The University of Sheffield System at TAC KBP 2010

Jingtao Yu, Omkar Mujgond and Rob Gaizauskas

Department of Computer Science, University of Sheffield, UK
{JYu3,OMujgond1,R.Gaizauskas}@sheffield.ac.uk

1 Introduction and Overview

This paper describes the University of Sheffield's entry in the 2010 TAC KBP entity linking and slot filling tasks. This was our first participation in the TAC KBP track. Given limited human resources and a relatively late decision to participate [1], we chose to view our participation as an exploratory effort, aimed at educating us in the issues surrounding the tasks. With that perspective we decided to first adopt a fairly naive approach, see where it went wrong, then refine the approach as time permitted.

Our first "naive" approach to the entity linking (EL) task was to build a text collection from the textual description portion of the KB nodes, index this collection using a search engine tool, convert the EL query into a search engine query and return the top ranked KB node whose name matched the entity name in the query as the answer to the query, provided the similarity score between the query and the KB node exceeded some threshold. Analysis of the failures of this approach suggested that a major problem was the insistence that a KB node name must match the query entity name exactly. The rest of our effort on the EL task went into exploring how we could relax this assumption.

Our first "naive" approach to the slot filling (SF) task was to treat it as a relation extraction task, which we tackled with a rule-based approach, given the shortage of training data and our limited development time and resource. We observed that the majority of slot values were one of the entity types PERSON, ORGANIZATION, GPE or TIMEX. We therefore chose to run a named entity recognition and classification (NERC) component that identified these entity types over the top ranked texts retrieved from the test corpus (which had been indexed previously by our search engine tool) using a query derived from the SF query name and associated document. For each slot a set of manually developed rules were applied to sentences containing the query entity name and another entity whose type indicated it was a candidate value for that slot. Candidate entities matched by the rules were returned as slot values. After implementing this approach little time was left for refinement. What limited time we had was spent analyzing what value of n should be chosen in selecting the top n documents returned in the retrieval stage for subsequent slot extraction.

The rest of this paper describes our approach and related investigations in more detail. Section 2 briefly describes existing language processing tools which we took "off-the-shelf", to reduce our development time and to allow us to concentrate on the most interesting aspects of the tasks. Sections 3 and 4 describe in detail our approaches to the EL and SF tasks respectively. Section 5 concludes the paper and discusses potential future work.

2 Off-the-Shelf Tools

This section briefly describes existing tools that we used in building our EL and SF systems. These include the Lucene search engine, the named entity and relation extration components in NLTK, Lingpipe's approximate string matching library and our own in-house TimeML annotation system, which recognises

[1]Our system development was carried out as a 6 month MSc dissertation project by the first two authors.

and grounds temporal expressions and event denoting expressions and adds temporal relational labels between time and event expressions.

2.1 Lucene

Lucene[2], which is used in our system as the core tool for searching knowledge base nodes, is a high performance information retrieval (IR) library which can be accessed via a Java API [1]. It allows us to easily add indexing and searching capabilities to our entity linking system. The data used for indexing and searching in Lucene should a textual format. Some sub-processes of Lucene, including the Analyzer, Indexing and Searching components, are applied in our EL system according to the requirements of the system.

2.2 NLTK

The Natural Language Toolkit (NLTK)[3] is a set of libraries and programs which is used for linguistic analysis of human language. It is implemented in the Python programming language.

2.2.1 NERC

NLTK contains a named entity recognizer and classifier, which is based on a classifier-based chunking approach. This NERC was retrained on CoNLL 2002 data. It was used to recognize alternative entity name form in the KB disambiguating text, according to defined rules.

2.2.2 Relation Extraction

NLTK also provides a relation extractor which finds relations between adjacent entities in the same sentence. This tool was used in a revised system developed after our official TAC submission with the intention of providing the user with an easier and more flexibile way of writing rules. However our post TAC experiments showed that the system using this extractor did not perform as well as the system we used for our official submission.

