MetaPAD: Meta Pattern Discovery from Massive Text Corpora

Meng Jiang[1], Jingbo Shang[1], Taylor Cassidy[2], Xiang Ren[1]
Lance M. Kaplan[2], Timothy P. Hanratty[2], Jiawei Han[1]
[1]Department of Computer Science, University of Illinois Urbana-Champaign, IL, USA
[2]Computational & Information Sciences Directorate, Army Research Laboratory, Adelphi, MD, USA
[1]{mjiang89, shang7, xren7, hanj}@illinois.edu [2]{taylor.cassidy.civ, lance.m.kaplan.civ, timothy.p.hanratty.civ}@mail.mil

ABSTRACT

Mining textual patterns in news, tweets, papers, and many other kinds of text corpora has been an active theme in text mining and NLP research. Previous studies adopt a dependency parsing-based pattern discovery approach. However, the parsing results lose rich context around entities in the patterns, and the process is costly for a corpus of large scale. In this study, we propose a novel *typed textual pattern structure*, called *meta pattern*, which is extended to a frequent, informative, and precise subsequence pattern in certain context. We propose an efficient framework, called MetaPAD, which discovers meta patterns from massive corpora with three techniques: (1) it develops a context-aware segmentation method to carefully determine the boundaries of patterns with a learnt pattern quality assessment function, which avoids costly dependency parsing and generates high-quality patterns; (2) it identifies and groups synonymous meta patterns from multiple facets—their types, contexts, and extractions; and (3) it examines type distributions of entities in the instances extracted by each group of patterns, and looks for appropriate type levels to make discovered patterns precise. Experiments demonstrate that our proposed framework discovers high-quality typed textual patterns efficiently from different genres of massive corpora and facilitates information extraction.

1 INTRODUCTION

Discovering *textual patterns* from text data is an active research theme [4, 7, 10, 12, 28], with broad applications such as attribute extraction [11, 30, 32, 33], aspect mining [8, 15, 19], and slot filling [40, 41]. Moreover, a data-driven exploration of *efficient* textual pattern mining may also have strong implications on the development of efficient methods for NLP tasks on massive text corpora.

Traditional methods of textual pattern mining have made large pattern collections publicly available, but very few can extract arbitrary patterns with semantic types. Hearst patterns like "*NP* such as *NP*, *NP*, and *NP*" were proposed and widely used to acquire hyponymy lexical relation [14]. TextRunner [4] and ReVerb [10] are blind to the typing information in their lexical patterns; ReVerb constrains patterns to verbs or verb phrases that end with prepositions. NELL [7] learns to extract noun-phrase pairs based on a fixed set of prespecified relations with entity types like country:president→$Country×$Politician.

One interesting exception is the SOL patterns proposed by Nakashole et al. in PATTY [28]. PATTY relies on the Stanford dependency parser [9] and harnesses the typing information from a knowledge base [3, 5, 29] or a typing system [20, 27]. Figure 1(a) shows how the SOL patterns are automatically generated with the shortest paths between two typed entities on the parse trees of individual sentences. Despite of the significant contributions of the work, SOL patterns have three limitations on mining typed textual patterns from a large-scale text corpus as illustrated below.

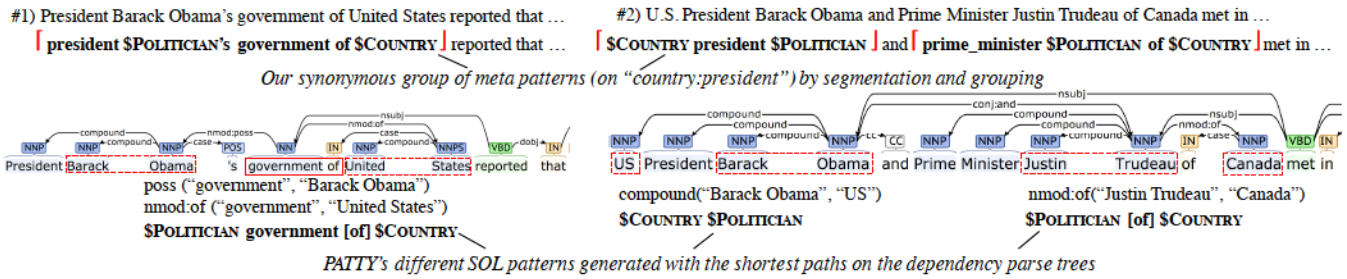
First, a good typed textual pattern should be of informative, self-contained context. The dependency parsing in PATTY loses the rich context around the entities such as the word "president" next to "Barack Obama" in sentence #1, and "president" and "prime_minister" in #2 (see Figure 1(a)). Moreover, the SOL patterns are restricted to the dependency path between two entities but do not represent the data types like $Digit for "55" (see Figure 1(b)) and $Month $Day $Year. Furthermore, the parsing process is costly: Its complexity is cubic in the length of sentence [23], which is too costly for news and scientific corpora that often have long sentences. We expect an efficient textual pattern mining method for massive corpora.

Second, synonymous textual patterns are expected to be identified and grouped for handling pattern sparseness and aggregating their extractions for extending knowledge bases and question answering. As quoted by red "⌈-⌉" pairs in Figure 1, country:president and person:age are two synonymous pattern groups: (1) {"president $Politician's government of $Country", "$Country president $Politician", ...} and (2) {"$Person, age $Digit", "$Person's age is $Digit", "$Person, a $Digit-year-old", ...}. However, the process of finding such synonymous pattern groups is non-trivial. Multi-faceted information should be considered: (1) synonymous patterns should share the same entity types or data types; (2) even for the same entity (e.g., Barack Obama), one should allow it be grouped and generalized differently (e.g., in ⟨United States, Barack Obama⟩ vs. ⟨Barack Obama, 55⟩); and (3) shared words (e.g., "president") or semantically similar contextual words (e.g., "age" and "-year-old") may play an important role in synonymous pattern grouping. PATTY does not explore the multi-faceted information at grouping syonymous patterns, and thus cannot aggregate such extractions into one collection.

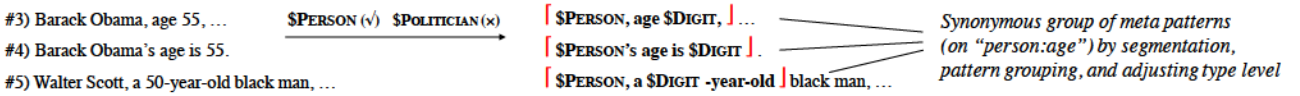
Third, the entity types in the textual patterns should be precise. In different patterns, even the same entity can be typed at different type levels. For example, the entity "Barack Obama" should be typed at a fine-grained level ($Politician) in the patterns generated from sentence #1–2, and it should be typed at a coarse-grained level ($Person) in the patterns from sentence #3–4. However, PATTY does not look for appropriate granularity of the entity types.

KDD'17, August 13–17, 2017, Halifax, NS, Canada.

DOI: http://dx.doi.org/10.1145/3097983.3098105



(a) MetaPAD considers rich contexts around entities and determines pattern boundaries by pattern quality assessment while dependency parsing does not.



(b) MetaPAD finds meta patterns consisting of both entity types and data types like $DIGIT. It also adjusts the type level for appropriate granularity.

Figure 1: Comparing the synonymous group of meta patterns in MetaPAD with that of SOL patterns in PATTY.

In this paper, we propose a new typed textual pattern called *meta pattern*, which is defined as follows.

