TruePIE: Discovering Reliable Patterns in Pattern-Based Information Extraction

Qi Li[1], Meng Jiang[2], Xikun Zhang[1], Meng Qu[1], Timothy Hanratty[3], Jing Gao[4], and Jiawei Han[1] *
[1]Dept of Computer Science, University of Illinois at Urbana-Champaign
[2]Dept of Computer Science and Engineering, University of Notre Dame
[3]US Army Research Laboratory
[4]Dept of Computer Science and Engineering, University at Buffalo
qili5@illinois.edu,mjiang2@nd.edu,xikunz2@illinois.edu,mengqu2@illinois.edu
timothy.p.hanratty.civ@mail.mil,jing@buffalo.edu,hanj@illinois.edu

ABSTRACT

Pattern-based methods have been successful in information extraction and NLP research. Previous approaches learn the quality of a textual pattern as relatedness to a certain task based on statistics of its individual content (e.g., length, frequency) and hundreds of carefully-annotated labels. However, patterns of good content-quality may generate heavily conflicting information due to the big gap between relatedness and correctness. Evaluating the correctness of information is critical in (entity, attribute, value)-tuple extraction. In this work, we propose a novel method, called TruePIE, that finds reliable patterns which can extract not only related but also correct information. TruePIE adopts the self-training framework and repeats the training-predicting-extracting process to gradually discover more and more reliable patterns. To better represent the textual patterns, pattern embeddings are formulated so that patterns with similar semantic meanings are embedded closely to each other. The embeddings jointly consider the local pattern information and the distributional information of the extractions. To conquer the challenge of lacking supervision on patterns' reliability, TruePIE can automatically generate high quality training patterns based on a couple of seed patterns by applying the arity-constraints to distinguish highly reliable patterns (i.e., positive patterns) and highly unreliable patterns (i.e., negative patterns). Experiments on a huge news dataset (over 25GB) demonstrate that the proposed TruePIE significantly outperforms baseline methods on each of the three tasks: reliable tuple extraction, reliable pattern extraction, and negative pattern extraction.

KEYWORDS

Information Extraction; Textual Patterns; Pattern Embedding; Pattern Reliability

*The first two authors contributed equally to this work and should be considered as joint first authors.

KDD 2018, August 19–23, 2018, London, United Kingdom
© 2018 Association for Computing Machinery.
ACM ISBN 978-1-4503-5552-0/18/08…$15.00
https://doi.org/10.1145/3219819.3220017

1 INTRODUCTION

Pattern-based methods have been popular in extracting structured information from text data for over two decades [1, 13]. Recently, with the achievement of high accuracy of entity typing systems, multiple pattern generation methods have been proposed [14, 21] to generate textual patterns with semantic types. These typed patterns such as "$Country president $Person" can help discover entity-attribute-value tuples beyond the predefined attribute schema of entities.

The existing pattern-based information extraction methods try to find high-quality patterns based on content-based criteria, such as frequency. However, the discovered "high-quality" patterns may still extract much incorrect information from the corpus. For example, consider the pattern "president $Person 's visit to $Country". It is likely to be considered as a high-quality pattern by the existing pattern-based information extraction methods for the task of finding country's president, since it appears frequently in the corpus, has a complete meaning, and contains the keyword "president". However, when considering its meanings, one can realize that this pattern is actually not proper for the targeted task. The extracted person *may not* be the president of the mentioned country in this pattern. In fact, this pattern always extracts incorrect information and should be excluded from the *president* extraction task. *Therefore, we propose to add another dimension to the pattern quality: pattern reliability, where we call a pattern is reliable if it is more likely to provide correct information.*

It is clear that when extracting information, one should rely more on the reliable patterns. It is especially important when there exist many patterns providing low-quality information: if those unreliable patterns can be detected and disregarded, then they will not introduce noise to the extracted information. However, it is a challenging problem to estimate the patterns' reliability degrees, since there is usually little supervision available. In practice, with the massive corpus, we cannot expect human to label every pattern's reliability and every piece of information's correctness. Therefore, it is critical to infer the patterns' reliability from the corpus without much human effort.

To achieve the goal, we develop a novel method called TruePIE (True-Pattern oriented Information Extraction). TruePIE tries to find both reliable information and patterns for the specific information extraction tasks. It adopts a **self-training** framework that can automatically generate training patterns, both positive and negative, and classify the massive candidate patterns.

There are two major challenges: 1) what features should be used to represent the patterns, and 2) how to automatically generate the training set? To conquer the first challenge, how to represent the patterns, we propose **pattern embeddings** as the features, where patterns with similar semantic meanings are embedded close to each other. The proposed pattern embeddings evaluate the pattern similarity from two aspects: the constructing words in a typed pattern and the extractions of the pattern. The idea is that if two patterns share similar words and/or their extractions share a similar relationship, then they are more likely to be similar. Such pattern embeddings consider both the local pattern information and the distributional information of the extractions, and consequently can nicely represent patterns.

To conquer the challenge of lacking supervision, we propose to automatically generate training patterns based on only a couple of seed patterns. Our basic principle is that the patterns which often extract correct information are more reliable, and the information extracted from the reliable patterns is more likely to be correct. However, this principle can only help us discover the reliable (i.e., positive) patterns for the training. Then how to detect the negative patterns under the open-world assumption? The existing pattern-based methods focus on discovering positive information and rarely consider the negative information or negative patterns. To tackle this challenge, we propose a novel approach which discovers the conflict information. Based on common sense and observations from the data, we find that the number of correct values of a specific entity is usually limited, and the number of entities that a specific value is associated with is also limited. For example, a country may have only a limited number of presidents in history and a president may serve in only one country. This information can be utilized to form constraints on the number of entities/values that a value/entity can be linked with, and we call such constraints "**arity-constraints**". The arity-constraints are effective in detecting incorrect information. For example, if we find that in the extractions one president is associated with two countries, then it is very likely that one of the countries is incorrect. Moreover, we define two types of arity-constraints, *hard* and *soft*, to allow flexibility under certain circumstances. We further propose an optimization problem to accurately estimate the reliability of the patterns and the correctness of the extractions.

Applying these ideas, TruePIE can start from a tiny amount of labeled information (our experiments show that only one seed pattern is needed in many cases) and gradually discover more and more reliable patterns and correct information. The proposed TruePIE method is tested on a massive corpus (over 25 GB in size) with 9.9 million documents and 4.0 billion words. The experiments show a significant improvement over the state-of-the-art information extraction methods in terms of finding both correct tuples and reliable patterns.

In summary, we make the following contributions in this paper:

- We identify the pitfall and challenge overlooked by existing pattern-based methods: the patterns' reliability. The reliable patterns are more likely to provide correct information for the specific information extraction tasks, and the negative patterns are the clues of likely wrong information.
- We formulate a pattern embedding approach where semantically similar patterns can be embedded close to each other. The pattern embeddings consider both the local pattern information as well as the distributional extraction information.
- We propose a novel approach using the arity-constraints to detect negative patterns under the open-world assumption. The automatically generated positive and negative patterns together can train a powerful classifier to identify more reliable patterns. The proposed TruePIE adopts a self-training framework and requires minimal human effort.

