OpenKN at TAC KBP 2016

Manling Li, Xinlei Chen, Yantao Jia, Yuanzhuo Wang, Xiaolong Jin, Zixuan Li, Juan Yao, Fan Yang, Yunqi Qiu, Jialin Su and Xueqi Cheng
CAS Key Laboratory of Network Data Science and Technology
Institute of Computing Technology
Chinese Academy of Science
jiayantao@ict.ac.cn

Abstract

This paper describes the system OpenKN which we established in the TAC KBP 2016. In TAC KBP 2016, we participated in one track: Cold Start KB Track. In order to complete the task, we developed a five-step system which solves the problem of building knowledge base from a document collection of unstructured text. These five steps to complete this task are document-processing, relation extraction, cross-document co-reference resolution, inference, and post-processing, where the relation extractor is the combination of four methods: rule-based pattern extractor, bootstrapping, OpenIE and an Implicit Relation Information Extractor.

1 Introduction

The goal of TAC KBP 2016 is to develop and evaluate technologies for building and populating knowledge bases (KBs) from unstructured text. It contains several tracks, and we participate in Cold Start Track (KB variant, English) this year.

The Cold Start KBP track builds a knowledge base from scratch using a given document collection and a predefined schema for the entities and relations that will compose the KB. Given a collection of documents, the Cold Start KB Construction system must find all PER, ORG, GPE, LOC, FAC entities in the collection, which are mentioned by named mentions or nominal mentions, and create a KB node for each entity as well as linking each name mention and each nominal mention to the correct entity node. Then, 41 slots are required to be filled for each entity node in the KB according to the given document collection.

For this purpose, we propose a system which includes five parts to finish this task: document-processing for preparing entities, relation extraction (regarded as slot filling, aims to extract relations by four methods), cross-document co-reference resolution to merge similar entities, inferring relations from already-known relations, and post-processing.

The paper is organized as follows. Section 2 describes the architecture of our system. The document processing module is explained in Section 3, including pre-processing, entity discovery and intra-document co-reference resolution. Section 4 describes the details of relation extractor, including the four methods and the combination strategy. And the cross-document co-reference resolution is introduced in Section 5, as well as inference module in Section 6 and post-processing module in Section7. Section 8 lists the evaluation results of the task. Finally, we conclude the paper in Section 9 and present related references.

2 System Architecture

Our proposed system contains five steps, i.e., document-processing, relation extraction, cross-document co-reference resolution, inference, and post-processing, as illustrated in Figure 1.

More specifically, in the document-processing step we extract all named mentions and nominal

mentions from the corpus and save their type and offset, as well as intra-document co-reference resolution included. Then the relations of entities are extracted from corpus (i.e. slot filling) in four ways: rule-based method, bootstrapping-based method, OpenIE-based method and implicit relation extractor. After this step, the task turns into co-reference resolution of the cross-document entities. It is implemented by linking entities to Wikipedia and nil clustering. What followed next is inference, which supplies the existing relations. Finally, we utilize the post-processing component to remove wrong results and format final results. In the following, we explain each step in detail.

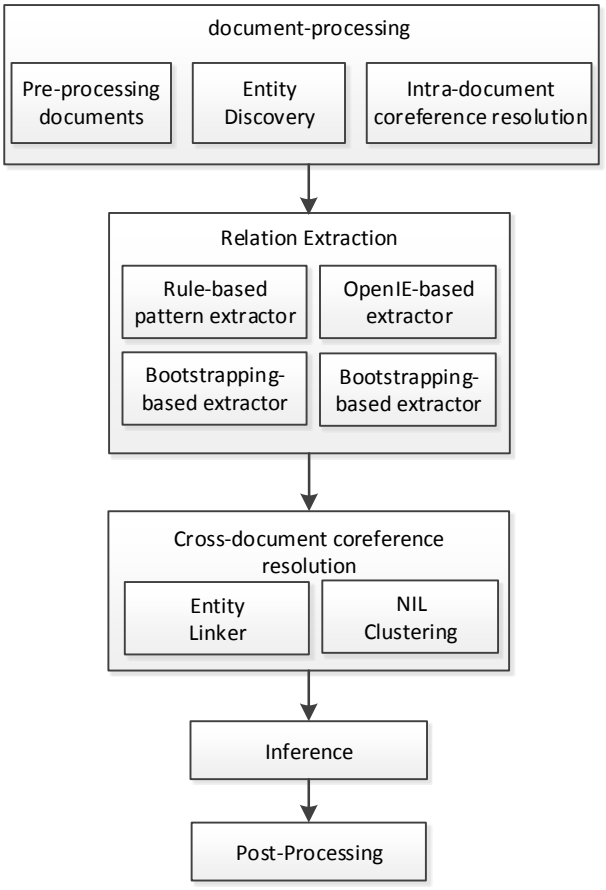

Figure 1 Architecture of the system

3 Document Processing

Document processing includes the pre-process of the corpus, entity discovery and intra-document co-reference resolution as following.