Definition (Meta Pattern). A meta pattern refers to a frequent, informative, and precise subsequence pattern of entity types (e.g., $PERSON, $POLITICIAN, $COUNTRY) or data types (e.g., $DIGIT, $MONTH, $YEAR), words (e.g., "politician", "age") or phrases (e.g., "prime_minister"), and possibly punctuation marks (e.g., ",", "("), which serves as an integral semantic unit in certain context.

We study the problem of mining meta patterns and grouping synonymous meta patterns. *Why mining meta patterns and grouping them into synonymous meta pattern groups?*—because mining and grouping meta patterns into synonymous groups may facilitate information extraction and turning unstructured data into structures. For example, given us a sentence from a news corpus, "President Blaise Compaoré's government of Burkina Faso was founded ...", if we have discovered the meta pattern "president $POLITICIAN's government of $COUNTRY", we can recognize and type new entities (i.e., type "Blaise Compaoré" as a $POLITICIAN and "Burkina Faso" as a $COUNTRY), which previously requires human expertise on language rules or heavy annotations for learning [26]. If we have grouped the pattern with synonymous patterns like "$COUNTRY president $POLITICIAN", we can merge the fact tuple ⟨Burkina Faso, president, Blaise Compaoré⟩ into the large collection of facts of the attribute type country:president.

To systematically address the challenges of mining meta patterns and grouping synonymous patterns, we develop a novel framework called MetaPAD (Meta PAttern Discovery). Instead of working on every individual sentence, our MetaPAD leverages *massive* sentences in which redundant patterns are used to express attributes or relations of *massive* instances. First, MetaPAD generates meta pattern candidates using efficient sequential pattern mining, learns a quality assessment function of the patterns candidates with a rich set of domain-independent contextual features for intuitive ideas (e.g., frequency, informativeness), and then mines the quality meta patterns by assessment-led context-aware segmentation (see Sec. 4.1). Second, MetaPAD formulates the grouping process of synonymous meta patterns as a learning task, and solves it by integrating features from multiple facets including entity types, data types, pattern context, and extracted instances (see Sec. 4.2). Third, MetaPAD examines the type distributions of entities in the extractions from every meta pattern group, and looks for the most appropriate type level that the patterns fit. This includes both top-down and bottom-up schemes that traverse the type ontology for the patterns' preciseness (see Sec. 4.3).

The major contributions of this paper are as follows: (1) we propose a new definition of typed textual pattern, called *meta pattern*, which is more informative, precise, and efficient in discovery than the SOL pattern; (2) we develop an efficient meta-pattern mining framework, MetaPAD of three components: generating quality meta patterns by context-aware segmentation, grouping synonymous meta patterns, and adjusting entity-type levels for appropriate granularity in the pattern groups; and (3) our experiments on news and tweet text datasets demonstrate that the MetaPAD not only generates high quality patterns but also achieves significant improvement over the state-of-the-art in information extraction.

2 RELATED WORK

In this section, we summarize existing systems and methods that are related to the topic of this paper.

TextRunner [4] extracts strings of words between entities in text corpus, and clusters and simplifies these word strings to produce relation-strings. ReVerb [10] constrains patterns to verbs or verb phrases that end with prepositions. However, the methods in the TextRunner/ReVerb family generate patterns of frequent relational strings/phrases without entity information. Another line of work, open information extraction systems [2, 22, 36, 39], are supposed to extract verbal expressions for identifying arguments. This is less related to our task of discovering textual patterns.

Google's Biperpedia [12, 13] generates *E-A patterns* (e.g., "*A* of *E*" and "*E* 's *A*") from users' fact-seeking queries (e.g., "president of united states" and "barack oabma's wife") by replacing entity with "*E*" and noun-phrase attribute with "*A*". ReNoun [40] generates *S-A-O patterns* (e.g., "*S*'s *A* is *O*" and "*O*, *A* of *S*,") from human-annotated

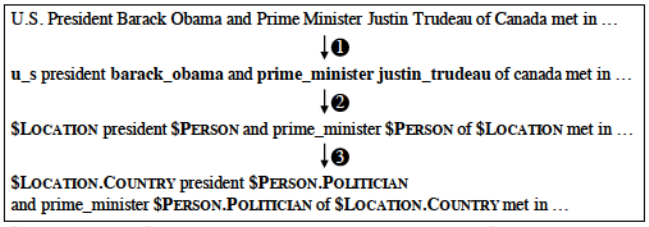


Figure 2: Preprocessing for fine-grained typed corpus: given us a corpus and a typing system.

corpus (e.g., "Barack Obama's wife is Michelle Obama" and "Larry Page, CEO of Google") on a pre-defined subset of the attribute names, by replacing entity/subject with "*S*", attribute name with "*A*", and value/object with "*O*". However, the query logs and annotations are often unavailable or expensive. Furthermore, query log word distributions are highly constrained compared with ordinary written language. So most of the *S-A-O* patterns like "*S A O*" and "*S*'s *A O*" will generate noisy extractions when applied to a text corpus. Textual pattern learning methods [38] including the above are blind to the typing information of the entities in the patterns; the patterns are not *typed textual patterns*.

NELL [7] learns to extract noun-phrase pairs from text corpus based on a fixed set of prespecified relations with entity types. OntExt [25] clusters pattern co-occurrences for the noun-phrase pairs for a given entity type at a time and does not scale up to mining a large corpus. PATTY [28] was the first to harness the typing system for mining relational patterns with entity types. We have extensively discussed the differences between our proposed *meta patterns* and PATTY's *SOL patterns* in the introduction: Meta pattern candidates are efficiently generated by sequential pattern mining [1, 31, 42] on a massive corpus instead of dependency parsing on every individual sentence; meta pattern mining adopts a context-aware segmentation method to determine where a pattern starts and ends; and meta patterns are not restricted to words between entity pairs but generated by pattern quality estimation based on four criteria: frequency, completeness, informativeness, and preciseness, grouped on synonymous patterns, and with type level adjusted for appropriate granularity.

3 META PATTERN DISCOVERY

3.1 Preprocessing: Harnessing Typing Systems

To find meta patterns that are typed textual patterns, we apply efficient text mining methods for preprocessing a corpus into *fine-grained typed corpus* as input in three steps as follows (see Figure 2): (1) we use a phrase mining method [21] to break down a sentence into phrases, words, and punctuation marks, which finds more real phrases (e.g., "barack_obama", "prime_minister") than the frequent n-grams by frequent itemset mining in PATTY; (2) we use a distant supervision-based method [34] to jointly recognize entities and their coarse-grained types (i.e., $PERSON, $LOCATION, and $ORGANIZATION); (3) we adopt a fine-grained typing system [35] to distinguish 113 entity types of 2-level ontology (e.g., $POLITICIAN, $COUNTRY, and $COMPANY); we further use a set of language rules to have 6 data types (i.e., $DIGIT, $DIGITUNIT[1], $DIGITRANK[2], $MONTH, $DAY, and $YEAR). Now we have a fine-grained, typed corpus consisting of the tokens as defined in the meta pattern: entity types, data types, phrases, words, and punctuation marks.

3.2 The Proposed Problem

Problem (Meta Pattern Discovery). Given a fine-grained, typed corpus of massive sentences $C = [\ldots, S, \ldots]$, and each sentence is denoted as $S = t_1 t_2 \ldots t_n$ in which $t_k \in \mathcal{T} \cup \mathcal{P} \cup \mathcal{M}$ is the k-th token ($\mathcal{T}$ is the set of entity types and data types, $\mathcal{P}$ is the set of phrases and words, and $\mathcal{M}$ is the set of punctuation marks), the task is to find **synonymous groups of quality meta patterns**. A *meta pattern mp* is a subsequential pattern of the tokens from the set $\mathcal{T} \cup \mathcal{P} \cup \mathcal{M}$. A *synonymous meta pattern group* is denoted by $\mathcal{MPG} = [\ldots, mp_i, \ldots, mp_j \ldots]$ in which each pair of meta patterns, mp_i and mp_j, are synonymous.

