SCU at TREC 2014 Knowledge Base Acceleration Track

Hung Nguyen, Yi Fang

Department of Computer Engineering
Santa Clara University
500 El Camino Real, Santa Clara, CA 95053, USA

Abstract

In this paper, we present our system we developed at Santa Clara University to address the SSF task in TREC KBA 2014. We used the pattern matching method to extract slot values for interested entities from relevant passages. We improved the approach we used last year to enhance the performance. Our system consists of the following steps: processing filtered corpus, retrieving relevant passages, pattern matching, computing scores, and generate results.

1. Introduction

The Knowledge Base Acceleration (KBA) track is a relatively new track in TREC and has been conducted for three years: 2012, 2013, and 2014. In 2012, KBA had only the CCR task. In 2013, KBA had two tasks, CCR and SSF. In 2014, KBA includes three tasks: Vital Filtering, SSF, and Accelerate & Create. Accelerate & Create is a new open task which invites participants to try novel ideas using the track data. In this paper, we present our approach to the SSF task.

2. Streaming Slot Filing

Large knowledge bases such as Wikipedia need to be updated on time and accurately. This task is very time consuming and error-prone if manually performed. The SSF task in KBA invites participants to create a system that can automatically detect the changes in slot value of interest entities. Such systems would be very useful to perform those updating task automatically.

The KBA Stream Corpus 2014 is 16.1 TB after XZ compression and GPG encryption, which is three times larger than the 2013 corpus. This year the organizers provided the filtered corpus which is a specially filtered subset of the full 2014 corpus for use in KBA CCR and KBA SSF tasks. It was filtered using surface form names and slot fill strings from the official query entities for KBA 2014. The total size of the filtered corpus is 639 GB after XZ compressed. This filtered corpus contains 20,494,260 StreamItems which were stored in 2,022,998 chunk files.

3. Our System

Our system consists of the following components:

a) Processing filtered corpus

- Accessing document
- Removing duplicated StreamItems

b) Retrieving relevant passages
   - Building document index
   - Generating seed patterns
   - Extracting relevant passages

c) Pattern Matching
d) Computing score
e) Generate results

The following subsections introduce each component in detail.

3.1 Processing Filtered Corpus

Since we had the filtered corpus, we could skip the filtering process and concentrated more on extracting slot values for entities. The text data in the corpus was serialized by Thrift library[1] into StreamItems, then zipped with XZ utility, and encrypted by GPG. To access data in the corpus, we process it in reverse order. First we decrypted data with GPG, and unzipped the files with XZ utility. After unzipping, the file size becomes more than 15x larger than the original file, and each file can contain one or more StreamItems. To access text data in StreamItem, we deserialized the chunk files to access StreamItem objects. The "clean_visible" text inside StreamItem is the content we would be working on to extract data.

Each StreamItem has a StreamID which is unique in the corpus. We have seen some duplicated StreamItem on the same day. Before proceeding to the next step, we removed those duplicated StreamItems.

3.2 Retrieving Relevant Passages

3.2.1 Building Index

With the *clean_visible* text from StreamItems, the TREC format files were built to be used as input for Indri[2] query language to perform phase matching [1]. From TREC files, Indri will build index of the documents.

```
<DOC>
<DOCNO>1325779697-fc463f58249d873a5b62b8345c1f8162</DOCNO>
<TIME>2012-01-05-16</TIME>
<TEXT>            CERN-PH-EP-2011-140; Submitted to EPJC Letters.

EPJ manuscript No. (will be inserted by editor)
</TEXT>
```

[1] https://thrift.apache.org/
[2] http://www.lemurproject.org/indri/

```
<SENTENCE>CERN-PH-EP-2011-140; Submitted to EPJC Letters.</SENTENCE>
<SENTENCE>EPJ manuscript No. (will be inserted by editor)</SENTENCE>
</DOC>
```

