Description of the BOUN System for the Trilingual Entity Detection and Linking Tasks at TAC KBP 2017

Arda Çelebi and Arzucan Özgür

Department of Computer Engineering
Boğaziçi University
Bebek, 34342 İstanbul, Turkey
{ arda.celebi, arzucan.ozgur } @boun.edu.tr

Abstract

This paper describes the BOUN system's participation in the Trilingual Entity Detection and Linking (EDL) track in 2017 TAC Knowledge Base Population (KBP) challenge. EDL is important aspect of text analysis and various tasks like sentiment analysis and opinion mining can benefit from discovering which named entities are mentioned in the context. For this challenge, we built a simple candidate generator and applied a Maximum Entropy-based approach for named entity linking. Our system has achieved an F1 score of 52.5% in the "strong typed all match" metric on the 2017 evaluation set.

1. Introduction

In this paper, we describe the BOUN system, which was designed to compete in the 2017 TAC KBP Trilingual Entity Discovery and Linking (EDL) task (Ji and Nothman, 2016) organized by NIST. Various natural language processing tasks like sentiment analysis and opinion mining can benefit from discovering which named entities are mentioned in the context. Hence, being able to achieve EDL with high accuracy affects the target performance of the dependent tasks.

In this challenge, participants are given a document collection in three languages (English, Chinese and Spanish) and it is required to automatically identify, classify, cluster and link entity mentions in those documents to the English Knowledge Base (KB). TAC-KBP-EDL challenge uses the BaseKB[1] as the reference KB, which is derived from the Freebase. However, we are not only expected to find known mentions from the reference KB, but also to be able to detect entity mentions that are not recorded in the KB, which are called NIL mentions. Moreover, the system should assign the same identification number to the different occurrences of the same unknown entity throughout the given documents. This is called NIL clustering. A system is required to identify and classify the following pre-defined entity types in a given text: Person (PER), Geo-political Entity (GPE), Organization (ORG), Location (LOC), and Facility (FAC). Entity mentions include both name mentions (NAM) and nominal mentions (NOM) for all entity types in all three languages. Nominal mentions are limited to specific and individual mentions. At the end, the system should output the starting and ending offset of each detected mention in a given document, its link to reference KB entry (or NIL link), its entity type, and mention type (NAM or NOM).

In the following sections, we first explain our method of generating candidates and selecting the best candidate. We describe our model and its features. Then we introduce the datasets and discuss our results of the submission and after improvements. Finally, we conclude with the future work.

2. Method

Our method is broken down to two major steps: candidate generation and selection of the correct candidate as the predicted mention, which is also called named entity disambiguation (NED). Before getting into the

[1] http://basekb.com

description of those two steps, the following section describes the overall design and the main components of the system.

2.1. System Design

In order to be able to develop our system fast, we break the system into multiple components. For speed- and memory-critical jobs, we implemented the corresponding components in C. Those are, candidate generator, disambiguator and memory server. The rest of the components were implemented in Python. In order to reduce the load time, we load embedding vectors via our memory server component into the shared memory, so that candidate generator and disambiguator can directly access and use them from shared memory, that is, without having to load them. This was especially useful as we frequently update and restart those components during the research period. Figure 1 depicts the flow of the system.

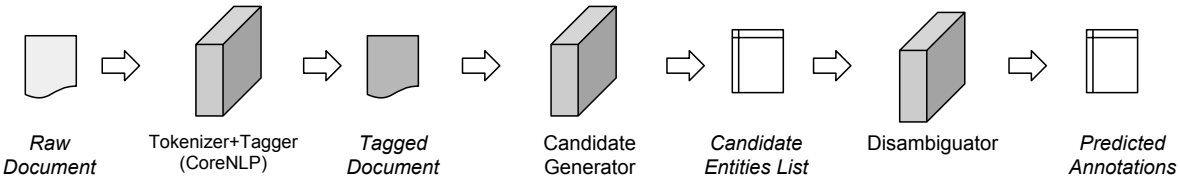

Figure 1: BOUN System Design.