What is a quality meta pattern? Here we take the sentences as sequences of tokens. Previous sequential pattern mining algorithms mine frequent subsequences satisfying a single metric, the minimum support threshold (min_sup), in a transactional sequence database [1]. However, for text sequence data, the quality of our proposed textual pattern, the meta pattern, should be evaluated similar to phrase mining [21], in four criteria as illustrated below.

Example. The quality of a pattern is evaluated with the following criteria: (the former pattern has higher quality than the latter)
Frequency: "$DIGITRANK president of $COUNTRY" vs. "young president of $COUNTRY";
Completeness: "$COUNTRY president $POLITICIAN" vs. "$COUNTRY president", "$PERSON's wife, $PERSON" vs. "$PERSON's wife";
Informativeness: "$PERSON's wife, $PERSON" vs. "$PERSON and $PERSON";
Preciseness: "$COUNTRY president $POLITICIAN" vs. "$LOCATION president $PERSON", "$PERSON's wife, $PERSON" vs. "$POLITICIAN's wife, $PERSON", "population of $LOCATION" vs. "population of $COUNTRY".

What are synonymous meta patterns? The full set of frequent sequential patterns from a transaction dataset is huge [1]; and the number of meta patterns from a massive corpus is also big. Since there are multiple ways to express the same or similar meanings in a natural language, many meta patterns may share the same or nearly the same meaning. Examples have been given in Figure 1. Grouping synonymous meta patterns can help aggregate a large number of extractions of different patterns from different sentences. And the type distribution of the aggregated extractions can help us adjust the meta patterns in the group for preciseness.

4 THE METAPAD FRAMEWORK

Figure 3 presents the MetaPAD framework for Meta PAttern Discovery. It has three modules. First, it develops a context-aware segmentation method to determine the boundaries of the subsequences and generate the meta patterns of frequency, completeness, and informativeness (see Sec. 4.1). Second, it groups synonymous meta patterns into clusters (see Sec. 4.2). Third, for every synonymous pattern group, it adjusts the levels of entity types for appropriate granularity to have precise meta patterns (see Sec. 4.3).

[1] $DIGITUNIT: "percent", "%", "hundred", "thousand", "million", "billion", "trillion"...
[2] $DIGITRANK: "first", "1st", "second", "2nd", "44th"...

4.1 Generating meta patterns by context-aware segmentation

Pattern candidate generation. We adopt the standard frequent sequential pattern mining algorithm [31] to look for pattern candidates that satisfy a min_sup threshold. In practice, one can set a maximum pattern length ω to restrict the number of tokens in the patterns. Different from syntactic analysis of very long sentences, our meta pattern mining explores pattern structures that are local but still of wide context: in our experiments, we set $\omega = 20$.

Meta pattern quality assessment. Given a huge number of pattern candidates that can be messy (e.g., "of $COUNTRY" and "$POLITICIAN and"), it is desired but challenging to assess the quality of the patterns with a very few training labels. We introduce a rich set of *contextual features* of the patterns according to the quality criteria (see Sec. 3.2) as follows and train a classifier to estimate the quality function $Q(mp) \in [0, 1]$ where mp is a meta pattern candidate:

1. Frequency: A good pattern mp should occur with sufficient count $c(mp)$ in a given typed text corpus. The other feature is the normalized frequency of mp by the size of the given corpus.

2. Concordance: If the collocation of tokens in such frequency that is significantly higher than what is expected due to chance, the meta pattern mp has good concordance. To statistically reason about the concordance, we consider a null hypothesis: the corpus is generated from a series of independent Bernoulli trials. Suppose the number of tokens in the corpus is L that can be assumed to be fairly large. The expected frequency of a pair of sub-patterns $\langle mp_l, mp_r \rangle$ under our null hypothesis of their independence is

$$\mu_0(c(\langle mp_l, mp_r \rangle)) = L \cdot p(mp_l) \cdot p(mp_r), \tag{1}$$

where $p(mp) = \frac{c(mp)}{L}$ is the empirical probability of the pattern. We examine all the possible cases of dividing mp to left sub-pattern mp_l and right sub-pattern mp_r. There is no overlap between the sub-patterns. We use Z score to provide a quantitative measure of a pair of sub-patterns $\langle mp_l, mp_r \rangle$ forming the best collocation (maximum Z score) as mp in the corpus:

$$Z(mp) = \max_{\langle mp_l, mp_r \rangle = mp} \frac{c(mp) - \mu_0(c(\langle mp_l, mp_r \rangle))}{\sigma_{\langle mp_l, mp_r \rangle}}, \tag{2}$$

where $\sigma_{\langle mp_l, mp_r \rangle}$ is the standard deviation of the frequency. A high Z score indicates that the pattern is acting as an integral semantic unit in the context: its composed sub-patterns are highly associated.

3. Informativeness: A good pattern mp should have informative context. We examine the counts of different kinds of tokens (e.g., types, words, phrases, non-stop words, marks). For example, the pattern "$PERSON's wife $PERSON" is informative for the non-stop word "wife"; "$PERSON was born_in $CITY" is good for the phrase "born_in"; and "$PERSON, $DIGIT," is also informative for the two different types and two commas. Besides the counts, we adopt Inverse-Document Frequency (IDF) to avoid the issue of over-popularity of some tokens.

4. Completeness: We use the ratio between the frequencies of the pattern candidate (e.g., "$COUNTRY president $POLITICIAN") and its sub-patterns (e.g., "$COUNTRY president"). If the ratio is high, the candidate is likely to be complete. We also use the ratio between the frequencies of the pattern candidate and its super-patterns. If the ratio is high, the candidate is likely to be incomplete. Moreover, we expect the meta pattern to be NOT bounded by stop words. For example, neither "and $COUNTRY president" nor "president $POLITICIAN and" is properly bounded. Note that completeness is different from concordance: For example, in the concordance test, "$COUNTRY president $POLITICIAN" cannot be divided into two sub-patterns because "$POLITICIAN" is not a valid sub-pattern, but the completeness features can tell that "$COUNTRY president $POLITICIAN" is more complete than any of the sub-patterns "$COUNTRY president" or "president $POLITICIAN".

5. Coverage: A good typed pattern can extract multiple instances. For example, the type $POLITICIAN in the pattern "$POLITICIAN's healthcare law" refers to only one entity "Barack Obama", and thus has too low coverage in the corpus. The count of entities referred to a type in the pattern is normalized by the size of the corpus.

We train a classifier based on random forests [6] for learning the meta-pattern quality function $Q(mp)$ with the above rich set of contextual features. Our experiments (not reported here for the sake of space) show that using only 100 positive pattern labels can achieve similar precision and recall as using 300 positive labels. Since the number of pattern candidate is often much more than the number of lables, we randomly pick a set of pattern candidates as negative labels. The numbers of positive labels and negative labels are the same. This part can be further improved by using ensemble learning for robust label selection [37]. Note that the learning results can be transferred to other domains: For example, if we transfer the learning model on news or tweets to the bio-medical corpus, the features of low-quality patterns "$POLITICIAN and $COUNTRY" and "$BACTERIA and $ANTIBIOTICS" are similar; the features of

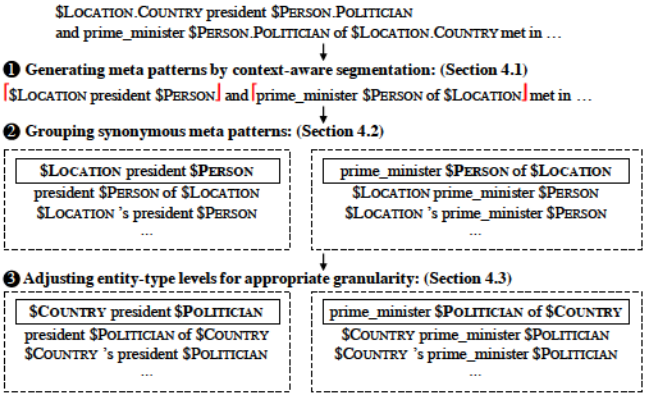


Figure 3: Three modules in our MetaPAD framework.