2.3 LingPipe

We made use of the string comparison module in LingPipe[4] to measure the similarity between strings. This was used for fuzzy entity name matching in the EL task. String similarity can be described as the degree of proximity between two strings. The proximity can be transformed into a distance or vice-versa by either negation or inversion. Damerau-Levenstein Distance was used to measure proximity [2].

2.4 Timex Identifier

To identify time and date related entities an identifier described in [4] was used. This system contains two components: a rule-based system which picks out and anchors temporal expressions and a maximum entropy classifier that assigns temporal relational links between times and events based on features that include descriptions of associated temporal signal words. In the recent TempEval2 challenge this component successfully identified temporal expressions and correctly classified their type in 90% of cases. In addition it determined the relation between event and time expressions in the same sentence with 63% accuracy.

[2]`http://lucene.apache.org`
[3]`http://www.nltk.org.`
[4]`http://www.lingpipe.com`

3 Entity Linking Task

The entity linking task at TAC-KBP 2010 can be seen as a knowledge base node retrieval problem. Thus we adopted information retrieval as the core technique to retrieve and rank KB nodes according to their degree of similarity to the query. The ranking strategy is based on two elements: (1) matching of the entity name from the query with the KB node name and (2) text similarity between the query document and the KB node description text, where similarity is calculated by Lucene using a variant of the cosine-distance or dot-product between the text of KB node and query document vectors in a vector space model. A critical issue in this process is how to match query and KB node entity names, since name used in the "correct" KB node may not be identical to that in the query. The method used in our system is to preprocess the KB collection to extract information about name variants which is then exploited at query time. We focused on three types of variation in names: names with variant spellings, aliases and name abbreviations.

3.1 Variant Name Extraction

In an EL query, the entity name is a critical clue to find the correct KB node and it is, accordingly, assigned a larger weight than other terms of the query in our system. It is necessary to find the KB node with the "same" entity name as that of a query. However, an entity name as it appears in a query may be an alias or a name abbreviation of an entity in the knowledge base. Table 1 shows some entity names which appear in queries in the form of aliases or abbreviations. It is difficult for an IR system to recognize the relation between two names with different forms. Therefore, potential name information in KB nodes needs to be mined and connected to entity names of KB nodes in advance, so that our system can associate a variant name form in a query with a KB node. These variant name forms can generally be found in the disambiguating text of KB nodes. Thus, we designed several strategies to extract these related names.

Original Entity Name in KB Node	Alias	Original entity name in KB node	Abbreviation
Charlotte North Carolina	Queen City	Movement for Democratic Change	MDC
Georgia country	Sakartvelo	Canadian National Railway	CN
Geelong Football Club	The Cats	Italian Communist Party	PCI
Nigeria national football team	The Eagles	Sikorsky Aircragft	SAC
Singapore national football team	The Lions	Uganda or University of Georgia	UGA

Table 1: Original entity names and their aliases and abbreviations

3.1.1 Pattern Matching

The disambiguating text of a KB node is commonly from a Wikipedia page and has a standard form. The critical name variant information is usually present at the beginning of the text, generally in the first few sentences. Therefore, one strategy is to extract possible aliases and abbreviations from the first three sentences of the disambiguating text. To do this we defined some patterns based on our observations of commonly recurring ways of expressing information about aliases and abbreviations. These patterns presume the NERC tool has been run over these sentences to recognize all occurrences of person, organization and GPE names within them.

Examples of the defined patterns are:

1. If *also known by* appears in the sentence, consider the name entity recognized after *known by* as an alias of the KB name.

