Streaming Slot Filling

Aju T. Scaria Anshul Mittal Bharath Bhat Advisor: Jeffrey Ullman

June 10, 2013

The web has an enormous amount of information stored as text content and it grows by the day. Building systems that could extract meaningful information from this large, freely available data repository would help us to maintain upto date information about different events and people. According to statistics released by the Text REtrieval Conference (TREC), it takes around 365 days on the average for a news article to be cited in Wikipedia after it has been published. This lag of almost a year is quite alarming considering the tremendous advancements in the field of data warehousing and information extraction in the recent years. TREC promotes research in this area by introducing a Knowledge Base Acceleration (KBA) track specifically targeting extracting information from a huge web corpus collected from different sources. The KBA systems must filter a large stream of text to find documents that can help update a knowledge base like Wikipedia. In this paper, we present our approach and models to extract information about a set of entities (people, facilities and organizations) released as part of the TREC KBA 2013 Streaming Slot Filling (SSF) task.

1 Problem statement

The SSF task under TREC KBA 2013 track targets extracting relevant information about target entities. Entities are identified by their Wikipedia or twitter page and can be of three types:

- Person (PER): These are people, for eg: Willilam H. Gates Sr., Aharan Barak
- Facility (FAC): These are facilites like schools, museums, railway etc. Eg. Stuart Powell Field, Lake Weir High School.
- Organization (ORG): These are establishments owned by a person or a group, Eg. Scotiabank Peru, Austral Group.

There are 125 entities of type PER, 24 of FAC and 21 of ORG.

The information about these entities are extracted by a predefined set of slots. A slot is an attribute of the target entity such as awards won by a person or the founder of an organization. For example, given the sentence, Bill Gates is the founder of the multi billion dollar company Microsoft, we could extract the slot value of Bill Gates for the slot Founder Of for the entity Microsoft. Given slots for each of the target entities, the SSF task is to detect changes to the slot value, such as location of next performance or founder for each hour in the corpus. The slots that needs to be filled for an entity will depend on its type. The list of slots per type is given in Table 1.

Some key points about the streaming slot filling task are:

Table 1: List of Slots

Entity Type	Slot Name	Description
PER	Affiliate	This slot captures affiliation between a person and a facility or an organization, like an employee, founder etc.
	AssociateOf	This slot captures the affiation of the target entity with other people.
	Contact_Meet_PlaceTime	This slot is used to capture the different locations at which the target entity was present along with the time.
	AwardsWon	This slot is to identify the awards won by the person.
	DateOfDeath	Teh date at which a person died.
	CauseOfDeath	The cause of a death of a person.
	Titles	Important titles held by a person, like Chairman of Yahoo! or CEO of Google etc.
	FounderOf	The organizations founded by the target entity.
	EmployeeOf	The organizations that the target entity has worked for.
FAC	Affiliate	This slot is to find the affiliation of the facility with another facility or organization or a person.
	Contact_Meet_Entity	This slot is to find the people that were present at the target facility.
ORG	Affiliate	This slot is to find the affiliation of the organization with another facility or organization or a person.
	TopMembers	This slot is to identify the top members of the target orgaziation like the founder, CEO, CTO, CFO, chairman, VP etc.
	FoundedBy	The people who founded the target organization.

- We need to determine the earliest time (on an hourly granularity) at which there is evidence for a slot change. This means that if a news article mentions a slot change on say 2012-06-15 between 10 am and 11 am, if our system doesnt capture this change during this hour, then we will be penalized for recall, but at the same time if we gather more evidence in the next hour and predict it, then it will be an incorrect prediction affecting our precision as well.
- Ground truth determined post-hoc: There is no training or development set provided as a part of the contest. The ground truth data will be generated post-hoc after the submissions from different teams have been pooled in and assessed manually by National Institute of Standards and Technology (NIST) assessors

2 Dataset

The dataset provided by TREC committee consists of approximately 4.5 TB of compressed and thrift serialized (1.046 billion documents) unstructured data obtained by crawling the web. It contains news articles, blog entries, social data, including twitter, yahoo answers etc. and

research papers from arxiv. On an average, there are around 10^5 articles per hour. In addition to the body of the pages, the dataset consists of NER and coreference information obtained using LingPipe (although the NER information is not really good).

In addition, we also used the Google Cross Lingual Dictionary (GCLD) dataset. It contains a mapping from common anchor texts to wikipedia concepts, spanning 7,560,141 concepts and 175,100,788 unique text strings.