As a preprocessing step, our system uses Stanford Tokenizer and POS Tagger (Toutanova and Manning, 2000). Then, candidate generator produces a list of possible candidate spans and entities in them. As described in the next section, POS tagged input helps the candidate generator detect the unknown named entities with a basic rule-based approach. As the disambiguator gets the candidates, it uses the Maximum Entropy (MaxEnt) model (Berger et al., 1996) to pick the best candidate and outputs the predicted annotations.

In addition to these components, we also developed two web-based user interfaces to maintain our title database and to examine the output of the disambiguator by running it on a given input text. We believe these interfaces help us explore our system and its output much faster, reducing development time.

2.2. Candidate Generation

Given input text, our candidate generator detects spans (sequence of words) in given text which possibly mentions a named entity. For each such span, it outputs all possible known named entities which that spans corresponds to. Not only known ones, but it also looks for unknown named entities which will be explained in detail in the Discovering NIL cases section below. Moreover, in order to be able to detect known entities with typos, we use the off-the-shelf library called "simstring"[2] (Okazaki and Tsujii, 2010) to index titles so that we can search them with partial match. As we get the search results, we compare the title of actual entities with searched span and look at whether the edit distance between them is lower than a threshold. We use variable threshold depending on the length of the title.

In addition to candidates, our candidate generator also extracts additional information and adds them into its output so that they can be used by the disambiguator at the next step. For example, the number of times two entities co-occur in Wikipedia articles is used as additional information.

Building Title Database

As the annotations are done with BaseKB entities, we consider the Freebase (FB) (Bollacker et al., 2008) as our main source of entities. We filter out entities that are not type of organization, person and location[3].

[2]Available at http://www.chokkan.org/software/simstring/

[3]Certain entries (eg. m.07k4cg1 which is named "Eurogroup") are seen in annotated data but their type information in FB is no specific than "common.topic". We believe this is due to the fact that we do not have the final version of FB.

Then we use the mapping of FB entity IDs to entity IDs on the DBPedia (Auer et al., 2007) and Wikipedia knowledge bases in order to be able to make use of information from DBPedia and Wikipedia. We mainly use this connection to gather as many titles as possible for each entity. Even though we did not use Wikipedia redirections in this process, DBPedia provides rich but noisy set of titles for each title, along with their DBPedia Frequencies. For example, "president" is one of the titles for the entity named "Barack Obama." If used, whenever we see the word president, our system should mark that word as a candidate mention of this particular entity. However, that leads to a lot of candidates which are most likely false. Hence, first we filter out the ones with the frequency of 1. Then, we extend the filtering with a semantic-based approach. By using word2vec (Mikolov et al., 2013), we learn title embedding of each title from Wikipedia by detecting the occurrence of each title in Wikipedia text (not anchors) and converting it into a single word. Then we calculate the cosine distance of each title against the Wikipedia title (converted from ID), which we assume as the main title of the entity. If the similarity is low, this means that the compared title occurred in different contexts and thus conveys not the same meaning as the main title. We filtered out the ones with similarity less than 0.6. We name this similarity measure as title probability, which will be used as a feature during modelling the disambiguation task.

Assigning Roles to Titles to Help Rule-based Candidate Detection

Named entities can be referred in numerous ways. Sometimes these refering text can be specific or unique (i.e. main title) to the entity or generic, like her first name, or nominal like "president." In most of the cases, these generic cases can be automatically generated from the main title. And in order to identify them better, we can also assign roles to them. Starting with firstname and surname lists, we look at person type named entities and check if the main title of the entity can be broken into firstname and surname parts. If so, we add the firstname and surname as individual titles for that entry and label them with FIRSTNAME and SURNAME roles. We also create a new title in "surname, firstname" format and label it as MAIN_TITLE. Moreover, if we detect a title that is also in the nominals list (extracted from 2015 training set), we also mark them with NOMINAL role. During candidate search, we do not use these titles because many entities may share these titles. Hence, we search them in the context after we detect the entity through its more complete title(s), like its main title. Note also that what we do is also a way of populating the title database automatically. However, due to limited amount of time, we did not apply this approach on entity types other than person.