3.1 Pre-processing

The documents are pre-processed in three aspects:

1) It should be noted that according to the Entity Linking Query Development Guidelines, when we detect discussion forum threads or web documents, the entity mentions occurring between <quote></quote> tags are ignored. As a result, we simply replace these pieces of content by whitespaces to avoid the errors of offsets.

2) As the publish dates of documents are helpful to infer the slots concerning date (in most cases is death date, see details in section 2.4), we extract the date information of each document. For newswires, the date is article's publish date; For forums, the date is post's date.

3) The length of each document is validated according to the given doc-length file.

3.2 Entity Discovery

Since Cold Start tasks require to return nominal entity mentions apart from the named entity mentions, there are two parts in our entity discovery module, ie, the named entity discovery and the nominal entity discovery.

3.2.1 Named Entity Discovery

There are three steps to detect all entity mentions from the target document. Firstly, we use Stanford NER (Finkel et al., 2005) to extract person (PER), organization (ORG), geo-political entity (GEP), facility (FAC) and location (LOC). Secondly, we use data provided by the reference KB (LDC2014T16 and LDC2015E42) to supplement the results of NER. Thirdly, we use the regular expression to extract post authors as persons. We combine these results as the final entity mentions.

In addition, as the Cold Start tasks requires two more types of entities, ie, facility (FAC) and location (LOC), we re-trained a named entity tagger model by Stanford NER tool based on the corpus from ERE (LDC2015E78, LDC2015E68, LDC2015E29, LDC2014E31, LDC2014E114, LDC2013E64) and the training and evaluation data of trilingual EDL 2015 (LDC2016E38).

3.2.2 Nominal Entity Discovery

To detect nominal entity mentions, we trained a model by Stanford NER tool as well, and the data

are same as the data used to train the named entity recognizer in Section 3.2.1, but only extract the NOM mentions which have <head> information. In our experiments, it is found that the training data without <head> information will introduce noise. And we also trained a model to detect name and nominal mentions at the same time, whose performance is not quite good enough. We also detect the nominal mentions of PER entities following RPI system (Hong et al., 2015).

3.2.3 Intra-document Co-reference Resolution

There are five steps in our intra-document co-reference resolution. Firstly, the mentions whose name are the same are linked together. Secondly, a co-reference chain is generated by Stanford CoreNLP (Manning et al., 2014). Thirdly, we combine the two chains as the final co-reference chain. Forthly, the entity types of mentions in one chain should be unified into the entity type same as the majority. Fifthly, the canonical mention is selected by following standards:

a) the canonical mention should appear in the main body of the document, ie, in the content between <text> tags, to prevent from lack of contextual information.

b) the start offset of the canonical mention should be as small as possible.

4 Relation Extractor

The relation extraction step consists of four modules, namely, the rule-based pattern extractor module, the bootstrapping-based relation extractor module, the OpenIE-based relation extractor module and Implicit Relation Information Extractor module.

4.1 Rule-based Relation Extractor

The rule-based pattern extractor module is based on our previous TAC system (Chen X et al., 2015; Lin H et al., 2014) and it works as follows. It firstly describes the rule-based patterns in terms of the context free grammar with respect to the target slot of entities. Then these grammars are compiled under the cascaded finite-state transducers (CFT). With the aid of CFT, the slots are extracted under the collision detection phrase. More specifically, the rule-based patterns are firstly represented in the form of triplets (Heads, Arguments, Body). For example, to extract the slot per:age, we can define the pattern as follows:

AGE(person,age):(DIST_5,"_person{NAME}",
"_age{AGE}","year's old"),

where the parameter DIST_n is used to define the distance (i.e., n words) between the followed two arguments. In particular, when the sentences match two or more rules, in order to boost the extraction efficiency, we adopt the inversed index to store the mapping from arguments to their locations in the corresponding patterns.

