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 ($\tilde{4}$.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.

   (b) If there are exactly two tokens, generate both combinations of names. t_1 t_2 and t_2 t_1

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:

1. 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

Once the documents have been indexed, our next task is to extract relevant sentences for each entity. A simple solution to this problem could be to look for an exact match of our entity name in the document. As said before, only 0.4% of the relevant documents for an entity dont contain an explicit mention, therefore this strategy is certainly good. However, there are three potential problems with this approach:

- The same entity is referred to with different names, e.g. William Gates as W. Gates or Bill Gates, Robert Stovall as Bob Stovall etc.

- Different people have the same name, e.g. Boris Berezovsky, the businessman and Boris Berezovsky, the pianist, or Basic Element, the company and Basic Element, the music group.

- A document mentions two people with a part of the name being common. For e.g. a document overall may mention Henry Gutierrez, however, it may also contain a mention for Priscilla Gutierrez. This may be a problem because if a sentence uses only the word Gutierrez, we need to know which Gutierrez it refers to.

In order to counter these problems, we use explicit mentions along with a combination of disambiguation strategies to find the relevant sentences.

Our first task is to find out documents which are relevant to our entity. In order to do this, we created a set of disambiguation words relevant to each entity. This set was created using the wikipedia pages for these entities. The process of creating the disambiguation sets had the following three steps:

- **Anchors**
  Find top 10 (by count) anchor texts on the wikipedia page of the entity and consider all tokens occurring in them. We consider only the top 10 to consider only the significant concepts representing the entity. The logic here is that the outlinks refer to concepts (people, places etc.) relevant to the entity and will therefore provide valuable disambiguation information. Table 2 shows the disambiguation set for two entities obtained using this method.

- **Categories**
  Every wikipedia page is annotated with information about the different categories that the entity belongs to on wikipedia. This information is valuable in providing most relevant concepts relating to the entity and we, therefore, consider all tokens occurring in these categories as well.

- **Pruning**
  While the above two methods together give us a very good disambiguation set, they generally include some very common words, which dont add any value to the set. We, therefore, pruned them manually. Although, we did this step manually, it can also be automated by considering the document frequency of the words in wikipedia.

Table 3 shows the final disambiguation set for two entities obtained after the three steps.

For every entity, while retrieving relevant documents, we estimate how many of the disambiguation words are contained in the document. Only when this number crosses a certain threshold do we consider the document to be relevant. We experimented with different threshold values and settled on a value of 0.1. However, in the absence of a training set, this value was based more on manual screening of several runs. Moreover, we also observed that the same threshold does not seem to work for different entities, however, there was no clear way to learn different threshold values for different entities.

Once the relevance of a document has been estimated, we need to extract all sentences within the document which are relevant to our entity. In order to this, we consider all possible unigrams of the expansions for that entity and retrieve all sentences which contain that. We do this because a sentence may refer to the entity only with the first or last name within the document. However, this necessitates disambiguation even at the sentence level. We do this using the following two methods:

Boris Berezovsky (Businessman)	Boris Berezovsky (Pianist)
Alexander Litvinenko	Hamish Milne
Roman Abramovich	Violin
Vladimir Putin	Sergei Rachmaninoff
Nikolai Glushkov	Moscow
Federal Security Service (Russia)	Russia
Yevgeny Primakov	Leopold Godowsky
Aeroflot	Piano Trio No. 2 (Shostakovich)
Paul Klebnikov	Virtual International Authority File
High Court of Justice	Pianist
The Guardian	Swedish Chamber Orchestra

Table 2: Disambiguation set obtained using anchors

Boris Berezovsky (Businessman)	Boris Berezovsky (Pianist)
nikolai, christian, mathematician, activist, abramovich, england, oil, financier, hanging, owner, polish, paul, suicide, wanted, aeroflot, descent, service, federal, writer, sunninghill, christianity, member, jewish, academy, litvinenko, local, kingdom, whitecollar, putin, orthodox, klebnikov, guardian, moscow, fugitive, emigrant, glushkov, billionaire, political	sergei, trio, violin, swedish, tchaikovsky, pianoforte, orchestra, classical, milne, international, piano, pianist, prizewinners, hamish, moscow, leeds, russian, shostakovich, leopold, conservatory, rachmaninoff, chamber, godowsky, russia

Table 3: Final disambiguation set for two entities

- **Coreference resolution**
  We use Stanford NLP parser to resolve the coreferences of the mentions so that we know that the pronominal references point to our entity and not some other entity. For example, in the sentence, Henry Gutierrez met his wife Priscilla Gutierrez while he was working at IBM, we conclude that he refers to Henry and is therefore relevant to us.