2 RELATED WORK

Given a text corpus, textual patterns leverage statistics (e.g., high frequency) by replacing words, phrases, or entities with symbols such as part-of-speech tags or entity types in order to extract a large collection of tuple-like information [27]. Hearst patterns like "*NP* such as *NP*, *NP*, and *NP*" were proposed to automatically acquire hyponymy relations from text data [13]. Later, machine learning experts designed the Snowball systems to propagate in plain text for numerous relational patterns [1, 7, 30]. Google's Biperpedia [11, 12] generated *E-A patterns* (e.g., "*A* of *E*" and "*E* 's *A*") from users' fact-seeking queries by replacing entity with "*E*" and noun-phrase attribute with "*A*". ReNoun [28] generated *S-A-O patterns* (e.g., "*S*'s *A* is *O*" and "*O*, *A* of *S*,") from human-annotated corpus on a pre-defined subset of the attribute names. Patty used parsing structures to generate relational patterns with semantic types [21]. The recent MetaPAD generated "meta patterns" based on content quality [14]. However, the above methods generated textual patterns based on either frequency or content quality. Our proposed TruePIE resolves conflicts for truth by evaluating *reliability* of patterns, which makes it distinctive among the pattern-based IE methods. There is also a study [24] trying to estimate the pattern reliability. However, this method only considers pattern extractions when calculating the reliability, and it cannot effectively detect negative information under the open world assumption. Our framework improves the reliability estimation by further considering the constructing words of patterns, and it can automatically generate training patterns, both positive and negative, based on several seed patterns.

Besides scoring textual patterns, there have been a few works on scoring tuples [15, 23, 25]. Bast et al. proposed to assign relevance scores for the tuples from type-like relations that are extracted from text or exported from Knowledge Bases [5, 6]. In our work, we evaluate the reliability of tuples based on the reliability of the sources (i.e., textual patterns) that provide them: If the patterns are reliable, their extractions are more likely to be correct; if the tuples are correct, the patterns are more likely to be reliable. Our studies show that these two processes can mutually enhance each other.

Open-domain IE systems can also extract facts by parsing individual sentences into (subject, relation/verb phrase, object)-tuples [4, 9, 10, 19, 26, 29]. Angeli et al. leveraged linguistic structure in every single sentence for IE and slot filling tasks [2, 3]. However, facts are often not expressed by whole sentences but segments of the sentences, which is why textual pattern-based IE often outperforms the Open IE systems on extracting values of specific attributes [14]. Therefore, the Open IE systems are not compared in our experiments.

3 PROBLEM DEFINITION

In this section, we describe the task and introduce some key concepts. Then we formally define our problem.

Definition 3.1 (Task definition). A target-attribute extraction task is an information extraction task for a specific attribute (e.g., president, capital). The task aims to **accurately** find all entities that have the target attribute and their corresponding attribute values. To reduce the ambiguity of the attribute, we assume that the entity type and the value type are also specified in the task. For example, a target-attribute extraction task can be set with the attribute goal as "president", where the entity type is \$LOCATION, and the value type is \$PERSON.

Definition 3.2 (EAV-Tuple). An EAV-tuple t is a piece of information extracted for the target-attribute extraction task, formating as $\langle e, a, v\rangle$, where e denotes the entity, a denotes the target attribute, and v denotes the attribute value.

Definition 3.3 (Pattern). A textual pattern is a sequential pattern of the tokens from the set of entity types (e.g., \$LOCATION, \$PERSON), data types (e.g., \$DIGIT, \$YEAR), phrases (e.g., "prime minister"), words (e.g., "president"), and punctuation marks. Here the *pattern* is a textual pattern of content quality, which means it has good frequency, concordance, completeness, informativeness, and coverage as defined in METAPAD [14]. We denote a pattern as p and the set of all the tuples it can extract as $\mathcal{T}_p$.

As discussed above, the pattern's quality assessment is usually defined on content-based criteria. However, in this quality definition, one important aspect is overlooked: the correctness of the information that it extracts. In real-world applications, there are many patterns that have high "quality" scores, but do not extract useful information for the given attribute. To address the shortcoming of the existing quality assessment for patterns, we further define the reliability of a pattern/tuple to reflect how likely the pattern/tuple is correct.

Definition 3.4 (Pattern Reliability). The pattern p's reliability is defined as how likely its extracted EAV-tuples $\mathcal{T}_p$ are correct. We use a score ρ_p to measure p's reliability. The higher the more reliable. Further, we call a pattern *positive* if it is reliable on attribute a, and a pattern *negative* if it is unreliable, i.e., it usually extracts wrong information for attribute a. Note here that, if a pattern does not contain correct entity or value types for the target-attribute extraction task, we consider it as irrelevant with attribute a, and do not assess its reliability on attribute a.

Definition 3.5 (Tuple Reliability). Similarly, we define the tuple t's reliability as how likely t is correct. We use a score τ_t to measure t's reliability.

With the aforementioned concepts, we formally define our problem in the target-attribute extraction task.

Problem. Given the text corpus and a specific target-attribute extraction task a, our goal is to generate reliable patterns $\mathcal{P}$, so that they can extract as many as possible reliable EAV-tuples.

To solve this problem in practice, the designed algorithm should also consider the human effort involved. With the massive text corpus, it is unrealistic to rely on the human to annotate the correctness for many tuples or patterns. Therefore, the designed algorithm should expect limited annotations to handle the massive text corpus.

Table 1 summarizes the frequently used notations in this paper; some will be introduced in the next section.

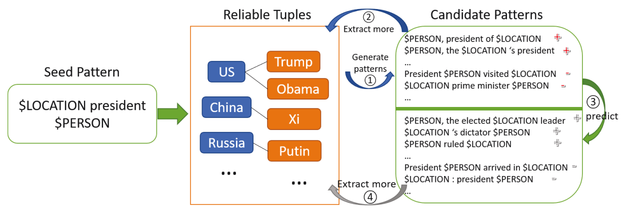
Table 1: Notations we use throughout this paper.

Symbol	Definition
e, a, v	An entity, an attribute, and a value, respectively
p	a pattern that contains a pair of typed entities
t	An EAV-tuple, in the format of $\langle e, a, v\rangle$
$vec(\cdot)$	embedding function.
$\mathcal{T}_p$	The tuples extracted by pattern p
ρ_p	The reliability score of pattern p
τ_t	The reliability score of tuple t
$\mathcal{P}$	The positive pattern set
$\mathcal{T}$	The reliable tuple set
$\bar{\mathcal{P}}$	The negative pattern set

4 THE TRUEPIE METHOD

In this section, we formally present the proposed algorithm TRUEPIE. Reliable patterns, i.e., the patterns that are more likely to extract correct information, are the key in the target-attribute extraction tasks. However, there is hardly any prior knowledge on which pattern is reliable and which tuple is correct. Therefore, the reliability of patterns and the tuples needs to be learned from the data with little human guidance. To achieve the goal, the proposed TRUEPIE adopts a self-training framework and gradually identifies more and more reliable tuples and positive patterns (and negative patterns).