Figure 1. An example file in TREC format

3.2.2 Extracting Relevant Passages

With the index from the previous step, we were able to execute the Indri search function to extract passages which are relevant to the entities and the interested slots. We used the Indri formula #od20(…) to perform the search for related passages. The formula #od20 will request Indri engine to search for the text in ordered windows. The terms must appear in order, with at most 19 words between consecutive terms in parentheses. The figure 2 below shows an example of the query to search for passages which are relevant to entity "Joby Shimomura" and the slot "PER_SCHOOL_ATTENDED". Figure 2 below shows a fragment of the xml input file which contains all queries for Indri engine.

```
<query>
   <number>5005</number>
   <text>#combine[sentence](#od20(Joby Shimomura attends))</text>
</query>
```

Figure 2. An example query in Indri query language

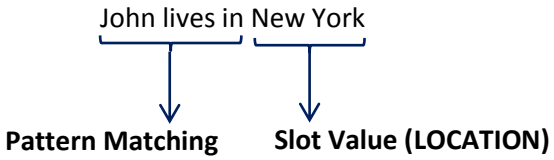
3.3 Pattern Matching

Our pattern matching process relies heavily on pattern generating. A good set of patterns is very important to extract information from passages. For each entity type and each slot name, we came up with a list of seed patterns and used that list to generate the pattern for matching. For example, for the entity type PER (person) and slot name PER_CITIES_OF_RESIDENCE, one of the seed patterns was "lives in". Assuming we have a sentence "John lives in New York", with this pattern, we are able to detect the location "New York" as a value for the slot PER_CITIES_OF_RESIDENCE for the entity "John".

Our pattern matching process this year is very similar to the method we used in KBA TREC 2013 [2] with some enhancement to obtain more accurate result. First, we did not apply the same pattern matching method for every slot type. For example, for the slot "PER_SCHOOLS_ATTENDED", we used the Stanford Named Entity Recognizer (NER) library[3] to obtain the slot value which is of the type <ORGANIZATION>. A slot value with the type <ORGANIZATION> is not necessary a school. We search for a few keywords such as "University", or "College", etc. to determine if this organization is a school. Another example is the slot "PER_COUNTRY_OF_BIRTH". The NER library only returns a slot value of the type <LOCATION>. This value could be a city, a state, a country, or any location. We needed to check the value against the list of all countries in the world to determine if the value is a country.

[3] http://nlp.stanford.edu/index.shtml

Example:



For the example above, we simply need to match the pattern in the sentence, and check if the next word(s) is of type <LOCATION>. If the word is tagged as <LOCATION>, New York in this case, we consider that word(s) is the candidate for slot value. One additional step was that we needed to check the slot value against the list of cities to find a match. In order to do that, we collected list of cities, list of states, and list of countries.

Besides generating seed patterns, the Pattern Matching method also relies on the ability of tagging the words correctly. The Stanford NER library only provides seven types of tag: LOCATION, TIME, PERSON, ORGANIZATION, MONEY, PERCENT, and DATE. Therefore, some slot types can be detected with Stanford NER library. One example is the slot PER_TITLE. For this case, we devised a list of possible titles, such as Mayor, Senator, and appended each of the title to each of the entity name, says Joby Shimomura, we would generate the patterns "Mayor Joby Shimomura", and "Senator Joby Shimomura". We searched for the exact match of those patterns. If we found a match, we could get the value for the slot PER_TITLE for the entity.

3.4 Computing Score

3.4.1 Confidence Score

In TREC KBA 2013, if a slot gets multiple values, the slot value which has the most votes would get the score of 1000 and the slot value which has the least votes would get the score of 1. This year we adjusted our formula to compute the confidence score. The new formula is as follows.

$$ConfidenceScore(V_i) = \frac{c_i * 1000}{\sum_1^n c_i}$$

where ***n*** is the number of different value V_i we retrieve for a slot type for an interested entity, and c_i is the number of votes of value V_i. As with this formula, for example, if a slot gets 10 results, 6 of them have the value of A, and 4 of those have the value of B, with this new formula, A will have the confidence score of 600, and B will have the confidence score of 400.