Discovering NIL Cases

While it is easy to detect occurrences of titles in given text, we should also be able to detect occurrences of unknown named entities, which do not have a record in the knowledge base. Such cases are called NIL mentions. To achieve this, we first use the Stanford Part-of-Speech (POS) Tagger to tag the given text and then detect the sequences of words that are marked with NNP tag.

In addition to this POS-based method, we also use two rule-based approaches. First, we manually crafted a short set of rules to catch simple cases like "University of @", "Mount @" etc. where @ is a slot to match with NNP-tagged word or phrase. With this approach, we not only detect such occurrences, but also assign appropriate entity type for each particular case (e.g. LOC for "University of @"). As a second rule-based approach, we curated a list of first names and surnames. Whenever we detect a first name and a surname following each other, we mark that span as a candidate NIL case with PER entity type unless there is already a known person name detected over that span. To make it more robust, we also allow OOV words to match with first name or surname slots of the rule.

Discovering Nominal Cases

We extracted the set of all nominals seen in the 2015 training set. We detect their occurrences in a given text and mark them as candidate nominal NIL cases. Since our model is simple, we did not connect those nominals to other occurring named entities in the context. Instead, we let the system to decide whether a detected nominal NIL candidate is valid or not. Nevertheless, if there is a nominal associated with a known named entity in our Title Database (eg. "president" for the entity "Barack_Obama"), in that case, after detecting that named entity in the context we specifically look for its associated nominal(s) in the context

and we connect them if exists.

2.3. Named Entity Disambiguation

We consider the disambiguation task as a classification task, where each candidate entity is considered whether it is an actual mentioned entity or not. In other words, we convert the disambiguation task to set of binary classification tasks and then pick the best scored candidate. We use Maximum Entropy (MaxEnt) model for the binary classification task. The selection mechanism also considers the case where none of the candidates has to be chosen by applying a threshold value. In official submission, we discard any candidate span if none of its candidate entities inside has higher than 0.5 score given by the MaxEnt model. After the challenge, we improved our decision mechanism by employing a conditional threshold: when the title of the highest scored named entity is the same as a Wikipedia title and its frequency is higher than 100 in DBPedia Frequency Dataset, or its frequency is just higher than 1000, then we drop the threshold to 0.1.

MaxEnt Model and Engineered Features

MaxEnt classifier models each case by its feature instances and class label. Basically, the case is converted and represented in terms of feature instances which can be used by the MaxEnt model to learn which feature instances are more likely to be seen in a certain class. Feature instances are extracted from the case based on pre-defined feature templates. In the context of our task, each case corresponds to one candidate named entity (CNE) in a given span. These feature templates can access various information regarding the case and generate the feature instances. We use off-the-shelf MaxEnt tool[4] in our experiments. Each annotation in the training data set is considered as a positive example during the training process. As we are also capable of producing candidates, we consider all candidate entities that are not the same as the annotated entity in a given span as negative examples.

In our experiments, we design two sets of feature templates, one for known named entities and other for unknown, namely NIL cases. As described below, we generate feature instances for each candidate. The MaxEnt model considers each feature instance as an identifier clue to model the associated class. To be precise, feature instances are no different than a string of characters. As given in the second column of the Tables 1 and 2, we generate instances in the "key=value" format, where "key" is the short name of the feature and "value" is the value of that feature for the particular case. The '=' in between is to make it easy to read.

FEATURE	**EXAMPLE FEATURE INSTANCE**
ContextScore(CNE)	cs=0.87
MaximumEntityContextScore(CNE)	mecs=0.70
TitleProbability(TITLE)	tp=0.3
TitleOccurrenceProbability(TITLE)	top=0.3
NumOfCooccurredEntitiesInContext(CNE)	cooc-count=10
NormalizedNumOfCooccurredEntitiesInContext(CNE)	norm-cooc-count=0.05
CooccurrenceDifferenceInSpan(NE)	cooc-diff=-10
LogFreq(TITLE) + *spanSize*(SPAN)	lf-ss=13+1
MaxCoocOverSpan(SPAN)	max-cooc-over=10
NumOfSeenTitlesWRoleInText(CNE)	roled-title-count=2
EditDistancePath(TITLE)	edp=.....+++....