4.1 TRUEPIE Overview



Seed Pattern
$LOCATION president $PERSON
Reliable Tuples
US
Trump
Obama
China
Xi
Russia
Putin
...
...
② Extract more
Generate patterns ①
Candidate Patterns
$PERSON, president of $LOCATION +
$PERSON, the $LOCATION 's president +
...
President $PERSON visited $LOCATION -
$LOCATION prime minister $PERSON -
...
③ predict
$PERSON, the elected $LOCATION leader +
$LOCATION 's dictator $PERSON +
$PERSON ruled $LOCATION +
...
President $PERSON arrived in $LOCATION -
$LOCATION : president $PERSON -
...
Extract more ④

Figure 1: The self-training framework of TRUEPIE

The basic idea of TRUEPIE is that reliable patterns for a target-attribute extraction task should convey similar semantic meanings. Therefore, we propose a pattern embedding approach so that patterns with similar semantic meanings are close to each other in the embedded space. However, without supervision, it is hard to judge how close is close enough. Therefore, to overcome the lack of supervision, TRUEPIE is a self-training model that can automatically generate training patterns from the data and the design runs with little human guidance.

The flow of the proposed TRUEPIE is shown in Figure 1. With the seed pattern, TRUEPIE first learns the arity-constraints (Section 4.4.1) to help identify the highly reliable (i.e., positive) and highly unreliable (i.e., negative) patterns (Section 4.4.2). With the identified positive and negative patterns, a classifier can be trained with the pattern embeddings (Section 4.3) as features and the pattern candidates can be classified. Combining the highly reliable predicted positive patterns with the previously found positive patterns, more positive and negative information can be extracted (Section 4.5). TRUEPIE will repeat this training-predicting-extracting process to discover more and more reliable patterns and information.

4.2 Pattern Candidate Generation

The pattern candidates used in TRUEPIE can be generated by any kind of pattern-based information extraction methods, for example, METAPAD [14], which uses a context-aware phrasal segmentation approach to generate pattern candidates of high content quality. Regardless of the pattern candidate generation methods, the same basic ideas apply: information extracted from the reliable patterns is more likely to be correct and the patterns which often extract correct information are more reliable. For example, in the large set of *S*-*A*-*O* patterns used in RENOUN [28], "*A* of *S* is *O*" is often more reliable than "*S A O*", and the pattern "*S*'s *A* and *O*" may even be a negative pattern; here *S* is the subject/entity, *A* is the attribute name, and *O* is the object/attribute value. From another perspective, TRUEPIE can be viewed as an enhancer for any pattern-based IE systems, which can better group synonymous patterns and improve the accuracy of the extracted information.

4.3 Pattern Embedding

Word embedding techniques have been widely used as features for NLP and machine learning tasks, thanks to its properties such as providing a distributional representation for words and preserving their semantic and syntactic relationships [17]. For example, similar words can have similar embeddings and the semantic relationships can be captured through algebraic operations (e.g., vec('Paris') − vec('France') + vec('Italy') ≈ vec('Rome')). These properties are also helpful to formulate pattern embeddings: good pattern embeddings should also preserve their semantic relationships. Therefore, we propose to learn a distributional representation for patterns using the word embedding techniques (word2vec) in [17, 18].

The key idea of the proposed pattern embedding is to map patterns with similar semantic meanings close to each other. The reason is that reliable patterns are likely to share similar meanings. When evaluating the similarity between patterns, there are two important perspectives: the words constructing the pattern and the pattern's extractions. For example, pattern "$LOCATION president $PERSON" has the constructing word "president" and has extractions such as ⟨USA, President, Trump⟩.

Suppose p has constructing words $(W_1^p, \cdots, W_m^p)$ and extractions $\{(e_i^p, v_i^p)\}_{i=1}^n$. Applying the above idea, we formulate pattern p's embedding, denoted as $vec(p)$, as follows.

$$vec(p) = (vec^p(w), vec^p(r)), \tag{1}$$

where

$$vec^p(w) = \frac{1}{m}\sum_{i=1}^{m} vec(W_i^p)$$
$$vec^p(r) = \frac{1}{n}\sum_{i=1}^{n} vec(e_i^p) - vec(v_i^p). \tag{2}$$

In the above definition, pattern embedding is a concatenation of two parts, which correspond to the aforementioned two perspectives: The first part is the mean of the constructing words' embedding, and the second part reflects the relationship between the extracted pairs. Using this pattern embedding, patterns with similar constructing words and similar relationships between the extracted pairs will have similar embeddings.

4.4 Generation of Training Patterns

With the proposed pattern embeddings, one straightforward approach to find reliable patterns is to conduct clustering or ranking based on the seed patterns, as its neighboring patterns may have similar semantic meanings and are likely to be reliable. However, such unsupervised methods have a strong shortcoming: it is hard to determine a proper threshold to justify how close is close enough for each target attribute. To overcome this problem, we propose to generate the training patterns with only a couple of seed patterns so that a classifier can be then used to distinguish positive and negative patterns.

4.4.1 Arity-Constraint. It is relatively easy to find positive patterns: if a pattern repeats significantly in the corpus and its extractions overlap significantly with the extractions from seed pattern set $\mathcal{P}_0$, it is likely to be reliable. However, having the positive patterns only is not enough for the training. Spotting the negative patterns/tuples plays a key role in completing the training set. Though important it may be, negative patterns/tuples are hard to detect as the close-world assumption is not valid under the real-life settings. In this section, we propose a novel method to answer this question: how to find the negative patterns/tuples?

To detect the negative tuples, we observe the following fact: Given a target attribute, the number of correct values of a specific entity may be limited, and the number of correct entities of a specific value may also be limited. For example, a country may have only a limited number of presidents in history, and a president may only serve in one country. If such observations can be modeled and applied, the false tuples can be effectively detected. To do so, we form the tuples into a bipartite graph of entity nodes and value nodes, where an edge between $\langle e, v\rangle$ indicates that $\langle e, a, v\rangle$ is extracted, then this observation can be formally defined as the arity-constraints on entities and values.

Definition 4.1 (Arity-Constraint). The arity-constraint for attribute a is equivalent to setting constraints on the degree of entities C_e^a (number of values an entity can associate with) and degree of values C_v^a (number of entities a value can associate with).

Given the arity-constraint, and suppose there is a reliable EAV-tuple set $\mathcal{T}$, then we can evaluate the reliability of the tuples more accurately. We call a tuple t is *positive* if $t \in \mathcal{T}$; a tuple t is *negative* if $t \notin \mathcal{T}$ and adding t to $\mathcal{T}$ can cause violation of C_e^a or C_v^a (we also call t a *conflict* of $\mathcal{T}$); a tuple t is *undecidable*, if t is neither positive nor negative.

