UALR at TREC 2011 Entity Track

Venkata Swamy Martha+, Halil Bisgin+, Stephen Wallace+, Hemant Joshi*, Nitin Agarwal+ and Xiaowei Xu+[1]
+Center for Advanced Research in Data-mining, Information Science Dept., University of Arkansas at Little Rock, Little Rock, Arkansas, 72204, USA
*Acxiom Corporation, Little Rock, Arkansas, 72201, USA
{vxmartha, hxbisgin, sxwallace, xwxu, nxagarwal}@ualr.edu,
Hemant.Joshi@acxiom.com

Abstract

The primary objective of the track is to accomplish a mechanism to answer entity related searches over web data. The TREC organizers provided a segment of web data i.e. ClueWeb09 data collection which is used for this work. An example of the entity related query is "*Countries other than Iceland to which Icelandair flies?*". To answer such queries, it needs two phases of processing, one is offline processing and the other is query time processing. During offline processing, the data is indexed using third party search engine tool. Query time processing involves query reformulation, extracting related documents from the index and document analysis to answer the query. Document analysis is the heart of the research which include entity identification and entity ranking based on a query. Since there is no readily available sophisticated entity identification tool, consensus approach is incorporated for entity identification. The consensus is achieved from Stanford NER tagger [1] Apache sentence detector with sentence parser [7] and DBpedia dataset lookup [9] A novel technique is developed to rank the entities related to a query. In other words, the identified entities from related documents are ranked based on their relevance to the query terms. Variations of query reformulations are: no reformulation, manual (human reformulated) and programmed using entity identification tool, resulted three submissions *CARDHTML, CARDHTMLM* and *CARDHTMLA* respectively. In addition to the above variations, we also used external web search engine (Bing) to extract related documents and submitted the run as *CARDHTMLB*. The submission also involved *LOD URI* look up which is achieved using organizers provided Sindice dataset [5] and its tool [10]

1. Introduction

University of Arkansas at Little Rock (UALR) in collaboration with Acxiom Corporation participates in entity track. The main task of this track is to develop a solution for finding relevant entities and subsequently their LOD-URI using Sindice dataset [5] An example of the query is *"Countries other than Iceland to which Icelandair flies?"*. To answer such queries we developed a network structure mining approach that is explained in the following.

[1]Contact author: Xiaowei Xu, email: xwxu@ualr.edu

2. System architecture

The system consists of two components: Indexing and Query Processing. Index is built off-line independent to queries. We indexed the Clueweb09 dataset [1] with Indri (part of Lemur) software [3] [4] for efficient retrieval. The off-line processing component is presented in Figure 1.

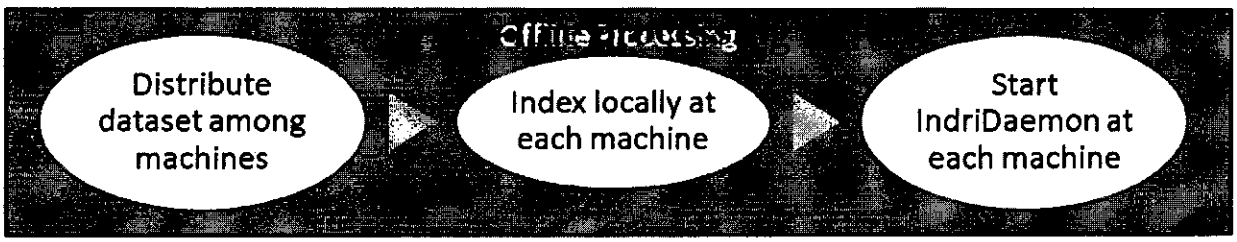

Figure 1: Off-line processing

Query processing leverages the index to retrieve query related documents. Further the query processing involves following steps: 1. Query restructuring, 2. Identifying sentences in the query related documents using HTML parser and Apache OPENNLP sentence detection library, 3. Entity recognition using an open source entity tagging libraries (Stanford NER tagger [2], Apache OPENNLP sentence parser [7]), 4. Entity validation with DBpedia dataset which also returns us entity type information, 5. Weighing sentences by comparing sentence terms with query terms, and finally 6. Ranking identified entities using a novel network mining approach. The flowchart of the steps is depicted as Figure 2. These steps are explained in detail in Section 3.