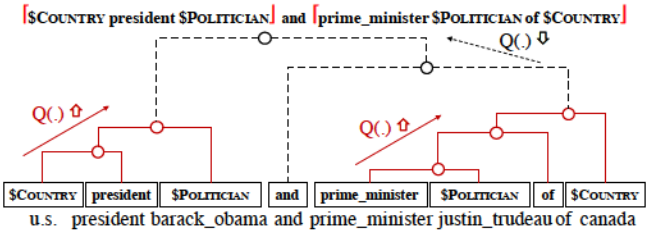


Figure 4: Generating meta patterns by context-aware segmentation with the pattern quality function $Q(.)$.

Table 1: Issues of quality over-/under-estimation can be fixed when the segmentation rectifies pattern frequency.

Pattern candidate	Before segmentation Count	Quality	Frequency rectified after segmentation Count	Quality	Issue fixed by feedback
$Country president $Politician	2,912	0.93	2,785	0.97	N/A
prime_minister $Politician of $Country	1,285	0.84	1,223	0.92	slight underestimation
$Politician and prime_minister $Politician	532	0.70	94	0.23	overestimation

high-quality patterns "$Politician is president of $Country" and "$Bacteria is resistant to $Antibiotics" are similar.

In our practice, we find the random forests model is effective and efficient. There could be space for improvement by adopting more complicated learning models such as Conditional Random Field (CRF) and Deep Neural Network (DNN) models. We would suggest practitioners who use the above models to keep considering (1) to use entity types in quality pattern classification and (2) to use the rich set of features we have introduced as above to assess the quality of meta patterns.

Context-aware segmentation using $Q(.)$ with feedback. With the pattern quality function $Q(.)$ learnt from the rich set of contextual features, we develop a bottom-up segmentation algorithm to construct the best partition of segments of high quality scores. As shown in Figure 4, we use $Q(.)$ to determine the boundaries of the segments: we take "$Country president $Politician" for its high quality score; we do not take the candidate "and prime_minister $Politician of $Country" because of its low quality score.

Since $Q(mp)$ was learnt with features including the raw frequency $c(mp)$, the quality score may be overestimated or underestimated: the principle is that every token's occurrence should be assigned to only one pattern but the raw frequency may count the tokens multiple times. Fortunately, after the segmentation, we can rectify the frequency as $c_r(mp)$, for example in Figure 4, the segmentation avoids counting "$Politician and prime_minister $Politician" of overestimated frequency/quality (see Table 1).

Once the frequency feature is rectified, we re-learn the quality function $Q(.)$ using $c(mp)$ as feedback and re-segment the corpus with it. This can be an iterative process but we found in only one iteration, the result converges. Algorithm 1 shows the details.

Algorithm 1 Context-aware segmentation using Q with feedback

Require: corpus of sentences C=[..., S, ...], $S = t_1 t_2 ... t_n$ (t_k is the k-th token), a set of meta pattern candidates MP_{cand}, meta-pattern quality function $Q(.)$ learnt by contextual features

1: Set all the rectified frequency $c_r(mp)$ to zero
2: **for** $S \in C$ **do**
3: Segment the sentence S into Seg=[..., mp, ...] by maximizing $\sum_{mp \in Seg} Q(mp)$ with a bottom-up scheme (see Figure 4), where $mp \in MP_{cand}$ is a segment of high quality score
4: **for** $mp \in Seg$ **do**
5: $c_r(mp) \leftarrow c_r(mp) + 1$
6: **end for**
7: **end for**
8: Re-learn $Q(.)$ by replacing the raw frequency feature $c(mp)$ with the rectified frequency $c_r(mp)$ as feedback
9: Re-segment the corpus C with the new $Q(.)$
10: **return** Segmented corpus, a set of quality meta patterns in the segmented corpus, and their quality scores in $Q(.)$

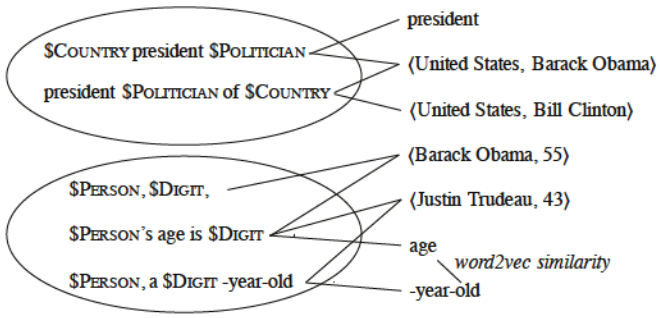


Figure 5: Grouping synonymous meta patterns with information of context words and extractions.

4.2 Grouping synonymous meta patterns

Grouping truly synonymous meta patterns enables a large collection of extractions of the same relation aggregated from different but synonymous patterns. For example, there could be hundreds of ways of expressing the relation country:president; if we group all such meta patterns, we can aggregate all the extractions of this relation from massive corpus. PATTY [28] has a narrow definition of their synonymous dependency path-based SOL patterns: two patterns are synonymous if they generate the same set of extractions from the corpus. Here we develop a learning method to incorporate information of three aspects, (1) entity/data types in the pattern, (2) context words/phrases in the pattern, and (3) extractions from the pattern, to assign the meta patterns into groups. Our method is based on three assumptions as follows (see Figure 5):

A1: Synonymous meta patterns must have the same entity/data types: the meta patterns "$Person's age is $Digit" and "$Person's wife is $Person" cannot be synonymous;

A2: If two meta patterns share (nearly) the same context words/phrases, they are more likely to be synonymous: the patterns "$Country president $Politician" and "president $Politician of $Country" share the word "president";

A3: If two patterns generate more common extractions, they are more likely to be synonymous: both "$Person's age is $Digit" and "$Person, $Digit," generate ⟨Barack Obama, 55⟩.

Since the number of groups cannot be pre-specified, we propose to first construct a pattern-pattern graph in which the two pattern nodes of every edge satisfy *A1* and are predicted to be synonymous, and then use a dense δ-clique detection technique to find all dense cliques as synonymous meta patten groups. We set up the density $\delta = 0.8$ as the common density clique detection technique does [17]. The density threshold could be derived and automatically set based on the principle of Minimum Description Length (MDL) [18]. Here each pair of the patterns (mp_i, mp_j) in the group $\mathcal{MPG} = [\ldots, mp_i, \ldots, mp_j \ldots]$ are synonymous.

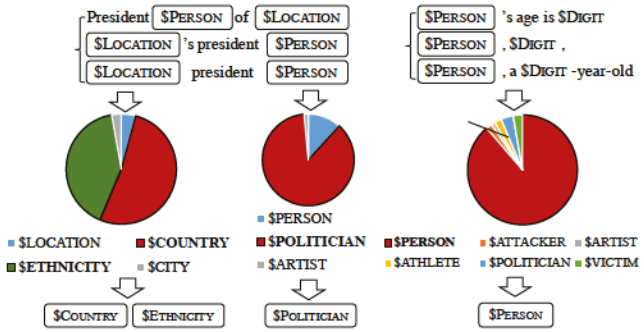


Figure 6: Adjusting entity-type levels for appropriate granularity with entity-type distributions.

