Unsupervised Person Slot Filling based on Graph Mining

Dian Yu **Heng Ji**
Computer Science Department
Rensselaer Polytechnic Institute
Troy, NY 12180, USA
{yud2,jih}@rpi.edu

Abstract

Slot filling aims to extract the values (*slot fillers*) of specific attributes (*slots types*) for a given entity (*query*) from a large-scale corpus. Slot filling remains very challenging over the past seven years. We propose a simple yet effective unsupervised approach to extract slot fillers based on the following two observations: (1) a trigger is usually a salient node relative to the query and filler nodes in the dependency graph of a context sentence; (2) a relation is likely to exist if the query and candidate filler nodes are strongly connected by a relation-specific trigger. Thus we design a graph-based algorithm to automatically identify triggers based on personalized PageRank and Affinity Propagation for a given (*query*, *filler*) pair and then label the slot type based on the identified triggers. Our approach achieves 11.6%-25% higher F-score over state-of-the-art English slot filling methods. Our experiments also demonstrate that as long as a few trigger seeds, name tagging and dependency parsing capabilities exist, this approach can be quickly adapted to any language and new slot types. Our promising results on Chinese slot filling can serve as a new benchmark.

1 Introduction

The goal of the Text Analysis Conference Knowledge Base Population (TAC-KBP) Slot Filling (SF) task (McNamee and Dang, 2009; Ji et al., 2010; Ji et al., 2011; Surdeanu and Ji, 2014) is to extract the values (***fillers***) of specific attributes (***slot types***) for a given entity (***query***) from a large-scale corpus and provide justification sentences to support these slot fillers. KBP defines 25 slot types for persons (*e.g.*, *spouse*) and 16 slots for organizations (*e.g.*, *founder*). For example, given a person query "*Dominich Dunne*" and slot type *spouse*, a SF system may extract a slot filler "*Ellen Griffin*" and its justification sentence $E1$ as shown in Figure 1.

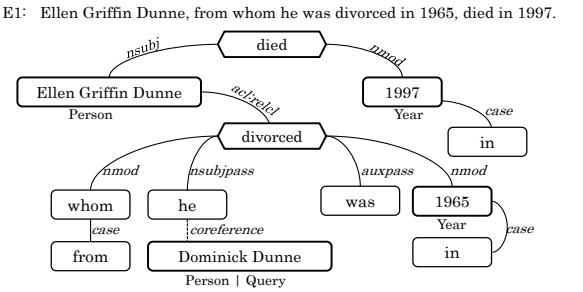

Figure 1: Extended dependency tree for E1.

Slot filling remains a very challenging task. The two most successful state-of-the-art techniques are as follows.

(1) Supervised classification. Considering any pair of query and candidate slot filler as an instance, these approaches train a classifier from manually labeled data through active learning (Angeli et al., 2014b) or noisy labeled data through distant supervision (Angeli et al., 2014a; Surdeanu et al., 2010) to predict the existence of a specific relation between them.

(2) Pattern matching. These approaches extract and generalize lexical and syntactic patterns automatically or semi-automatically (Sun et al., 2011; Li et al., 2012; Yu et al., 2013; Hong et al., 2014). They usually suffer from low recall due to numerous different ways to express a certain relation type (Surdeanu and Ji, 2014). For example, none of the top-ranked patterns (Li et al., 2012) based on dependency paths in Table 1 can capture the *spouse* slot in *E1*.

Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, pages 44–53,
Berlin, Germany, August 7-12, 2016.

Query	poss^{-1}	Slot Filler
Query	poss^{-1} [**wife**-**widow**-**husband**] appos	Slot Filler
Query	nsubj^{-1} **married** dobj	Slot Filler
Query	appos **wife** prep_of	Slot Filler
Query	nsubjpass^{-1} **survived** agent	Slot Filler

Table 1: Dependency patterns for slot *spouse*.

Both of the previous methods have poor portability to a new language or a new slot type. Furthermore, both methods focus on the flat relation representation between the query and the candidate slot filler, while ignoring the global graph structure among them and other facts in the context.

When multiple facts about a person entity are presented in a sentence, the author (e.g., a news reporter or a discussion forum poster) often uses explicit *trigger* words or phrases to indicate their relations with the entity. As a result, these interdependent facts and query entities are strongly connected via syntactic or semantic relations.

Many slot types, especially when the queries are person entities, are indicated by such *triggers*. We call these slots ***trigger-driven*** slots. In this paper, we define a ***trigger*** as the smallest extent of a text which most clearly indicates a slot type. For example, in *E1*, "*divorced*" is a trigger for *spouse* while "*died*" is a trigger for death-related slots.

Considering the limitations of previous flat representations for the relations between a query (Q) and a candidate slot filler (F), we focus on analyzing the whole dependency tree structure that connects Q, F and other semantically related words or phrases in each context sentence. Our main observation is that there often exists a trigger word (T) which plays an important role in connecting Q and F in the dependency tree for trigger-driven slots. From the extended dependency tree shown in Figure 1, we can clearly see that "*divorced*" is most strongly connected to the query mention ("*he*") and the slot filler ("*Ellen Griffin Dunne*"). Therefore we can consider it as a trigger word which explicitly indicates a particular slot type.

