IIIT Hyderabad in Guided Summarization and Knowledge Base Population

Vasudeva Varma, Praveen Bysani, Kranthi Reddy, Vijay Bharath Reddy, Sudheer Kovelamudi, Srikanth Reddy Vaddepally, Radheshyam Nanduri, Kiran Kumar N, Santhosh Gsk, Prasad Pingali

International Institute of Information Technology, Hyderabad, India.

Abstract

In this report, we present details about the participation of IIIT Hyderabad in Guided Summarization and Knowledge Base Population tracks at TAC 2010. This year, we enhanced our summaization system with knowledge based measures and utilized domain and sentence tag models to score sentences to suit guided summarization track. We have used an external tool, WikiMiner to identify key concepts in the documents and extract important sentences. We adopted an Information Retrieval based approach for the Entity Linking and Slot Filling tasks. In Entity Linking we identified potential nodes from the Knowledge Base and then identified the mapping node using tf-idf similarity. We achieved very good performance in the Entity Linking task. For Slot Filling task we identified documents from the document collection that might contain attribute information. We extracted the attribute information using a rule based approach.

Part I
Guided Summarization Track

1 Introduction

The Summarization track at Document understanding Conferences (DUC) and later at Text Analysis Conferences(TAC) has introduced different flavors of summarization like generic multi document summarization, query focused multi document summarization, opinion summarization, and most recently update summarization. This year the track has evolved into a more challenging and interesting task, Guided summarization. Unlike previous tasks, Guided summarization task promotes summarization systems to make a deeper linguistic and semantic analysis of source documents to extract important information. The guided summarization task is defined as to generate summaries for a set of newswire articles on a particular topic that is classified into a set of predefined categories. Each category has a list of important aspects and the summary is expected to answer all these aspects while it may also contain other relevant information about the topic.

The term guided summarization is used for the first time in literature, as the summary is guided by a template of aspects for each category. The description of the task is similar to the problem of information extraction, where specific information from unstructured text is identified and consequently classified into a set of semantic labels (templates) making the information more suitable for other information processing tasks. While information extraction (IE) systems select only specific information nuggets to fill the slots of templates, a guided summarization system has to produce a readable summary encompassing all the information about the templates. Coupling information extraction techniques with summarization is a relatively less explored area, making it hard to find any relevant literature. Very few investigations have explored the potential of merging summarization with information extraction techniques. RIPTIDES [11] hypothesize that IE supported summarization will enable the generation of more accurate and targeted summaries in specific domain than that is possible with domain independent techniques. Authors combine information extraction, extractive summarization and natural language generation to support user directed multi document summaries. RIPTIDES identify domain specific relations among relevant entities in the text using a weakly supervised learning algorithm, which are then used to extract slot fillers for the domain. The extracted templates are then processed by a summarizer, that merges templates into a event oriented structure using heuristics and assigns score to each sentence based on the frequency and membership of each template. Finally the summary is generated from the resulting content pool using a combination of top-down, schema-like text building rules and surface-oriented revisions. Similarly [2] built a news event summarizer to provide consistent and succint summaries about an event. They detect useful event snippets based on language model learned from train-

ing corpus. Authors used fusion techniques to merge information from multiple sources.

In this work, we generate guided summaries via information extraction measures. A sufficiently large set of relevant articles are selected manually from Wikipedia for each category. These articles are used to build domain knowledge and extract important sentences containing events mentioned in the template. We also experimented with knowledge based measures, through WikiMiner for extracting concepts from documents and using them instead of simple document words to estimate importance.

2 Approach

1. **Domain model**: Domain Knowledge is an essential part of information extraction systems to identify and extract the relationships between entities. We built a domain knowledge model for each topic category by collecting a set of relevant Wikipedia articles for that category. A group of indviduals are given the task of manually selecting approximately 500 Wikipedia pages for each category and then extracting the first passages from that article (the style of Wikipedia is such that first paragraph of most articles serve as a summary containing answers to most of the questions in the template).