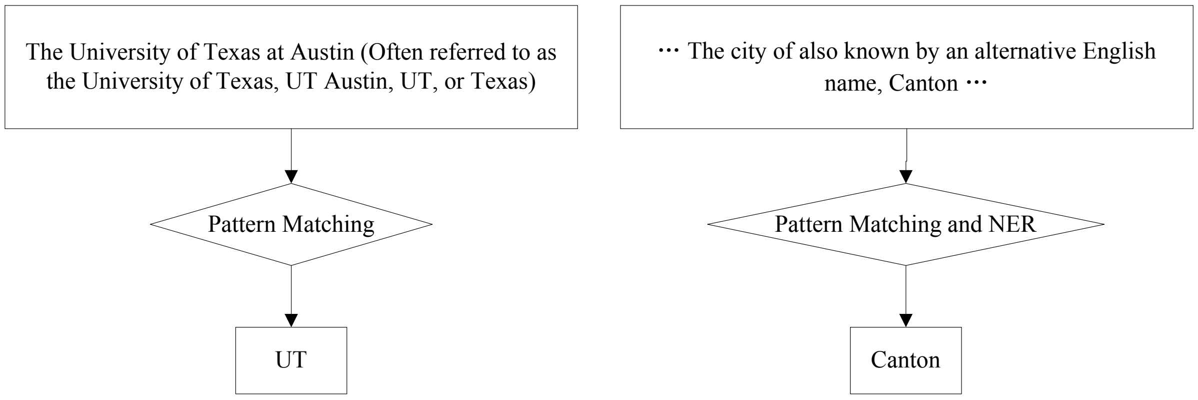

The University of Texas at Austin (Often referred to as the University of Texas, UT Austin, UT, or Texas)
Pattern Matching
UT
··· The city of also known by an alternative English name, Canton ···
Pattern Matching and NER
Canton

Figure 1: Extract name abbreviation from the first three sentences of the KB description text

2. If *(...)* occurs after an entity name and a token in the *()* consists of entirely of uppercase characters, consider the item in *()* as an abbreviation of the entity name before it.
3. If *also called* appears in the sentence, consider the name entity recognized after *also called* as an alias of the KB name.
4. If *nickname is* appears in the sentence, consider the name entity recognized after *nickname is* as an alias of the KB name.
5. If *is its nickname* appears in the sentence, consider the name entity recognized before *is its nickname* as an alias of the KB name.

The process of extracting related names using pattern matching can be seen in Figure 2. In the first example, the original entity name is *University of Texas at Austin.* As we can see, the first sentence of the disambiguating text includes the entity name and some potential variant name information in parentheses. Pattern 2 matches this example so the text in brackets is extracted and the abbreviation *UT* found.

In the second example, the system makes use of the NERC tool to recognize all the named entities in the sentence and pattern 1 then applies and returns the entity name *Canton.* We defined 12 patterns like these, covering most of the common ways in which we observed variant names being expressing in the KB disambigating texts.

3.1.2 KB Attribute Selection

Another method to extract related name information is to mine it from the attributes of the KB node because these attributes are from Wikipedia infoboxes which frequently contain name information, such as *official name*, *native name*,*full name*, *cartoon name*, etc. By observing a large number of the attribute names, we only extracted three important attributes as the source of aliases: *nickname*, *nicknames* and *past name.*

For instance, *Geelong Football Club* is a KB entity name which is unrelated with *the Cats* which appears in a query. However, from the attributes of the KB node, we can determine that the nickname of the entity is *the Cats*, which can be stored in the entity name field. By mining the attributes in advance we can store the two names together for use in the EL task.

3.1.3 Potential Abbreviation

Since name abbreviations often appear at any position in the disambiguating text of a KB node and are seldom present in the attributes, the previous methods are not effective to extract abbreviations in many situations. For this problem, we selected another more direct strategy.

Most commonly, an abbreviation is the combination of the capital letters of an entity name, such as *Movement for Democratic Change (MDC)*, so we extract the capital letters that appear in an entity name as its abbreviation. In order to make these abbreviations meaningful, we require the number of capital letters in an abbreviation to be greater than 2. In addition, for some special situations, we need to define extra rules. For example, when the proper noun *United States* appears in an entity name, as in cases such as *United States Department of Health and Human Services*, "US" may get mothballed as abbreviation letters. Thus we get two entity name abbreviations, *USDHHS and DHHS*.

3.1.4 Name Edit Distance

Some entities names may be written in different ways because of regions or custom or history. In order to not miss the KB nodes with different spelled names, we compared the query name with all the possible names of a KB node during searching and judged the relatedness of two names based on Damerau-Levenstein Distance.