The arity-constraint can be set by human for a given target-attribute extraction task or can be learned from the data automatically. Either way, the arity-constraints should fit for "average" entities and values. To learn the arity-constraints from the data, due to the ubiquitous long-tail phenomenon in word distributions, median may be a good estimation of "average". Therefore, we suggest the arity-constraints as follows.

$$C_e^a : deg(e) \leq median(f_e); \tag{3}$$
$$C_v^a : deg(v) \leq median(f_v), \tag{4}$$

where f_e and f_v represent the empirical distributions of the degree of entities and values, respectively.

However, there may exist exceptions that should be considered. If the arity-constraint is too tight, it may cause high false negative rate when estimating tuples' reliability. Therefore, we further refine the definition of arity-constraint by differentiating between hard constraint and soft constraint. For the former, no violation is allowed, while for the latter, a violation is allowed if there is enough evidence that a tuple is positive. Mathematically, we define a hard constraint if $median(f_e) = Q_{F_e}(1-\alpha)$. If $median(f_e) < Q_{F_e}(1-\alpha)$, then the arity-constraint is a soft constraint. $Q_F(\cdot)$ is the quantile function for distribution function $F(x) = \Pr(X \leq x)$, defined as $Q_F(p) = inf\{x \in \mathbb{R} : F(x) \geq p\}$, and α is the significant level. Note that the $Q_F(\cdot)$ is non-decreasing and median is equivalent to $Q_F(0.5)$, so $median(f_e) \leq Q_{F_e}(1-\alpha)$. In our experiment, α is set to 0.1.

4.4.2 Pattern Reliability and Tuple Reliability Estimation. With the detected negative tuples, the estimation of the reliabilities of patterns and tuples can be further improved. Moreover, the pattern reliability and tuple reliability are closely related: the patterns which often extract correct information are more reliable, and the information extracted from the reliable patterns is more likely to be correct. Therefore, we propose a unified model to estimate pattern and tuple reliability together.

We first define pattern and tuple reliability. Given the arity-constraints and the reliable EAV-tuple set $\mathcal{T}$, we calculate the pattern reliability ρ_p as:

$$\rho_p = \frac{N_+ + \frac{1}{2}N_U}{N_+ + N_- + N_U}, \tag{5}$$

where N_+, N_-, and N_U denote the number of positive tuples, negative tuples, and undecidable tuples p extracts, respectively. This formulation is a natural extension of precision, where the undecidable tuples are given partial credits. Compared with precision, it is more suitable to the open-world assumption.

This pattern reliability score has the ability to distinguish patterns with different reliability characteristics. For a pattern p, if ρ_p is close to 1 (e.g., greater than a threshold θ: $\rho_p > 0.8$), it means that p is highly reliable and always provides correct information. Therefore, it should be considered as a positive pattern. On the other hand, if ρ_p is close to 0 (e.g., $\rho_p < 0.2$), it means that p always provides information that conflicts with the reliable EAV-tuple set, so p should be a negative pattern. If p is irrelevant with the target-attribute extraction task, then very likely, it will provide many undecidable tuples, and thus ρ_p would be close to 0.5.

With the reliable pattern set $\mathcal{P}$ and the arity-constraints C_e^a, C_v^a, the extracted tuples can form a bipartite graph of entity and value nodes, where an edge between $\langle e, v\rangle$ indicates that $\langle e, a, v\rangle$ is extracted. The reliability score of the tuple is then represented as the edge weight, which is defined as:

$$\tau_t = \sum_{p:p\in\mathcal{P}} \rho_p \times n_t^p - b, \tag{6}$$

where ρ_p is the reliability score of pattern p, n_t^p is the count of t extracted by p, and b is a small positive parameter to reduce the randomness in extractions. Here, we only consider tuples from the reliable patterns since unreliable patterns have high noise and may have harmful influence to the reliability estimation of the tuples.

Since the tuples with higher weights are more likely to be correct, the problem is then equivalent to forming the bipartite graph with the maximum sum of edge weights subject to the arity-constraints. Mathematically, it is an optimization problem to find the best assignment such that:

$$\begin{aligned} &\max \sum_t \left(\tau_t - \max(\beta_1 \mathbb{1}(\neg C_e^a), \beta_2 \mathbb{1}(\neg C_v^a)\right) \\ &s.t. \rho_p \geqslant \theta, \forall p \in \mathcal{P}, \mathcal{P}_0 \subseteq \mathcal{P}, \end{aligned} \tag{7}$$

where $\mathbb{1}(\cdot)$ is the indicator function, $\mathcal{P}_0$ refers to the seed pattern set, and β_i is a positive penalty parameter if the corresponding arity-constraint is soft and β_i is ∞ for hard constraint. For soft constraints, if an edge causes violation but its weight τ_t is large enough, then this edge should still be kept. However, for hard constraint, no violation can be kept.

This optimization problem jointly models the reliability of the tuples and patterns. If τ_t's are given, this problem can be reduced to a network flow problem. Since the correct information usually appears more frequently than the incorrect information in the reliable pattern's extractions, therefore, we propose a greedy method to speed up the optimization process. Algorithm 1 shows the generation of the positive/negative pattern set and the reliable EAV-tuple set.

4.5 The Self-Training Framework

Now with the generated training patterns, we can then train a classifier to predict the reliability of the candidate patterns who are neither positive nor negative, where the features are the patten embeddings. Considering the characteristics of the proposed pattern embedding, in our experiments, we use K-nearest-neighbors approach to make predictions, with cosine distance and the inverse distance as the weight. The candidate patterns which are closer to the positive patterns will be predicted as positive. To avoid over fitting, the negative patterns that are close (cosine similarity > 0.9) to any positive patterns will be removed from the training set. In practice, there are more negative patterns than the positive patterns, so the value of K should be chosen proportional to the number of positive patterns to ensure the effectiveness of the classification results.