   For example consider the following article about "Bhopal Tragedy"[1] that is categorised under Accidents/Disaster category,

   The Bhopal disaster (also referred to as the Bhopal gas tragedy) is the world's worst industrial catastrophe. It occurred on the night of December 2, 1984 at the Union Carbide India Limited (UCIL) pesticide plant in Bhopal, Madhya Pradesh, India. A leak of methyl isocyanate (MIC) gas and other chemicals from the plant resulted in the exposure of several thousands of people. Estimates vary on the death toll. The official immediate death toll was 2,259 and the government of Madhya Pradesh has confirmed a total of 3,787 deaths related to the gas release. Other government agencies estimate 15,000 deaths. Others estimate that 3,000 died within weeks and that another 8,000 have since died from gas-related diseases. A government affidavit in 2006 stated the leak caused 558,125 injuries including 38,478 temporary partial and approximately 3,900 severely and permanently disabling injuries.

   This excerpt from Wikipedia article has a summary of all the statistics about Bhopal gas tragedy and its aftermath. Similar articles are collected for each category and a domain knowledge model is built over these articles using a naive bayes classifier.

   We have a multi class classifier built over all the categories given in TAC 2010 task description. The confidence (probability) of a sentence selected from the document of particular category being classified into the same category is regarded as the score of that particular sentence. This normalized domain model score is considered as a feature in our experiments.

2. **Sentence Annotation**: The articles collected from Wikipedia are split into sentences and annotated with appropriate template tags given in the task description. These annotations include both objective *(when,where,who)* and subjective *(how,why,countermeasures)* tags. As any standard NER can only tag objective tags, we chose to manually annotate all the articles with all possible tags. For example consider a sentence from the "Bhopal Tragedy" article,

   It occurred on <when> the night of December 23, 1984<when> at the <where>Union Carbide India Limited (UCIL) pesticide plant in Bhopal, Madhya Pradesh, India<where>. <how>A leak of methyl isocyanate (MIC) gas and other chemicals from the plant resulted in the exposure of several thousands of people<how>

   A sentence is tagged with multiple tags if it has more than one answer to the template. All the annotated sentences are used for building a model by a multi class classifier (naive bayes classifier is used in this work).

3. **Concept Mining**: Words are conventionally considered to be the units of text to calculate importance. Simple word counts and frequencies in the document collection have proved to work very well in the context of summarization [10]. This year we attempted to use semantic concepts instead of simple word frequencies in computing sentence importance.

   Wikipedia is a vast, constantly evolving resource of interlinked articles providing a giant multilingual database of concepts and semantic relations. It serves as a promising resource for natural language processing and many other research areas. Wikipedia Miner [5] is a freely available toolkit for navigating and making use of content of Wikipedia. It provides simplified, object-oriented access to Wikipedia's structure and content and offers several services to help users to search for entities, comparing the relation between entities and wikifying snippets of texts. Assuming that each Wikipedia topic serves

[1] http://en.wikipedia.org/Bhopal_disaster

as a semantic concept, we make use of Wikify service of the toolkit to annotate the document collection with links to relevant Wikipedia concepts. Wikiminer also provides a semantic relatedness measure between two concepts using category hierarchy and textual content of respective concepts. After the Wikification of the document collection, we use the relatedness measure (RM) of a concept with all other concepts in the document collection as its importance measure. Sentence score is calculated as the normalized concept relatedness measure of all the concepts it contains.

A detailed description of the process of linking entities with Wikipedia and calculating the relatedness measure between two concepts can be found in [7] and [6].

4. **Role of prepositions in estimating sentence importance**: In English grammar, a preposition is a part of speech that links nouns, pronouns to other phrases in a sentence. A preposition generally represents the temporal, spatial or logical relationship of its object to the rest of the sentence. Observe the role of prepositions *on,of,to,in,from* in the below sentences,

   *The book is **on** the table*
   *President **of** India lives **in** delhi*
   *The indian cricket team is travelling **from** australia **to** new zealand*

   It is very interesting to observe how prepositions are implicitly capturing the key elements in a sentence. The preposition **on** in first sentence is conveying that there is a book, a table and some relation between them. Similarly, the other two sentences have some key information about one or more entities and connecting prepositions. To the best of our knowledge, the role of prepositions has never been explored before to calculate sentence importance.