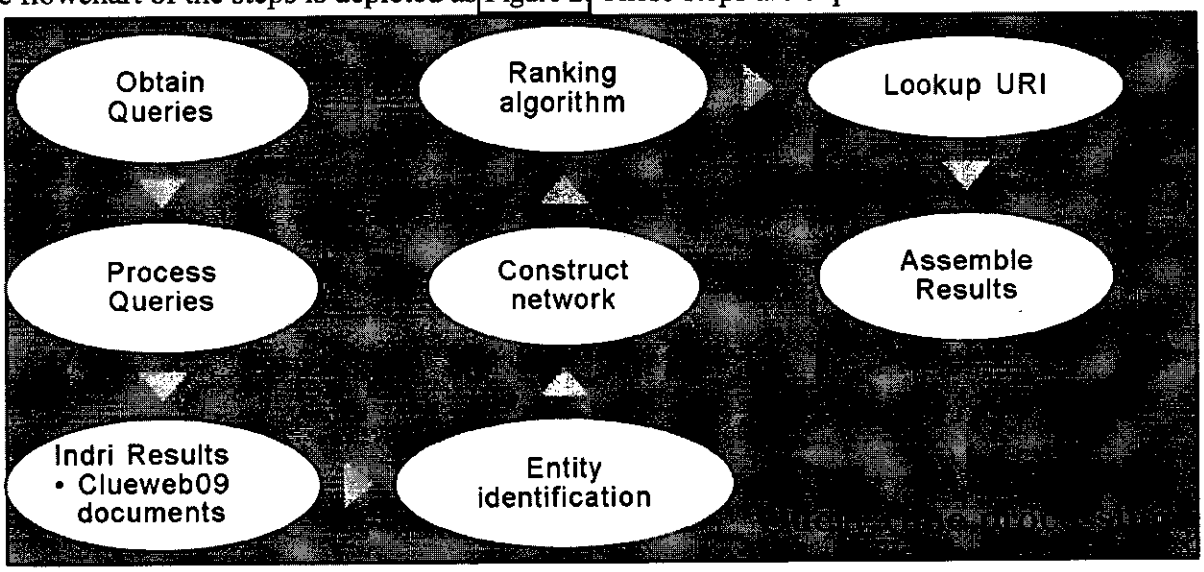

Figure 2 Query time processing

The overall architecture of our system is illustrated in Figure 1 and Figure 2. Architecturally significant components of the system are explained in Sections 3 and 4 respectively.

3. Procedure to extract related documents

3.1 Index

Indri search engine is used to index the ClueWeb09 dataset. Since the dataset is large enough to not to fit in a single machine, we segmented the dataset into 46 partitions (each partition with *en<x>*, *en<x+1>*, *en<x+2>* directories) and distributed them on to 46 machines which further fed to indri to index locally with standard indri parameters at each machine. With this infrastructure, we were able to achieve faster document retrieval. We configured indri to store documents along with the index; the rational behind doing it is discussed in next section. Indri daemon (server) is started on all the machines and kept available for further steps.

3.2 Topics analysis

The given queries from the organizers include narrative, type. To avoid inconsistency we mapped the given types to generic DBpedia classes. For example, "*school*" to "*EducationalInstitute*".