4.2 Bootstrapping-based Relation Extractor

Apart from defining these rule-based patterns, we also consider the bootstrapping technique for the purpose of recursive extraction. This can be accomplished by training the data sets and tuning the parameters appearing in the body of the rules. For example, these parameters may include the distances between arguments, the order of arguments, the distances between arguments by analyzing the results of dependency parsing. In this system, the bootstrapping-based relation extractor is built based on ICE [the Integrated Customization Environment] (He et al., 2015), which is an information extraction customization tool.

We use ICE to bootstrap slot filling rules on the cold start 2015 source corpus (LDC2016E39) and cold start 2016 source corpus. The ICE bootstrapper firstly conduct NLP pre-processing on the corpus, including named entity recognition, dependency parsing, POS tagging, etc., so that all dependency patterns are generated at once. Then, for each slot, we provide ICE with two or three seed dependency patterns derived from all dependency paths, and the bootstrapper searches the most similar dependency patterns and returns them to the user for review, where the positive and negative patterns tagged by the user are then sent back to the ICE bootstrapper for the next iteration of bootstrapping patterns. Finally, the bootstrapper generated rules for 22 slots, and the rules extracted about twelve thousand fillers in total.

4.3 OpenIE-based Relation Extractor

OpenIE [Open Information Extraction] is another method we used to extract relations. We use OpenIE v4.0 (Mausam et al., 2012) on the cold

start 2016 source corpus, and it produces tuples of the form (arg1, rel, arg2) for the given sentence, where rel is the phrase related to arg1 and arg2 in the sentence, which may or may not align to the 41 slots of TAC. As a result, the main challenge is to align the rel to TAC slots as well as mapping the arg1 and arg2 to entities.

To this end, a more automated approach using embedding techniques is adopted, apart from the rule-based aligning approach following (Soderland et al., 2013, Jia et al. 2016). In order to better calculate the similarity of rels and TAC slots, the automated approach makes use of the low-dimensional embeddings of words in rels and the embeddings of TAC slots (Weston J et al., 2013), which are learned according to the constrains of the form:

$$Score(r,s) > 1 + \max_{s' \in \{S-s\}} Score(r,s'),$$

where r denotes the rel and s denotes the true target TAC slot that the rel should be align to. S denotes the set of all 41 TAC slots, and the $\{S-s\}$ denotes the set of other 40 TAC slots excluding s. The score function is defined to score the similarity of a rel r and a TAC slot s,

$$Score(r,s) = \boldsymbol{f}(r)^T \boldsymbol{s}$$

where the bold face denotes the embedding vector in the low-dimensional space $\mathbb{R}^d$, where d is the dimension of the embedding vector space. Here f is a function mapping words into $\mathbb{R}^d$, $\boldsymbol{f}(r) = W^T \emptyset(r)$, where $\boldsymbol{W}$ is the matrix of $\mathbb{R}^{|V|}$, containing all word embeddings $\boldsymbol{w}$ and V is the set of all words, $\emptyset(r)$ is the (sparse) binary representation of $\boldsymbol{r}$ ($\in \mathbb{R}^{|V|}$) indicating absence or presence of words, and $\boldsymbol{s}(\in \mathbb{R}^d)$ is the embedding of the TAC slot s.

In this setup, the embedding vectors are learned by minimizing a margin-based loss function,

$$L = \max(0, 1 - Score(r,s) + Score(r,s'))$$

namely, optimizing the hinge loss.

More precisely, there are three steps in the automated approach. Firstly, the embeddings of the words in rels and the embeddings of TAC slots are learned according to a margin-based loss function, based on the mapping information between rels and TAC slots. Secondly, given a rel r, the similarity of the rel r and TAC slots s are calculated by score function $Score(r,s)$, and the TAC slot with highest score which is bigger than a threshold is returned as the TAC slot that the rel align to. Thirdly, the arg1 and arg2 are mapped to the entities through the named and nominal mentions extracted in Section 3.

In order to determine the mapping information between rels and TAC slots, the training and evaluation data of previous cold start task (LDC2016E39) and slot filling task (LDC2015E46) are processed, where the rel extracted from the provenance are positive examples for the given slot, and negative examples for other slots.

4.4 Implicit Relation Information Extractor

We extract the implicit relations, ie, the relations which are not represented by explicit relation phrase like OpenIE, but are noun phrases or adjective phrases. For example, given a phrase "American journalist Ann", the triple (Ann, per:origin, America) should be extracted. These implicit relations are detected by IMPLIE (Soderland et al, 2015), which firstly extracts the features of sentence using Stanford NLP tools like the Part-Of-Speech tags (Wu and Weld, 2007, Toutanova et al. 2003) and the dependency parser (Marneffe and Manning, 2008), etc., and secondly extracts substring of the sentence, which contains both arg1 and arg2 of the relation, by dependency path rules, and outputs tuple (arg1, rel, arg2) finally.