   We propose using the frequency of a small set of prepositions as a sentence scoring feature. The frequency of prepositions is indirectly achieving the effect of performing a Named Entity Recognition (NER) on a sentence, but without any additional cost of processing or using any POS tags. Score of a sentence (s) calculated by PrepImp is given as,

$$PrepImp(s) = \frac{\sum_{w_i \in s} IsPrep(w_i)}{|s|}$$

   The list of prepositions used for calculating sentence importance are limited to simple single word prepositions like *in,on,of,at,for,from,to,by,with*, after a careful observation over the data.

```
for sentence in RankedList:
        while(summary.length <=100):
                annotation=getMaxAnnotation(sentence)
                if(!summary.contains(annotation))
                        summary.add(sentence,annotation)
```

Fig. 1: *Summary Extraction Algorithm*

5. **Sentence Ranking**: After feature extraction, we estimate a final rank of each sentence using a regression model similar to our system of Update summarization in TAC 2009 [10]. We modeled sentence rank as a dependent variable that is estimated from a set of features. Each sentence in the training phase is represented as a tuple of sentence importance estimate and feature vector. The sentence importance is estimated as the ROUGE 2 score of that sentence with the model summaries. Final rank of the sentence is calculated as,

$$i_s = q(F_s)$$

   where i_s is the sentence importance (rank) of sentence s, q is the regression function and F_s is the feature vector that comprises of all the extracted features.

6. **Summary Extraction**: Normally, during summary extraction a subset of ranked list of sentences satisfying redundancy checks, length constraints and other conditions are selected for the summary. To adapt to the requirements of guided summarization, we modified our extraction algorithm to Figure 1. Among the top ranked sentences, as predicted by the regression model, we select the sentence having maximum aspect score (MaxAnnotation) for the summary. Once an aspect of the template has been answered, we select the next sentence with the maximum score for the remaining aspects in the template. These tag scores are predicted by the Sentence annotation model we built over Wikipedia articles in step 2.

3 Experiments and Results

The test dataset released by NIST composed of approximately 46 topics, divided into five categories:

- Accidents and Natural Disasters
- Health and Safety

- Attacks
- Endangered Resources
- Investigations and Trials

Each topic has a topic ID, category, title, and 20 relevant documents which have been equally divided into 2 clusters A and B in chronological order. Summary for cluster A is a normal multi document summary while summary for cluster B is an update summary, generated under the assumption that the user has already read the earlier articles (cluster A) about the topic.

The information about the category and its aspects already define what information the reader is looking for and replaces the need of narrative in general query focused summarization. Each of the above mentioned categories had a template of aspects that the summary had to answer. For example, the accident category has the following template:

- WHAT: what happened
- WHEN: date, time, other temporal placement markers
- WHERE: physical location
- WHY: reasons for accident
- WHO_AFFECTED: casualties (death, injury), or individuals otherwise negatively affected by the accident
- DAMAGES: damages caused by the accident
- COUNTERMEASURES: countermeasures, rescue efforts, prevention efforts

We used successful features from our earlier work [10] and [3], Document Frequency Score (DFS), Sentence Location1(SL1), PHAL, Kullback Leibler divergence (KL) and Novelty Factor (NF) to produce an initial run. The features described in the previous section are used incrementally in this initial experiment. The summaries are manually evaluated by a group of indviduals and ranked on a scale of 1 to 5, with 5 being the highest quality. After analysing these manual evaluations, the run that achieved maximum consensus amongst the annotators was selected. The final submission has two runs,

Run1 : PHAL, KL, DFS and SL1 are used as sentence scoring features for cluster A and NF, KL, SL1 as features for cluster B. Sentence rank is estimated through regression model from the training feature vectors build over TAC 2009 training data.

Run2 : Along with the sentence scoring features used by Run1, PrepImportance is used as an additional feature and the sentences are picked with the modified summary extraction algorithm as described in Section 2. The evaluation results of these runs are provided in Table 1

The results shown in Table 1 are the average scores over all the categories and all the aspects of the template. A detailed analysis at the category and aspect level of the results is provided in Table 2 and Table 3. The Avg-pyramid and overall responsiveness scores of Run1 are split by category. Interestingly, the scores for Health and Safety category are much lower compared to other categories.