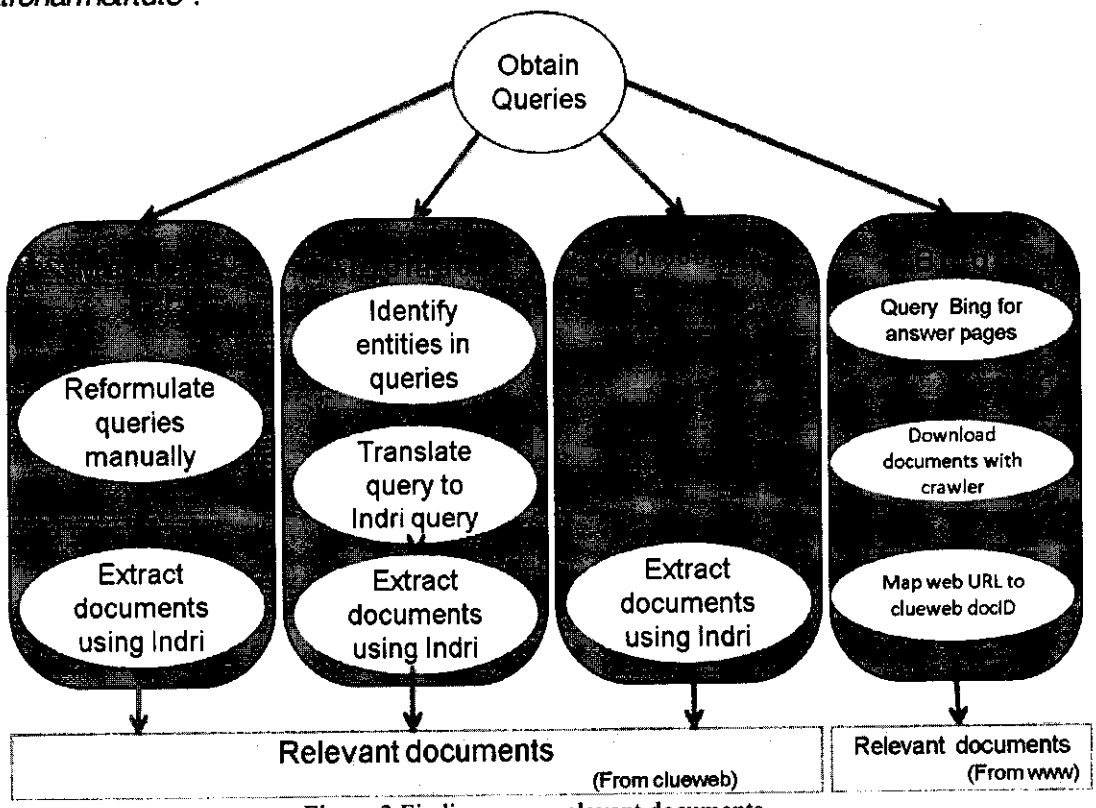

Figure 3 Finding query relevant documents

3.2.1 Query reformulation (Manual)

As part of the manual run, we reformulated the query narrative into indri query language. The main reason behind human intervention is to remove/replace more generic terms, to keep specific terms together as phrase (particularity entities with two terms). For instance, the query (1) is reformulated as

"#combine(#1(IcelandAir) #1(destinations))"

3.2.2 Query reformulation (Automatic)

This run involves complex processing of the query narrative. First, entities are identified using methodology discussed in Section-4.3. The stop words are removed except the term "of", doubting it is part of entities. The phrases are called "*query_key_words*". The *query_key_words* plays an important role in weighing sentences in documents. Now the *query_key_words* are translated to indri query language. Here we included weight which is the length of the terms included in the specific frame. The automatic reformulation turned the query (1) into

"#weight(7 #1(iceland) 10 #1(icelandair) 54 #1(countries other than iceland to which icelandair flies) 9 #1(countries) 9 #1(countries) 33 #1(iceland to which icelandair flies) 16 #1(iceland to which) 7 #1(iceland) 16 #1(icelandair flies) 5 #1(flies))"

3.2.3 Query relevant documents

The reformulated query is requested to indri which returns us query related document id's (we use this *doc_id* later for network construction). Using *dumpindex* utility of indri, corresponding raw documents are retrieved. For the four runs we submitted for organizers, we followed four different approaches to obtain relevant documents. The variations are clearly depicted as Figure 3. The raw documents are digested to extract the answer entities for a given query. The document processing involves complex operations and uses other third party tools, further detailed in section 4.

4. Documents processing

There are 150 documents from the Clueweb09 collection which can contain answer. For the discussion, let us consider "clueweb09-en00-123456" is one of the result documents i.e one of the 150 documents. Figure 4 shows the steps involved in the document processing for ranking entities.

4.1 HTML parsing

The document is parsed using HTML parser obtained from [6] (to retrieve text nodes along with it's html tree positions. For each text node, there exists an "*html_tree_tag_id*" which is unique and represents position of the text node in the document.

4.2 Sentence detection

A text node might contain more than one sentence. For the reason, each text node is further drilled to segment into sentences. Each sentence is assigned a unique numeric id called "*sentence_id*". The segmentation is achieved through Apache Incubator's *OPENNLP* sentence detection tool [7]. The *OpenNLP Sentence Detector* can detect that a punctuation character marks the end of a sentence or not. In this sense a sentence is defined as the longest white space trimmed character sequence between two punctuation marks. The first and last sentences make an exception to this rule. The first non whitespace character is assumed to be the beginning of a sentence, and the last non whitespace character is assumed to be a sentence end. The OpenNLP Sentence Detector cannot identify sentence boundaries based on the contents of the sentence.

