Appears in Text REtrieval Conference (TREC) 2013 Proceedings

Bootstrapping Knowledge Base Acceleration

Tushar Khot, Ce Zhang,
Sriraam Natarajan[+], Christopher Re[*] and Jude Shavlik

University of Wisconsin-Madison, [+]Indiana University, [*]Stanford University

Abstract

The Streaming Slot Filler (SSF) task in TREC Knowledge Base Acceleration track involves detecting changes to slot values (relations) over time. To handle this task, the system needs to extract relations to identify slot-filler values and detect novel values. Being the first attempt at KBA, the biggest challenge that we faced was the scale of the data. We present the approach used by University of Wisconsin for the SSF task and the large scale challenge. We used Elementary, a scalable statistical inference and learning system, developed in University of Wisconsin as our core system. We used Stanford NLP Toolkit to generate parse trees, dependency graphs and named-entity recognition information. These were then converted to features for the logistic regression learner of Elementary. To handle the lack of early SSF training data, we used our existing Knowledge Base Population system to bootstrap a logistic regression model and added rules to handle the new relations.

1 Introduction

The Knowledge Base Acceleration (KBA) track seeks to help humans expand knowledge bases like Wikipedia by automatically recommending edits based on incoming content streams. To this end, KBA systems must filter a large stream of text to find changes to a knowledge base. To recognize changes in knowledge base profiles for particular entities of interest, a KBA system will have to extract relations (slot fillers) for these entities from the corpus stream and find novel relations from these extractions.

Given the scale of the streaming corpus, running a relation extraction system on all the documents was infeasible. Hence, our system had to filter down the documents for relation extraction. Moreover, the filtering step needs to be much faster than the relation extraction step. Even after filtering the documents, we need a scalable learning and inference system to perform relation extraction. Since many of the relations in this task were new, we also needed a flexible system which allows us to easily specify new features / rules for these relations.

We used basic string search to quickly filter out irrelevant documents (that do not mention the entities of interest). From the filtered list of documents, we extracted relations using Elementary [6, 7], a statistical inference and learning system. A key advantage of the Elementary system is that it is a prototype system that scales to very large corpuses. A secondary advantage of this system is that it uses the richer representation of a probabilistic logic formation called Markov Logic [2] allowing to model and capture rules that are likely but not certain to be correct. We used the Stanford NLP toolkit[1] to extract parse trees, dependency graphs and named entities to generate the features necessary for Elementary. We used the model learned for TAC-KBP 2010 [4] by mapping the relations from KBP domain to the KBA task. We designed features to handle the new relations and entity types.

We also learned various lessons in our first attempt at KBA. Although we filtered down the doc-

[1] http://nlp.stanford.edu/downloads/corenlp.shtml

uments for relation extraction, we still had to download all the data and decrypt it locally which was time-consuming. In hindsight, performing the filtering on Amazon Cloud would have been more efficient. We also assumed given the size of the corpus, most of the relations would be mentioned multiple times. Hence we did not rely on having all the possible rules for each relation, but on capturing the common cases. But since the set of entities were not popular, we were not able to extract many slot values for these entities.

The rest of the paper is organized as follows. In Section 2, we present few details about the Streaming Slot-Filling task (SSF) that we worked on. Following that, we present the details of our approach in Section 3. Finally, we briefly discuss the results of our approach and future steps to further improve them.

2 Background

The KBA track provides a large streaming corpus by breaking the documents up into hourly chunks which can then be processed sequentially. The track also provides a list of target entities represented as links to Wikipedia or Twitter pages. Given the corpus and entity list as input, the track consists of two tasks:

- Cumulative Citation Recommendation (CCR) task. The CCR task involves filtering documents worth citing in a profile of a target entity. The system needs to recognize whether a document is *useful* (time-invariant e.g., place of birth) or *vital* (timely e.g. title).
- Streaming Slot Filling (SSF) task. The SSF task involves detecting changes to slot filler values (relations) for target entities. The system needs to extract slot filler values for target entities and then recognize changes to slot values. We concentrated only on this task in our system.

