Markov logic networks (MLN) (Richardson and Domingos, 2006), which subsume logistic regression (LR) and conditional random fields (CRF). Discriminative models can incorporate arbitrary useful features without strong independence assumptions as made in generative models, like Naive Bayes (NB) and Hidden Markov Models (HMM). By incorporating general patterns, StatSnowball can perform both traditional relation extraction like Snowball to extract pre-specified relations and Open IE (Banko et al., 2007) to identify general types of relations.

3 Feature-based Methods for RC-SF

Following the definition by Bach et al. (Bach and Badaskar, 2007), RC-SF tasks take a sentence $S = w_1, w_2, ..., e_1, w_j, e_2, ..., w_n$ as input, where $e_1$ and $e_2$ are the entities, a mapping function $f(.)$ can be given as

$$f_R(T(S)) = \begin{cases} +1, & \text{if } R(e_1, e_2) \\ -1, & \text{otherwise} \end{cases}$$

where $T(S)$ are features that are extracted from $S$. Essentially the mapping function $f(.)$ decides whether the entities in the sentence are in a relation or not. If a labeled set of positive and negative relation examples are available for training, the function $f(.)$ can be constructed as a discriminative classifier like Perceptron, Voted Perceptron or Support Vector Machines (SVMs). These classifiers can be trained using a set of features selected after performing textual analysis (like POS tagging, dependency parsing, etc.) of the labeled sentences. On the other hand, input to the classifiers can also take the form of rich structural representations like parse trees.

Syntactic features extracted from the sentence include (1) the entities themselves, (2) the types of the two entities, (3) word sequence between the entities, (4) number of words between the entities and (5) path in the parse tree containing the two entities. Semantic cues include the path between the two entities in the dependency parse. Both the semantic and syntactic features extracted are presented to the classifier in the form of a feature vector, for training or classification. Kambhatla et al. (Kambhatla, 2004) trains a log-linear model using the features described, for the task of entity classification. On the other hand, Zhao et al. (Zhao and Grishman, 2005) and Zhou et al. (GuoDong et al., 2005) use SVMs trained on these features using polynomial and linear kernels respectively for classifying different types of entity relations. In 2006, Culotta et al. (Culotta et al., 2006) proposed an integrated supervised machine learning method that learns both contextual and relational patterns to extract relations. In particular, we construct a linear-chain conditional random field (CRF) to extract relations from biographical texts while simultaneously discovering interesting relational patterns that improve extraction performance.

Feature based methods involve heuristic choices and the features have to be selected on a trial-and-error basis in order to maximize performance. Since NLP applications in general and relation extraction in particular involve structured representations of the input data, it can be difficult to arrive at an optimal subset of relevant features. To remedy the problem of selecting a suitable feature-set, specialized kernels are designed for relation extraction in order to exploit rich representations of the input data like shallow parse trees etc. These kernels explore the input representations exhaustively in an implicit manner in a higher dimensional space. The kernels used for relation-extraction (or relation-detection) are based on string-kernels described in Lodhi et al. (Lodhi et al., 2002). Bach et al. (Bach and Badaskar, 2007) gave a comprehensive introduction of the use of string-kernels in relation classification problems, which we will not repeat here. In kernel approaches, the sentence $S$ and the target entities $e_1$ and $e_2$ can be represented in two ways: (1) word sequences around the entities under question or (2) parse trees containing the entities. Depending on the choice of representation, we discuss two major kernel approaches: Bag of features kernels and Tree kernels.

Mooney et al. (Mooney and Bunescu, 2006) use the word-context around the named entities for extracting protein interactions from the MEDLINE abstracts. A sentence $S = w_1, w_2, ..., e_1, w_j, e_2, ..., w_n$ containing related entities $e_1$ and $e_2$ can be described as $S = s_b e_1 s_{m1} e_2 s_{na}$. Here $s_b$, $s_m$ and $s_a$ are portions of word-context before, middle and after the related entities respectively. Now given a test sentence $S_t$ containing entities $e_1'$ and $e_2'$, the similarity of its before, middle and after portions with those of sentence $S$ is computed using the sub-sequence kernel. The kernel computes similarity between two sequences at the word level rather than character level (as in string-kernels). In Mooney et al. (Mooney and Bunescu, 2006), three sub-sequence kernels are defined, one each for matching the before, middle and after portions of the entities context and the combined kernel is simply the sum of all the subkernels. Using subse-
quence kernels in conjunction with SVMs improves both the precision and recall of the relation classification task over a set of 225 MEDLINE abstracts when compared to a existing rule based system. Further, they also augment the words in the context with their respective part-of-speech (POS) tags, entity types etc to improve over the dependency tree kernel approach of Culotta et al. (Culotta and Sorensen, 2004).