For example, *Macao* and *Macau* are two differently spelled names which describe the same entity. Their Damerau-Levenstein distance is 1 because it uses a substitution of an *o* for an *u*. *Blufton* and *Bluffon* are another example of the same phenomeon. Thus we suppose two entity names link to the same entity if the DL distance between them is 1 unless one or other is an abbreviation. To do this we compare the entity name of a query and an entity name of a KB node, not including related names (aliases and abbreviations) during the searching. However, for some special situations, this method is not effective. For instance, *Mohammad Daud* and *Mohammed Daoud Khan* are linked to the same entity but their Damerau-Levenstein distance is 6.

After extracting variant names, the analyzer provided by Lucene is used to filter redundant content from free text and extract effective information. After that, all the terms, including entity names, disambiguating texts and extracted related names, are stored in different fields, a kind of indexing structure in Lucene.

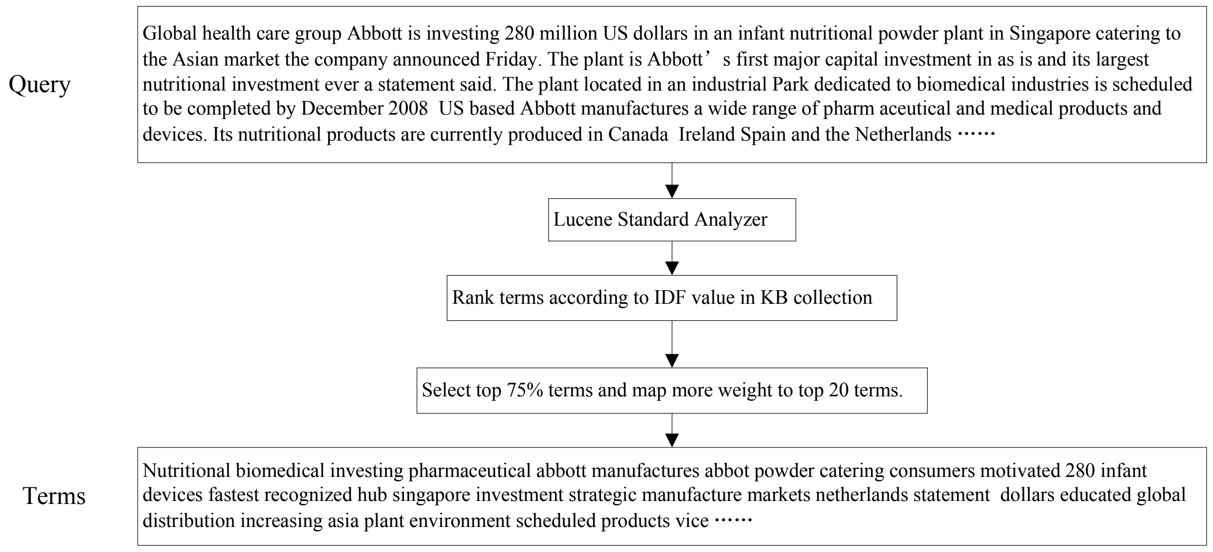
3.2 Search-time processing

3.2.1 Query Formulation

Lucene ignores terms in queries beyond a certain length. Thus the full text of the associated document in the query cannot sensibly be used verbatim as a query string. To select which terms from the associated document should be used in the Lucene query we first remove stop words and special symbols and then chose terms with the highest IDF values.

Lucene ignores terms in queries beyond a certain length which can be set by ourselves. The number of terms in every query document differs and the full text of the associated document in the query cannot be used verbatim as a query string, so we need to extract the more critical terms to use in our Lucene query. Take the case of entity *Abbott*. Some terms, such as *nutritional biomedical manufactures powder*, are clearly more important than others terms. such as*ever, meet, wide, the.* To select which terms from the associated document to use in the Lucene query we first remove stop words and special symbols and then chose terms with IDF values above a threshold. The threshold is set based on the size of the collection and the distribution of all the terms.

At the same time terms derived from the entity name are assigned a higher weight than the terms from the associated document so that the disambiguating documents of KB nodes which contain these terms will be ranked higher. The process of query formulation is shown in Figure 2. We did not process entity names in queries and just searched for the matching name in the KB name field which stores all variants of a KB entity name. The terms extracted from the query document are used to calculate the similarity with every KB node's disambiguating text.