For the graph construction, we train Support Vector Regression (SVR) to learn the following features of a pair of patterns based on *A2* and *A3*: (1) the numbers of words, non-stop words, phrases that each pattern has and they share; (2) the maximum similarity score between pairs of non-stop words or phrases in the two patterns; (3) the number of extractions that each pattern has and they share. The similarity between words/phrases is represented by the cosine similarity of their word2vec embeddings [24, 38]. The regression results provide us a scores of mixed similarities for each pair of patter nodes.

4.3 Adjusting type levels for preciseness

Given a group of synonymous meta patterns, we expect the patterns to be precise: it is desired to determine the levels of the entity types in the patterns for appropriate granularity. Thanks to the grouping process of synonymous meta patterns, we have rich type distributions of the entities from the large collection of extractions.

As shown in Figure 6, given the ontology of entity types (e.g., $LOCATION: $COUNTRY, $STATE, $CITY, ...; $PERSON: $ARTIST, $ATHLETE, $POLITICIAN, ...), for the group of synonymous meta patterns "president $PERSON of $LOCATION", "$LOCATION's president $PERSON", and "$LOCATION president $PERSON", are the entity types, $LOCATION and $PERSON, of appropriate granularity to make the patterns precise? If we look at the type distributions of entities in the extractions of these patterns, it is clear that most of the entities for $LOCATION are typed at a fine-grained level as $COUNTRY (e.g., "United States") or $ETHNICITY (e.g., "Russian"), and most of the entities for $PERSON also have the fine-grained type $POLITICIAN. Therefore, compared with "$LOCATION president $PERSON", the two fine-grained meta patterns "$COUNTRY president $POLITICIAN" and "$ETHNICITY president $POLITICIAN" are more precise; we have the same claim for other meta patterns in the synonymous group. On the other hand, for the group of synonymous meta patterns on person:age, we can see most of the entities are typed at a coarse-grained level as $PERSON instead of $ATHLETE or $POLITICIAN. So the entity type in the patterns is good to be $PERSON. From this observation, given an entity type T in the meta pattern group, we propose a metric, called *graininess*, that is defined as the fraction of the entities typed by T that can be fine-grained to T's sub-types:

$$g(T) = \frac{\sum_{T' \in subtype\_of(T)} num\_entity(T')}{\sum_{T' \in subtype\_of(T) \cup \{T\}} num\_entity(T')}. \quad (3)$$

Table 2: Two datasets we use in the experiments.

Dataset	File Size	#Document	#Entity	#Entity Mention
APR (news)	199MB	62,146	284,061	6,732,399
TWT (tweet)	1.05GB	13,200,821	618,459	21,412,381

If $g(T)$ is higher than a threshold θ, we go down the type ontology for the fine-grained types.

Suppose we have determined the appropriate type level in the meta pattern group using the graininess metric. However, not every type at the level should be used to construct precise meta patterns. For example, we can see from Figure 6 for the patterns on president, very few entities of $LOCATION are typed as $CITY, and very few entities of $PERSON are typed as $ARTIST. Comparing with $COUNTRY, $ETHNICITY, and $POLITICIAN, these fine-grained types are at the same level but have too small support of extractions. We exclude them from the meta pattern group. Based on this idea, for an entity type T, we propose another metric, called *support*, that is defined as the ratio of the number of entities typed by T to the maximum number of entities typed by T's sibling types:

$$s(T) = \frac{num\_entity(T)}{\max_{T' \in sibling-type\_of(T) \cup \{T\}} num\_entity(T')}. \quad (4)$$

If $s(T)$ is higher than a threshold γ, we consider the type T in the meta pattern group; otherwise, we drop it.

With these two metrics, we develop a *top-down* scheme that first conducts segmentation and synonymous pattern grouping on the coarse-grained typed meta patterns, and then checks if the fine-grained types are significant and if the patterns can be split to the fine-grained level; we also develop a *bottom-down* scheme that first works on the fine-grained typed meta patterns, and then checks if the patterns can be merged into a coarse-grained level.

4.4 Complexity analysis

We develop three new components in our MetaPAD. The time complexity of generating meta patterns with context-aware segmentation is $O(\omega|C|)$ where ω is the maximum pattern length and $|C|$ is the corpus size (i.e., the total number of tokens in the corpus). The complexity of grouping synonymous meta patterns is $O(|\mathcal{MP}|)$, and the complexity of adjusting type levels is $O(h|\mathcal{MP}|)$ where $|\mathcal{MP}|$ is the number of quality meta patterns and h is the height of type ontology. The total complexity is $O(\omega|C| + (h+1)|\mathcal{MP}|)$, which is linear in the corpus size.

PATTY [28] is also scalable in the number of sentences but for each sentence, the complexity of dependency parsing it adopted is as high as $O(n^3)$ where n is the length of the sentence. If the corpus has many long sentences, PATTY is time-consuming; whereas our MetaPAD's complexity is linear to the sentence length for every individual sentence. The empirical study on the scalability can be found in the next section.

5 EXPERIMENTS

This section reports our essential experiments that demonstrate the effectiveness of the MetaPAD at (1) typed textual pattern mining: discovering synonymous groups of meta patterns, and (2) one application: extracting tuple information from two datasets of different genres. Additional results regarding efficiency are reported as well.

Table 3: Entity-Attribute-Value tuples as ground truth.

Attribute	Type of Entity	Type of Value	#Tuple
country:president	$COUNTRY	$POLITICIAN	1,170
country:minister	$COUNTRY	$POLITICIAN	1,047
state:representative	$STATE	$POLITICIAN	655
state:senator	$STATE	$POLITICIAN	610
county:sheriff	$COUNTY	$POLITICIAN	106
company:ceo	$COMPANY	$BUSINESSPERSON	1,052
university:professor	$UNIVERSITY	$RESEARCHER	707
award:winner	$AWARD	$PERSON	274

5.1 Datasets

Table 2 presents the statistics of two datasets from different genres:

- APR: news from The Associated Press and Reuters in 2015;
- TWT: tweets collected via Twitter API in 2015/06–2015/09.

The news corpus often has long sentences, which is rather challenging for textual pattern mining. For example, the component of dependency parsing in PATTY [28] has cubic computational complexity of the length for individual sentences.

The preprocessing techniques in our MetaPAD adopt distant supervision with external databases for entity recognition and fine-grained typing (see Sec. 3.1). We use DBpedia [3] and Freebase [5] as knowledge bases for distant supervision.

5.2 Experimental Settings

We conduct two tasks in the experiments. The first task is to *discover typed textual patterns from massive corpora and organize the patterns into synonymous groups*. We compare with the state-of-the-art SOL pattern synset mining method **PATTY** [28] on both the quality of patterns and the quality of synonymous pattern groups. Since there is no standard ground truth of the typed textual patterns, we report extensive qualitative analysis on the three datasets.

The second task is to *extract* ⟨*entity, attribute, value*⟩ (*EAV*) *tuple information*. For every synonymous pattern set generated by the competitive methods from news and tweets, we assign it to one attribute type from the set in Table 3 if appropriate. We collect 5,621 EAV-tuples from the extractions, label them as true or false, and finally, we have 3,345 true EAV-tuples. We have 2,400 true EAV-tuples from APR and 2,090 from TWT. Most of them are out of the existing knowledge bases: we are exploring new extractions from new text corpora.