Compared to the previous approach, Zelenko et al. (Zelenko et al., 2003) replace the strings in the kernel with structured shallow parse trees built on the sentence. They use the tree kernels and plug them into SVM and Voted Perceptron for the task of extracting person-affiliation and organization location relations from text. Thus, the kernel is designed to compute the similarity between two entity-augmented shallow parse tree structures. Given a sentence, its shallow parse is constructed first. The rationale for using shallow parse trees instead of full parse trees is that they tend to be more robust and reliable. Apart from phrasal information, the shallow parser augments the tree with entity information.

To sum up, Culotta et al. (Culotta and Sorensen, 2004) and Zelenko et al. (Zelenko et al., 2003) use rich structural information in the form of trees in order to obtain a decent performance in the relation extraction task. In contrast, Bunescu et al. (Bunescu and Mooney, 2005) make an interesting observation that the shortest path between the two entities in a dependency parse encodes sufficient information to extract the relation between them. If \( e_1 \) and \( e_2 \) are two entities in a sentence and \( p \) their predicate, then the shortest path between \( e_1 \) and \( e_2 \) passes through \( p \). This is because, \( e_1 \) and \( e_2 \) are the arguments of \( p \). Given a sentence \( S \), a dependency tree is first extracted. Then the shortest path between the entity pairs in a sentence are computed. Let one such path be \( P = e_1 \rightarrow w_1 \rightarrow \ldots \rightarrow w_i \leftarrow \ldots \leftarrow w_n \leftarrow e_2 \). Here, \( w_i \) are the words in the shortest path and arrows indicate the direction of dependency as extracted from the tree. Using \( P \) alone as a feature would lead to bad performance due to data sparsity. Therefore word-classes like POS are extracted from each of the words and the feature vectors is given by a Cartesian product. The kernel over this feature space is defined as

\[
K(x, y) = \begin{cases} 
0, & \text{if } |x| \neq |y|; \\
\prod_{i=1}^{|x|} f(x_i, y_i), & \text{otherwise},
\end{cases}
\]

where \( f(x_i, y_i) \) is the number of word-classes common to \( x_i \) and \( y_i \). The advantage of this kernel over the ones proposed by Zelenko et al. (Zelenko et al., 2003) and Culotta et al. (Culotta and Sorensen, 2004) is that its computation requires linear time apart from having a simplified feature space. At the same time, the performance of the relation extraction is shown to be better than Culotta’s tree based approach. In fact the recall of the shortest-path dependency kernel is found to be significantly better than Culotta’s tree based kernel while having a comparable precision.

4 Distant Supervision for RC-SF

Supervised methods are limited by a lack of large-scale training data, which requires intensive manual annotation. Pattern-based bootstrapping methods significantly reduce the number of training examples by iteratively extracting patterns and identifying entity relations with a small number of seeds, but are often limited by low recall and error aggregation.

To tackle these problems, Mintz et al. proposed an alternative paradigm, distant supervision (Mintz et al., 2009). The intuition of distant supervision is that any sentence that contains a pair of entities that participate in a known relation in a knowledge base (e.g., Freebase) is likely to express that relation in some way. They then align text data with these knowledge bases. For each pair of entities, they aggregate the features from the many different sentences in which that pair appeared into a single feature vector. The extracted features and corresponding relation labels are then fed to a relation classifier, for instance, a logistic regression classifier. This approach allows them to extract 10,000 instances of 102 relations at a precision of 67.6%, using Freebase as supervision source. They argue that since the algorithm is supervised by a database, rather than by labeled text, it does not suffer from the problems of over-fitting and domain-dependence that plague supervised systems.