Based on these observations, we propose a novel and effective **unsupervised graph mining** approach for person slot filling by deeply exploring the structures of dependency trees. It consists of the following three steps:

- **Step 1 - Candidate Relation Identification:** Construct an extended dependency tree for each sentence including any mention referring to the query entity. Identify candidate slot fillers based on slot type constraints (*e.g.*, the *spouse* fillers are limited to person entities) (Section 2).
- **Step 2 - Trigger Identification:** Measure the importance of each node in the extended dependency tree relative to Q and F, rank them and select the most important ones as the trigger set (Section 3).
- **Step 3 - Slot Typing:** For any given new slot type, automatically expand a few trigger seeds using the Paraphrase Database (Ganitkevitch et al., 2013). Then we use the expanded trigger set to label the slot types of identified triggers (Section 4).

This framework only requires name tagging and dependency parsing as pre-processing, and a few trigger seeds as input, and thus it can be easily adapted to a new language or a new slot type. Experiments on English and Chinese demonstrate that our approach dramatically advances state-of-the-art results for both pre-defined KBP slot types and new slot types.

2 Candidate Relation Identification

We first present how to build an extended dependency graph for each evidence sentence (Section 2.1) and generate query and filler candidate mentions (Section 2.2).

2.1 Extended Dependency Tree Construction

Given a sentence containing N words, we construct an undirected graph $G = (V, E)$, where $V = \{v_1, \ldots, v_N\}$ represents the words in a sentence, E is an edge set, associated with each edge e_{ij} representing a dependency relation between v_i and v_j. We first apply a dependency parser to generate basic uncollapsed dependencies by ignoring the direction of edges. Figure 1 shows the dependency tree built from the example sentence. In addition, we annotate an entity, time or value mention node with its type. For example, in Figure 1, "*Ellen Griffin Dunne*" is annotated as a person, and "*1997*" is annotated as a year. Finally we perform co-reference resolution, which introduces implicit links between nodes that refer to the same entity. We replace any nominal or pronominal entity mention with its coreferential name mention. For example, "*he*" is replaced by "*Dominick Dunne*" in Figure 1. Formally, an extended dependency tree is an annotated tree of entity mentions, phrases and their links.

2.2 Query Mention and Filler Candidate Identification

Given a query q and a set of relevant documents, we construct a dependency tree for each sentence. We identify a person entity e as a query mention if e matches the last name of q or e shares two or more tokens with q. For example, "*he/Dominick Dunne*" in Figure 1 is identified as a mention referring to the query *Dominick Dunne*. For each sentence which contains at least one query mention, we regard all other entities, values and time expressions as candidate fillers and generate a set of entity pairs (q, f), where q is a query mention, and f is a candidate filler. In Example $E1$, we can extract three entity pairs (*i.e.*, {*Dominick Dunne*} $\times$ {*Ellen Griffin Dunne*, 1997, 1965}). For each entity pair, we represent the query mention and the filler candidate as two sets of nodes Q and F respectively, where $Q, F \subseteq V$.

3 Trigger Identification

In this section, we proceed to introduce an unsupervised graph-based method to identify triggers for each query and candidate filler pair. We rank all trigger candidates (Section 3.1) and then keep the top ones as the trigger set (Section 3.2).

3.1 Trigger Candidate Ranking

As we have discussed in Section 1, we can consider trigger identification problem as finding the important nodes relative to Q and F in G. Algorithms such as Pagerank (Page et al., 1999) are designed to compute the global importance of each node relative to all other nodes in a graph. By redefining the importance according to our preference toward F and Q, we can extend PageRank to generate relative importance scores.

We use the random surfer model (Page et al., 1999) to explain our motivation. Suppose a random surfer keeps visiting adjacent nodes in G at random. The expected percentage of surfers visiting each node converges to the PageRank score. We extend PageRank by introducing a "back probability" β to determine how often surfers jump back to the ***preferred nodes*** (*i.e.*, Q or F) so that the converged score can be used to estimate the relative probability of visiting these preferred nodes.

Given G and a set of preferred nodes R where $R \subseteq V$, we denote the relative importance for all $v \in V$ with respect to R as $I(v \mid R)$, following the work of White and Smyth (2003).

For a node v_k, we denote $N(k)$ as the set of neighbors of v_k. We use $\pi(k)$, the k-th component of the vector $\boldsymbol{\pi}$, to denote the stationary distribution of v_k where $1 \leq k \leq |V|$. We define a preference vector $\boldsymbol{p_R} = \{p_1, ..., p_{|V|}\}$ such that the probabilities sum to 1, and p_k denotes the relative importance attached to v_k. p_k is set to $1/|R|$ for $v_k \in R$, otherwise 0. Let $\boldsymbol{A}$ be the matrix corresponding to the graph G where $A_{jk} = 1/|N(k)|$ and $A_{jk} = 0$ otherwise.