We also link arg1 and arg2 to entities like Section 4.3, and link the rel to the TAC slots according to the corresponding entity types and keyword, as the rels in the output are fairly standard and there is no need to align through embedding-based method.

4.5 The Combination Strategy

The results of four methods are combined by simply taking the union. If these systems had different outputs for a functional relation, we calculate the confidence score by a linear-weighted method.

To be more precise, each method has different weight designed by the performance conducted on evaluation data of previous years, and the method with better performance has higher weight. For each triple (ie, entity1, slot, entity2) we extracted, the confidence score is the normalization result

based on the sum of the weights of the methods related to the triple.

For slots that admit only a single value (e.g., country_of_birth), we select the triple with highest confidence. For slots that can have more than one value (e.g, per:parents), we select the top 10 triples as the best set of values for the slot to avoid noises.

5 Cross-document Co-reference Resolution

To address the tagging of entities, the system employs two steps to cluster the cross-document entities across target documents. Firstly, it employs entity linker to link entities to Wikipedia, and the entities linked to the same Wikipedia
employs the hierarchical clustering method to cluster the entities in terms of the context similarity and name similarity between entity mentions. Secondly, the heuristic rule aiming to augment the cluster results is proposed.

We use acronym expansion matching in the document text. For example, PA is expanded to "Pennsylvania" state in several different documents in the training data set. Thus, they are mapped to the same identifier

5.1 Candidate entity generator

In order to reduce the time complexity of the linking process, a small set of candidate entities that may link to an entity mention detected from target documents should be generated in an appropriate manner. Namely, we regard the entity mention as a query to obtain the candidate set from the reference KB (i.e., Wikipedia). Specifically, we search the five fields, i.e., the page title, entity alias name, entity acronym and document text provided by the Wikipedia dump.

5.2 Entity linker

It takes two steps to generate the linking results. Firstly, coreference resolution is used to cluster the entity mentions that are referred as the same entity. Secondly, based on the cluster results, the system employs seven features to measure the similarity between the reference entity and an entity mention. These features are listed as follows:

1) Name similarity. Namely, the string similarity between the entity mention in the document text and the candidate entity in the reference KB.
2) Context similarity. We selected K words window surrounding an entity mention as its context, and compute the similarity between the entity mention and the candidate entity in the reference KB using the biterm model presented in (Yan et al., 2013), which is an extension to the commonly used cosine similarity.
3) Wikipedia redirect page with identical titles. Entity mention in the document matches the candidate entity with page referred by Wikipedia redirect page with identical titles.
4) Acronym matching, which indicates whether the entity mention is an acronym of the candidate entity and whether the candidate entity appears in the document text.
5) Number of different entity mentions in the target document that lead to the same candidate entity in the KB (Cucerzan, 2011).
7) Type indicator. A binary indicator determines whether the type of the entity mention and the candidate entity are different.

The optimal entity is finally calculated by maximizing the linear combination of these similarity features:

$$\arg \max_{e_i \in E(\mathrm{em}_i)} \sum_{j=1...|\mathrm{F}|} \lambda_j f_j(e_i, D),$$

where em_i denotes one of the entity mention detected from the target document D, $E(\mathrm{em}_i)$ denotes the set of candidate entities for em_i, F denotes the feature set and $\mathrm{f}(e, D)$ denotes the score of each feature. We learned the best linear coefficient λ by using PSO algorithm (Eberhart, R. C., & Kennedy, J. (1995, October)) on the TAC 2015 training data set.

5.3 NIL entity cluster

For the NIL entities, three steps are used to cluster the NIL entities across target documents. Firstly, the similarities between entities are measured in terms of the context similarity, name similarity and type indicator between entity mentions, where the context similarity and type indicator are defined in the same way as above. As for the name similarity, some heuristic rules are combined to promote the performance of clustering. The heuristic rules can be stated from three aspects.

1) Coreference resolution within one document. For example, the entity mention “Mandela” and “Nelson Mandela” where the latter contains the former as a substring, refer to the same entity if they are persons appearing in one document in the training data set. Therefore, all entity mentions of “Mandela” and “Nelson Mandela” are assigned with the same NIL identifier.
2) Acronym expansion matching in the document text. For example, "UK" is expanded to "the United Kingdom" in several different documents in the training data set. Thus, they are mapped to the same NIL identifier.
3) Furthermore, we only calculate the similarities between entities of the same type, otherwise the similarities are equal to zero.