With the availability of evaluation data after the announcement of results, we had the chance to experiment with all possible combinations of our techniques. As we are succesful in producing informative summaries for cluster A, we focused our post-TAC experiment to improve the quality of update summaries (cluster B). We carried out experiments with combinations of techniques proposed in Section 2 along with a new feature Hybrid Kullback Leibler Information divergence (HKLID).

Hybrid Kullback-Leibler Information Divergence (HKLID) : Kullback-Leibler Information divergence (KLID) is a popular technique to measure the difference between two probability distributions. We used an extension of KLID to measure the divergence between Hybrid Language Models (LM) of two sentences that are built over $clusterA$ and $clusterB$. A hybrid language model is the combination of document and sentence language models, for better divergence calculations. HKLID between LM's of two sentences s_i in $clusterB$ and s_j in $clusterA$ is calculated using,

$$\sum_{w \in s_i} P(w|s_i)P(w|clusterB) * log \frac{P(w|s_i)P(w|clusterB)}{P(w|s_j)P(w|clusterA)}$$

HKLID measures the importance of a sentence in ndocs conditioned over the sentences in pdocs. So more the divergence between these hybrid language models, greater the novelty of that sentence. Average HKLID between a sentence s_i in $clusterB$ and all the sentences in $clusterA$ is used as its novelty score. Probability distributions are smoothed using Dirichlet principle [12].The results of the post-TAC experiments on cluster B is provided in Table 4

4 Discussion

Evaluation results show that our Run1 has secured first position in ROUGE-2, ROUGE-SU4 and Pyramid scores and second position in terms of overall responsiveness. In spite of developing a more sophisticated system by building a category model over re-

System	ROUGE-2	ROUGE-SU4	Avg-Pyramid Score	Overall Responsiveness
Run1(id: 22)	0.09574 (1/41)	0.13014 (1/41)	0.425 (1/41)	3.130 (2/41)
Run2 (id: 40)	0.0695 (23/41)	0.10788 (22/41)	0.347 (21/41)	2.804 (21/41)

Table 1: *Evaluation results released by NIST for cluster A*

Category	Avg-Pyramid Score	Overall Responsiveness
Accidents	0.445	3.429
Attacks	0.524	3.286
Health&Safety	0.300	2.583
Endangered resources	0.396	3.100
Investigations	0.520	3.500

Table 2: *Pyramid and Overall Responsiveness scores of Run1 for each category*

lated Wikipedia documents and manually annotating template aspects for each sentence, the evaluation results of Run2 are not as good as Run1. As the language model of Wikipedia is quite different from the news articles, the Wikipedia domain model has not produced desired results. We believe that the proposed techniques would work better if the model was built over a sufficiently large news corpus. A closer look at the results reveals that the difficulty of task varies with the type of category. The pyramid scores and overall responsiveness for Accidents, Attacks and Investigations are very high compared to Health and Endangered Resources cateogories. Also, there is a huge disparity between the pyramid scores of various aspects. Guided summaries are able to answer objective questions like WHEN, WHERE, WHO with more ease than subjective aspects like HOW, WHY, and others. It is intutive that subjective questions are more difficult to answer than the objective questions. The new feature Preposition importance uses a simple frequency of the prepositions in a sentence to estimate the importance, and that did not perform well in Run2. But, we strongly believe that the idea can be used in a more sophisticated way to devise an effective sentence scoring feature.

Since we focused on the guided summarization in this submission, it resulted in relatively poor performance in the update task. We used succesful techniques from our previous work and also devised a new feature HKLID to produce update summaries for cluster B. During our post-TAC experiments we evaluated different combinations of all the techniques described in this work. The concepts mined from Wikipedia using wikiminer enhanced the quality of update summaries. We achieved significant improvement in ROUGE scores of cluster B (around 9% increase) during our post-TAC experiments.