3.4.2 Vital Ratings

Vital ratings can have the following values

- -1, Garbage
- 0, Neutral
- 1, Useful

- 2, Vital

We submitted multiple runs with different confidence scores as cutoff values to determine the Vital ratings.

4 Preliminary Results

With a few adjustments this year, our system has improved over the last year's. Part of results is shown in Figure 3 and 4, and Table 1.

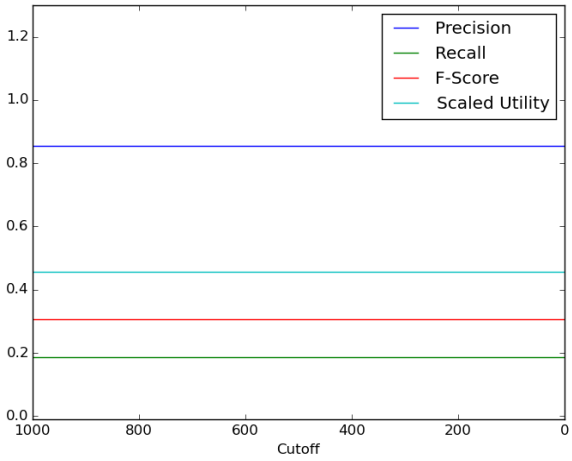


Figure 3. ORG Vital + Useful results

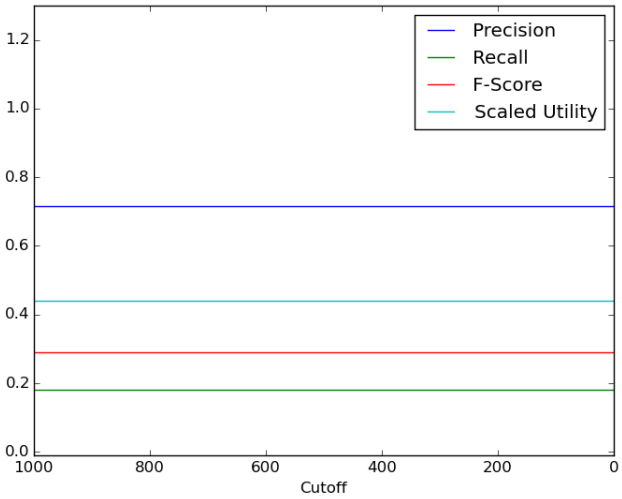


Figure 4. PER Vital + Useful results

Table 1. Our top 3 runs for various evaluation metrics

Sokalsneath	SCU-ssf_6.gz 36.5750775602 SCU-ssf_11.gz 36.5750775602 SCU-ssf_2.gz 36.5750775602
Cosine	SCU-ssf_9.gz 54.5623548418 SCU-ssf_12.gz 54.5623548418 SCU-ssf_8.gz 54.5623548418
Dot	SCU-ssf_6.gz 423.0 SCU-ssf_11.gz 423.0 SCU-ssf_2.gz 423.0
C_TT	SCU-ssf_6.gz 269.0 SCU-ssf_11.gz 269.0 SCU-ssf_2.gz 269.0

5 Future Work

We used the pattern matching method to extract the slot values. Although this approach can detect slot values from passages, it relies on seed patterns. Without comprehensive seed patterns, we may miss some important slot values. We will explore to combine the proposed method with machine learning. Moreover, external knowledge bases such as DBpedia could also be incorporated into the process, which would enhance the corpus filtering process as well as slot value extraction [3].

References

[1] T. Strohman, D. Metzler, and H. Turtle, and B. Croft. Indri: A Language Model-based Search Engine for Complex Queries. In Proceedings of the International conference on Intelligence Analysis, 2004.

[2] H. Nguyen, Y. Fang, S. Gade, V. Mysore, J. Hu, S. Pandit, A. Srinivasan, M. Jiang. A Pattern Matching Approach to Streaming Slot Filling, KBA Track, TREC 2013.

[3] R. Abbes, K. Pinel-Sauvagnat, N. Hernandez, and M. Boughanem. IRIT at TREC Knowledge Base Acceleration 2013: Cumulative Citation Recommendation Task, KBA Track, TREC 2013.