3 Our Approach

The three key challenges that we faced with Streaming Slot Filling task were:

1. Handling the large scale of data
2. Extracting slot values for target entities
3. Detecting novel slot values

To solve each of these problems, we developed heuristic approaches that we outline in this section.

3.1 Large scale

Due to the scale of the proposed task, it is not feasible to perform relation extraction on the entire corpus. Similarly, employing many of the standard, publicly available natural language processing tools would not be feasible as well. In order to make the corpus size manageable, we searched for target entities (and variants of their names) in the articles. For e.g., if "William Smith" is a person of interest, we will accept any document that mentions "William", "Bill" or "Smith". If an article contains any useful information about a target entity, we assume that some variant of the target entity name will be mentioned in the article. This heuristic is very fast as it does not require any processing to be done on the article and we only need to find the first match for a target entity in the article. We use a simplified version of the Aho-Corasick algorithm [1] since we only need to find the first match. This reduces the size of the corpus from ∼60TB to a manageable ∼1TB. We implemented parallel version of the filtering algorithm on these articles to further speed up our system. Since we ran this step on our local cluster with limited storage space, this step still took a week to run. Going forward, we will move this step to the Amazon cloud service and reduce the amount of data downloaded to our local cluster.

3.2 Extracting slot values

To perform relation extraction, we first extracted parse trees and dependency graphs for sentences in the filtered articles. Although TREC does provide some basic NLP annotations as part of the corpus, we employed Stanford toolkit since we needed dependency graphs as features for relation extraction. We then used the Elementary system developed at University of Wisconsin-Madison to perform relation extraction. At a high level, Elementary first creates the

potential mentions based on the named entities. It then creates a list of potential relations between pairs of mentions in the same sentence. For each potential relation, the path in the dependency graph and parse tree is calculated and used as features for a logistic regression model. The weights are learned for each feature and each relation type using gold-standard training data as well as distant supervision examples [5]. We also manually can specify rules which can be as simple as just specifying feature weights or even constraints on the relations. For further details about Elementary, please refer to Niu et al. [6].

Once these features were obtained, we needed to learn the weighs for features in Elementary. It should be mentioned that for the initial runs of the SSF task, there was no available training data. Hence, we used a model that had been learned using distant supervision on a subset of the relations for the Knowledge Base Population (TAC KBP 2010) task [4]. We found mappings between the relation names from KBP task to the KBA task. The mapping between the KBA and KBP relations is shown in Table 1. For the relations in KBA that couldn't be mapped, we manually created features based on few sample sentences. To do so, we searched for sample sentences containing these relations and added the corresponding dependency path or parse tree features to the model. We assumed that capturing few features for these new relations would be sufficient. Given the scale of the data, we assumed that at least one mention of a valid relation will be extracted using a simple relation extractor. Once we extracted the relations, we employed Elementary's entity-linking model to link the slot filler entities with the target entities. We filtered out the relations that did not include any target entities.

KBP Relations	KBA Relation
$per : date\_of\_death$	DateOfDeath
$per : title$	Titles
$per : spouse$	SignificantOther
$per : employee\_of$, $per : member\_of$, $org : member\_of$, $org : top\_employees$	EmployeeOf
$org : top\_members$	TopMembers
$org : subsidiaries$, $per : schools\_attended$	Affiliate

Table 1: Mapping between KBP and KBA relations. All extractions of the relation type on the left were marked as KBA relations of the type on right.

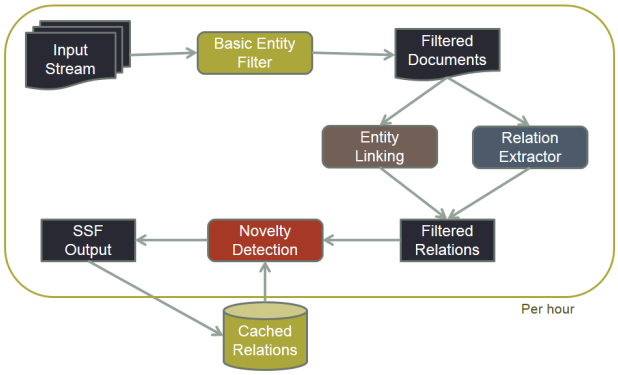

Figure 1: Overall System Design.