Experiment	R-2	R-SU4
Run1	0.06998 (7/43)	0.11107 (4/43)
Run2	0.05894 (23/43)	0.10024 (20/43)
NF+SL1+HKLID	0.07899	0.12000
NF+SL1+ ConceptMining	0.07508	0.11746
NF+SL1+HKLID+ ConceptMining	0.07773	0.11822

Table 4: *ROUGE scores of post-TAC experiments for cluster B*

Category	What	When	Where	Who	Subjective
Accidents	0.621	0.468	0.33	0.457	0.289
Attacks	0.745	0	0.753	0.622	0.491
Health	0.322	–	–	0.172	0.166
Endangered resources	0.52	–	–	–	0.128
Investigations	–	–	–	0.785	0.371

Table 3: *Average Pyramid scores of Run1 for each aspect in the template*

Part II

Knowledge Base Population (KBP)

1 Introduction

The rise of web 2.0 technology has provided a platform to generate content on the web through blogs, forums etc. This has lead to information overload and users face difficulty in finding the information they are looking for. Structured Knowledge Bases (KB) like Wikipedia, act as a rich source of information. The problem with Knowledge Bases like Wikipedia is that they have to be created and maintained manually. Manual effort is not only time consuming but can also lead to erroneous values being fed, outdated information and inconsistency. Automatically updating Knowledge Bases from news source, where the latest information is available, is a possible solution. The Knowledge Base Population (KBP) track at TAC-2010 proposes the problem of automatically updating Knowledge Bases from textual content. The task has been broken down into 2 fundamental sub-problems:

1. **Entity Linking** : The task of Entity Linking is to determine for each query entity, which node is being referred to in the KB or if the query entity is not present in the KB. The query consists of a named entity and an associated document from the Document Collection (DC) using which we need to link the named entity to it's corresponding node in the KB, if any. The purpose of the associated document is to provide context that might be useful in linking it. We need to return the node id if the query entity is present in the KB else NIL.

2. **Slot Filling** : Slot Filling involves mining information about entities from textual content. Slot Filling shares similarities with traditional Information Extraction and Question Answering tasks. Slot Filling involves learning a predefined set of relationships and attributes for target entities based on the documents in the Document Collection. A query in the Slot Filling task contain a name-string, doc-id, entity-type, node-id, an optional list of slots to ignore. As in the Entity Linking task the doc-id is intended to provide the context for the entity. We need to return the slot name and slot value along with the document in which it occurs.

2 Our approach to Entity Linking

For a given query entity, we identify a set of possible nodes from the KB that can be linked with it. We then use a Tf-idf ranking model to rank these set of nodes and identify the correct mapping node. We have broken down the task of entity linking into 3 sub tasks:

a) **Building a Knowledge Repository (KR)** : Since named entities can be referred using different forms like nick names, alias names, acronyms and spelling variations, we need to have a knowledge repository which contains all these variations of the named entities. Since Wikipedia has better coverage of named entities[8], we used it to build this knowledge repository. The features used are

- **Redirect Pages** : Redirect pages are an aid to navigation in the Wikipedia. When a page redirects to another, it simply means that both the referred page and the referee page are describing the same named entity. It is used to identify synonyms and in addition also provides information about abbreviations, scientific or common terms and alternative spellings etc.

- **Disambiguation pages** : Disambiguation pages have been specifically created for ambiguous entities. They contain links to the pages which describe a unique real world entity. They are useful in homonym resolution and help in extracting abbreviations etc.

- **Bold text from first paragraph** : The first paragraph of a Wikipedia article in general is a summary of the entire article. It contains few phrases which are written in bold. We observed that these bold phrases, invariably are nick names, alias names, full names etc.

b) **Candidate List Identification** : In this phase, we identify all potential nodes from the KB that can be linked to our query entity. First, we obtain all possible variations of the query entity using different heuristics. We use these variations to identify the potential nodes from the KB. In order to identify NIL valued queries we add documents from Wikipedia as well. The heuristics used are

- We obtain all possible variations of the query entity from the knowledge repository built in the previous step i.e we have obtained all the possible variations of the query entity present in Wikipedia.

- We use Stanford Named Entity Recognizer(NER), we identify other possible variations of the query entity from the associated document from the document collection. Here we are using the contextual information present surrounding our query entity to identify further variations. Phrases tagged as "Person/Location/Organization" by Stanford NER and having our query entity present in it are also considered as possible variations. For Example, if Stanford NER tags "Columbus, Ohio" as a location in the document associated with the query entity "Columbus", we consider it as a variation.

- We give the query entity to the Google Search engine and retrieve spell suggestions if any. This heuristic helps us in identifying the spelling variations of the query entity.