Since the automatically generated training set has high standard on their reliability scores, the size may limit its power to discover

Algorithm 1 Algorithm of Generating Training Patterns

Input: The corpus, seed pattern set $\mathcal{P}_{\prime}$, arity-constraints, and parameters β_1, β_2, and θ.
Output: Reliable EAV-tuple set $\mathcal{T}$, positive pattern set $\mathcal{P}$, and negative pattern set $\bar{\mathcal{P}}$.

```
1:  Initialization: $T = \emptyset$, $\mathcal{P} = \mathcal{P}_{\prime}$, and $\bar{\mathcal{P}} = \emptyset$. For
    $p \in \mathcal{P}$, $\rho_p = \max(\beta_1, \beta_2)/2$, $T = \mathcal{T}_{\mathcal{P}}$
2:  if Arity-constraints are not given then
3:      Learn $C_e^a$ and $C_v^a$ from $\mathcal{T}_{\mathcal{P}}$;
4:      Set $\beta_i = \infty$ if corresponding arity-constraint is hard;
5:  end if
6:  repeat
7:      Calculate the reliability score of the tuples using Eq.(6)
8:      Sort $t$ with $\tau_t$ in decreasing order
9:      for each $t$ with $\tau_t > 0$ do
10:         if adding $t$ to $\mathcal{T}$ satisfy the arity-constraints then
11:             Add $t$ to $\mathcal{T}$;
12:         else if $\tau_t > \max(\beta_1, \beta_2)$ then
13:             Add $t$ to $\mathcal{T}$;
14:         end if
15:     end for
16:     Generate a new set of pattern candidates using MetaPAD;
17:     for each candidate pattern $p$ do
18:         Calculate $\rho_p$ based on Eq.(5)
19:         if $\rho_p > \theta$ then
20:             $\mathcal{P} = \mathcal{P} \cup p$;
21:         else if $\rho_p < 1 - \theta$ then
22:             $\bar{\mathcal{P}} = \bar{\mathcal{P}} \cup p$;
23:         end if
24:     end for
25: until $\mathcal{P}$ is stable
```

reliable patterns. In this case, a self-training framework can be adopted, where the "good" predicted positive patterns will be added into the training patterns.

Since the predictions may contain errors, further estimation of their reliability is necessary to ensure the quality of the training set. We first combine the extractions of the predicted positive patterns with the existing reliable pattern set and update the reliable tuple set. In this step, since the reliabilities of the newly added positive patterns are unknown, we use their prediction probabilities instead: $\tau_t^{new} = \tau_t^{old} + \sum_{t \in \mathcal{T}_p} \text{Prob}(p)$. Now, with the updated reliable tuple set, the reliability of the candidate patterns can be calculated. Note here, since the reliable tuple set already contains the extractions from the predicted positive patterns, their ρ_p's should be calculated under the close-world assumption (i.e., the vanilla precision). Finally, the patterns with high reliability scores will be added into the reliable pattern set.

In summary, the proposed self-training TruePIE framework will gradually enlarge the set of reliable EAV-tuples and the set of positive (reliable) patterns through repeating the following training-predicting-extracting steps until the stopping criterion is met.

Step 1: Given the reliable tuple set, generate training patterns;
Step 2: With the training patterns, classify the candidate patterns, where the features are the pattern embeddings;
Step 3: Combine the extractions of the positive patterns and update the reliable tuple set. Repeat from Step 1.

4.6 Time Complexity of TruePIE

The proposed TruePIE method runs in an iterative manner, discovering more and more positive patterns and tuples in each iteration. In practice, we find that TruePIE can find sufficient positive patterns within 2 to 3 iterations. In each iteration, if the arity-constraint is not given, then it needs $O(|\mathcal{T}_{\mathcal{P}}|)$ to learn. The construction of the reliable tuple set needs $O(|\mathcal{T}_{\mathcal{P}}| log(|\mathcal{T}_{\mathcal{P}}|))$. The pattern generation step can be done by calling the module in MetaPAD [14] once to generate all pattern candidates, which takes $O(|C|)$, where $|C|$ is the corpus size. Then for each pattern, we estimate the correctness of all tuples it extracted and calculate its reliability scores, so totally this step can be done in $O(\sum_p |\mathcal{T}_p|)$. When conducting the classification step, the time complexity for KNN is linear in the size of the training set $O(|\bar{\mathcal{P}}| + |\mathcal{P}|)$. Overall, besides the pattern generation step, TruePIE runs in $O(\sum_p |\mathcal{T}_p| + |\mathcal{T}_{\mathcal{P}}| log(|\mathcal{T}_{\mathcal{P}}| + |\bar{\mathcal{P}}| + |\mathcal{P}|)$. However, since the number of positive tuples and number of patterns are usually much less than the total number of tuples, in practice, TruePIE runs in $O(\sum_p |\mathcal{T}_p|)$.

5 EXPERIMENTS

In this section, we first introduce a huge text corpus (as our dataset) and the competitive methods. Then we report the experimental results on multiple tasks including a) reliable tuple extraction, b) reliable pattern generation, c) case studies, and d) error analysis.

5.1 Experimental Setup

We adopt a huge set of language resources from *English Gigaword Fourth Edition* LDC2009T13 [22]. There are news articles from different news sources spanning from mid-1990s to 2010. The six distinct international sources of English newswire are Agence France-Presse, Associated Press Worldstream, Central News Agency of Taiwan, Los Angeles Times/Washington Post, New York Times, and Xinhua News Agency. The total size of the text corpus is 26,348 MB (25.7 GB) including 9.9 million documents and 4.0 billion words. The named entities are recognized and typed using Stanford NER tool [16].

We compare the proposed approach with the following baseline methods that represent state-of-the-art pattern-based information extraction methods. As discussed in Section 2, the open IE systems are not compared.

- Patty [21] relies on the Stanford dependency parser [8] and formulates the textual patterns with semantic types if the parsing path between entity and value is short and the patterns appear frequently. Then the pattern taxonomy is constructed based on patterns' extractions.
- MetaPAD [14] assesses the quality of textual patterns according to content-based criteria such as frequency, concordance, completeness, and informativeness. It adopts context-aware phrasal segmentation to generate patterns of good content quality and groups synonymous patterns by high agreement on trigger words (e.g., "president") or extractions. This method is adopted by TruePIE to generate the candidate patterns.
- REPEL [24] is a co-training method which also estimates pattern reliabilities. When calculating the pattern reliability, it only considers the pattern extractions, and the constructing words are ignored.

In our experiments, PATTY and METAPAD are compared for the tuple extraction task, and REPEL is compared for the reliable pattern generation task.

5.2 Results on EAV-Tuple Extraction

In this section, we evaluate the extracted EAV-tuples from different methods. We focus on four tasks, namely, the Leader of a country (i.e., ⟨$LOCATION, leader, $PERSON⟩), the President of a country (i.e., ⟨$LOCATION, president, $PERSON⟩), the Capital of a location including country, state and province (i.e., ⟨$LOCATION, capital, $LOCATION⟩), and the Director of an organization (i.e., ⟨$ORGANIZATION, director, $PERSON⟩).

The seed patterns are "$LOCATION leader $PERSON", "$LOCATION president $PERSON", "$LOCATION capital $LOCATION" and "$LOCATION , the capital of $LOCATION", and "$ORGANIZATION director $PERSON", respectively. For the first three tasks, the arity-constraints are learned from the data, which state that one location can have one or more leader/president but only one capital, and one person/city can be the leader/president/capital for one location. For the Director task, the arity-constraint is given as "1-soft:1-soft", meaning that a person can be director for one or more organizations and one organization can have one or more directors. The $k = 15$ for the kNN classification since there are many positive patterns.

