Temporal taggings of Entities in Text

Haozhou Pang
Department of Computing Science
University of Alberta
haozhou@ualberta.ca

Yourui Guo
Department of Computing Science
University of Alberta
yourui@ualberta.ca

Abstract

TODO

1 Introduction

Entities in text can be related to some time points or intervals mentioned in an article, and the demand of knowing temporal expressions that are significant to entities in articles is increasing over the years. For example, utilizing the temporal expressions can help us to look for the named entity it truly refers to. When we read a newspaper about Bush, we can interpret the named entity as George W. Bush knowing his birthdate being July 6, 1946 mentioned in the article. Also, extracting temporal expressions related to entities helps us to construct knowledge bases. As we know that most of knowledge bases contain a small quantity of temporal taggings for named entities. Thus, the content of existing knowledge bases can be expanded by providing temporal information that was taken from any source of text.

Temporal tagging detection is the process of looking for expressions in text that refers to time points or time intervals. Few works have studied on detecting temporal taggings [1, 2]. SUTime [1] is a library for recognizing and normalizing time expressions. It transforms time expressions like October 1963 to the normalized value of 1963-10 and type of DATE. Normalizing temporal expressions reduces the chance of mis-recognizing the same date value to different ones, and the result of detecting the pairs of entity and temporal tagging can be more accurate as well.

Named entities recognition seeks phrases or strings that refer to named entities like organizations, person name and places and so on. Natural Language ToolKit (NLTK) [3] is a platform that provides us with interfaces to work with human natural language data. NLTK is utilized for natural language processing such as named entity recognition. The basic idea of recognizing named entities is to consider the task as a noun-phrase chunking. Raw text will be split into sentences first, and each sentences will be chunked by word tokenizer. Then, next step is to seeking noun phrases among chunks by tagging each chunk with pos-tagger. The last step is to use relation detection to search for potential related entities to ensure the detected entity is meaningful.

The problem addressed in this paper aims at searching for temporal taggings that are significant to entities mentioned in articles. The input can be a corpus containing multiple raw texts such as news articles, web data and social media. The output is a list of pairs that consist of both entities and temporal taggings where for each pair the entity is highly related to the temporal tagging. In this work, we need to find out all temporal taggings mentioned in plain text, and also look for entities related to them. Then, we should sort out the list of result by ranking the relatedness of entity and temporal tagging in each pair among all the pairs, and retrieve top k result with highest relatedness.

As shown in figure.1, our task could be formulated in such way: given a set of articles with entities and meaningful temporal expres-

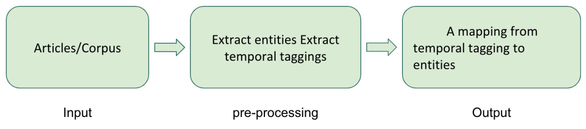
Articles/Corpus
Extract entities Extract temporal taggings
A mapping from temporal tagging to entities
Input
pre-processing
Output

Figure 1: The visualization of the process

sions, since we are dealing with raw text data of any format and any length, we first use the nltk sentence tokenizer (CITE) to split the text into sentences, then we pre-process the sentences by extracting the candidate entities and temporal expressions, we calculate the importance score by using a series of features for each pair of entity and temporal expression so that the output will be a list of ranked pairs. The importance score is measured by adding syntactic feature and lexical feature. In this way, we can extract the most important temporal information along with their corresponding entities from a article (plain text) without assistance of any external resources.

The remainder of the paper is organized as follows: Section 2 describes the related work that addresses the same problem using other methods. Section 3 introduces the dataset and methodologies used for detecting and ranking important pairs of entity and temporal tagging,. Section 4 describes the experiments and reports the results and analysis about out experiments. Finally comes the summary and conclusion about our contributions as well as directions for future work.

2 Related Work

There are some papers that studied on this problem, and the concept of knowledge base is addressed to solve it where individual entities and relationships among them are stored in a large repository. Knowledge base is widely used to store complex structured and unstructured information, and knowledge bases like YAGO [4], DBPedia [5] and freebase [6] are popular and public. The format of relation stored in knowledge base is a Subject-Predicate-Object (SPO) triple. Having the property that Knowledge base provides entities along with important facts about entities and other important temporal information, Kuzey et al (2016) [7] proposed a method to detect temponyms and map a temporal tagging into a fact or event if present in knowledge base.

The entire process of detecting temponyms is divided into two parts. The first step is to extract important phrases from the free-text input and detect temponyms from extracted phrases. The second step is to disambiguate temponyms onto events stored in Knowledge base. Temponyms extracted from above methods consist of a large set of candidate entities. To prune candidates with low possibility of being mapped into events, mention-entity Dictionary is introduced to rank them. Top-k candidates are selected and a set of facts from Yago2 is selected if it contains the entity mentioned in top-k candidates. Some measures are applied to describe the relatedness for the mapping between temponym and facts.