- **Word Expansion**
  Since we retrieve sentences which have all possible unigrams, it is possible that the sentence actually does not refer to our entity at all. For example, the sentence, Priscilla Gutierrez works at Microsoft, will be retrieved for the entity Henry Gutierrez. However, it is not relevant to our entity. In order to handle such scenarios, we expand the entity mention into the noun phrase and consider it relevant only if all the tokens are present within our entitys expansion set. In this example, Gutierrez will be expanded to Priscilla Gutierrez, and since the token Priscilla doesnt occur in Henry Gutierrezs expansion set, this sentence will be ignored.

While retrieving relevant sentences, we also clean them up (non-ascii characters etc.) and split them further wherever necessary, especially for social documents like twitter.

7 Finding slot values

Now that the relevant sentences for each entity have been extracted, we need to find out whether those sentences contain any information about our slots and find the value for those slots. In order to do this, we define the following properties of each slot:

- **TargetNERTypes:** set of possible NER Types that the slot value can have; NONE if it may not have an NER value
- **Patterns:** we build patterns for each slot using the data provided to us and use those patterns to find slot values.
- **Threshold:** We associate a threshold with each slot which denotes above what confidence score we should predict a value for the slot.

Now, lets talk about how we build the pattern sets and find confidence scores for each slot.

7.1 Pattern finding for slots

We use a combination of bootstrapping and rule based approaches to construct our patterns.

Bootstrapping
We use the bootstrapping method to find patterns for the different slots given to us as part of the problem. Since the index we created based on high recall filter had only those documents that mentioned the 170 entities, we created a separate index for around 21 days that indexed all the documents during this period (April 1 to 21, 2012). The choice of 21 days was made so that the dataset fit into the 1 TB hard disk size supported by the machines we used for indexing. The way the bootstrapping method works with is as follows:

1. We start with a manual seed-set, i.e. (entity, slot-value) pairs, for each slot
2. We extract documents that contain the entity and the slot-value and find the pattern connecting the entity and its slot value in the dependency parse of the sentence.

The seed-set used for AwardsWon is shown in Table 4. The seed-set is constructed manually and can be (and most likely is) biased, despite our attempts to make it as representative as possible. We refer to the entity whose slot value has to be found as the source. The target is the desired slot-value we wish to extract from the sentence for any slot. And the pattern word is the word in the sentence that indicates the slot. For instance, the word married is a pattern word for AssociateOf relation between two people. Now, a pattern is defined by the pattern word and a set of at most two dependency rules that connect the source, target and the pattern word. The patterns we obtained using bootstrapping falls under four primary categories (based on the location of the pattern word in the path):

1. **WordInBetween:** pattern word in the middle. These are the cases in which the pattern word appears in between the source and target entities in the dependency tree.Meryl Streep has won the Oscar Award for best actress at the Academy Award ceremony currently taking place at the Kodak Theatre in Hollywood.
2. **SourceInBetween:** In this case, the source word connects the target and the pattern word. Oscar award winner Mery Streep will talk in the ceremony on Monday.

Entity	Award
Meryl Streep	Oscar
Meryl Streep	Academy award
Meryl Streep	award
Geno Auriemma	Wooden Award
Anthony Davis	Naismith Award
Michel Hazanavicius	Oscar
Eli Sanders	Pulitzer prize
Mary Schmich	Pulitzer prize
Wesley Morris	Pulitzer prize
Hilary Mantel	Booker prize
Judea Pearl	Turing award
Saul Perlmutter	Nobel Prize
Tawakkol Karman	Nobel Prize
Leymah Gbowee	Nobel Prize

Table 4: Seed set for the slot AwardsWon

3. **TargetInBetween:** In this case, the target word connects the source and the pattern word. Microsoft founder Bill Gates is popular for his acts of philanthropy.
4. **NoPatternWord:** source word connected directly to target by a dependency graph edge. Goldwasser's Weizmann Institute page.

Some patterns found using bootstrapping for the slot EmployeeOf are shown in Table 5.

Pattern Word	Pattern Type	Rule 1	Rule 2
ceo	WordInBetween	appos	appos
vp	SourceInBetween	nn	nn
engineer	WordInBetween	appos	prep_at
reporter	SourceInBetween	nn	nn
spokesperson	WordInBetween	nsubj	prep_at

Table 5: Some patterns obtained using bootstrap for the slot EmployeeOf

Computing confidence thresholds

The confidence with which we can tell that a pattern represents a desired slot depends on:

– How common it is used for a specific seed set entry?

– How prevalent it is across different seed sets?