Query
Global health care group Abbott is investing 280 million US dollars in an infant nutritional powder plant in Singapore catering to the Asian market the company announced Friday. The plant is Abbott's first major capital investment in as is and its largest nutritional investment ever a statement said. The plant located in an industrial Park dedicated to biomedical industries is scheduled to be completed by December 2008 US based Abbott manufactures a wide range of pharm aceutical and medical products and devices. Its nutritional products are currently produced in Canada Ireland Spain and the Netherlands ……
Lucene Standard Analyzer
Rank terms according to IDF value in KB collection
Select top 75% terms and map more weight to top 20 terms.
Terms
Nutritional biomedical investing pharmaceutical abbott manufactures abbot powder catering consumers motivated 280 infant devices fastest recognized hub singapore investment strategic manufacture markets netherlands statement dollars educated global distribution increasing asia plant environment scheduled products vice ……

Figure 2: Extract name abbreviation from the first three sentences of the KB description text

3.2.2 Answer Selection

The query formulation and searching strategy described above was used in our system. Lucene then returns a ranked list of documents, however the EL task requires us to return a unique KB node or NIL (no answer) for each query not a ranked list of KB nodes. Here, we adopted the Difference method to decide whether to return a non-NIL answer from the top ranked entries in the list.

In the Difference method approach we calculate the difference in value between the similarity score of the first retrieved document (KB node) and the query and that of the second or third retrieved document and the query. If this difference value is above a defined threshold, the system returns the first answer, otherwise it returns NIL (i.e. no related KB node). The reason for this strategy is that if the query has an obvious related KB node, the similarity score of that document should be much higher than others. On the contrary, if there is little difference among the top documents if the query doesn't have any answer. The threshold was determined by testing on the 2009 EL evaluation data.

3.3 Evaluation Results

We evaluated our EL system on the TAC evaluation data by setting different thresholds for NIL-non-NIL response. We also investigated the performance of the system across queries with different entity types. These results are summarized in Tables 2 and 3, respectively.

As we can seen in Table 2, the number of correct NILs returned increased as the threshold became larger. At the same time, the number of correct KB nodse returned decreased to some extent. The overall accuracy was the highest at the threshold value of 0.1, where the difference value was obtained by subtracting the similarity score of the second KB node in the list from the first one.

Different threshold	KB	NIL	All
0.1	0.6231	0.4235	0.7886
0.15	0.6178	0.4471	0.7593
0.2	0.6089	0.4706	0.7236

Table 2: Results of the accuracy entity linking system on the evaluation queries.

Table 3 illustrates the EL system's accuracy for each entity type (PER, ORG and GPE). Three entity types in the queries occur in the same proportion. Since there is more name disambiguation information about PER entities in the knowledge base than for the others, this entity type has the highest accuracy, 73.9%. The approach adopted for entity disambiguation does not prove very appropriate for ORG and GPE entity. The accuracy of Non-NIL ORG entities, which appear in the queries and have entity linking in KB, is just 37.5% while that of Non-NIL GPE entities is more or less the same at 35.2%. The accuracies for NIL ORG entity and NIL GPE entity are 79% and 83% respectively.

Amount	Type	Accuracy
750	ORG	0.6200
304	non-NIL ORG	0.3750
446	NIL ORG	0.7870
749	GPE	0.5100
503	non-NIL GPE	0.3519
246	NIL GPE	0.8333
751	PER	0.7390
213	non-NIL PER	0.6620
538	NIL PER	0.7695

Table 3: The accuracy of each category of entity with threshold 0.1

3.4 Analysis

3.4.1 Alternative Formulation Strategies

In the process of query formulation, there are multiple alternative methods to limit the number of terms in a query. Our system discarded the terms that have low IDF values from a query document. These terms seem unlikely to represent important features of a KB node because they appear in many nodes. However, some terms, particularly dates, might appear in the term vectors and affect the performance of system.