  Once all the variations of our query entity have been obtained, we use them to identify candidate nodes by searching on the KB and Wikipedia titles. This is done by using token search. Since we need to identify query entities that don't have an entry in the KB, we take the help of Wikipedia articles. This addition of Wikipedia articles to the candidate list helps us in identifying entities that don't have an entry in the KB i.e. if the query entity is linked to Wikipedia article, it means that the article isn't present in the KB and hence we return NIL.

c) **Entity Linking using Ranking** : Once the set of candidate nodes are obtained, we rank them using Tf-idf[9]. The best ranked node is returned as the mapping node for our query entity. Tf-idf is the most popular weighting function used in the field of Information Retrieval. Given a query Q={t1,t2...tn} and a document D={w1,w2,...,wn}, the similarity between them is given by

$$Similarity(D,Q) = \sum_{t_i inQ} tf(t_i, D) * idf(t_i) \quad (1)$$

where term frequency (tf) is simply the number of times the term t_i appears in the document D. The inverse document frequency (idf) is a measure of the general importance of the term.

We index the text of the candidate nodes using Lucene. Lucene is a full-featured, open source text search engine. We then form a boolean "OR" query from the associated document given for our query entity. Querying the index built using this "OR" query, would return a ranked set of nodes.

We further re-rank these ranked documents, using query expansion. We use pseudo relevance feedback to expand the query. We consider the top "3" documents for query expansion and build an Hyperspace Analogue to Language Model (HAL)[4]. We consider only those tokens that are present within a window size of four for the query entity present in the top "3" documents. The intuition behind this is that named entities derive their meaning from the accumulated experience of the context in which they appear. The HAL weighting for a term *w* and any other term $w^{'}$ is given by:

$$HAL(w^{'}/w) = \sum_{k=0}^{K} W(k) n(w, k, w^{'}) \quad (2)$$

where *n(w, k, $w^{'}$)* is the number of times term $w^{'}$ occurs a distance *k* away from *w*, and *W (k) = K − k + 1* denotes the strength of the relationship between the two terms given *k*.

We search the Google using query "wiki" and the query entity. We rank the snippets and title of the results using the boolean "OR" query generated from the associated document. We consider the top most ranked title and snippet tokens for query expansion.

Once the query is expanded we re-rank the indexed documents. The final relevance score for a candidate document is a linear combination of the ranking score and re-rank score. The equation is given by:

$$Final\ Score = \lambda * RankingScore + (1-\lambda) * Re-rankedScore \quad (3)$$

For all our experiment we set λ to 0.7 .

d) **Description of Runs** : We have submitted 3 runs for the Entity Linking task. The description of each run is provided below

Run-1 : Without the web : For this run we used the knowledge repository built and Stanford NER for identifying candidate list documents. Once these candidate documents are identified we rank them. We also re-rank these documents using query expansion.

Run-2 : Using the web and phrase search : For this run we used the knowledge repository, Stanford NER and Google web search engine for identifying the candidate list documents. We used Google for spell suggestion for identifying incorrect spellings. Using these variations we did a phrase search on the titles of Wikipedia and KB nodes. All the nodes whose titles exactly or partially match these phrases are considered as candidate nodes. We then rank

these nodes using Tf-idf and re-rank using pseudo relevance feedback. We use Google search results snippet, title and HAL for query expansion. The final relevance score for each node is a linear combination of the rank and re-rank scores.

Run-3 : Using the web and token search : This run is similar to above run, except that we do token search on the titles of Wikipedia and KB to identify candidate nodes. All those nodes whose titles contain the variation tokens or have token variation and an extra token are considered as candidate nodes. The ranking and re-ranking is same as in run-2.

Table shows the micro-average score of our runs.

3 Entity Linking Optional Task

We have participated in the Entity Linking optional task as well. In the primary task, the systems were allowed to consult the text of the nodes present in the KB, where the optional task involves linking the query entity without using the text from the nodes and use only the slot values.

Our approach towards optional task is similar to what we used for the primary task. The only difference is that instead of indexing the text of the nodes we index the slot values. We extracted the attribute value pairs from Wikipedia infoboxes while indexing them. The runs submitted for the optional task are the same as for primary task. The results of optional task are summarized in Table 2.