We conduct quantitative evaluation in terms of *precision* and *coverage rate*. Precision is defined as the percentage of the extracted EAV tuples that are correct. To evaluate precision, we randomly sample 50 extracted tuples from each method and label their correctness. To insure the quality of the evaluation, the labeling is conducted by four hired students (one for each task) manually by looking up the information from Google search. We repeat this random sampling for 10 times and report the average precisions with standard deviations. Since all methods can provide reliability scores for tuples, we also compare the precisions on the top tuples. Another important aspect of the extracted tuple quality is the completeness (recall). However, because of the corpus size, it is unrealistic to get a complete list of all correct tuples. Therefore, we use coverage rate to evaluate how complete the extractions are. To evaluate the coverage rate, we first sample 100 correct tuples extracted from each method to form a ground truth tuple set, and then combine them to examine how many of these 300 tuples are covered by each method. We report the percentage. For both precision and coverage rate, the values are the higher the better.

Table 2 summarizes the comparison results on the extracted tuples. It is clear that the proposed TRUEPIE method achieves significant improvement in precision. Since in the TRUEPIE method, only the reliable patterns are used in information extraction, TRUEPIE is less prone to the noise. The precision on the tuples with high reliability scores even achieves 100% or almost 100% accuracy for many tasks. The coverage rate of the TRUEPIE method is also competitive with the baseline methods. Even though for some tasks, the baseline methods achieve higher coverage rate than TRUEPIE, they may need to extract much more tuples to achieve the improvement (e.g., for the President task, PATTY extracts 423% times more tuples to gain 22% improvement, for the Capital task, METAPAD extracts 921% times more tuples to gain 28% improvement, and for the Director task, METAPAD extracts 173% times more tuples to gain 20% improvement in the coverage rate). Moreover, for the Leader extraction task, which involves many sub-relations and thus more diverse expressions on reliable patterns, TRUEPIE achieves the highest precision and coverage rate. The discovered reliable patterns by TRUEPIE contain key words such as president, prime minister, chancellor, dictator, and ruler, in addition to "leader". Yet for this task, only one seed pattern is used by TRUEPIE.

For the baselines, PATTY and METAPAD perform similarly. For both methods, the key factors to assess the pattern reliability are the frequency and the trigger words of the patterns. Though these two factors are important, as shown in the results that the top extractions have higher precisions, they are not accurate and sufficient. For example, pattern such as "president $PERSON in $LOCATION" is a frequent pattern and contains trigger word "president". However, this pattern is not reliable in the president extraction task. When many of these kind of patterns are used to extract information, the results will unavoidably suffer from high noise.

5.3 Results on Reliable Pattern Extraction

In this section, we focus on the following three tasks to evaluate the generated reliable patterns: the Spouse of a person (i.e., ⟨$PERSON, spouse, $PERSON⟩), the Parent of a person (i.e., ⟨$PERSON, parent, $PERSON⟩), and the Death Year of a person (i.e., ⟨$PERSON, year of death, $YEAR⟩). These three tasks are much sparser in the given news corpus and has many diverse expressions.

For the Spouse extraction task, the seed patterns are "$PERSON and his/her wife/husband , $PERSON ,", "$PERSON ' s wife/husband , $PERSON ,", and "$PERSON married $PERSON". For Parent extraction tasks, the seed patterns are "$PERSON ' s father/mother/parents , $PERSON ,". For the Death year extraction task, the seed patterns are "$PERSON died in $YEAR" and "$PERSON ' s death in $YEAR". The seed patterns are also used in the baseline method REPEL. The arity-constraints are set as "1-soft:1-soft", "2-soft:2-soft", and "1-soft:20-soft" respectively. They are set according to common sense for average cases. For example, one person may have one spouse, but should have exceptions as he/she may have a divorce. The $k = 3$ for the kNN classification since the positive patterns are relatively sparse for these tasks in the dataset. Since REPEL returns a ranked list of generated patterns on their reliability scores, we compare its top 100 reliable patterns in the experiment.

To evaluate the discovered reliable patterns, we ask a human labeler to read the patterns and provide her judgment on the pattern correctness for each task. We compare the True Positive Rate for the discovered reliable patterns. To evaluate the discovered unreliable patterns, we report the True Negative Rate. For both measurements, the values are the higher the better.

Table 3 summarizes the details of some of the labeling results. Overall, the discovered reliable patterns by TRUEPIE enjoy high accuracy, with true positive rate higher than 90%. Moreover, the true positive rates on the training patterns are even higher, with 99%, 94% and 94% for the three tasks. Those automatically generated training patterns ensure the good results of the self-training TRUEPIE. From limited seed patterns, TRUEPIE detects "widower", "divorce", "bride", "couple", etc, for Spouse task; "son", "daughter", etc, for Parent task;

Table 2: Comparison of the extracted EAV-tuples. Precisions are calculated based on 10 sets of 50 random samples from the extracted tuples of each method. K=400 for Capital and K=1000 for other tasks. Coverage rates are calculated based on 300 true tuples where each method provides 100 unique tuples.

	Task	Patty	MetaPAD	TruePIE	Task	Patty	MetaPAD	TruePIE
#Extracted Tuples		2752	4067	2317		7801	4917	1490
Average Precision		0.59 ± 0.05	0.43 ± 0.07	**0.87** ± 0.05		0.38 ± 0.08	0.30 ± 0.06	**0.89** ± 0.05
Top 10% Precision	Leader	0.89 ± 0.17	0.66 ± 0.30	**0.99** ± 0.03	President	0.59 ± 0.29	0.42 ± 0.15	**1** ± 0
Top K Precision		0.67 ± 0.12	0.56 ± 0.10	**0.99** ± 0.01		0.56 ± 0.27	0.33 ± 0.07	**0.95** ± 0.04
Coverage Rate		0.56	0.59	**0.61**		**0.87**	0.63	0.71
#Extracted Tuples		1316	4371	428		10313	14234	5205
Average Precision		0.37 ± 0.07	0.27 ± 0.10	**0.97** ± 0.02		0.54 ± 0.08	0.56 ± 0.07	**0.86** ± 0.05
Top 10% Precision	Capital	0.54 ± 0.25	0.47 ± 0.16	**1** ± 0	Director	0.63 ± 0.31	0.65 ± 0.20	**0.93** ± 0.12
Top K Precision		0.51 ± 0.18	0.47 ± 0.16	**0.98** ± 0.02		0.63 ± 0.32	0.67 ± 0.31	**0.89** ± 0.10
Coverage Rate		0.67	**0.92**	0.68		0.52	**0.6**	0.50

Table 3: Comparison of the discovered reliable patterns. "Positive in Training" refers to the positive patterns generated for the initial training set; "All Reliable" refers to all the discovered reliable patterns; "Unreliable Positive" refers to the discovered unreliable patterns but with positive predictions from the classifier. TPR/TNR refers to the True Positive or Negative Rate.