For a given $\boldsymbol{p_R}$, we can obtain the personalized PageRank equation (Jeh and Widom, 2003):

$$\boldsymbol{\pi} = (1 - \beta)\boldsymbol{A\pi} + \beta\boldsymbol{p_R} \quad (1)$$

where $\beta \in [0, 1]$ determines how often surfers jump back to the nodes in R. We set $\beta = 0.3$ in our experiment. The solution π to Equation 1 is a steady-state importance distribution induced by $\boldsymbol{p_R}$. Based on a theorem of Markov Theory, a solution π with $\sum_{k=1}^{|V|} \pi(k) = 1$ always exists and is unique (Motwani and Raghavan, 1996).

We define relative importance scores based on the personalized ranks described above, *i.e.*, $I(v \mid R) = \pi(v)$ after convergence, and we compute the importance scores for all the nodes in V relative to Q and F respectively.

A query mention in a sentence is more likely to be involved in multiple relations while a filler is usually associated with only one slot type. Therefore we combine two relative importance scores by assigning a higher priority to $I(v \mid F)$ as follows.

$$\mathcal{I}(v \mid \{Q, F\}) = I(v \mid F) + I(v \mid F) \cdot I(v \mid Q) \quad (2)$$

We discard a trigger candidate if it is (or part of) an entity which can only act as a query or a slot filler. We assume a trigger can only be a noun, verb, adjective, adverb or preposition. In addition, verbs, nouns and adjectives are more informative to be triggers. Thus, we remove any trigger candidate v if it has a higher $\mathcal{I}(v \mid \{Q, F\})$ than the first top-ranked verb/noun/adjective trigger candidate.

For example, we rank the candidate triggers based on the query and slot filler pair ("*Dominick Dunne*", "*Ellen Griffin Dunne*") as shown in Figure 2.

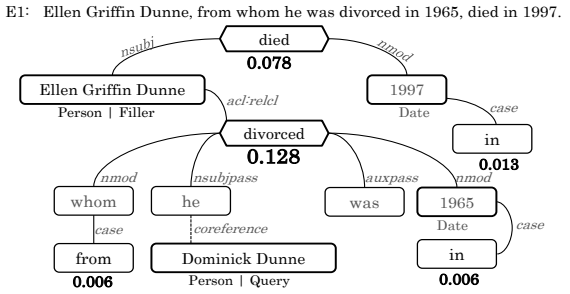

Figure 2: Importance scores of trigger candidates relative to query and filler in E1.

3.2 Trigger Candidate Selection

Given Q and F, we can obtain a relative importance score $\mathcal{I}(v \mid \{Q, F\})$ for each candidate trigger node v in V as shown in Section 3.1. We denote the set of trigger candidates as $T = \{t_1, \cdots, t_n\}$ where $n \leq |V|$.

Since a relation can be indicated by a single trigger word, a trigger phrase or even multiple non-adjacent trigger words, it is difficult to set a single threshold even for one slot type. Instead, we aim to automatically classify top ranked candidates into one group (*i.e.*, a trigger set) so that they all have similar higher scores compared to other candidates.

Therefore, we define this problem as a clustering task. We mainly consider clustering algorithms which do not require pre-specified number of clusters.

We apply the affinity propagation approach to take as input a collection of real-valued similarity scores between pairs of candidate triggers. Real-valued ***messages*** are exchanged between candidate triggers until a high-quality set of ***exemplars*** (centers of clusters), and corresponding clusters gradually emerges (Frey and Dueck, 2007).

There are two kinds of messages exchanged between candidate triggers: one is called ***responsibility*** $\gamma(i, j)$, sent from t_i to a candidate exemplar t_j; the other is ***availability*** $\alpha(i, j)$, sent from the candidate exemplar t_j to t_i.

The calculation of each procedure iterates until convergence. To begin with, the availabilities are initialized to zero: $\alpha(i, j) = 0$. Then the responsibilities are computed using the following rule:

$$\gamma(i, j) \leftarrow s(i, j) - \max_{j's.t.j' \neq j} \{\alpha(i, j') + s(i, j')\} \quad (3)$$

where the similarity score $s(i, j)$ indicates how well t_j is suited to be the exemplar for t_i. Whereas the above responsibility update lets all candidate exemplars compete for the ownership of a trigger candidate t_i, the following availability update gathers evidence from trigger candidates as to whether each candidate exemplar would make a good exemplar:

$$\alpha(i, j) \leftarrow \min \Big\{0, \gamma(j, j) + \sum_{i's.t.i' \notin \{i,j\}} \max\{0, \gamma(i', j)\}\Big\} \quad (4)$$

Given T, we can generate an $n \times n$ affinity matrix $\boldsymbol{M}$ which serves as the input of the affinity propagation. M_{ij} represents the negative squared difference in relative importance score between t_i and t_j (Equation 5).

$$M_{ij} = -(\mathcal{I}(i \mid \{Q, F\}) - \mathcal{I}(j \mid \{Q, F\}))^2 \quad (5)$$

We compute the average importance score for all the clusters after convergence and keep the one with the highest average score as the trigger set. For example, given the query and slot filler pair in Figure 3, we obtain trigger candidates $T = \{died, divorced, from, in, in\}$ and their corresponding relative importance scores. After the above clustering, we obtain three clusters and choose the cluster $\{divorced\}$ with the highest average relative importance score (0.128) as the trigger set.