4 Our Approach to Slot Filling

Our Slot Filling model basically identifies documents from the Document Collection that might contain information about the query entity. We then pick sentences from these documents that are likely to contain attribute information relevant to the given query entity. Once the sentences are obtained we extract attribute information using a rule based approach. We then filter and process the extracted attribute information to address the requirements of the task. We used training annotated data provided by TAC-KBP-2010 for constructing our rules. We have annotated training samples for 6 entities as contribution to the community.

For the run submitted for TAC-2010 Slot Filling task, we considered documents which contained partial match of the query entity as well for extracting attribute information. This has resulted in a lower precision score. In order to overcome this lower precision, we considered only those documents which had the exact query phrase in our post-TAC experiments. We report our official scores obtained for the run submitted for TAC-2010 and post-TAC results as well. Our Slot Filling system can be broken down into the following modules

1. **Pre-processing** : Since the Document Collection and KB have documents in the order of millions, we index them using Lucene to enable search and fast retrieval of documents.

2. **Document Retrieval** : For every query entity, we search the Document Collection index for relevant documents. We select the documents that satisfy the following conditions

   - Only documents that contain exact phrase match of the query entity.
   - Documents that contain partial query entity are also considered.

3. **Sentence Retrieval** : Once the relevant documents are obtained, we pick sentences from each document that contain mapping words for a slot, depending on the entity type.

   For example, for the slot org:alternate_names, we select the documents that contain the complete name/partial-name of that organization. We then pick the sentences that contain the mapping-phrases like “also known as”, “also called as”, “referred as”, “renamed as” etc.

4. **Attribute information extraction** : In this phase our system extracts attribute information from the sentences that are obtained in the previous step. We use a “Rule-based Ordered Co-occurrence”[1] strategy for extracting attribute information. This method follows the simple definition of the term Co-occurrence “an event or situation that happens at the same time as or in connection with another”. It seeks only the existence of the query entity and keywords in the same sentence.

   We formed rules to extract attribute information for each slot using syntactic structure of sentences. These rules were formed using the annotated training data. We used POS tags, Named-Entities, sentence-window-length and the order of occurrence of entities for constructing the rules. For example, for the slot Org:alternate_names, patterns like

Run No.	**Heuristics Used**	Micro-Average Score
1	Entire Wikipedia in building KR	81.73
2	Entire Wikipedia in building KR + Stanford NER + Google Search + Phrase Search	83.60
3	Entire Wikipedia in building KR + Stanford NER + Google Search + Token Search	83.73

Table 5: *Micro-average score for each run.*

Run No.	**Heuristics Used**	Micro-Average Score
1	Entire Wikipedia in building KR	60.53
2	Entire Wikipedia in building KR + Stanford NER + Google Search + Phrase Search	65.16
3	Entire Wikipedia in building KR + Stanford NER + Google Search + Token Search	66.00

Table 6: *Micro-average score for each run.*

<org1, mapping-phrase, org2, [verb phrase], [org|loc], EOS>

are matched with the sentences obtained from the Document Retrieval phase with the help of Stanford POS tagger and Stanford NER. For the above example, if org1 is the query entity then org2 is taken as the alternate name for org1 and the corresponding org:alternate_names slot is filled. Appropriate conditions to handle the redundancy in list-valued slots, ignore slots etc are also taken. Different patterns that include dates, locations, entities and cardinal numbers are formed to handle different slots like per:date_of_birth, per:age, org:founded, per:date_of_death, per:place_of_birth, org:city_of_headquarters, etc.

For example, consider the slot per:age, if there is no time window for that query, then we have given the largest of all the cardinal numbers that we found using our patterns for age. Whenever there is a conflict in terms of Location whether it is a city, state or country, we used lists of variations in names of cities, states and countries. Similarly we resolved for the slots like per:country_of_birth, per:stateorprovince_of_birth, per:city_of_birth, org:city_of_headquarters, org:stateorprovince_of_headquarters, org:country_of_headquarters, etc.

5 Results

In our post-TAC experiments, we added a few new rules and modified the existing ones. In our post-TAC experiments we considered only the documents from the Document Collection which have the exact phrase of the query entity. For example, if the query entity is "Samsung", we gather all the documents that contain the entity name Samsung by searching the index. These documents can also contain the documents relevant to "Samsung Semiconductors". In our post-TAC experiments we discarded all the documents related to "Samsung Semiconductors" as both are not the same organizations. Table 7, shows F-measure for our official run and also post-TAC experiments. Statistics for post-TAC experiments about the number of rules, mappings etc are provided in Table 8.