Through distant supervision, one can easily fetch large-scale training data with little cost. Considerable amount of works have taken this setting (Lin et al., 2016; Zeng et al., 2014, 2015; Riedel et al., 2010). However, the coverage of distant supervision is somehow limited by the schema of its source knowledge base. This can be avoided by using language itself as the source of the schema, as the approach taken by OpenIE (Banko et al., 2007). Here surface patterns between mentions of concepts serve as relations. This approach requires no supervision and has tremendous flexibility, but lacks the ability to generalize for lack of insights of semantic correlations between textual relation mentions (Riedel et al., 2013). In response of these problems, Riedel et al. proposed the universal schema (Riedel et al., 2013): the union of surface form predicates as in OpenIE and relations in the schemas of pre-existing databases. They represent the probabilistic knowledge base as a ma-
A matrix factorization approach based on distant supervision fills up a database of universal schema (Riedel et al., 2013). Dark circles are observed facts, shaded circles are inferred facts. The rows come from running cross-document entity resolution across pre-existing structured knowledge base and textual corpora. The columns come from the union of surface forms and KB relations. Similar to Nickel et al.’s work on factorizing YAGO to predict new links (Nickel et al., 2012), they extend matrix factorization and collaborative filtering models (Collins et al., 2002; Koren, 2008; Rendle et al., 2009) to predict surface patterns relationships which do not appear explicitly in text, and to learn latent representations of entities, tuples and relations simultaneously. Through this, they model outperforms the state-of-the-art (in 2012) distant supervision method (Surdeanu et al., 2012) by 10% points Mean Average Precision. Similar idea is presented in Weston et al.’s work (Weston et al., 2013) on connecting free text and existing knowledge base by low-dimensional embedding. CoType uses these learned embeddings and estimates the types of unknown entity and relation mentions. The objective function is designed under three hypothesis. (1) Mention-Feature co-occurrence: two relation mentions tend to share similar types (closer in the embedding space) if they share many text features in the corpus, and the converse way also holds. (2) Partial-label association: a relation mention’s embedding vector should be more similar (closer in the low-dimensional space) to its “most relevant” candidate type, than to any other non-candidate type. (3) Entity-relation interaction: for a relation mention \( m = (e_1, e_2, r) \), embedding vector of \( e_1 \) should be a nearest neighbor of the embedding vector of \( e_2 \) plus the embedding vector of relation mention \( m \). By doing so, text features, as complements to mention’s candidate types, also participate in modeling the relation mention embeddings, and help identify a mention’s most relevant type – mention-type relevance is progressively estimated.
during model learning. Hypothesis(3) poses constraints on the search space for the relation types of the relation mention with entity types of the relation arguments. (e.g., it is unlikely to find an author_of relation between an organization entity and a location entity). They then use a stochastic sub-gradient descent algorithm to optimize the objective function. With the learned embeddings of both entities and relations, they perform nearest neighbor search in the target relation type set for relation classification. As experiments show, they obtain over 10% enhancement on both two datasets over the next best method.

5 Deep Learning Methods for RC-SF

Recently, neural network approaches have been widely used for relation classification, which aim at reducing the need of hand-crafted features. These approaches are broadly divided into two categories (Xiao and Liu, 2016): one explores the effectiveness of using dependency paths and its attached subtrees between two nominals, and various neural networks are adopted to model the shortest dependency paths and dependency subtrees (Xu et al., 2015a,b; Liu et al., 2015; Yang et al., 2016); the other exploits deep neural networks to learn syntactic and semantic features from raw sentences (Zeng et al., 2014; Santos et al., 2015; Zhang et al., 2015). Popular models include convolutional neural networks (Liu et al., 2013; Zeng et al., 2014; Santos et al., 2015; Xu et al., 2015b,a; Nguyen and Grishman, 2015; Zeng et al., 2015; Lin et al., 2016; Wang et al., 2016; Yang et al., 2016; Shen and Huang, 2016; Jiang et al., 2016; Ji et al., 2017), recurrent neural networks (Zhang et al., 2015; Xu et al., 2015b; Miwa and Bansal, 2016; Xiao and Liu, 2016), recursive neural networks (Socher et al., 2012; Hashimoto et al., 2013; Liu et al., 2015), and their combinations (Liu et al., 2015; Cai et al., 2016).

5.1 Convolutional Neural Networks for RC-SF

The Convolutional Neural Network (CNN) is the most popular deep learning model for relation classification. Instead of hand-craft features, it takes word vectors as input. Assume that we embed word vectors as input. Assume that we take into account that at least one sentence in the bag that mentions these two entities will express their relation between an author and a location entity. They then use a stochastic sub-gradient descent algorithm to optimize the objective function. With the learned embeddings of both entities and relations, they perform nearest neighbor search in the target relation type set for relation classification. As experiments show, they obtain over 10% enhancement on both two datasets over the next best method.