Table 1: Description of the Features to Model Known Named Entities.

Table 1 lists the features to model known named entities. *ContextScore*(·) measure strength of the semantic relation between the named entity and its surrounding context window of words. We use the BridgeGap toolkit[5] (Cao et al., 2017) and trained[6] it on Wikipedia in order to get the word embeddings and context

[4]Available at http://homepages.inf.ed.ac.uk/lzhang10/maxent.html

[5]Available athttps://github.com/TaoMiner/bridgeGap

[6]We use following settings: window=10, vector dimension=100, ns=100

vectors of entities. When text is given, we combine the word embeddings around the entity and compare it with the context vector of that entity, which gives the context score. *MaximumEntityContextScore*(·) (MECS) is a similar measure but between co-occurring named entities in a given context. We generate a context vector for each Freebase (FB) entity by taking the mean of other entity vectors co-occurring in the Wikipedia documents. This may be interpreted as keeping the co-occurring statistics in the form of a vector rather than count, which is expected to be more robust. We use Google's Freebase entity vectors which have the dimension of 1000. After obtaining the context vector, we calculate cosine distance between that vector and the vector of all other entities in a given context. This feature calculates the highest score. *TitleProbability*(·) (TP) measures the word embedding-based similarity of entity's title in given input with respect to its main title. *TitleOccurrenceProbability*(·) (TOP) is pre-calculated probability of seeing the title based on the frequency counts provided by DBPedia. Similar to MECS, *NumOfCooccurredEntitiesInContext* (COOC-COUNT) calculates how many named entities in the context are seen together with the target entity in the same document in Wikipedia. Both MECS and COOC-COUNT values represent the strength of the connection between the target entity and the context from different aspects. The higher they are, the more likely that the target entity is the valid mention. While these values consider the candidate entity within its span alone, we also consider a feature that compares that candidate entity with respect to other candidates under the same span. While COOC-COUNT value increases proportionally to the length of the context, we also include its normalized version by dividing that value with the number of candidate spans seen in given text, which is taken into account with *NormalizedNumOfCooccurredEntitiesInContext* feature. Another feature *CooccurrenceDifferenceInSpan* (COOC-DIFF) calculates the difference between COOC-COUNT value of the candidate and the highest COOC-COUNT value seen in the same span. The more negative the value is, the less likely that it is not a valid mention as the other candidate with the higher COOC-COUNT value has better context connections. In order for our system to better model the frequent entities, we designed a feature (LF-SS) that looks at log frequency of the target entity's title and number of candidate entities in the span. The higher the first number is and the lower the second number is, the more likely that that target entity is valid mention. In order the model to consider the cases where one span is overlapping with another, *MaxCoocOverSpan*(·) checks the overlapping spans and find the highest COOC-COUNT and uses it. Another feature to model known named entity candidate considers how many times we see its occurrence in given text with the title that has assigned role in our Title Database. The last feature in the table considers the case when we detect a title with a typo. In that case, we generate the edit distance traversal path, which represents where the characters are the same (marked with '.'), added ('+'), removed ('-'), or replaced ('*'). This is especially useful in cases like where we detect an extra middle name in between the firstname and surname of a known person.

FEATURE	**EXAMPLE FEATURE INSTANCE**
TitleEncoding(TITLE)	title-enc=3:2_3:1
OrthographicShape(TITLE)	ortho=CcccCccc
SurroundingOrthographicShape(TEXT)	surr-ortho=ccc-ccc
NILPersonPOSTagSequence(TAGGED_TITLE)	per-tag-seq=NNP_NNP
NILPersonSurroundingPOSTagSequence(TAGGED_TEXT)	per-surr-tag-seq=IN_,
NILNonPersonPOSTagSequence(TAGGED_TITLE)	nil-tag-seq=NNP_NNP
NILNonPersonSurroundingPOSTagSequence(TAGGED_TEXT)	nil-surr-tag-seq=IN_,
NILPatternItself()	pattern=Univesity_of_@
NominalItself()	nominal=uncle

Table 2: Description of the Features to model NIL Cases.