Runs	Precision	recall	f-score
Official Run	0.363	0.054	0.094
post-TAC experiments	0.76	0.098	0.174

Table 7: *Results of official submission and post-TAC experiments*

Twelve new rules are added for the person entity type which identified 8 additional slot values and corrected 19 false positives. Two new rules are added for the organization entity type also resulted in achieving better accuracy. Overall 46 new slots are correctly filled and other 20 spurious slot responses are rectified in the post-TAC experiments.

6 Conclusion

In this paper, we presented our attempt to develop a system for automatically populating a structured Knowledge Base. Our system for Entity Linking is based on Information Retrieval approach. We identify a set of potential nodes for the KB that can be linked to the query entity. We rank these potential nodes using Tf-idf ranking model and re-rank them using pseudo relevance feedback for query expansion. The final score for a node is the a linear combination of the ranking and re-rank score. Our system achieves

Total no. of Slots	No. of Slots addressed	Total no. of mappings used	Total no. of rules used
42	35	82	70

Table 8: *Statistics of mappings and rules*

a very good performance and is scalable because of the continuous development of Wikipedia, which is used for building our Knowledge Repository.

Our approach to Slot Filling being a rule-based model required a lot of human effort. The system is able to achieve high precision because the rules written are generalizable and intuitively correct. The recall of our system is low because we couldn't write rules for all the slots. With some minor tweaking and addition of few more rules for frequently occurring cases, our post-TAC score is above the median. Factors that limited our performance in KBP Slot Filling 2010 include sentence selection that was too broad, noisy patterns, poorer domain models, low training data for most of the slots, quality of the training annotated data. We plan to do boot strapping to identify more rules automatically using the rules written manually.

References

[1] R. Feldman, Y. Aumann, M. Finkelstein-Landau, E. Hurvitz, Y. Regev, and A. Yaroshevich. A comparative study of information extraction strategies. *Computational Linguistics and Intelligent Text Processing*, pages 21–34, 2010.

[2] Y. Han, F. Li, K. Liu, and L. Liu. Template based chinese news event summarization. *Semantics, Knowledge and Grid, International Conference on*, 0:53, 2006.

[3] J. Jagarlamudi, P. Pingali, and V. Varma. Query independent sentence scoring approach to duc 2006. In *In proceedings of DUC 2006*. DUC, 2006.

[4] K. Lund and C. Burgess. Producing high-dimensional semantic spaces from lexical co-occurrence. *Behavior Research Methods, Instruments, and Computers*, (28):203–208, 1996.

[5] D. Milne. An open-source toolkit for mining wikipedia. 2009.

[6] D. Milne and I. H. Witten. An effective, low-cost measure of semantic relatedness obtained from wikipedia links. 2008.

[7] D. Milne and I. H. Witten. Learning to link with wikipedia. In *CIKM '08: Proceeding of the 17th ACM conference on Information and knowledge management*, pages 509–518, New York, NY, USA, 2008. ACM.

[8] M. Remy. Wikipedia: The free encyclopedia. *Reference Reviews*, 16(6):5, 2002.

[9] K. Sparck Jones. A statistical interpretation of term specificity and its application in retrieval. pages 132–142, 1988.

[10] V. Varma, P. Bysani, K. Reddy, V. Bharat, S. GSK, K. Kumar, S. Kovelamudi, K. K. N, and N. Maganti. iiit hyderabad at tac 2009. Technical report, Gaithersburg, Maryland USA, 2009.

[11] M. White, T. Korelsky, C. Cardie, V. Ng, D. Pierce, and K. Wagstaff. Multidocument summarization via information extraction. In *HLT '01: Proceedings of the first international conference on Human language technology research*, pages 1–7, Morristown, NJ, USA, 2001. Association for Computational Linguistics.

[12] C. Zhai and J. Lafferty. A study of smoothing methods for language models applied to information retrieval. *ACM Trans. Inf. Syst.*, 22(2):179–214, 2004.