We evaluate the performance in terms of *precision* and *recall*. Precision is defined as the fraction of the predicted EAV-tuples that are true. Recall is defined as the fraction of the labelled true EAV-tuples that are predicted as true EAV-tuples. We use (1) the F1 score that is the harmonic mean of precision and recall, and (2) the Area Under the precision-recall Curve (AUC). All the values are between 0 and 1, and a higher value means better performance.

In the second task, besides **PATTY**, the competitive methods for tuple extraction are: **Ollie** [36] is an open IE system that extracts relational tuples with syntactic and lexical patterns; **ReNoun** [40] learns "*S-A-O*" patterns such as "*S A*, *O*," and "*A* of *S* is *O*" with annotated corpus. Both methods ignore the entity-typing information. We develop four alternatives of MetaPAD as follows:

1. MetaPAD-T only develops segmentation to generate patterns in which the entity types are at the *top* (coarse-grained) level;
2. MetaPAD-TS develops all the three components of MetaPAD including synonymous pattern grouping based on MetaPAD-T;
3. MetaPAD-B only develops segmentation to generate patterns in which the entity types are at the *bottom* (fine-grained) level;
4. MetaPAD-BS develops all the three components of MetaPAD including synonymous pattern grouping based on MetaPAD-B.

For the parameters in MetaPAD, we set the maximum pattern length as $\omega = 20$, the threshold of graininess score as $\theta = 0.8$, and the threshold of support score as $\gamma = 0.1$. We tuned the parameters to achieve the best performance. We would like to point out that it would be more effective to automatically find the best parameters by statitical analysis on the corpus distribution.

5.3 Results on Typed Textual Pattern Discovery

Our proposed MetaPAD discovers high-quality meta patterns by context-aware segmentation from massive text corpus with a pattern quality assessment function. It further organizes them into synonymous groups. With each group of the truly synonymous meta patterns, we can easily assign an appropriate attribute type to it, and harvest a large collection of instances extracted by different patterns of the same group.

Table 4 presents the groups of synonymous meta patterns that express attribute types country:president and company:ceo. **First**, we can see that the meta patterns are generated from a typed corpus instead of the shortest path of a dependency parse tree. Thus, the patterns can keep rich, wide context information. **Second**, the meta patterns are of high quality on informativeness, completeness, and so on, and practitioners can easily tell why the patterns are extracted as an integral semantic unit. **Third**, though the patterns like "$POLITICIAN was elected as the president of $COUNTRY" are relatively long and rare, they can be grouped with their synonymous patterns so that all the extractions about one entity-attribute type can be aggregated into one set. That is why MetaPAD successfully discovers who is/was the president of a small country like Burkina Faso or the ceo of a young company like Afghan Citadel. **Fourth**, MetaPAD discovered a rich collection of person:date_of_birth information from the new corpus that *does not often exist in the knowledge bases*, thanks to our meta patterns use not only entity types but also data types like $MONTH $DAY $YEAR.

Figure 7 shows the SOL pattern synsets that PATTY generates from the four sentences. First, the dependency path loses the rich context around the entities like "president" in the first example and "ceo" in the last example. Second, the SOL pattern synset cannot group truly synonymous typed textual patterns. We can see the advantages of generating meta patterns and grouping them into synonymous clusters. In the introduction section we also show our MetaPAD can find meta patterns of rich data types for the attribute types like person:age and person:date_of_birth.

5.4 Results on EAV-Tuple Extraction

Besides directly comparisons on the quality of mining synonymous typed textual patterns, we apply patterns from different systems, Ollie [36], ReNoun [40], and PATTY [28], to extract tuple information from the two general corpora APR (news) and TWT (tweets). We attempt to provide quantitative analysis on the use of the typed textual patterns by evaluating how well they can facilitate the tuple

Table 4: Synonymous meta patterns and their extractions that MetaPAD generates from the *news corpus* APR on country:president, company:ceo, and person:date_of_birth.

A group of synonymous meta patterns	$Country	$Politician
$Country president $Politician	United States	Barack Obama
$Country's president $Politician	United States	Bill Clinton
president $Politician of $Country	Russia	Vladimir Putin
$Politician, the president of $Country,	France	François Hollande
...	...	...
president $Politician's government of $Country	Comoros	Ikililou Dhoinine
$Politician was elected as the president of $Country	Burkina Faso	Blaise Compaoré

A group of synonymous meta patterns	$Company	$Businessperson
$Company ceo $Businessperson	Apple	Tim Cook
$Company chief executive $Businessperson	Facebook	Mark Zuckerburg
$Businessperson, the $Company ceo,	Hewlett-Packard	Carly Fiorina
$Company former ceo $Businessperson	Yahoo!	Marissa Mayer
...	...	...
$Businessperson was appointed as ceo of $Company	Infor	Charles Phillips
$Businessperson, former interim ceo, leaves $Company	Afghan Citadel	Roya Mahboob

A group of synonymous meta patterns	$Person	$Day $Month $Year
$Person was born $Month $Day, $Year	Willie Howard Mays	6 May 1931
$Person was born on $Day $Month $Year	Robert David Simon	29 May 1941
$Person (born on $Month $Day, $Year)	Phillip Joel Hughes	30 Nov 1988
$Person (born on $Day $Month $Year)	...	...
$Person, was born on $Month $Day, $Year	Carl Sessions Stepp	8 Sept 1956
...	Richard von Weizsaecker	15 April 1920

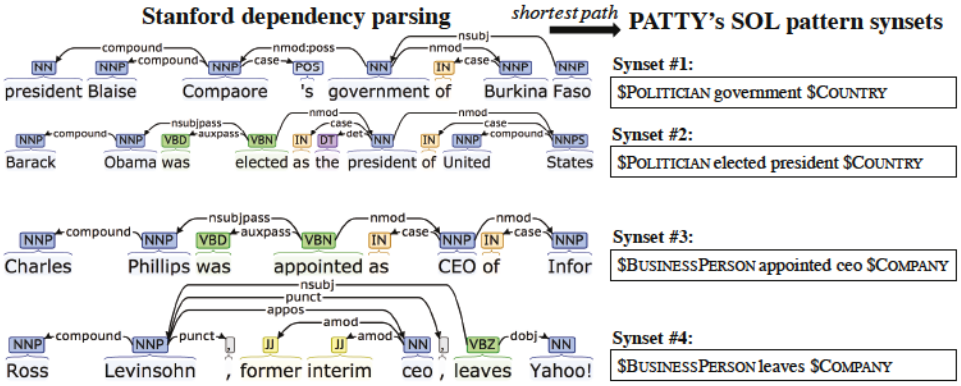
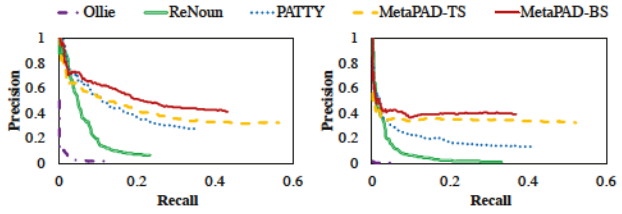


Figure 7: Compared with our meta patterns, the SOL pattern mining does not take the rich context into full consideration of pattern quality assessment; the definition of SOL pattern synset is too limited to group truly synonymous patterns.

Table 5: Reporting F1, AUC, and number of true positives (TP) on tuple extraction from news and tweets data.