Secondly, the cluster centroids are automatically spotted and excluded from the analysis（Rodriguez A, Laio A，2014）. Namely, we assumed that the centroid of entities are surrounded by neighbor synonymous entities with lower densities, and located with a relatively large distance away from entities with higher densities.

The density ρ_i of entity i is defined as

$$\rho_i = \sum_j \chi(d_{ij} - d_c),$$

where $\chi(x) = 1$ if $d_{ij} - d_c < 0$ and $\chi(x) = 0$ otherwise, and d_c is a cut-off distance and d_{ij} is the distance between entity i and entity j ,which can be simply obtained by the similarity between two entities introduced in the first step.

Moreover, we define the density-based distance with respect to an entity, denoted by δ_i ,measured by computing the minimum distance between the entity i and any other entities with higher density:

$$\delta_i = \min_{j:\rho_j > \rho_i} (d_{ij})$$

Thus, the centroid of many entities is recognized as the one among them for which its values of δ and ρ are both high. Obviously, we regard the entity with a relatively high δ and a low ρ as isolated entity, and an isolated entity forms a cluster itself.

Thirdly, after the cluster centroids have been found, each remaining entity is assigned recursively to the same cluster of its nearest neighbor entity with higher density.

6 Inference

The inference module is aimed to infer more triples based on the generated ones in Section 4, and it is conducted by mainly following these rules:

1) Rules for place-related slots. For example, for an entity that has value about slot “city”, we can infer corresponding "stateorprovince" and "country" by Gazetteer. Similarly, "country" can be inferred from "stateorprovince".

2) Rules for date-related slots. For example, for per:date_of_birth, per:date_of_death, per:age, given two of these three slots, the third one can be inferred (except birth && age -> death, because someone who has birthdate and age may not die yet). For example, if A died in 2010 at age 78, so we can infer that A was born in 1932.

3) Rules for family relationships, which is illustrated in Table 1.

Table 1 Rules for inferring family relationships

A --- B	B --- C	A --- C
children	siblings	children
children	spouse	other_family
children	children	other_family
spouse	children	children
spouse	parents	other_family
spouse	siblings	other_family
parents	parents	other_family
parents	siblings	other_family
parents	spouse	parents

4) Rules for implicit-date results. For results for the slots describing date which doesn’t express year/month/day explicitly, such as “died in Tuesday”, we transform it into standard date format according to the calendar.

5) Rules for employee-related slot. For example, for a person entity whose title is CEO, president, vice-president, or other titles which represent top employees, and this person entity has slot “per:employee_or_member_of”, we can infer slot “org:top_members_employees”.

6) Rules for inverse slot. For every slot which has inverse slot, we add the inverse relation of this slot according to the Slot Description Guideline.

7 Post-processing

To correct the errors in the slots extracted by the Filler component, we introduce the post-processing step. Specifically, we mainly use some rules, which are listed as below.

1) The values of some slots must be of certain particular type. For example, when slots describe relations between people (e.g. spouse, children), the type of the results must be person (PER). This can be examined by means of the Stanford NER tool (Finkel et al., 2005).

2) Standardization of dates. Convert all answers which represent dates to standard date format "XXXX-XX-XX".

3) Delete unreasonable answers. For example, the results for the slots describing age should be a number usually larger than 0 and smaller than 130 respectively.

8 Evaluation results

The results for Cold Start KB track are divided into two dimensions (Entity Discovery & Linking and Slot Filling) and are illustrated in Table 2 and 3.

For Cold Start KB Task, we submitted four runs, in which the second run performs best in Entity discovery and linking and the first run performs best in slot filling. These four runs are:

Run1: named mention + without nominal mention + NIL cluster + different weight in relation extractor combination.

Run2: named mention + nominal mention + NIL cluster + different weight in relation extractor combination.

Run3: named mention + without nominal mention + without NIL cluster + different weight in relation extractor combination.

Run4: named mention + without nominal mention + NIL cluster + same weight in relation extractor combination.

Table 2 and Table 3 are the results of Entity Discovery and Slot Filling dimension respectively.