		REPEL	TruePIE		
			Positive in Training	All Reliable	Unreliable Positive
	# Patterns	100	429	1223	97
Spouse	TPR/TNR	0.94	0.99	0.91	0.54
	# Patterns	100	261	321	17
Parent	TPR/TNR	0.45	0.94	0.92	0.53
	# Patterns	100	54	235	37
Death Year	TPR/TNR	0.79	0.94	0.93	0.51

and "assassinated", "killed", "suicide", etc, for Death Year task. We further examine how much those false positive patterns can affect the results and find that most of them are infrequent patterns with 10 or less extracted tuples.

The baseline method REPEL performs well on Spouse task but not satisfying on Parent and Death Year tasks. On the Parent task, REPEL discovers "grandmother", "sister", "nephew", "wife" and other family relationships and cannot distinguish their difference with "parent". On the Death Year task, most wrong patterns are about a person's family's death year, such as "$PERSON 's mother died in $YEAR". Such mistakes may be caused by the sparsity of the named entities in the corpus, which leads to noisy entity embeddings. Compared with REPEL, which only considers the pattern extractions' embeddings, TruePIE further takes into consideration the pattern constructing words' embeddings and detects negative patterns. Thus, it is more robust against the embedding noise.

When evaluating the unreliable patterns, we randomly sample 200 patterns and the true negative rate is nearly 1 for all three tasks. Note that this true negative rate may be not precise for all the detected unreliable patterns since there are much more unreliable patterns than reliable patterns. Therefore, we further examine a special set of unreliable patterns (Unreliable Positive in Table 3). These patterns are considered unreliable based on their estimated reliability scores, but they are labeled positive by the classification algorithm. The result shows quite high true negative rate (above 50%), which strongly suggests that the extracted tuples are of high quality and demonstrates the robustness of the TruePIE method.

5.4 Case Study on Reliable/Unreliable Patterns

We also run TruePIE method on four additional tasks, namely Vice President, Governor, Mayor, and Prime Minister. Due to space limit, we do not discuss their results in details. However, from the ten tasks, We find that for some tasks, the trigger words are more important than for others. For example, in Vice President, Governor, and Mayor tasks, the reliable patterns usually contain these trigger words, but in tasks such as Death Year, Parent, Spouse, and Leader, the expressions are more diverse.

Table 4 shows some interesting examples that TruePIE method detects. Items in bold refer to the entity and items in italic refer to the value. We provide our insights and analysis as follows.

First, the trigger words are not always reliable. The semantic meanings can still differ a lot with the same trigger words. Not relying on the trigger words may give TruePIE a big advantage in discovering reliable patterns. Second, TruePIE discovers reliable patterns that may explicitly or implicitly indicate similar attributes. For example, reliable patterns may contain the hyponym of the

Table 4: Examples of positive patterns and negative patterns ("$Bold" denotes the entity and "*$Italian*" denotes the value).

Task	Positive Patterns	Negative Patterns
Leader	**$Location** president ***$Person*** **$Location** prime minister ***$Person*** **$Location** military ruler ***$Person*** **$Location** 's chancellor , ***$Person*** ,	**$Location** leader told ***$Person*** **$Location** scoring leader ***$Person*** ***$Person*** , son of the **$Location** leader **$Location** 's cricket chief , ***$Person***
Governor	***$Person*** , the **$Location** administrator	**$Location** senator $Person
Capital	**$Location** 's central government in ***$Location*** president sworn in ***$Location*** , **$Location**	**$Location** leader $Person will visit ***$Location*** embassy of **$Location** in ***$Location***
Spouse	***$Person*** 's widower **$Person** $Location president ***$Person*** and first lady **$Person** wedding of prince ***$Person*** and princess **$Person**	***$Person*** 's lover **$Person** , ***$Person*** 's affair with **$Person** ***$Person*** 's girlfriend , **$Person** ,
Parent	**$Person** 's son ***$Person*** **$Person** to his daughter ***$Person***	***$Person*** 's brother , **$Person** , ***$Person*** 's husband **$Person**
Death Year	king **$Person** ($Year - ***$Year***) **$Person** 's ***$Year*** suicide **$Person** 's ***$Year*** funeral killed **$Person** in ***$Year***	**$Person** 's trial in ***$Year*** **$Person** fired him in ***$Year*** **$Person** 's husband died in ***$Year*** **$Person** left in ***$Year***

trigger words, such as "president" and "prime minister" for "leader"; and the two examples for Capital tasks show that the attribute is implicitly referred. Third, the arity-constraints can play an important role in detecting negative patterns. For example, "***$Person*** 's affair with **$Person**" is considered an unreliable pattern because its extractions conflict with many reliable tuples. This pattern, however, is considered reliable for Spouse and IsMarriedTo relationships by Patty [20]. TruePIE successfully distinguishes the subtle difference and finds that the two persons extracted from this pattern are not married.

5.5 Error Analysis and Future Work

To provide further insights of the proposed TruePIE, we also examine the reason of the errors and provide some guidelines for using TruePIE.

One of the main reasons causing errors in TruePIE is that the embeddings are not distinguishable enough to classify positive and negative patterns, especially for the patterns with sparse or ambiguous named entities, and the low-coverage low-frequency patterns. As discussed in Section 5.3, for the Parent task, REPEL generates many patterns with other family relationships. Adding the pattern constructing word embeddings significantly helps TruePIE to find reliable patterns, but still a small amount of other family relationships are generated. This also explains why there are unreliable patterns labeled positive in the classification step. To overcome this issue, we plan to investigate more sophisticated pattern embedding methods.

Another reason of errors comes from the named entity recognizer. The errors of the NER results can propagate to the pattern generation step and then affect the accuracy of the extractions. For example, in the Leader task, most of the errors are caused by the incorrect typing: the NER tool types some company names as locations. We also find that our candidate pattern generator, MetaPAD, can achieve better results if a reliable fine-grained typing system can be applied. Such fine-grained typing system can also help TruePIE to find more reliable patterns by reducing the ambiguity of the patterns. For example, "**$Country** senator $Person" and "**$State** senator $Person" may derive different arity-constraints.

To avoid the aforementioned errors, we suggest to apply TruePIE on the corpus that contains considerably dense information for the extraction tasks. Proper entity linking may also help to improve the results by reducing entity sparsity and entity ambiguity. To ensure the quality of the generated training examples, the selected seed patterns should provide a good amount of extractions with high precision. In our experiments, we choose the seed patterns as the most frequent patterns containing the key words, which works nice in the examined tasks. There are relation types that TruePIE may fail, such as extracting the high/low blood pressure from medical records and n-ary relations, where the current definitions of pattern reliability score and arity-constraints are not suitable. We leave those relation extraction tasks as our future work.

One potential extension of the proposed TruePIE is to use it for building taxonomy of attributes automatically. Currently, TruePIE can discover that "leader" includes "president", "prime minister", "ruler" and several other attributes, which shows that it is promising in finding hypernym-hyponym and synonym attributes. We also observe a hierarchical structure in the pattern embedding space, such as the parent, family, general relationships between persons.