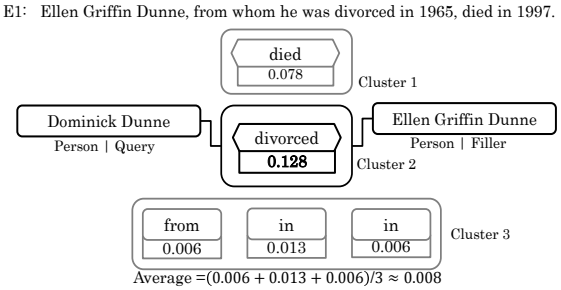

Figure 3: Trigger candidate filtering for E1.

4 Slot Type Labeling

In this section, we will introduce how to label the slot type for an identified relation tuple (Q, T, F). The simplest solution is to match T against existing trigger gazetteers for certain types of slots. For

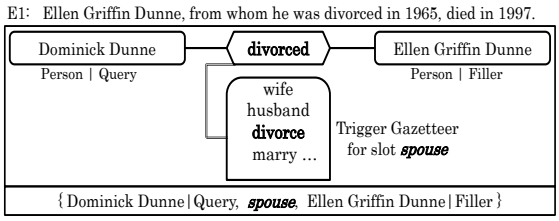

Figure 4: Example of slot type labeling.

example, Figure 4 shows how we label the relation as a *spouse* slot type.

In fact, some trigger gazetteers have already been constructed by previous work such as (Yu et al., 2015). However, manual construction of these triggers heavily rely upon labeled training data and high-quality patterns, which would be unavailable for a new language or a new slot type.

Inspired by the trigger-based event extraction work (Bronstein et al., 2015), we propose to extract trigger seeds from the slot filling annotation guideline [1] and then expand them by paraphrasing techniques. For each slot type we manually select two trigger seeds from the guideline and then use the Paraphrase Database (PPDB) (Ganitkevitch et al., 2013; Pavlick et al., 2015) to expand these seeds. Specifically, we select top-20 lexical paraphrases based on similarity scores as our new triggers for each slot type. Some examples are shown in Table 2.

Seeds	Slot Types	Expanded Triggers
assassinate	*death*	kill, die, slay, murder
graduate	*schools*	PhD, supervisor, diploma
sister	*siblings*	twin, half-brother, sibling
marriage	*spouse*	married, spouse, matrimony

Table 2: PPDB-based trigger expansion examples.

5 Filler Validation

After we label each relation tuple, we perform the following validation steps to filter noise and remove redundancy. For many slot types, there are some specific constraints on entity types of slot fillers defined in the task specification. For example, *employee_or_member_of* fillers should be either organizations or geopolitical entities, while family slots (*e.g.*, *spouse* and *children*) expect person entities. We apply these constraints to further validate all relation tuples.

[1] http://www.nist.gov/tac/2015/KBP/ColdStart/guidelines/TAC_KBP_2015_Slot_Descriptions_V1.0.pdf

Moreover, ***single-value*** slots can only have a single filler (*e.g.*, *date_of_birth*), while ***list-value*** slots can take multiple fillers (*e.g.*, *cities_of_residence*). However, we might extract conflicting relation tuples from multiple sentences and sources. For each relation tuple, it can also be extracted from multiple sentences, and thus it may receive multiple relative importance scores. We aim to keep the most reliable relation tuple for a single-value slot.

For a single-value slot, suppose we have a collection of relation tuples R which share the same query. Given $r \in R$ with a set of relative importance scores $\mathcal{I} = \{i_1, i_2, \cdots, i_n\}$, we can regard the average score of $\mathcal{I}$ as the credibility score of r. The reason is that the higher the relative importance score, the more likely the tuple is to be correct. In our experiments, we use the weighted arithmetic mean as follows so that higher scores can contribute more to the final average:

$$\bar{i} = \frac{\sum_{k=1}^{n} w_k \cdot i_k}{\sum_{k=1}^{n} w_k} \quad (6)$$

where w_k denotes the non-negative weight of i_k. When we regard the weight w_k equal to the score i_k, Equation 6 can be simplified as:

$$\bar{i} = \frac{\sum_{k=1}^{n} w_k^2}{\sum_{k=1}^{n} w_k} \quad (7)$$

We calculate the weighted mean $\bar{i}$ for each $r \in R$ and keep the relation tuple with the highest $\bar{i}$.

6 Experiments

6.1 Data and Scoring Metric

In order to evaluate the quality of our proposed framework and its portability to a new language, we use TAC-KBP2013 English Slot Filling (ESF), TAC-KBP 2015 English Cold Start Slot Filling (CSSF) and TAC-KBP2015 Chinese Slot Filling (CSF) data sets for which we can compare with the ground truth and state-of-the-art results reported in previous work. The source collection includes news documents, web blogs and discussion forum posts. In ESF there are 50 person queries and on average 20 relevant documents per query; while in CSF there are 51 person queries, and on average 5 relevant documents per query.