Table 2 Entity Discovery Result (Engilish)

	strong_mention_match			strong_typed_metion_match			mention_ceaf			b_cubed		
	P	R	F1	P	R	F1	P	R	F1	P	R	F1
1	0.921	0.652	0.763	0.874	0.619	0.725	0.749	0.531	0.621	0.854	0.443	0.584
2	0.915	0.680	0.780	0.870	0.647	0.742	0.721	0.536	0.615	0.851	0.452	0.590
3	0.923	0.610	0.735	0.880	0.582	0.700	0.664	0.439	0.528	0.877	0.343	0.493
4	0.916	0.680	0.781	0.833	0.619	0.710	0.714	0.531	0.609	0.852	0.452	0.591

Table 3 Slot Filling Result (LDC-MAX, English)

	hop0_P	hop0_R	hop0_F	hop1_P	hop1_R	hop1_F	All_P	All_R	All_F
1	0.2215	0.1176	0.1537	0.0534	0.0357	0.0428	0.1563	0.0902	0.1144
2	0.2195	0.1176	0.1532	0.0483	0.0325	0.0388	0.1533	0.0891	0.1127
3	0.2837	0.0964	0.1439	0.1092	0.0422	0.0609	0.2202	0.0783	0.1155
4	0.2410	0.1095	0.1506	0.0420	0.0162	0.0234	0.1814	0.0783	0.1093

From these results we can conclude that nominal mention detection can help improve the recall at the cost of precision. In addition, NIL cluster in cross-document co-reference resolution can improve the performance of entity linking. Moreover, combining results of different relation extractors with different weights can improve the performance of slot filling by a little.

9 Conclusion

In this paper, we present the OpenKN system for the Cold Start KB Track of the KBP 2016. The proposed system contains pre-processing, relation extraction, cross-document co-reference resolution, inference, post-processing five steps corresponding to the task. The official evaluation results are also provided.

References

Finkel J. Rose, Grenager T. and Manning C. 2005. Incorporating non-local information into information extraction systems by gibbs sampling, Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics, 363-370.

He Y. and Grishman R.. 2015. ICE: Rapid in- formation extraction customization for nlp novices. In Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Demonstrations, pages 31–35, Denver, CO. Association for Computational Linguistics.

He Y. and Grishman R. 2015. The NYU Cold Start System for TAC 2015. In Proc. Text Analysis Conference (TAC2015).

Hong Y., Lu D., Yu D., Pan X., Wang X., Chen Y., Huang L., and Ji H.. 2015. Rpi blender tac-kbp2015 system description. In Proc. Text Analysis Conference (TAC2015).

Mausam, Schmitz M., Bart R., Soderland S., and Etzioni O. 2012. Open language learning for information extraction. In Proceedings of EMNLP.

Soderland S., Gilmer J., Robert Bart, Oren Et- zioni, and Daniel S. Weld. 2013. Open information extraction to KBP relations in 3 hours. In Proceedings of TAC-KBP 2013.

Jia Y, Wang Y, H Lin, et al. Locally Adaptive Translation for Knowledge Graph Embedding, AAAI, 2016.

Weston J, Bordes A, Yakhnenko O, et al. Connecting Language and Knowledge Bases with Embedding Models for Relation Extraction[J]. 2013.

Lin H, Zhao Z, Jia Y, et.al. OpenKN at TAC KBP 2014. In Proceedings of TAC-KBP 2014.

Chen X, Jia Y, Wang Y, eet al., OpenKN at TAC KBP 2015. In Proceedings of TAC-KBP 2015.

Toutanova K., Klein D., Manning C., and Singer Y. 2003. Feature-rich Part-of-Speech Tagging with a cyclic dependency network. Proceedings of HLT-NAACL 2003, 252-259.

Soderland S., Hawkins N., Kim G. L., and Weld D. S. 2015. University of Washington System for 2015 KBP Cold Start Slot Filling. In Proceedings of TAC-KBP 2015.

Marneffe M. D. and Manning C. 2008. Stanford Dependencies manual.

Wu F. and Weld D. 2007. Autonomously semantifying Wikipedia. Proceedings of the sixteenth ACM conference on information and knowledge management, 41-50.

Manning, Christopher D., Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J. Bethard, and David McClosky. 2014. The Stanford CoreNLP Natural Language Processing Toolkit In Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations, pp. 55-6

Rodriguez A, Laio A. Clustering by fast search and find of density peaks[J]. Science, 2014, 344(6191): 1492-1496.

Yan X, Guo J, Lan Y, et al. A biterm topic model for short texts[C]//Proceedings of the 22nd international conference on World Wide Web. ACM, 2013: 1445-1456.

Eberhart R C, Kennedy J. A new optimizer using particle swarm theory[C]//Proceedings of the sixth international symposium on micro machine and human science. 1995, 1: 39-43.