Other alternative methods include discarding terms with low TF · IDF or TF values. Table 4 shows results for these different strategies on the All accuracy (NIL and non-NIL responses) of the system. The accuracy of the TF method is lower than others. In this method, some unhelpful common verbs or nouns might be kept in the term vector, such as *stay, serve, weather, and etc.* In the TF · IDF method the All accuracy increased somewhat as compared to the TF method, but KB accuracy is still much lower than the IDF method.

Our approach – selecting query terms by IDF value – can select terms, such person names and organization names, each of which appears in a small node of KB nodes. These terms are more useful than others in the process of entity disambiguating. Some other possible strategies, such as keeping all terms that are NEs will be considered in further work.

Strategies	KB	NIL	All
IDF	62.31%	42.35%	78.86%
TF	34%	75%	57.4%
TF · IDF	38%	86.1%	65.3%

Table 4: Different query formulation strategies

3.4.2 Ranked Nodes Distribution

In order to observe the distribution of correct KB nodes within the top ranked KB nodes returned by our search strategy, we investigated the top 30 retrieved answers in the ranked list. The correct answers for 24% queries which are linked to a KB node (i.e. non-NIL answers) were not at the first position. Most of them lay in the top 5 positions, particular at position 2. In these cases, the difference values in similarity score between the top two KB nodes in the ranked list are very small and the threshold value of 0.1 or 0.2 that we used in variants of our system is much larger than these different values. Table 5 shows some examples from the 2009 EL data where the entity names of the top two KB nodes are very similar. More complex term weighting strategies are needed to address this problem.

Entity in query	Entity of the first node	Entity of the second node	Different value
ABC	American Broadcasting Company	ABC newspaper	0.0096854
Garden City	Garden in the City	Garden City New York	0.06992
CSS	Cadillac Sixty Special	Cansei de Ser Sexy	0.1017442
Galatasaray SK	Galatasaray S.K. PAF	Galatasaray A.	0.00396497

Table 5: Proximate first node and second ranked nodes

4 Slot Filling

We pursued a traditional approach of (1) finding the documents which are relevant to the query (2) identifying in those documents the named entities which might end up as slot answers (3) finding relations between these entities to determine which values should fill the slots.

4.1 Offline processing

All the text from the document collection and the knowledge base provided was indexed using Lucene for faster processing. The XML markup of the document collections was used to decide on which part of the document should be included or excluded from indexing. Material within the following tags was excluded on the assumption that it would not help in our slot filling task.

1. ELEMENT HEADER - - (DATE—PCDATA)
2. ELEMENT TRAILER - - (PCDATA)
3. ELEMENT SLUG - - (PCDATA)
4. ELEMENT FOOTER - - (PCDATA)
5. ELEMENT QUOTE – (PCDATA)

Note that no tags from KB entities were excluded.

4.2 Search Time Processing

4.2.1 Retrieval

This stage of in the slot filling system requires us to retrieve only those documents from the document collection which might contain answers for the slots. Lucene's searching component was used. One simple way of constructing a Lucene query from the SF query is to include only the entity name in the Lucene query. However this is unlikely to be adequate. We constructed our query using the name of the entity, text from the related document provided with the entity and text from the knowledge base document (if present in the entity). The name of the entity which acts as an important element was made mandatory in the Lucene query. The combined text was further processed, as we did in our EL system, to exclude stop words and to include only those words with high inverse document frequency. Terms relating to specific slots which were found during the generation of manual rules were also included. For example, if we have a query which needs two slots per:alternatenames and per:schoolsattended to be filled we include *known as, called as, graduate of, graduated from, studying at, studied in, r earned his.*

To decide how many of the top documents in the ranked search results to select for further processing we carried out some investigations over the TAC 2009 data using the answer coverage measure introduced in [3] in the context of the related problem of how many of the top ranked documents returned by an IR system should be investigated during answer extraction stage of a question answering (QA) system: "Let Q be the question set, D the document (or passage) collection,$A_{D,q}$ the subset of D which contains correct answers for $q \epsilon Q$ and $R_{D,q,n}$ be the n top-ranked documents (or passages) in D retrieved by a retrieval system S given question q. The coverage of a retrieval system S for a question set Q and document collection D at rank n is defined as":

$$Coverage^{s}\left(Q, D, n\right) \equiv \frac{|q \epsilon Q| R^{s}_{D,q,n} \cap A_{D,q} \neq \emptyset|}{|Q|}$$