Slot Type	Our Approach	Roth'13	Angeli'14
siblings	**62.9**	48.0	40
other_family	**42.4**	11.8	0
spouse	58.7	40.0	**66**
children	**66.7**	27.3	27
parents	43.1	**47.8**	39
schools_attended	**81.4**	30.2	60
date_of_birth	87.0	60.0	**92**
date_of_death	**73.2**	3.2	48
state_of_birth	**55.6**	30.8	17
state_of_death	**88.2**	53.3	0
city_of_birth	**70.0**	64.0	25
city_of_death	72.7	**73.7**	30
country_of_birth	**75.0**	0.0	0
country_of_death	**70.0**	46.2	18
states_of_residence	**57.1**	25.6	12
cities_of_res.	**61.4**	38.8	38
countries_of_res.	**45.7**	20.0	41
employee_of	**43.8**	18.5	38
Overall	**57.4**	32.3	–

Table 3: English Slot Filling F_1 (%) (KBP2013 SF data set).

Slot Type	Our Approach	Angeli'15
siblings	**48.0**	26.1
other_family	0.0	**33.3**
spouse	14.3	**15.4**
children	**72.8**	0.0
parents	**25.0**	14.3
schools_attended	**63.6**	42.1
date_of_birth	0.0	**80.0**
date_of_death	**44.0**	0.0
state_of_birth	0.0	**33.3**
state_of_death	0.0	**15.4**
city_of_birth	0.0	**85.7**
city_of_death	0.0	0.0
country_of_birth	0.0	**66.7**
country_of_death	**100.0**	0.0
states_of_residence	0.0	0.0
cities_of_res.	0.0	**50.0**
countries_of_res.	0.0	0.0
employee_of	**60.0**	26.7
Overall	**39.2**	27.6

Table 4: English Cold Start Slot Filling F_1 (%) (KBP2015 CSSF data set).

We only test our method on 18 trigger-driven person slot types shown in Table 3. Some other slot types (*e.g.*, *age*, *origin*, *religion* and *title*) do not rely on lexical triggers in most cases; instead the query mention and the filler are usually adjacent or seperated by a comma. In addition, we do not deal with the two remaining trigger-driven person slot types (*i.e.*, *cause_of_death* and *charges*) since these slots often expect other types of concepts (*e.g.*, a disease or a crime phrase).

We use the official TAC-KBP slot filling evaluation scoring metrics: *Precision* (P), *Recall* (R) and *F-score* (F_1) (Ji et al., 2010) to evaluate our results.

6.2 English Slot Filling

We apply Stanford CoreNLP (Manning et al., 2014) for English part-of-speech (POS) tagging, name tagging, time expression extraction, dependency parsing and coreference resolution. In Table 3 we compare our approach with two state-of-the-art English slot filling methods: a distant supervision method (Roth et al., 2013) and a hybrid method that combines distant and partial supervision (Angeli et al., 2014b). Our method outperforms both methods dramatically. KBP2015 English cold start slot filling is a task which combines entity mention extraction and slot filling (Surdeanu and Ji, 2014). Based on the released evaluation queries from KBP2015 Cold Start Slot Filling, our approach achieves 39.2% overall F-score on 18 person trigger-driven slot types, which is significantly better than state-of-the-art (Angeli et al., 2015) on the same set of news documents (Table 4).

Compared to the previous work, our method discards a trigger-driven relation tuple if it is not supported by triggers. For example, "*Poland*" is mistakenly extracted as the country of residence of "*Mandelbrot*" by distant supervision (Roth et al., 2013) from the following sentence:

A professor emeritus at Yale University, ***Mandelbrot*** *was born in* ***Poland*** *but as a child moved with his family to* ***France*** *where he was educated.*

maybe because the relation tuple (*Mandelbrot*, *live_in*, *Poland*) indeed exists in external knowledge bases. Given the same entity pair, our method identifies "*born*" as the trigger word and labels the slot type as *country_of_birth*.

When there are several triggers indicating different slot types in a sentence, our approach performs better in associating each trigger with the filler it dominates by analyzing the whole dependency tree. For example, given a sentence:

Haig *is survived by his wife of 60 years, Patricia; his children Alexander, Brian and* ***Barbara****; eight grandchildren; and his brother, the Rev. Francis R. Haig.*

(*Haig*, *sibling*, *Barbara*) is the only relation tuple extracted from the above sentence by the previous method. Given the entity pair (*Haig*, *Barbara*), the relative importance score of "*children*" (0.1) is higher than the score of "*brother*" (0.003),

and "*children*" is kept as the only trigger candidate after clustering. Therefore, we extract the tuple (*Haig*, *children*, *Barbara*) instead. In addition, we successfully identify the missing fillers for other slot types: *spouse* (*Patricia*), *children* (*Alexander, Brian and Barbara*) and *siblings* (*Francis R. Haig*) by identifying their corresponding triggers.

In addition, flat relation representations fail to extract the correct relation (*i.e.*, *alternate_names*) between "*Dandy Don*" and "*Meredith*" since "*brother*" is close to both of them in the following sentence:

In high school and at Southern Methodist University, where, already known as ***Dandy Don*** *(a nickname bestowed on him by his brother) ,* ***Meredith*** *became an all-American.*

6.3 Adapting to New Slot Types

Our framework can also be easily adapted to new slot types. We evaluate it on three new person list-value slot types: *friends*, *colleagues* and *collaborators*.