3 Related work

Although the Streaming Slot Filling task was introduced by TREC this year, another closely related task - Cumulative Citation Recommendation (CCR) has been ongoing for a couple of years on a subset of the same corpus. The CCR task deals with having to find relevant documents for a given set of entities, and hence is an important subset of our task as well. We went through the reports from previous submissions to get a sense of the tools and approaches most commonly used.

1. A common thread across all teams was the use of a search engine to index documents and make querying faster. This led us to the open source search engine Indri from the Lemur Project, details of which are in the subsequent sections. http://www.lemurproject.org/indri/

2. In the current corpus, only 0.4% of the documents that do not explicitly mention an entity are relevant. This means that we dont have to worry about retrieving documents that contain an indirect reference to our entity of interest. On the other hand, out of documents that mention an entity, 23.8% are garbage, 35.3% are garbage or neutral, and the rest relevant. This means that our focus should be on aggressively pruning out documents.

3. The existence of training data for this task enabled teams to use supervised learning approaches (SVMs, Markov Random Fields). The SSF task does not have any training data and hence we have to depend on unsupervised approaches such as bootstrapping.

Bootstrapping as a technique for extracting relations from text has been around for some time. Some of the previous works use Freebase, a large semantic database of several thousand relations, to provide distant supervision. Mintz et. al(2009) uses a classifier based approach to extract relations. For each pair of entities that appears in some Freebase relation, they find all sentences containing those entities in a large unlabeled corpus and extract textual features to train a relation classier. This method is advantageous because it automatically extracts relations which is line with what what we wanted to do. At the same time, some of the slot values were not entities (e.g. AwardWon, Contact_Meet_PlaceTime). Sergey Brin (1998) used a bootstrapping approach for extracting book-author pairs from web corpuses. But, his methods were based on co-occurrence of pattern words. Banko et. al. uses a classifier based approach to extract relations based on the words that appear in between. These methods wouldnt scale for corpora of the size of the TREC KBA as classifiers had to be trained and features extracted for a several combinations of entity pairs. As a result, we use dependency parse of a sentence for relation extraction, which makes it easier to just look at the desired edges between patterns and extract the information we need. Another motivation to use the dependency parse was that the state of the art NLP parsers perform quite well and makes it easier to extract patterns in comparison to classifier based approaches.

4 overview

The streaming slot filling task serves as a middle ground for techniques and ideas from both the information retrieval and natural language processing community. There were a lot of key factors that affected the design decisions and algorithms we devised during the course of the project. In this section, we first introduce some the unique challenges that the SSF task poses and how it influences our design choices and then we cover the high level system design.

4.1 Challenges

The task of streaming slot filling based on the content from the web is challenging because of several reasons:

- Different ways to represent the same entity: The same entity can be referred by different names. Hence, if we just rely on the actual name based on the task definition, the system would have poor recall.
- Size of data: The size of the corpus was so large that the data couldnt be uncompressed and stored at a single place for learning models based on it. Also, the sheer volume of data makes it impossible to make multiple passes over the entire dataset throughout the course of the project. Hence the data had to be filtered to consider only relevant documents.
- Disambiguation: There could be multiple entities with the same name. Hence, the appearance of the name of an entity in a document doesnt necessarily mean that the document talks about the target entity of our interest.
- Finding relationships for slots: Extracting slot values from text is a hard task because the same relationship could be expressed in a variety of ways and also, finding the boundaries for the slot value poses several challenges as well.
- Nature of data: The dataset was collected by crawling the web and most of the documents had ill formed social content from blogs, facebook and twitter. There were embedded html content, advertisements, links to many pages, etc within the corpus.
- Nature of slots: Slots like Affiliate, Contact_Meet_PlaceTime were not very well defined and drawing lines as to what qualifies as an appropriate slot value was hard.
- Parsing algorithms: Natural language is complex to deal with and the sytem we built relies heavily on the performance of the constituency and dependency parsing algorithms. As a result, all the incorrect parses generated affect the performance of our system.

Our methods and algorithms described in the paper are designed to overcome the challenges mentioned above.

4.2 High Level Design

Our solution to the challenges discussed above can be broadly broken down into three major components:

Indexing
In order to perform fast retrieval of documents, it is imperative that we index them. We use Indri to index the documents in our corpus. However, the size of our corpus is

huge (~4.5 TB uncompressed), and not all documents (in fact very few) are relevant to our set of entities. Therefore, in order to facilitate faster querying of relevant documents, we indexed documents based on a high recall filter, the details of which are discussed in Section 6. This filter ensures that almost all the relevant documents are indexed and are available at our disposal to learn patterns and to predict slot changes.