3.3 Novelty detection

Given the extracted relations, we processed them chronologically based on the document time to check for any change in slot value. Unlike the previous steps, this step can not be performed in parallel. For every relation, we check if we have already extracted any relations of the same type for the same entity. If not, we accept this relation as a novel slot value. If we already have a slot value extracted, we compare the slot values and accept the new relation as novel if the edit distance between the two values is large enough.

Figure 1 shows the overall system design as a flowchart. As mentioned before, we filter the documents using a basic entity filter. We perform Entity Linking and Relation Extraction using Elementary. We only accept relations over mentions linked to the target entities. We then perform our basic novelty detection over the stream of extracted relations where the previous extraction are cached. Only the novel relations are then used to create the output for the SSF task.

4 Results

We now discuss the results of our system on the SSF task. There were two evaluation measures used by TREC for this task: (1) average F-1 score and (2) average Scaled Utility (SU) [3]. Each score is calculated over four stages in a pipeline:

- **SSF-DOCS**: In this stage, the system's capability to identify the slot type in a document is evaluated. The scores are calculated by comparing the slot types of the output run against the annotations (the slot values are ignored).
- **SSF-OVERLAP**: This stage evaluates the slot values of the output run by checking for overlap with the annotations, but only considers the true positives from the previous stage.
- **SSF-FILL**: This stage too takes as input the true positives from the previous stage but checks if the output run recognizes the equivalence between the same slot values.
- **SSF-DATE_HOUR**: This stage checks if the system is able to recognize the first occurrence of the slot value and ignores the duplicates.

Since our approach concentrated on a subset of the relations, the F1 score on any of these measures averaged over all the relations was very low. In general for all the stages, our F1 score was close to zero but the scaled utility was close to the median value. Also, our basic assumption that most of the relations would be mentioned multiple times in multiple ways was flawed. Since we only relied on capturing the common cases, we missed many extractions for the uncommon entities resulting in a low recall.

5 Conclusion

We present the design of our system for the streaming slot-filling task. We use a high-throughput and high-recall filter to get a smaller corpus for the computationally intensive relation extraction step. We use a scalable and highly flexible system, Elementary to perform relation extraction. We bootstrapped this system by transferring the learned model from TAC KBP 2010 thereby circumventing the need for training examples.

We have learned various lessons in our first attempt at this task. Going forward, we will move the filtering task to Amazon Cloud to make it much more computationally efficient. We will work on training a model for all the relations of this domain via distant supervision. We also will work on manually evaluating our results to check for the sources of errors in our system.

Acknowledgements

The authors gratefully acknowledge support of the DARPA DEFT Program under the Air Force Research Laboratory (AFRL) prime contract no. FA8750-13-2-0039. Any opinions, findings, and conclusion or recommendations expressed in this material are those of the authors and do not necessarily reflect the view of the DARPA, AFRL, ARO, or the US government.

References

[1] A. V. Aho and M. J. Corasick. Efficient string matching: an aid to bibliographic search. *Communications of the ACM*, 18(6), 1975.

[2] P. Domingos and D. Lowd. *Markov Logic: An Interface Layer for AI*. Morgan & Claypool, San Rafael, CA, 2009.

[3] D. Hull and S. Robertson. The TREC–8 filtering track final report. In *TREC*, 1999.

[4] H. Ji, R. Grishman, H. Dang, and K. Griffit. An overview of the TAC 2010 knowledge base population track. In *TAC*, 2010.

[5] M. Mintz, S. Bills, R. Snow, and D. Jurafsky. Distant supervision for relation extraction without labeled data. In *ACL*, 2009.

[6] Feng Niu, Ce Zhang, Christopher Ré, and Jude Shavlik. Elementary: Large-scale knowledge-base construction via machine learning and statistical inference. *IJSWIS Special Issue on Web-Scale Knowledge Extraction*, 2012.

[7] Ce Zhang and Christopher R. Towards high-throughput gibbs sampling at scale: A study across storage managers. 2013.