We use "*friend*" as the slot-specific trigger for the slot *friends* and "*colleague*" for the slot *colleagues*. "*collaborate*", "*cooperate*" and "*partner*" are used to type the slot *collaborators*.

We manually annotate ground truth for evaluation. It is difficult to find all the correct fillers for a given query from millions of documents. Therefore, we only calculate precision. Experiments show we can achieve 56.3% for *friends*, 100% for *colleagues* and 60% for *collaborators* (examples shown in Table 5).

6.4 Impact of Trigger Mining

In Section 3.2, we keep top-ranked trigger candidates based on clustering rather than threshold tuning. We explore a range of thresholds for comparison, as shown in Figure 5. Our approach achieves 57.4% F-score, which is comparable to the highest F-score 58.1% obtained by threshold tuning.

We also measure the impact of the size of the trigger gazetteer. We already outperform state-of-the-art by using PPDB to expand triggers mined from guidelines as shown in Table 6. As the size of the trigger gazetteer increases, our method (marked with a $\star$) achieves better performance.

6.5 Chinese Slot Filling

As long as we have the following resources: (1) a POS tagger, (2) a name tagger, (3) a dependency parser and (4) slot-specific trigger gazetteers, we can apply the framework to a new language. Coreference resolution is optional.

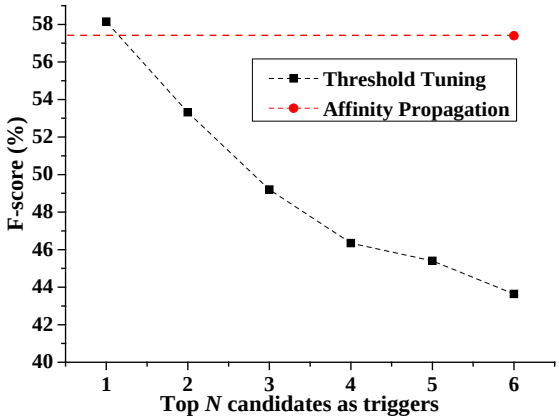

Figure 5: The effect of the number of trigger candidates on ESF.

Method	Size	F_1 (%)
State-of-the-art (Roth et al., 2013)	–	32.3
Guideline seeds*	20	27.3
Guideline seeds + PPDB expansion*	220	**38.9**
Manually Constructed Trigger Gazetteers*	7,463	**57.4**

Table 6: The effect of trigger gazetteers on ESF (size: the number of triggers).

We demonstrate the portability of our framework to Chinese since all the resources mentioned above are available. We apply Stanford CoreNLP (Manning et al., 2014) for Chinese POS tagging, name tagging (Wang et al., 2013) and dependency parsing (Levy and Manning, 2003). To explore the impact of the quality of annotation resources, we also use a Chinese language analysis tool: Language Technology Platform (LTP) (Che et al., 2010). We use the full set of Chinese trigger gazetteers published by Yu et al. (2015). Experimental results (Table 7) demonstrate that our approach can serve as a new and promising benchmark. As far as we know, there are no results available for comparison.

However, the performance of Chinese SF is heavily influenced by the relatively low performance of name tagging since our method returns an empty result if it fails to find any query metnion. About 20% and 16% queries cannot be recognized by CoreNLP and LTP respectively. One reason is that many Chinese names are also common words. For example, a buddhist monk's name "觉醒"(*wake*) is identified as a verb rather than a person entity.

Evidence Sentence	Slot Type	Query	Extracted Fillers
Many of ***his*** subjects were friends from his previous life , such as ***Elizabeth Taylor*** and ***Gloria Vanderbilt*** .	*friends*	Dominick Dunne	Gloria Vanderbilt; Elizabeth Taylor
Toby Keith hit an emotional note with a performance of "Cryin' For Me (Wayman's Song)," dedicated to ***his*** late friend, jazz artist and former basketball star ***Wayman Tisdale***, who died last May.	*friends*	Wayman Tisdale	Toby Keith
"I think all of her writing came from ***her*** heart," ***Michael Glaser***, a longtime colleague at St. Mary's and former Maryland poet laureate, said last week.	*colleagues*	Lucille Clifton	Michael Glaser
Cunningham has collaborated on two books: "Changes: Notes on Choreography," with Frances Starr, and "The Dancer and the Dance," with ***Jacqueline Lesschaeve***.	*collaborators*	Merce Cunningham	Jacqueline Lesschaeve

Table 5: Examples for new slot types.

A dependency parser is indispensable to produce reliable rankings of trigger candidates. Unfortunately, a high-quality parser for a new language is often not available because of language-specific features. For example, in Chinese a single sentence about a person's biography often contains more than five co-ordinated clauses, each of which includes a trigger. Therefore a dependency parser adapted from English often mistakenly identifies one of the triggers as a main predicate of the sentence.