While all described features are specific to known named entity candidates, we also designed features for NIL cases as listed in Table 2. *TitleEncoding*(·) represents each word of the title with its length and log frequency. Frequencies are calculated over the Wikipedia. Words having the same length and similar frequency can be seen as clustered together in this representation. The model is expected to use this information to figure out which words make the NIL candidate less likely the valid one. *OrthographicShape*(·) converts the title to its orthographic shape by replacing upper cased letters with 'C', lower cased ones with 'c', digits with 'd' and keep the other characters as they are. At the end, we also clip character repetitions to three characters, if they are longer. We also take into account the orthographic shapes of the previous and next words of the span with *SurroundingOrthographicShape*(·) feature. Among these features, we also divided

some features into two groups; the ones for unknown/NIL person and non-person ones. For the first case, *NILPersonPOSTagSequence*(·) considers concatenation of POS tags of words in the title and *NILPersonSurroundingPOSTagSequence*(·) combines POS tags of previous and next words of the span. We also have the same two features for non-person NIL cases, named *NILNonPersonPOSTagSequence*(·) and *NILNonPersonSurroundingPOSTagSequence*(·), respectively. We expect the model to use this in order to differentiate the POS tag sequence of person and non-person cases. Another feature *NILPatternItself*() takes into account the pattern itself we use to detect NIL cases. Finally, to better model nominals, *NominalItself*() feature uses the nominal itself.

NIL Clustering

During our submission period, we only performed NIL clustering within a document, but not between documents. However, we later applied a basic string-based approach to cluster NIL entries with the same title. In our experiments, we run this approach after performing the disambiguation step.

3. Experiments

In these experiments, we consider running our system on English-only documents due to short amount of research time and lack of expertise in non-English content. Hence the following discussion and results are only for English.

3.1. Dataset

For training and evaluation, we use datasets provided at the TAC-KBP-EDL challenge. Organizers made available annotated documents from 2015 and 2016 challenges. We have access to training set from 2015 (2015t), evaluation set from 2016 (2016e) and 2017 (2017e). During the development period, we used the annotated documents from 2015e for training and 2016e for evaluation purposes. The number of documents and annotated mention statistics are given in Table 3. Each dataset contains documents from two genres: newswire (NW) articles and discussion form (DF) text. While NWs exhibit well-written and less noisy language usage, DFs may contain more casual usage of the language, not to mention quoted text segments which were supposed to be discarded by our system. In the table, *Linkable Name* and *Linkable Nominal* refers to known entities that are mentioned with their actual name and nominal, respectively. *NIL Name* and *NIL Nominal* are the same cases for unknown entities.

	2015t		**2016e**		**2017e**	
Type	NW	DF	NW	DF	NW	DF
Linkable Name	4,123	4,833	3,329	2,289	2,280	1,465
Linkable Nominal	520	297	1,051	538	619	270
NIL Name	688	2531	372	1,019	388	1,118
NIL Nominal	369	180	458	175	479	358
ALL	5,700	7,841	5,210	4,021	3,704	3,211

Table 3: Mention statistics for English documents in 2015, 2016 and 2017 data sets.

Table 3 reveals the fact that number of *NIL Name* cases are quite high in case of DFs. Specifically the ratio of them to *Linkable Name* is getting higher for the newer datasets. Since most of such cases are name mentions that are very easy to detect with a rule-based approach, such aspect of the dataset makes the final results look higher.

In addition to these annotated documents, organizers also share a large set of raw documents for further training purposes. However, we did not make use of these documents in our experiments.