6 CONCLUSIONS

In this paper, we propose a novel method, called TruePIE, to discover reliable EAV-tuples and patterns from text data. By adding reliability into pattern quality assessment, TruePIE can significantly improve the precision of information extraction. We propose to represent the patterns by pattern embeddings so that semantically similar patterns are close to each other. To tackle the lack of supervision challenge, the proposed TruePIE is a self-training framework that automatically generates training patterns based on a couple of seed patterns, and gradually discovers more and more reliable patterns and EAV-tuples from the corpus with little human effort.

To better estimate the pattern reliability under the open-world assumption, arity-constraints are proposed to detect negative tuples and patterns. Extensive experiments on a massive corpus clearly demonstrates the effectiveness of the proposed TruePIE method.

7 ACKNOWLEDGMENTS

The work was sponsored in part by U.S. Army Research Lab. under Cooperative Agreement No. W911NF-09-2-0053 (NSCTA), DARPA under Agreement No. W911NF-17-C-0099, National Science Foundation IIS-16-18481, IIS-17-04532, IIS-17-41317, IIS-17-47614, and IIS-13-19973, DTRA HDTRA11810026, and grant 1U54GM114838 awarded by NIGMS through funds provided by the trans-NIH Big Data to Knowledge (BD2K) initiative. Any opinions, findings, and conclusions or recommendations expressed in this document are those of the author(s) and should not be interpreted as the views of any U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation hereon.

REFERENCES

[1] Eugene Agichtein and Luis Gravano. 2000. Snowball: Extracting relations from large plain-text collections. In *ACM DL*.
[2] Gabor Angeli, Sonal Gupta, Melvin Jose, Christopher D Manning, Christopher Ré, Julie Tibshirani, Jean Y Wu, Sen Wu, and Ce Zhang. 2014. Stanford's 2014 slot filling systems. *TAC KBP* 695 (2014).
[3] Gabor Angeli, Melvin Johnson Premkumar, and Christopher D Manning. 2015. Leveraging linguistic structure for open domain information extraction. In *ACL*.
[4] Michele Banko, Michael J Cafarella, Stephen Soderland, Matthew Broadhead, and Oren Etzioni. 2007. Open information extraction from the Web. In *IJCAI*.
[5] Hannah Bast, Björn Buchhold, and Elmar Haussmann. 2015. Relevance Scores for Triples from Type-Like Relations. In *SIGIR*.
[6] Hannah Bast, Björn Buchhold, Elmar Haussmann, and others. 2016. Semantic Search on Text and Knowledge Bases. *Foundations and Trends® in Information Retrieval* (2016).
[7] Andrew Carlson, Justin Betteridge, Bryan Kisiel, Burr Settles, Estevam R Hruschka Jr, and Tom M Mitchell. 2010. Toward an architecture for never-ending language learning. In *AAAI*, Vol. 5. 3.
[8] Marie-Catherine De Marneffe, Bill MacCartney, Christopher D Manning, and others. 2006. Generating typed dependency parses from phrase structure parses. In *Proceedings of LREC*, Vol. 6. Genoa, 449–454.
[9] Luciano Del Corro and Rainer Gemulla. 2013. Clausie: Clause-based open information extraction. In *WWW*.
[10] Anthony Fader, Stephen Soderland, and Oren Etzioni. 2011. Identifying relations for open information extraction. In *EMNLP*.
[11] Rahul Gupta, Alon Halevy, Xuezhi Wang, Steven Euijong Whang, and Fei Wu. 2014. Biperpedia: An ontology for search applications. In *VLDB*.
[12] Alon Halevy, Natalya Noy, Sunita Sarawagi, Steven Euijong Whang, and Xiao Yu. 2016. Discovering structure in the universe of attribute names. In *WWW*. 939–949.
[13] Marti A Hearst. 1992. Automatic acquisition of hyponyms from large text corpora. In *Proceedings of the 14th conference on Computational linguistics-Volume 2*. 539–545.
[14] Meng Jiang, Jingbo Shang, Taylor Cassidy, Xiang Ren, Lance M Kaplan, Timothy P Hanratty, and Jiawei Han. 2017. MetaPAD: Meta pattern discovery from massive text corpora. In *KDD*.
[15] Taesung Lee, Zhongyuan Wang, Haixun Wang, and Seung-won Hwang. 2013. Attribute extraction and scoring: A probabilistic approach. In *ICDE*.
[16] Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J. Bethard, and David McClosky. 2014. The Stanford CoreNLP Natural Language Processing Toolkit. In *Association for Computational Linguistics (ACL) System Demonstrations*. 55–60.
[17] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781* (2013).
[18] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *NIPS*. 3111–3119.
[19] Thahir P Mohamed, Estevam R Hruschka Jr, and Tom M Mitchell. 2011. Discovering relations between noun categories. In *EMNLP*. 1447–1455.
[20] Ndapandula Nakashole, Gerhard Weikum, and Fabian Suchanek. 2012. Discovering and exploring relations on the web. *Proceedings of the VLDB Endowment* 5, 12 (2012), 1982–1985.
[21] Ndapandula Nakashole, Gerhard Weikum, and Fabian Suchanek. 2012. PATTY: A taxonomy of relational patterns with semantic types. In *EMNLP*.
[22] Robert Parker, David Graff, Junbo Kong, Ke Chen, and Kazuaki Maeda. 2009. English Gigaword Fourth Edition LDC2009T13. *Linguistic Data Consortium, Philadelphia* (2009).
[23] Simon Parsons. 1996. Current approaches to handling imperfect information in data and knowledge bases. *TKDE* (1996).
[24] Meng Qu, Xiang Ren, Yu Zhang, and Jiawei Han. 2017. Overcoming Limited Supervision in Relation Extraction: A Pattern-enhanced Distributional Representation Approach. *arXiv preprint arXiv:1711.03226* (2017).
[25] Sebastian Riedel, Limin Yao, Andrew McCallum, and Benjamin M Marlin. 2013. Relation extraction with matrix factorization and universal schemas. In *HLT-NAACL*.
[26] Michael Schmitz, Robert Bart, Stephen Soderland, Oren Etzioni, and others. 2012. Open language learning for information extraction. In *EMNLP*.
[27] Kristina Toutanova, Danqi Chen, Patrick Pantel, Hoifung Poon, Pallavi Choudhury, and Michael Gamon. 2015. Representing Text for Joint Embedding of Text and Knowledge Bases.. In *EMNLP*, Vol. 15. 1499–1509.
[28] Mohamed Yahya, Steven Whang, Rahul Gupta, and Alon Y Halevy. 2014. ReNoun: Fact extraction for nominal attributes. In *EMNLP*.
[29] Alexander Yates, Michael Cafarella, Michele Banko, Oren Etzioni, Matthew Broadhead, and Stephen Soderland. 2007. Textrunner: Open information extraction on the web. In *ACL*.
[30] Jun Zhu, Zaiqing Nie, Xiaojiang Liu, Bo Zhang, and Ji-Rong Wen. 2009. StatSnowball: A statistical approach to extracting entity relationships. In *WWW*.