In addition, Chinese is a very concise language. For example, a "[Person Name][Organization Suffix]"structure can indicate various different types of relations between the person name and the organization: "杨明牙医诊所"(*Yang Ming Clinic*) indicates ownership, "邵逸夫图书馆"(*Shao Yifu Library*) indicates sponsorship, "丰子恺研究中心"(*Feng Zikai Research Center*) indicates research theme, and "罗京治丧委员会"(*Luojing Commemoration Committee*) indicates commemoration. None of them includes an explicit trigger nor indicates employment relation. It requires more fine-grained dependency relation types to distinguish them.

Finally, compared to English, Chinese tends to have more variants for some types of triggers (e.g., there are at least 31 different titles for "*wife*"in Chinese). Some of them are implicit and require shallow inference. For example, "投奔"(*to seek shelter or asylum*) indicates a residence relation in most cases.

Slot Type	CoreNLP-based	LTP-based
siblings	40.0	**57.1**
other_family	**40.0**	0.0
spouse	40.0	**48.0**
children	19.0	**21.4**
parents	0.0	**25.0**
schools_attended	11.1	**17.1**
date_of_birth	**42.4**	0.0
date_of_death	**48.5**	0.0
state_of_birth	38.1	**52.2**
state_of_death	55.6	**70.0**
city_of_birth	**28.6**	26.7
city_of_death	33.3	**42.9**
country_of_birth	11.8	11.8
country_of_death	0.0	0.0
states_of_residence	**30.8**	29.6
cities_of_residence	27.3	**34.8**
country_of_residence	**6.5**	0.0
employee_of	31.0	**31.2**
Overall	**29.6**	28.3

Table 7: Chinese Slot Filling F_1 (%) (KBP2015 CSF data set).

7 Related Work

Besides the methods based on distant supervision (*e.g.*, (Surdeanu et al., 2010; Roth et al., 2013; Angeli et al., 2014b)) discussed in Section 6.2, pattern-based methods have also been proven to be effective in SF in the past years (Sun et al., 2011; Li et al., 2012; Yu et al., 2013). Dependency-based patterns achieve better performance since they can capture long-distance relations. Most of these approaches assume that a relation exists between Q and F if there is a dependency path connecting Q and F and all the words on the path are equally regarded as trigger candidates. We explore the complete graph structure of a sentence rather than chains/subgraphs as in previous work. Our previous research focused on identifying the relation between F and T by extracting filler candidates from the identified scope of a trigger (*e.g.*, (Yu et al., 2015)). We found that each slot-specific trigger has its own scope, and corresponding fillers seldom appear outside its scope. We did not compare with results from this previous approach which did not consider redundancy removal required in the official evaluations.

Soderland et al. (2013) built their SF system based on Open Information Extraction (IE) technology. Our method achieves much higher recall since dependency trees can capture the relations among query, slot filler and trigger in more complicated long sentences. In addition, our triggers are automatically labeled so that we do not need to design manual rules to classify relation phrases as in Open IE.

8 Conclusions and Future Work

In this paper, we demonstrate the importance of deep mining of dependency structures for slot filling. Our approach outperforms state-of-the-art and can be rapidly portable to a new language or a new slot type, as long as there exists capabilities of name tagging, POS tagging, dependency parsing and trigger gazetteers.

In the future we aim to label slot types based on contextual information as well as sentence structures instead of trigger gazetteers only. There are two primary reasons. First, a trigger can serve for multiple slot types. For example, slot *children* and its inverse slot *parents* share a subset of triggers. Second, a trigger word can have multiple different meanings. For example, a *sibling* trigger word "*sister*" can also represent a female member of a religious community. We attempt to combine multi-prototype approaches (*e.g.*, (Reisinger and Mooney, 2010)) to better disambiguate senses of trigger words.

Besides considering the cross-sentence conflicts, we also want to investigate the within-sentence conflicts caused by the competition of triggers. A trigger identified by our approach is the most important node in the dependency tree relative to the given entity pair. However, this trigger might be more important to another entity pair, which shares the same filler, in the same sentence. A promising solution is to rank all the entities in the sentence based on their importance relative to the identified trigger and the filler candidate.

Acknowledgement

We would like to thank Chris Callison-Burch for providing English and Chinese paraphrase resources. This work was supported by the DARPA LORELEI Program No. HR0011-15-C-0115, DARPA DEFT Program No. FA8750-13-2-0041, ARL NS-CTA No. W911NF-09-2-0053, NSF CAREER Award IIS-1523198. 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 U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation here on.

References

G. Angeli, S. Gupta, M. Jose, C. Manning, C. Ré, J. Tibshirani, J. Wu, S. Wu, and C. Zhang. 2014a. Stanford's 2014 slot filling systems. In *Proc. Text Analysis Conference (TAC 2014)*.

G. Angeli, J. Tibshirani, J. Wu, and C. Manning. 2014b. Combining distant and partial supervision for relation extraction. In *Proc. Empirical Methods on Natural Language Processing (EMNLP 2014)*.

G. Angeli, V. Zhong, D. Chen, J. Bauer, A. Chang, V. Spitkovsky, and C. Manning. 2015. Bootstrapped self training for knowledge base population. In *Proc. Text Analysis Conference (TAC 2015)*.