The Convolutional Neural Network (CNN) is the most popular deep learning model for relation classification. Instead of hand-craft features, it takes word vectors as input. Assume that we have the following sequence of words (Zeng et al., 2014): S : [People]_0 have_1 been_2 moving_3 back_4 into_5 [downtown]_6, where people and downtown are target entities. All of the word tokens of the sentence S are then represented as a list of vectors \( [x_0, x_1, ..., x_6] \), where \( x_i \) corresponds to the word embedding of the \( i \)-th word in the sentence. Words in a context window of size \( w \) are combined together. When we take \( w = 3 \), the word feature WF of the third word “moving” is expressed as \( [x_2, x_3, x_4] \). Considering the whole sentence, the WF can be represented as \( \{ [x_5, x_6, x_7], [x_0, x_1, x_2], ..., [x_5, x_6, x_7] \} \). In addition to traditional word embedding, relative distances between words, named position features or position embedding, are included to capture structural information. These features, along with lexical features generated by external NLP tools, are concatenated to form the final word embedding. A standard convolution layer is used to merge all the features. They then perform a max operation (max pooling) to select the most useful features. Another advantage of max pooling is that the dimension of feature vectors is no longer related to the sentence length. At the top level, a softmax classifier assigns probabilities of candidate relation types. The CNN model significantly outperform state-of-the-art at that time. Feature contribution experiments show that by adding position features, the system achieves approximately 9.2% F1-score improvement.

Distant supervision provides large-scale training data with little cost, while also brings potentially noisy data (4). Multi-instance learning methods are proposed to alleviate this problem, where sentences mentioning two entities are grouped together in a bag. Zeng et al. (Zeng et al., 2015) assume that at least one sentence in the bag that mentions these two entities will express their relation, and only selects the most likely relation for each entity pair in training and prediction. They show that this setting brings significant improvements to CNN models. However, they also lose rich information contained in neglected sentences (Lin et al., 2016). To solve this, Lin et al. (Lin et al., 2016) introduce a selective attention mechanism. The feature of a sentence bag is calculated by a weighted sum of all sentences inside. The weight of a sentence is related to a query-based function which scores how well the input sentence and the predicted relation matches. Through this they make full use of all informative sentences, and outperform the piecewise CNN model. Sim-
ilarly, Jiang et al. (Jiang et al., 2016) relax the at-least-one assumption to “a relation holding between two entities can be either expressed explicitly or inferred implicitly from all sentences that mention these two entities”. Therefore, after automatically extracting features within each sentence using a convolutional architecture, they employ cross-sentence max-pooling to select features across different sentences, and then aggregate the most significant features into a vector representation for each entity pair. Since the resultant representation consists of features from different sentences, they successfully make full use of all available information contained in these sentences. To tackle the label overlapping problem, that more than one relations can hold between two entities, they propose a multi-instance multi-label setting, where each candidate relation type is assigned an independent probability with a sigmoid function. The loss functions are defined as

$$\text{Loss}_\sigma = -\sum_{i=1}^{l} y_i \log(p_i) + (1 - y_i)\log(1 - p_i)$$

and

$$\text{Loss}_{\text{square}} = \sum_{i=1}^{l} (y_i - p_i)^2,$$

where $l$ is the number of relation types, $p_i$ is the probability of the $i$-th relation holds, and $y_i \in \{0,1\}$ is the true value on that relation. Experiments show that both cross-sentence max-pooling and the multi-label setting contributes to improving performance.

Structural information is also important for RC-SF tasks. Various neural networks are adopted to model the shortest dependency paths and dependency subtrees (Socher et al., 2012), which have been proved effective. However, these models can hardly capture long-distance dependencies, and may suffer from irrelevant subsequences of clauses. One solution to this is through the shortest dependency path. Figure 3 shows the shortest dependency path representation for an example sentence from SemEval08. Xu et al. (Xu et al., 2015a) try to capture long-distance information based on the shortest path hypothesis: suppose $e_1$ and $e_2$ are two nominal entity mentions in a sentence, then (1) if $e_1$ and $e_2$ are arguments of the same predicate, then their shortest path should pass through that predicate; (2) if $e_1$ and $e_2$ belong to different predicate-argument structures, their shortest path will pass through a sequence of predicates, and any consecutive predicates will share a common argument. Note that, the order of the predicates on the path indicates the proper assignments of subjects and objects for that relation. For example, in Figure 3, the dependency path consecutively passes through carried and receives, which together implies that in the Instrument-Agent relation, the subject and object play a sender and receiver role, respectively. The shortest dependency path (i.e., the words, dependency edge directions, and dependency labels) are taken successively from the subject to the object as input. A convolution layer produces local features around each node on the dependency path, and combines these features into a global feature vector that are then fed to a softmax classifier for relation classification.

The ability of CNN to capture salient features from a sequence of objects makes it a powerful component when combined with other models, for instance, Recursive Neural Networks (Liu et al., 2015), and Recurrent Neural Networks (Cai et al., 2016).