We dont want to give a lot of importance to those patterns that just appear in one seed set. If we used absolute counts as weights, it could lead to one pattern that appeared for one seed set to get unusually high weight. So, we adopted the following strategy:

1. Normalize weights: The total weight for a seed set was fixed to 1 and each pattern found splits this weight proportionally.
2. Discard the lowest weight from across alll seed set. If a pattern appears just once across all the seed sets, it is likely that the pattern was not indicating the slot that we want to capture. So, we discount the lowest weight found.

3. Up weight frequent patterns: If a pattern appears in almost all seedsets, it should receive high confidence score. So, we upweight the total weight accumulated over the different patterns by multiplying it with the number of seed sets that extracted the patterns. Overall formula for computing confidence

Rule based
For some slots, the dependency parsing alone does not provide enough evidence. For example, for the slot Titles, the title can also be mentioned as a coreference to the entity in the sentence. For example, in the sentence, Marissa Mayer, VP of Google, was also seen at the event. Another such example is Nobel Laureate Amartya Sen gave a lecture at the event. Such values cannot be captured using the dependency parse of the sentence. Therefore, we use other rulesalso, in addition to the dependency parser, to find slot values.

It is important to note that the same strategy does not work for all kinds of documents. For example, while the sentences in the news articles are generally well formed, and therefore suitable for dependency parsing, the social documents can be very badly formed and therefore, not conducive to such strategies. The arxiv documents are another example where such dependency parses wont work as the research papers have valuable associate information in the authors and references sections which cannot be obtained using a dependency parse. Therefore, we follow different strategies for different types of documents: for arxiv documents we exploit the systematic nature of paper writing and explicitly extract the information in different sections whereas for social documents, instead of relying on a dependency parse, we do a range query on our entity, the pattern word and a target word of appropriate entity type. News documents are generally well formed and we use the bootstrapped patterns directly on them.

7.2 Finding slot values

We use the patterns, rules and confidence thresholds computed in the previous step to find slot values in the set of relevant sentences. This is done using simple pattern matching, i.e. we find if a similar path exists in our sentence connecting our entity to a target word of the appropriate NER type. We do this for all sentences for the entity during the hour and accumulate the confidence scores of each pattern that matched and gave a slot value. In theory, this confidence score can be used to prune all slot values which lie below a certain threshold. However, in practice, we observed that the number of relevant sentences actually generating something useful for a slot was very low (1̃-2 per hour) and therefore, not significant enough for a threshold cutoff.

Another important step here is that of categorizing slot values into equivalence classes. As mentioned before, the same entity can be referred to in very different ways, these variations being even more prominent for facilities and organizations. However, when predicting slot values, we want to collapse all values which refer to the same entity into one equivalence class. Not only is this a requirement of the TREC-KBA submissions, it also helps in accumulating confidence scores for an equivalence class and making a better decision. We followed two major approaches to do this:

Dictionary based
We used the Google Cross Lingual Dictionary, which contains the different anchor texts used to point to a concept on wikipedia, to map a word to a wikipedia concept. While this system performed well, especially for facilities and organizations (it maps both Seagram Company and Seagram Inc Ltd. to Seagram, I.I.T. Delhi to Indian Institue of Technology

Delhi, both Oscar and Oscar winner to Academy Award etc.), it is not the best way for us for two primary reasons. Firstly, because this method is dictionary based, it doesnt take context into account. Therefore, even though William may refer to different people in different contexts, it will always map them to the same person. It is also not suitable for situations where the target entity may not have a corresponding wikipedia page. Secondly, for the purpose of our submission, we need only the equivalence class and not the concept representing the equivalence class.

Approximate matching based
Due to the drawbacks associated with the above method, we adopt a simple strategy where we simply find the k-gram jaccard similarity between two candidate values to predict whether they belong to the same equivalence class or not, the assumption again being that the document will have an explicit mention of the entity, and our system will therefore catch it.

8 Results

The models that we built cannot be evaluated directly against the dataset as the TREC KBA SSF task hasnt released any training or development sets to work with. As a result, one approach we used throughout the project to track progress is to eyeball the results generated. The different design decisions we adopted were also keeping in view the performance of the models on the corpus. We were mostly focusing on precision to begin with, as it is easier to measure in case of extremely large corpora with no training data. While eyeballing the results was a good approach to iteratively improve the models, we decided to use another approach to evaluate the performance of the model. To this end, we scraped the Wikipedia articles for 40 entities that had a Wikipedia page and ran our models on the sentences extracted from this. We manually evaluated the results generated and tagged the prediction to be correct or incorrect by looking at the actual sentence from which the slot value was extracted.