O. Bronstein, I. Dagan, Q. Li, H. Ji, and A. Frank. 2015. Seed-based event trigger labeling: How far can event descriptions get us? In *Proc. Association for Computational Linguistics (ACL 2015)*.

W. Che, Z. Li, and T. Liu. 2010. Ltp: A chinese language technology platform. In *Proc. Computational Linguistics (COLING 2010)*.

B. Frey and D. Dueck. 2007. Clustering by passing messages between data points. *science*.

J. Ganitkevitch, B. Van Durme, and C. Callison-Burch. 2013. PPDB: The paraphrase database. In *Proc. North American Chapter of the Association for Computational Linguistics - Human Language Technologies (NAACL-HLT 2013)*.

Y. Hong, X. Wang, Y. Chen, J. Wang, T. Zhang, J. Zheng, D. Yu, and Q. Li. 2014. Rpi blender tac-kbp2014 knowledge base population system. In *Proc. Text Analysis Conference (TAC 2014)*.

G. Jeh and J. Widom. 2003. Scaling personalized web search. In *Proc. World Wide Web (WWW 2003)*.

H. Ji, R. Grishman, H. Dang, K. Griffitt, and Joe Ellis. 2010. An overview of the tac2010 knowledge base population track. In *Proc. Text Analysis Conference (TAC 2010)*.

H. Ji, R. Grishman, and H. Dang. 2011. An overview of the tac2011 knowledge base population track. In *Proc. Text Analysis Conference (TAC 2011)*.

R. Levy and C. Manning. 2003. Is it harder to parse chinese, or the chinese treebank? In *Proc. Association for Computational Linguistics (ACL 2003)*.

Y. Li, S. Chen, Z. Zhou, J. Yin, H. Luo, L. Hong, W. Xu, G. Chen, and J. Guo. 2012. Pris at tac2012 kbp track. In *Proc. Text Analysis Conference (TAC 2012)*.

C. Manning, M. Surdeanu, J. Bauer, J. Finkel, S. Bethard, and D. McClosky. 2014. The Stanford CoreNLP natural language processing toolkit. In *Proc. Association for Computational Linguistics (ACL 2014)*.

P. McNamee and H. Dang. 2009. Overview of the tac 2009 knowledge base population track. In *Proc. Text Analysis Conference (TAC 2009)*.

R. Motwani and P. Raghavan. 1996. Randomized algorithms. *ACM Computing Surveys (CSUR)*.

L. Page, S. Brin, R. Motwani, and T. Winograd. 1999. The pagerank citation ranking: Bringing order to the web. Technical report, Stanford InfoLab.

E. Pavlick, P. Rastogi, J. Ganitkevitch, and C. Van Durme, B.and Callison-Burch. 2015. Ppdb 2.0: Better paraphrase ranking, fine-grained entailment relations, word embeddings, and style classification. In *Proc. Association for Computational Linguistics (ACL 2015)*.

J. Reisinger and R. Mooney. 2010. Multi-prototype vector-space models of word meaning. In *Proc. North American Chapter of the Association for Computational Linguistics - Human Language Technologies (NAACL-HLT 2010)*.

B. Roth, T. Barth, M. Wiegand, M. Singh, and D. Klakow. 2013. Effective slot filling based on shallow distant supervision methods. In *Proc. Text Analysis Conference (TAC 2013)*.

S. Soderland, J. Gilmer, R. Bart, O. Etzioni, and D. Weld. 2013. Open ie to kbp relations in 3 hours. In *Proc. Text Analysis Conference (TAC 2013)*.

A. Sun, R. Grishman, B. Min, and W. Xu. 2011. Nyu 2011 system for kbp slot filling. In *Proc. Text Analysis Conference (TAC 2011)*.

M. Surdeanu and H. Ji. 2014. Overview of the english slot filling track at the tac2014 knowledge base population evaluation. In *Proc. Text Analysis Conference (TAC 2014)*.

M. Surdeanu, D. McClosky, J. Tibshirani, J. Bauer, A. Chang, V. Spitkovsky, and C. Manning. 2010. A simple distant supervision approach for the tac-kbp slot filling task. In *Proc. Text Analysis Conference (TAC 2010)*.

M. Wang, W. Che, and C. Manning. 2013. Joint word alignment and bilingual named entity recognition using dual decomposition. In *Proc. Association for Computational Linguistics (ACL 2013)*.

S. White and P. Smyth. 2003. Algorithms for estimating relative importance in networks. In *Proc. Knowledge discovery and data mining (KDD 2003)*.

D. Yu, H. Li, T. Cassidy, Q. Li, H. Huang, Z. Chen, H. Ji, Y. Zhang, and D. Roth. 2013. Rpi-blender tac-kbp2013 knowledge base population system. In *Proc. Text Analysis Conference (TAC 2013)*.

D. Yu, H. Ji, S. Li, and C. Lin. 2015. Why read if you can scan: Scoping strategy for biographical fact extraction. In *Proc. North American Chapter of the Association for Computational Linguistics - Human Language Technologies (NAACL-HLT 2015)*.