	APR (news, 199MB)			TWT (tweets, 1.05GB)		
	F1	AUC	TP	F1	AUC	TP
Ollie [36]	0.0353	0.0133	288	0.0094	0.0012	115
ReNoun [40]	0.1309	0.0900	562	0.0821	0.0347	698
PATTY [28]	0.3085	0.2497	860	0.2029	0.1256	860
MetaPAD-T	0.3614	0.2843	799	0.3621	0.2641	880
MetaPAD-TS	0.4156	0.3269	**1,355**	**0.4153**	**0.3554**	**1,111**
MetaPAD-B	0.3684	0.3186	787	0.3228	0.2704	650
MetaPAD-BS	**0.4236**	**0.3525**	1,040	0.3827	0.3408	975



(a) APR (news, 199MB) (b) TWT (tweets, 1.05GB)

Figure 8: Precision-recall on tuple information extraction.

extraction which is similar with one of the most challenging NLP tasks called *slot filling* for new attributes [16].

Table 5 summarizes comparison results on tuple information that each texutal pattern-driven system extracts from news and tweet datasets. Figure 8 presents precision-recall curves that further demonstrate the effectiveness of our MetaPAD methods. We provide our observation and analysis as follows.

1) Overall, our MetaPAD-TS and MetaPAD-BS outperform the baseline methods, achieving significant improvement on both datasets

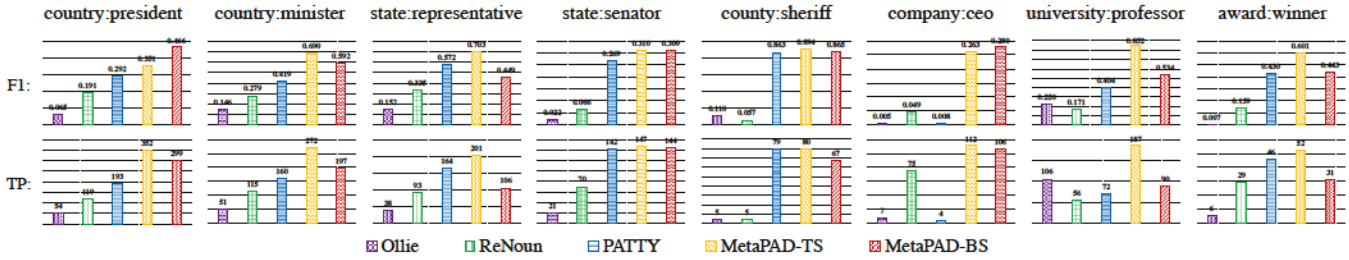


Figure 9: Performance comparisons on concrete attribute types in terms of F1 score and number of true positives.

(e.g., relatively 37.3% and 41.2% on F1 and AUC in the APR data). MetaPAD achieves 0.38–0.42 F1 score on discovering the EAV-tuples of new attributes like country:president and company:ceo. In the TAC KBP competition, the best F1 score of extracting values of traditional attributes like person:parent is only 0.3430 [16]. MetaPAD can achieve reasonable performance when working on the *new* attributes. MetaPAD also discovers the largest number of true tuples: on both datasets we discover more than a half of the labelled EAV-tuples (1,355/2,400 from APR and 1,111/2,090 from TWT).

2) The best of MetaPAD-T and MetaPAD-B that only segment but do not group meta patterns can outperform PATTY relatively by 19.4% (APR) and 78.5% (TWT) on F1 and by 27.6% (APR) and 115.3% (TWT) on AUC. Ollie parses individual sentences for relational tuples in which the relational phrases are often verbal expressions. So Ollie can hardly find exact attribute names from words or phrases of the relational phrases. ReNoun's *S-A-O* patterns like "*S*'s *A O*" require human annotations, use too general symbols, and bring too much noise in the extractions. PATTY's SOL patterns use entity types but ignore rich context around the entities and only keep the short dependency path. Our meta patten mining has context-aware segmentation with pattern quality assessment, which generates high-quality typed textual patterns from the rich context.

3) In MetaPAD-TS and MetaPAD-BS, we develop the modules of grouping synonymous patterns and adjusting the entity types for appropriate granularity. They improve the F1 score by 14.8% and 16.8% over MetaPAD-T and MetaPAD-B, respectively. We can see the number of true positives is significantly improved by aggregating extractions from different but synonymous meta patterns.

4) On the tweet data, most of the person, location, and organization entities are NOT able to be typed at a fine-grained level. So MetaPAD-T(S) works better than MetaPAD-B(S). The news data include a large number of entities of fine-grained types like the presidents and CEOs. So MetaPAD-B(S) works better.

Figure 9 shows the performance on different attribute types on APR. MetaPAD outperforms all the other methods on each type. When there are many ways (patterns) of expressing the attributes, such as country:president, company:ceo, and award:winner, MetaPAD gains more aggregated extractions from grouping the synonymous meta patterns. Our MetaPAD can generate more informative and complete patterns than PATTY's SOL patterns: for state:representative, state:senator, and county:sheriff that may not have many patterns, MetaPAD does not improve the performance much but it still works better than the baselines.

In our study, we find false EAV-tuple cases from quality meta patterns because the patterns are of high quality but not consistently reliable on specific attributes. For example, "president $PRESIDENT spoke to $COUNTRY people" is a quality pattern but it is only highly reliable to extract who-spoke-to-whom relations but less reliable to claim the person is the country's president. We can often see correct cases like (American, president, Barack Obama) from "President Barack Obama spoke to American people" but we can also find false cases like (Iraqi, president, Jimmy Carter) from "President Jimmy Carter spoke to Iraqi people". We would suggest to use either truth finding models or more syntatic and lexical features to find the trustworthy tuples in the future.

Table 6: Efficiency: time complexity is linear in corpus size.

	APR (news)	TWT (tweets)
File Size	199 MB	1.05 GB
#Meta Pattern	19,034	156,338
Time Cost	29 min	117 min

5.5 Results on Efficiency

The execution time experiments were all conducted on a machine with 20 cores of Intel(R) Xeon(R) CPU E5-2680 v2 @ 2.80GHz. Our framework is implemented in C++ for *meta-pattern segmentation* and in Python for *grouping synonymous meta patterns* and *adjusting type levels*. We set up 10 threads for MetaPAD as well as all baseline methods. Table 6 presents the efficiency performance of MetaPAD on the datasets: both the number of meta patterns and time complexity are linear to the corpus size. Specifically, for the 31G tweet data, MetaPAD takes less than 2 hours, while PATTY that requires Stanford parser takes 7.3 hours, and Ollie takes 28.4 hours. Note that for the smaller news data that have many long sentences, PATTY takes even more time, 10.1 hours.

6 CONCLUSIONS

In this work, we proposed a novel typed textual pattern structure, called *meta pattern*, which is extened to a frequent, complete, informative, and precise subsequence pattern in certain context, compared with the SOL pattern. We developed an efficient framework, MetaPAD, to discover the meta patterns from massive corpora with three techniques, including (1) a context-aware segmentation method to carefully determine the boundaries of the patterns with a learnt pattern quality assessment function, which avoids costly dependency parsing and generates high-quality patterns, (2) a clustering method to group synonymous meta patterns with integrated information of types, context, and instances, and (3) top-down and bottom-up schemes to adjust the levels of entity types in the meta patterns by examining the type distributions of entities in the

instances. Experiments demonstrated that MetaPAD efficiently discovered a large collection of high-quality typed textual patterns to facilitate challenging NLP tasks like tuple information extraction.

7 ACKNOWLEDGEMENTS

Research was sponsored in part by the U.S. Army Research Lab. under Cooperative Agreement No. W911NF-09-2-0053 (NSCTA), National Science Foundation IIS-1320617 and IIS 16-18481, and grant 1U54GM114838 awarded by NIGMS through funds provided by the trans-NIH Big Data to Knowledge (BD2K) initiative (www.bd2k.nih.gov).