3.2. Evaluation

Evaluation of the system is performed based on three different aspects: mention evaluation, linking evaluation and clustering evaluation. The evaluation metric called *"strong_typed_mention_match"* (NERC) (Hachey et al., 2014) evaluates mention detection and classification, by considering how accurately it detects spans and their type. The linking evaluation metric called *"strong_typed_all_match"* (NERLC) adds linking performance on top of NERC by measuring how accurate each detected known entity is linked to the right entry at the reference KB. To evaluate the clustering, Mention CEAF (Luo, 2005) method is used, which finds the optimal alignment between system and gold standard clusters and evaluates the precision and recall micro-averaged over mentions. In our experiments, special form of this method called *"typed_mention_ceaf"* (CEAFmC) is used, which further constrain clustering evaluation to require correct mention type classification. In our experiments, we use the scorer script[7] provided by the organizers.

3.3. Results

Our best submission scores are shown in Table 4. Without using any off-the-shelf Named Entity Recognizer (NER), we achieved F_1-score of 52.9 on all English-only documents based on NERC evaluation metric. When counting the accuracy of the linking capability on top of NERC, which is NERLC, our score drops to 49.4. As we also include the accuracy of clustering, our results drop further to 44.4. Note that we obtained these results by training out system on only NWs from 2015t and 2016e data sets. Adding DFs on top of NWs did not change the results much. We believe this is due to the fact that DFs are more noisy data and most of the annotated cases are NILs. As we explore the results in Table 4, it is easy to notice the high NERC and NERLC scores in DF documents compared to NW, but comparatively low CEAFmC score in DF due to high number of NIL cases in DF. In all cases, our recall is much lower than precision.

	Newswire			Discussion Forms			Overall		
Metric	Pre	Rec	F_1	Pre	Rec	F_1	Pre	Rec	F_1
NERC	62.2	34.8	44.7	75.5	52.3	61.8	69.1	42.9	52.9
NERLC	55.3	30.9	39.7	72.9	50.5	59.7	64.4	40.0	49.4
CEAFmC	56.9	31.9	40.9	59.0	40.9	48.3	58.0	36.0	44.4

Table 4: Our submission scores in TAC-KBP-EDL Challenge on English-only documents in 2017e.

There are number of factors to explain the low scores. While not using an off-the-shelf NER but instead using simple candidate generator might be the main reason, we also detected certain discrepancies after submission. First, our FB dataset is not up-to-date as there are 164 (out of 6,915; i.e. 2.3%) occurrences of 58 named entities in 2017e that do not exist in our version of FB. Secondly, we did not model FAC type named entities so we automatically missed to identify 399 (i.e. 5.7% of all) such occurrences in 2017e test set. Thirdly, our model is not tuned to detect nominals, especially Linkable Nominals (270 such occurrences in 2017e; i.e. 3.9% of all). As we add them up, our system already misses up to 12% of the annotations in the 2017e from the get-go.

After our submissions, we continued to explore new features and improve our system. The final results on the 2017 test set are given in Table 5. We improved our results 3-4 points in all metrics. This improvement can be attributed to a number of updates. First, we switched from fix threshold for selecting the best scored candidate entity to conditional threshold as described in Section 2.3. Secondly, we re-run BridgeGap on Wikipedia with higher negative sampling value which helped make the context score (CS) feature values more discriminative. Thirdly, we add new features like LS-SS, NORM-COOC-COUNT, and EDP. Also, we cleaned our firstname and surname lists. Finally, we implemented simple string based clustering for NIL cases, which affects only the results in CEAFmC metric.

Table 6 breaks down the results based on entity types in terms of the NERLC evaluation metric. Other than not handled FAC-type entities, the results indicate that our system is not good at identifying and linking organization and specifically location type entities.

[7] Available at https://github.com/wikilinks/neleval.

Metric	Newswire			Discussion Forms			Overall		
	Pre	Rec	F_1	Pre	Rec	F_1	Pre	Rec	F_1
NERC	63.8	38.5	48.1	78.9	56.5	65.8	71.5	46.9	56.6
NERLC	56.1	33.9	42.2	76.1	54.4	63.5	66.2	43.4	52.5
CEAFmC	57.6	34.8	43.4	67.7	48.5	56.5	62.5	41.0	49.5

Table 5: Our current best results on TAC-KBP-EDL Challenge on English-only documents in 2017e.