Number of entities: 40 Number of slot values extracted: 69 Number of correct predictions: 57 Precision: 57/69 = 82.6

There is no direct way to measure recall, but on manually inspecting the sentences that contained mentions of the entities of interest, we found that there were many slot values that were not found. It looks like the recall of the system is lower than the precision. This is expected as we mostly rely on bootstrapping based approach with a fixed seed set. Here are some sentences from which our algorithm extracted slot values correctly:

On the other side, our model extracted the slot value Senate for the entity Bernard Kenny for the slot FounderOf from the sentence Jon Corzine had left the state or been incapacitated during Kenny's Senate Presidency, Kenny would have served as acting governor. Here, the pattern Kennys Senate which is similar to Sergey Brins Gogle results in an incorrect prediction. To prevent this, possessive references between an organization and an entity could be downweighted. In another instance, our model finds the value throat cancer as CauseOfDeath for Nassim Nicholas Taleb from the sentence Though a non-smoker, Taleb suffered from throat cancer in the mid-1990s, which he overcame. Here, the pattern suffered was not really indicative of death in itself unless accompanied by stronger clues. This could be fixed either by adjusting the threshold values or by ensuring that strong patterns like death or die appear in close proximity.

Since the patterns that we extracted are based on a fixed seed set using bootstrapping they have high precision. We found that the slots that have a target NER type are easier to capture as we could validate the output of the slot value extracted by comparing it with the NER tag

Entity	Annie Laurie Gaylor
Sentence	Gaylor and her mother, Anne Nicol Gaylor, and the late John Sontarck, founded the Freedom From Religion Foundation in a meeting around the Gaylors’ dining room table in 1978.
Slot	FounderOf
Value	Freedom From Religion Foundation
Entity	Barbara Liskov
Sentence	In 2004, Barbara Liskov won the John von Neumann Medal for ”fundamental contributions to programming languages, programming methodology, and distributed systems”.
Slot	AwardsWon
Value	the John von Neumann Medal

Table 6: Sentences found correctly

assigned by Stanford NER tagger. News articles have well formed English sentences and can be relied more for the slot values extracted and generally works well with the models we built. Social data on the other hand is much harder to work with as the sentences are not well formed and has a lot of embedded links, advertisements and spam. But, news articles are generally slower to report updates to slot, where as the first sight of new happenings begin in the social media. As a result, we had to adapt the models and patterns to work with social data which was not well structured. Thresholds for different slots had to be tuned separately. Some slots like Date of death, Cause of death etc. are relatively rarer and have highly precise rules and hence the thresholds were chosen to be lower. On the other hand, more commonly occuring slots like ContactMeetPlaceTime, AffiliateOf etc required a higher threshold to be met to update a slot value.

9 Conclusion

The Streaming Slot Filling task is extremely challenging given the size of the corpus, the nature of the data, need for disambiguation, need for indexing and the uncertainity in what qualifies as a slot update. Each of these individually are hard problems. The project encompassed different techniques from both the information retrieval and natural language processing community. They are both based on different philosophies but tasks like SSF provides an ideal middle ground where methodologies from both sides are equally important and one compliments the other. We feel that this is a very promising field of research and is key to unlocking the question as to how the enormous web corpora can be used to build automated systems that can keep track of the events occuring and updates to entities.

10 Acknowledgements

We would like to extend our thanks to Prof. Jeffrey Ullman who guided us throughout the course of the project and his suggestions and obervations were key to building system from scratch. We also thank the entire 341 course staff, Prof. Jure Leskovec, Prof. Anand Rajaraman and Robi, for providing us with the opportunity and framework to work on this very interesting

and challening project.

11 References

References

[1] Extracting Patterns and Relations from the World Wide Web. Brin, Sergey (1999) Extracting Patterns and Relations from the World Wide Web. Technical Report. Stanford InfoLab. (Publication Note: WebDB Workshop at EDBT'98)

[2] Mike Mintz, Steven Bills, Rion Snow, Daniel Jurafsky: Distant supervision for relation extraction without labeled data. ACL/AFNLP 2009: 1003-1011

[3] Michele Banko, Michael J. Cafarella, Stephen Soderland, Matthew Broadhead, Oren Etzioni: Open Information Extraction from the Web. IJCAI 2007: 2670-2676

[4] Limin Yao, Sebastian Riedel, Andrew McCallum: Unsupervised Relation Discovery with Sense Disambiguation. ACL (1) 2012: 712-720

[5] *TREC-KBA 2012 overview slides* http://trec-kba.org/trec-kba-2012-overview.ppt