The coverage gives the proportion of the question set, for which a correct answer can be found within the top n documents retrieved for each question. Considering all the (247 + 483) slots in the TAC 2009 SF data (treating these as questions in this case) there are a total of 168 slots which contain at least one correct answer bearing document in the human judgments provided for the TAC 2009 data. We can now compute coverage at different ranks (Table 6).

Top n documents	**Queries covered**	**Coverage**
30	113	113/168 = 0.6726
50	148	148/168 = 0.8988

Table 6: Coverage of queries over top n documents

Given these coverage scores, we had to decide whether to select, e.g. the top 30 or top 50 documents for further processing. Even though it is clear that we can cover up to 89 % of the queries by selecting top 50 documents we could end up with a lower overall SF result since processing 50 documents could lead to the inclusion of more noise. To avoid this risk only top 30 documents were selected for further processing, though clearly more experimentation is needed here.

4.2.2 Named Entity Recognition

NLTK's named entity recognizer was used to categorize named entities present in the documents into LOCATION, ORGANIZATON and PERSON entities. It covers most out of the 43 TAC 2010 slots. Out of the 43 TAC 2010 slots 31 slots accept their answer as one of these three entities. No major modification where made to the nltk recognizer. Note that nltk's named entity recognizer does not recognize Timex entities. [4, Derczynski, L and Gaizauskas, G. (2010)] provided one such Timex identifier which helped us to cover the following slots.

1. per:date-of-birth
2. per:date-of-birth
3. orgfounded
4. org:dissolved

We wanted to evaluate NLTK's Named entity recognizer to assess its likely impact on our relation extractor. For example, the per: employeeof slot expects its answer to be an entity of type sc ORGANIZATION. There are 301 training sentences which cover most of the (20+17) slot types and are extracted from the annotated documents and are judged as correct. Text from each annotated document is subjected to sentence splitting which yields a list of sentences. Only those sentences which contain the slot answer are selected from the list. Named entities are then recognized in all these sentences. Table 7 depicts the number of sentences where the answer string is correctly categorized according to entity type.

Attributes	**Results**
Total training sentences	301
Slot answers which are correctly categorized	142
Accuracy	0.4717

Table 7: Entity categorization accuracy

From table 7 it is clear that named entity recognizer is only 47.17% accurate on this data. We did not have time to investigate this problem further, but clearly it is a serious issue.

4.2.3 Relationship Extraction

Once the documents are tagged with named entities, the final stage in our system attempts to extract relations between the entities . This stage is completely based on hand-crafted rules developed using the training sentences provided. All rules are based on standard regular expressions. Care is taken to generalize the rules by carefully analyzing the training sentences. In cases where NLTK's named entity recognizer could not recognize slot value entity types (as in case of per:title) standard look-ups were used. Similarly look-ups were used to further classify GPE entities which could be values for the per:cityofbirth, per:stateorprovinceofbirth, per:countryofbirth slots into city, state or province and country. The per:alternatenames slot was filled first because we found that in many training sentences alternate names are used instead of person names. The per:title slot was the next slot to be fill because a combination of per:title and per:alternatenames or just the name of the entity helps to fill slots like per:employee_of (*FMF's Captain Li Jie admitted China lacked experience, but said they had the talent to be competitive*) and org:topmembers (*We recognise the importance of both Heathrow and Gatwick to the UK economy," said CC deputy chairman Christopher Clarke*).