Extracting Relevant Documents
Once the documents have been indexed, we need to extract the relevant documents for every entity. This task is more tricky than it may sound because of the challenges mentioned above. Specifically, there can be (and generally are) multiple people with the same name, e.g. Boris Berezovsky, the businessman and Boris Berezovsky, the pianist, and the same person may be referred to with different names, e.g. William Gates Jr. as Bill Gates etc. Therefore, we use a disambiguation logic, discussed in Section 7, to deal with such issues.

Finding Slot Values
Once the relevant documents have been extracted, we need to find whether they contain information for any of the slots. In order to this, we need to develop a mechanism to detect whether a sentence contains information about any slot, and if it does, we need to extract the value of the slot from that sentence. Moreover, since the sentences may not be very well formed, especially in the social context, our method needs to be flexible enough to accommodate that. In Section 8, we delve deeper into how we obtain slot values.

Our system relies heavily on natural language parsing and we use Stanfords NLP package for the purpose of our project as it is widely considered to be the best off-the-shelf NLP package. An underlying philosophy behind most of our design decisions has been automation and scalability. We have sacrificed accuracy at times to ensure that the solution can be implemented at scale. For instance, for disambiguation and entity expansion, we used automated methods that extracted information from Wikipedia and GCLD, which could be noisy. Doing a manual pruning of the entities would make it more specific and help us achieve better performance, but is not scalable when we have a relatively larger list of entities.

5 Indexing

Indexing is a central part of the infrastructure of our solution. It is needed to solve our problem in a reasonable time so that it is useful in a streaming setting. However, to support faster querying, it is also important that we minimise our index size. Therefore, we index only those documents which are potentially relevant to our entities. In order to do this, we first created a high recall entity expansion set as explained below.

5.1 High Recall filter using entity expansion

Documents across the web refer to the same person in different ways. Some articles may be very formal in introducing an entity and may have their full names contained in it, whereas some other documents may have part of their names or may contain initials. For example the entity Alexandar McCall Smith could be refered by his full name or by Alexandar Smith or just A. Smith. To overcome this, we adopted an approach to expand entities based on:

1. Google Cross Lingual Dictionary (GCLD). The GCLD is a ready-made resource associating Wikipedia entries to strings. For each Wikipedia based entity, we extracted the

different anchor texts that are used across the web to refer to the Wikipedia page. Since there were many anchors that contained junk like Click here to navigate to Wikipedia article, we adopted a character based Jaccard similarity metric to weed out anchor text entries that were not relevant. A GCLD entry that has more than 0.5 character Jaccard similarity with the target entity were used as expansions of the entity.

2. Manual expansion: Some entities had very few expansions. In addition, the twitter entities were specified by their handles and had to be manually expanded.

3. Permuting tokens in names: In many instances, for entities with long names, only a part of the entities name was present in the document. In order to cover these documents, we generated permutations of two tokens based on the following strategy:

(a) If there were more than two tokens, for example, Alexander McCall Smith, we generate combinations of two tokes $t_i and t_j so that i < j. If there are exactly two tokens, generate both combinations of n$

In addition, for all entities, we are adding the last name alone as an expansion. This turns out to be useful specifically in the arxiv documents that includes author names in shorthand forms like S. Kachru.

5.2 Indexing with high recall filter

We used the Indri indexing system to index our documents. We chose Indri as it has very good indexing and querying performance, as has been successfully by other TREC participants in the past. The indexing workflow involved the following steps:

(i) For each hour, download all the documents and build a larger temporary index with all the documents.

2. Query this index to obtain relevant documents for each of our entities. This is done by performing an OR query over the entity expansions described in the previous section.

3. Index only the documents retrieved in step (ii) and delete the temporary index. This index is backed up on S3 and includes all documents that are relevant for a given hour.

4. To avoid downloading the relevant documents again for any subsequent processing, we keep the set of relevant documents in a serialized format on S3.

Steps (i) and (ii) of the process were parallelized to take advantage of the multi-core machines on EC2. The entire workflow itself was run in parallel for multiple hours on the biggest machines on EC2. (16 cores and 60 GB RAM).

For each hour, the high recall filter captured about 1 in every 20 documents. The indexing process took five minutes on average to process each hour.

6 Extracting relevant sentences

7 Finding slot values

8 Results

9 Conclusion

10 References

References

[1] John W. Dower *Readings compiled for History 21.479.* 1991.

[2] The Japan Reader *Imperial Japan 1800-1945* 1973: Random House, N.Y.

[3] E. H. Norman *Japan's emergence as a modern state* 1940: International Secretariat, Institute of Pacific Relations.

[4] Bob Tadashi Wakabayashi *Anti-Foreignism and Western Learning in Early-Modern Japan* 1986: Harvard University Press.