However, the method that we proposed in this paper doesnt build on the same ideas proposed in this paper. Knowledge base like DBPedia has a small number of temporal taggings for each entity, but there might be many time points or intervals detected in raw text that we cannot retrieve from knowledge base. So, many of results extracted from plain text will be lost as the knowledge base lacks of temporal information.

Niu et al (2012) [8] proposed the system of Deep Dive to construct knowledge base. Unlike the construction of DBPedia [5] who extracts structured content from Wikipedia, Deep Dive makes the advantage of applying Natural Language Processing (NLP) to extract useful linguistic features from plain text. Deep Dive also performs web-scale statistical learning and inferences using data-management and optimization tools [8].

The basic working process of Deep Dive is to first extract useful relational features from input text, and then train the statistical model that presents the correlations between linguistic patterns and target relations with those extracted features. It then uses markov logic program to combine both statistical model and additional knowledge and transform relational features to relationships of entities. In this paper, both lexical and syntactic features are used to train statistical relation-extraction model. Lexical and syntactic features are both introduced in detail in the work proposed by Mintz et al. [9], and we will also present both of them that we applied in this paper in section 3.

3 Method

The section presents information about how we approached the current task with more detailed description about the data set.

3.1 Data

TODO

3.2 Baseline Algorithm

We first observed some articles that contains entities and meaningful temporal expressions, we find that one entity and its corresponding temporal expression are very likely to occur in the same sentences. For example, one sentence in our dataset is that: *In 2008, Barack Obama became the first African American to be elected president of the United States.*, the entity *Barack Obama* and the temporal expression *2008* appear in the same sentence, and we find this hypothesis are very likely to hold in a well-written article. Therefore, our baseline algorithm uses this simple heuristic by assigning a score to such pair of entity and temporal expression. At the end, all the pairs will be ranked based on the frequency of the co-occurence in one sentence.

However, the main flaws of the baseline algorithm are that: first, the baseline does not catch the situation when the name entity and temporal expression are not in the same sentence. Second, the entities extracted by nltk are noisy, since we observed that some meaningless entities (*e.g D, St*) are also extracted by nltk, which results in a significant degrade of the performance since such meaningless entities will be involved in the output. In this project, we try to tackle these problems by introducing the lexical feature and syntactic feature, which will be discussed more in detail in next section.

3.3 Lexical Feature

In order to solve the situation that the entity and temporal expression are not in the same sentence, we introduce our lexical feature. It is formally defined as:

$$score = \sum(w/d)$$

where w is the weight of lexical feature, and d is the distance (number of sentences) from entity to temporal expressions. For example, one sentence in the dataset is that: Barack Obama was elected over Republican John McCain and was inaugurated on January 20, 2009. Nine months later, he was named the 2009 Nobel Peace Prize laureate. Then entity Barack Obama and the temporal expression 2009 are not in the same sentence, but if we look at the context we will know that they are actually related. In this case, by applying the lexical feature, the pair <Barack Obama, 2009> will also receive partial weight. The way that we implement the lexical feature is summarized by the pseudo code below:

TODO: pseudo code

3.4 Syntactic Feature

We try to tackle the problem of meaningless entities caused by nltk by introducing the syntactic feature. After observed some sentences in the dataset, we found that the entities usually appear at the subject position of a sentence. Therefore, we use the Stanford CoreNLP sentence parser (CITE) to extract the subject of a sentence. If the subject of a sentence happens to be an entity name, then we assign extra weight to such paris that contain the particular entity. In the example discussed in previous section, the sentence: In 2008, Barack Obama became the first African American to be elected president of the United States., the pair <Barack Obama, 2008> will be assigned a extra weight, since the entity Barack Obama is also the subject of the sentence.

Additionally, we take a sample of 10 sentences and we find that the entities appear in the 10 sentences are all proper nouns. The intuition is that the entities of a sentences are usually person names, locations, organizations, and etc. Syntactically, such entities typically have a pos-tagging of NNP (proper noun). Based on this observation, we use nltk pos-tagger (CITE) to extract all the proper nouns that appear in the dataset, and we extend our syntactic feature such that we assign extra weight to the pairs that the entity is a proper noun.

In this way, since the meaningless entities such as D are neither proper nouns nor subjects of a sentence. Assuming the weight given to syntactic feature is appropriate, we can filter out those meaningless entities.

3.5 Evaluation

The evaluation was conducted based on the top-50 results returned from each algorithm. We manually assess each result and report the precision estimated from the top-50 results.

4 Result

The section present the results of current task followed by detailed error analysis.

5 Conclusion

TODO

Acknowledgments

First and far most, we would like to thank prof. Davood Rafiei for the advice and help on this project, also special thank to Michael Su for helpful discussions.