Entity Type	**Pre**	**Rec**	F_1
PER	70.1	60.9	65.2
ORG	70.1	28.9	40.9
LOC	12.1	17.8	14.4
GPE	76.7	44.2	56.1
FAC	00.0	00.0	00.0

Table 6: Breakdown of the 2017e set results in entity types based on the NERLC metric.

4. Conclusion

In this study, we developed a system that is capable of generating candidate named entities from a given text and then picking the best possible candidate by using a maximum entropy model. We defined the disambiguation task as a set of binary classification tasks where we score each candidate based on whether it can be the right one and then choose the one with the highest score. While such a simple ranking approach ignores comparing candidates within the same span with each other, we designed a couple of features that take into account that in-span comparison. As we investigate our submission results, we observed that there are a number of factors that contribute to their low nature. Our system misses up to 12% of annotations from the get-go. Our further experiments did not address those issues but explored new features. We increase our results up to 3-4%.

5. Future Work

We will explore the boundaries of the MaxEnt-based model. First, we will address the factors that play major role in the results. We will update our FB version to remedy the case of missing named entities. We will model FAC-type named entities and design better model to handle nominal cases. Secondly, we will replace our candidate generator with an off-the-shelf NER tool. This way, we will be able to assess the performance of the MaxEnt-based model better.

6. Acknowledgements

This research is supported by Boğaziçi University Research Fund Grant Number 11170. Arda Çelebi is also supported by ASELSAN Graduate Scholarship for Turkish Academicians.

7. References

Auer, Soren, Bizer, Christian, Kobilarov, Georgi, Lehmann, Jens, Cyganiak, Richard, and Ives, Zachary. (2007). Dbpedia: A nucleus for a web of open data. *In Proceedings of the 6th International Semantic Web Conference*.

Berger, Adam L., Pietra, Stephen A. Della, and Pietra, Vincent J. Della. (1996). A maximum entropy approach to natural language processing. *Computational Linguistics, 22(1):3971*.

Bollacker, Kurt, Evans, Colin, Paritosh, Praveen, Sturge, Tim, and Taylor, Jamie. (2008). Freebase: A collaboratively created graph database for structuring human knowledge. *In proceedings of ACM SIGMOD International Conference on Management of Data*.

Cao, Yixin, Huang, Lifu, Ji, Heng, Chen, Xu, and Li, Juanzi. (2017). Bridging text and knowledge by learning multi-prototype entity mention embedding. *In proceedings of ACL*.

Hachey, Ben, Nothman, Joel, and Radford, Will. (2014). Cheap and easy entity evaluation. *In Proc. of the 52nd Annual Meeting of the Association for Computational Linguistics (Vol. 2)*.

Ji, Heng and Nothman, Joel. (2016). Overview of tac-kbp2016 tri-lingual edl and its impact on end-to-end kbp. *Proc. Text Analysis Conference (TAC2016)*.

Luo, Xiaoqiang. (2005). On coreference resolution performance metrics. *Proc. HLT/EMNLP*.

Mikolov, Tomas, Sutskever, Ilya, Chen, Kai, Corrado, Greg, and Dean, Jeffrey. (2013). Distributed representations of words and phrases and their compositionality. *Proceedings of Neural Information Processing Systems (NIPS)*.

Okazaki, Naoaki and Tsujii, Jun'ichi. (2010). Simple and efficient algorithm for approximate dictionary matching. In *Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010)*, pages 851–859, Beijing, China, August.

Toutanova, Kristina and Manning, Christopher D. (2000). Enriching the knowledge sources used in a maximum entropy part-of-speech tagger. *In Proceedings of the Joint SIGDAT Conference on Empirical Methods in Natural Language Processing and Very Large Corpora (EMNLP/VLC-2000)*.