Research was sponsored by the Army Research Laboratory and was accomplished under Cooperative Agreement Number W911NF-09-2-0053 (the ARL Network Science CTA). The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the Army Research Laboratory or the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation here on.

This research was supported by grant 1U54GM114838 awarded by NIGMS through funds provided by the trans-NIH Big Data to Knowledge (BD2K) initiative (www.bd2k.nih.gov).

REFERENCES

[1] Rakesh Agrawal and Ramakrishnan Srikant. 1995. Mining sequential patterns. In *ICDE*. 3–14.

[2] Gabor Angeli, Melvin Johnson Premkumar, and Christopher D Manning. 2015. Leveraging linguistic structure for open domain information extraction. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics (ACL 2015)*.

[3] Sören Auer, Christian Bizer, Georgi Kobilarov, Jens Lehmann, Richard Cyganiak, and Zachary Ives. 2007. Dbpedia: a nucleus for a web of open data. In *The semantic web*. 722–735.

[4] Michele Banko, Michael J Cafarella, Stephen Soderland, Matthew Broadhead, and Oren Etzioni. 2007. Open information extraction from the web.. In *IJCAI*, Vol. 7. 2670–2676.

[5] Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. 2008. Freebase: a collaboratively created graph database for structuring human knowledge. In *SIGMOD*. 1247–1250.

[6] Leo Breiman. 2001. Random forests. *Machine learning* 45, 1 (2001), 5–32.

[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] Zhiyuan Chen, Arjun Mukherjee, and Bing Liu. 2014. Aspect extraction with automated prior knowledge learning. In *ACL*.

[9] 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.

[10] Anthony Fader, Stephen Soderland, and Oren Etzioni. 2011. Identifying relations for open information extraction. In *EMNLP*. 1535–1545.

[11] Rayid Ghani, Katharina Probst, Yan Liu, Marko Krema, and Andrew Fano. 2006. Text mining for product attribute extraction. *SIGKDD Explorations* 8, 1 (2006), 41–48.

[12] Rahul Gupta, Alon Halevy, Xuezhi Wang, Steven Euijong Whang, and Fei Wu. 2014. Biperpedia: an ontology for search applications. *PVLDB* 7, 7 (2014), 505–516.

[13] Alon Halevy, Natalya Noy, Sunita Sarawagi, Steven Euijong Whang, and Xiao Yu. 2016. Discovering structure in the universe of attribute names. In *Proceedings of the 25th International Conference on World Wide Web*. International World Wide Web Conferences Steering Committee, 939–949.

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

[15] Minqing Hu and Bing Liu. 2004. Mining and summarizing customer reviews. In *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 168–177.

[16] Heng Ji, Ralph Grishman, Hoa Trang Dang, Kira Griffitt, and Joe Ellis. 2010. Overview of the TAC 2010 knowledge base population track. In *Third Text Analysis Conference (TAC)*, Vol. 3. 3–3.

[17] Meng Jiang, Peng Cui, Alex Beutel, Christos Faloutsos, and Shiqiang Yang. 2016. Inferring lockstep behavior from connectivity pattern in large graphs. *Knowledge and Information Systems* 48, 2 (2016), 399–428.

[18] Meng Jiang, Christos Faloutsos, and Jiawei Han. 2016. CatchTartan: Representing and Summarizing Dynamic Multicontextual Behaviors. In *Proceedings of the 22rd ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM.

[19] Anitha Kannan, Inmar E Givoni, Rakesh Agrawal, and Ariel Fuxman. 2011. Matching unstructured product offers to structured product specifications. In *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 404–412.

[20] Xiao Ling and Daniel S Weld. 2012. Fine-grained entity recognition. In *AAAI*.

[21] Jialu Liu, Jingbo Shang, Chi Wang, Xiang Ren, and Jiawei Han. 2015. Mining quality phrases from massive text corpora. In *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data*. ACM, 1729–1744.

[22] Christopher D Manning, Mihai Surdeanu, John Bauer, Jenny Rose Finkel, Steven Bethard, and David McClosky. 2014. The stanford corenlp natural language processing toolkit.. In *ACL (System Demonstrations)*. 55–60.

[23] Ryan McDonald, Koby Crammer, and Fernando Pereira. 2005. Online large-margin training of dependency parsers. In *Proceedings of the 43rd annual meeting on association for computational linguistics*. Association for Computational Linguistics, 91–98.

[24] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*. 3111–3119.

[25] Thahir P Mohamed, Estevam R Hruschka Jr, and Tom M Mitchell. 2011. Discovering relations between noun categories. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, 1447–1455.

[26] David Nadeau and Satoshi Sekine. 2007. A survey of named entity recognition and classification. *Lingvisticae Investigationes* 30, 1 (2007), 3–26.

[27] Ndapandula Nakashole, Tomasz Tylenda, and Gerhard Weikum. 2013. Fine-grained semantic typing of emerging entities. In *ACL*. 1488–1497.

[28] Ndapandula Nakashole, Gerhard Weikum, and Fabian Suchanek. 2012. PATTY: a taxonomy of relational patterns with semantic types. In *EMNLP*. 1135–1145.

[29] Vivi Nastase, Michael Strube, Benjamin Börschinger, Cäcilia Zirn, and Anas Elghafari. 2010. WikiNet: a very large scale multi-lingual concept network. In *LREC*.

[30] Marius Pasca and Benjamin Van Durme. 2008. Weakly-supervised acquisition of open-domain classes and class attributes from web documents and query logs.. In *ACL*. 19–27.

[31] Jian Pei, Jiawei Han, Behzad Mortazavi-Asl, Jianyong Wang, Helen Pinto, Qiming Chen, Umeshwar Dayal, and Mei-Chun Hsu. 2004. Mining sequential patterns by pattern-growth: The prefixspan approach. *TKDE* 16, 11 (2004), 1424–1440.

[32] Katharina Probst, Rayid Ghani, Marko Krema, Andrew Fano, and Yan Liu. 2007. Semififftsupervised learning of attributefifftvalue pairs from product descriptions. In *AAAI*.

[33] Sujith Ravi and Marius Paşca. 2008. Using structured text for large-scale attribute extraction. In *Proceedings of the 17th ACM conference on Information and knowledge management*. ACM, 1183–1192.

[34] Xiang Ren, Ahmed El-Kishky, Chi Wang, Fangbo Tao, Clare R Voss, and Jiawei Han. 2015. Clustype: Effective entity recognition and typing by relation phrase-based clustering. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 995–1004.

[35] 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. In *KDD*.

[36] Michael Schmitz, Robert Bart, Stephen Soderland, Oren Etzioni, and others. 2012. Open language learning for information extraction. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*. Association for Computational Linguistics, 523–534.

[37] Jingbo Shang, Jialu Liu, Meng Jiang, Xiang Ren, Clare R Voss, and Jiawei Han. 2017. Automated Phrase Mining from Massive Text Corpora. *arXiv preprint arXiv:1702.04457* (2017).

[38] 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.

[39] Fei Wu and Daniel S Weld. 2010. Open information extraction using Wikipedia. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, 118–127.

[40] Mohamed Yahya, Steven Whang, Rahul Gupta, and Alon Y Halevy. 2014. ReNoun: fact extraction for nominal attributes. In *EMNLP*. 325–335.

[41] Dian Yu and Heng Ji. 2016. Unsupervised person slot filling based on graph mining. In *ACL*.

[42] Ning Zhong, Yuefeng Li, and Sheng-Tang Wu. 2012. Effective pattern discovery for text mining. *IEEE transactions on knowledge and data engineering* 24, 1 (2012), 30–44.