### 5.2 Recurrent Neural Networks for RC-SF

To the best of our knowledge, Recurrent Neural Network models were fist used to tackle the RC-SF task in 2015, by Zhang et al. (Zhang et al., 2015) and Xu et al. (Xu et al., 2015a). They both use a LSTM model to encode the entire sentence, while incorporating dependency path as features to capture long-distance information. However, they didn’t outperform CNN models (Santos et al., 2015) at that time (Miwa and Bansal, 2016). In 2016, Miwa et al. (Miwa and Bansal, 2016) presented an end-to-end model to extract relations between entities. In their model, the word sequence is represented by a bidirectional LSTM, and the dependency parse tree is encoded by bidirectional tree-structure LSTMs.

In these models, the sentence is encoded as a whole. Xiao et al. (Xiao and Liu, 2016), however, divide into three context subsequences according to two target entities, which allows the model to encode each context subsequence independently so as to selectively focus as on the important context information. They propose a hierarchical model consisting of two recurrent neural networks: the first one learns context representations of the three context subsequences respectively, and the second one computes semantic composition of these three representations and produces a sentence representation for relation classification. The attention mechanism is adopted in both RNNs to encourage the model to concentrate on the important information when learning the sentence representations. The model shows comparable performance to the state-of-the-art on the SemEval-2010 Task 8 dataset without using any costly hand-crafted features. A significant improvement can be made by incorporating CNN for feature extraction (Cai et al., 2016).
Recursive Neural Networks for RC-SF

Word vector models alone are severely limited since they do not capture compositionality (Socher et al., 2012), the important quality of natural language that allows speakers to determine the meaning of a longer expression based on the meanings of its words and the rules used to combine them (Frege, 1994). These models fail when words function as operators that modify the meaning of another word: the meaning of “extremely strong” cannot be captured as the sum of word representations nor “extremely” and “strong”. This prevents them from gaining a deeper understanding of the semantics of longer phrases or sentences.

Recursive neural networks are good at modeling hierarchical structures (Liu et al., 2015), and are often used to handle structural information, like syntactic parse tree, to capture structural compositionality between words. In 2012, Socher et al. (Socher et al., 2012) proposed a Matrix-vector Recursive Neural Network (MV-RNN). Figure 5 shows the architecture of MV-RNN. The target sentence, or subsentence, is first parsed to a binary syntactic parse tree. Each word and phrase is represented by a vector and a matrix. The vector representation of a parent node $p$ can be calculated by the combination of its child nodes:

$$p = f_{A,B}(a, b) = f(Ba, Ab) = g(W [Ba, Ab])$$

where $a$ and $b$ are word vectors of $p$’s left child and right child, $A$ and $B$ are their matrices respectively. The element-wise function $g$ could be simply the identity function but they use instead a non-linearity such as the sigmoid or hyperbolic tangent $\tanh$, since such a non-linearity will allow it to approximate a wider range of functions beyond purely linear functions. MV-RNN takes as input a binary parse tree of a phrase or sentence and also compute matrices at each nonterminal parent node. The function $f$ can be readily used for phrase vectors since it is recursively compatible ($p$ has the same dimensionality as its children). For computing nonterminal phrase matrices, they define the function

$$P = f_M(A, B) = W_M [A, B]$$

where $P$ has the same size with $A$ and $B$. After two words form a constituent in the parse tree, this constituent can be merged with another one by applying the same functions $f$ and $f_M$. The model computes vectors and matrices in a bottom-up fashion, applying the functions $f$, $f_M$ to its own previous output (i.e. recursively) until it reaches the top node of the tree which represents the entire sentence. In the end, each node of a tree has associated with it a distributed vector representation. Then a softmax classifier is to predict a relation class distribution. Liu et al. (Liu et al., 2015) fur-
ther developed the recursive neural network model by incorporating a CNN layer to extract features of the shortest dependency path, and achieved significant performance improvements.

6 Conclusion
In this paper, we give an overview of existing methods and benchmarks of relation-classification slot filling. We categorize proposed methods on addressing the task of SL-SF over the last two decades into four main kinds, and plot the percentage of the research works in Figure 4. We also highlight representative publications for each category. We observe that recently, deep learning methods have been the most popular over other three kinds of models.

References

Nguyen Bach and Sameer Badaskar. 2007. A review of relation extraction. Literature review for Language and Statistics II.


Liyuan Liu, Xiang Ren, Qi Zhu, Shi Zhi, Huan Gui, Heng Ji, and Jiawei Han. 2017. Heterogeneous supervision for relation extraction: A representation learning approach. *arXiv preprint arXiv:1707.00166*.


Xiang Ren, Wenqi He, Meng Qu, Clare R Voss, Heng Ji, and Jiawei Han. 2016. Label noise reduction in entity typing by heterogeneous partial-label embedding. *arXiv preprint arXiv:1602.05307*.