4.3 Evaluation Results

Recall	0.02417795
Precision	0.0030917635
F1	0.005482456

Table 8: Evaluation Results

Our slot filling system was aggressive in filling slots resulting in a very low score. The following were the reasons identified for the failure of our slot filling system:

- Cases were found where responses are wrong because of spelling mismatches and in some cases organization names found in response were abbrevations of names found in key.
- In some cases our named entity recognizer failed to recognize the entity completely and correctly. For example, a slot expects *Antonio Basilio* to be a PERSON entity but our recognizer recognized

it as [PERSON] *Antonio* [PERSON] [ORGANIZATION] *Basilio* [ORGANIZATION]. Hence our relation extractor took only *Antonio* into account.

- Normally in rule-based systems precision will be better than recall but in our system precision is less than recall. This is because rules were generalized. Rules which make use of pattern terms like *founded by* for org: foundedby slots are specific, but some of our rules were so general that they ended up throwing up many responses for a list valued slot hence affecting the overall precision.

  For per:alternate-names (List-Valued slot) we had a rule which considers first name as alternate-name of a person. To do this we extract all possible name of the target person per document and blindly consider the first word as an answer. This is true in some cases but false in many cases. Similarly for org:top-members or employees (List-Valued slot) to find top members we considered all the person entities which follow a title. This rule also ended up throwing up many responses. Likewise for per:siblings there is a rule which considers names which include the surname of the target entity as slot answers. For example, for person *John Smith* entities which consist of *Smith (Andy Smith)* was considered to be *John Smith's* sibling. However care was taken not to include his children in the siblings list.

4.4 Post Submission Experiments

Out of the three stages we had found tools for two of our stages and our rule based relation extraction stage was not user friendly. All the rules were implemented programmatically; we wanted a tool which helps us to write our rules in a friendlier, more declarative manner. NLTK's relation extractor is one such tool which helps us to write rules comfortably. With few modifications NLTK's relation extractor was used to extract relations between the entities. Table 9 compares both NLTK and our slot filling system's performance. Note that these results are obtained using TAC 2009 data and TAC 2009 queries.

Attributes	**NLTK**	**Our System**
SF value score	0.39316	0.59790
Single-Valued slot's accuracy	0.39271	0.69230
List-Valued slot's accuracy	0.39362	0.50359

Table 9: comparision of NLTK and Our system outputs on TAC 2009 data using TAC 2009 queries

We noticed that our system performed well since it provides flexibility in writing rules when compared to NLTK's relationship extractor which restricts us to write rules between two adjascent entities. For example, given a named entity tagged sentence *[per:Ali-Akbar Salehi], was born on [Date: 1.sept 1944] in [Loc: Italy]*, suppose we want to extract information about the birth country of *Ali-Akbar Salehi.* The NLTK extractor ends up finding following relations separately which makes it difficult to write a rule.

1. Per: Ali-Akbar Salehi] was born on [Date: 1.sept 1944]
2. Date: 1.sept 1944] in [Loc: Italy].

our extractor can handle such cases efficiently.

We also performed an experiment which justified our decision to use the top 30 rather than top 50 retrieved documents. The following table compares both top 50 and top 30 selected documents.

From table 10 it is clear that our system performed well on tselecting top 30 but we can see that when the number of top documents being selected increases, accuracy gradually decreases. From this observation we can see that selecting the top 30 retrieved documents for further processing was a good decision.

Attributes	**top 30**	**top 50**
SF value score	0.59790	0.59745
Single-Valued slot's accuracy	0.69230	0.69131
List-Valued slot's accuracy	0.50359	0.50359

Table 10: slot filling systems performance on selecting top 30 and top 50 documents using TAC 2009 data and TAC 2009 queries

References

[1] Hatcher, E. and Gospodnetic, O. Lucene in Action. Manning Publications Co. 2005.

[2] A guided tour to approximate string matching. ACM Computing Surveys (CSUR) archive 33(1), 31-88, 2001.

[3] Roberts, I. and Gaizauskas, R. (2004), Evaluating Passage Retrieval Approaches for Question Answering In Advances in Information Retrieval: Proceedings of the 26th European Conference on Information Retrieval (ECIR'04), 72-84.

[4] Derczynski, L and Gaizauskas, R. (2010). USFD2: Annotating Temporal Expresions and TLINKs for TempEval-2 In Proceedings of the Workshop on Semantic Evaluations, ACL.