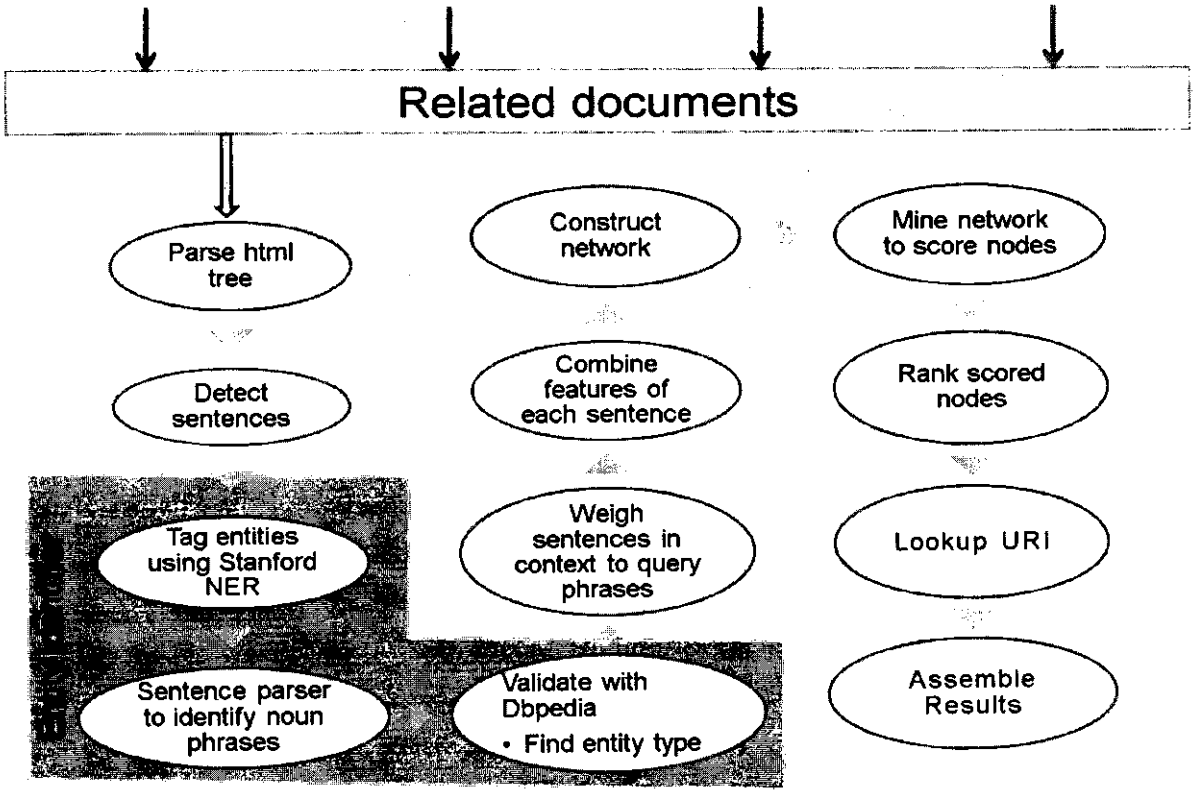

Figure 4 Document processing and ranking entities

4.3 Entity detection

A sentence has set of terms which have specific meaning. Sentence analysis is critical and important task which can affect results significantly. We followed following steps to achieve concise entities and semantic meanings among them.

4.3.1 Stanford NER tagger

This is very naive approach to identify entities of primitive types person, location, organization. Stanford NER tagger [2] takes a sentence and tags recognized entities. We ignore entity types assigned the tagger because the types are more generic to filter out target entities. We forwarded the knowledge (identified entities) to next step. To forward the knowledge, the entities in the sentence are replaced by indices (here we used *XYZ<index>*) which are recognized as single term in the next step, we call the resulting sentence "virtual sentence".

4.3.2 Apache Incubator's Parser

The sentence (virtual sentence) from entity tagger is forwarded to *Apache Incubator's Parser* [8]. The parser identifies terms with its parts of speech tags. The reason behind using parser instead of *POS tagger* is to retain tag tree structure. To focus on the given task (entity identification), only phrases tagged as *NP, NN, NNP, NNS, NNPS* are considered as entities. Since *NP* is a phrase clause and lead to its descendants, so all descendant terms of NP tag is considered as an entity. The rational behind using parser instead of POS tagger is to get the descendant terms of *NP* tag. Here is an example tags obtained from the parser

(TOP (NP (NP (DT The) (JJ quick) (JJ brown) (NN fox) (NNS jumps)) (PP (IN over) (NP (DT the) (JJ lazy) (NN dog))) (. .)))

Once all the nouns are extracted, the entities found in earlier step (section 4.3.1) are reverted back by replacing indices (*XYZ<index>*) with corresponding phrases. By the end of the step we had entities in the sentence.

4.3.3 Entity validation and Entity type lookup

The entities recognized are needed to be validated to avoid noise. The validation is carried out using DBPEDIA dataset [9]. Advantage in using the dataset is that it not only validates but also gives fine grained entity types. For faster access, we indexed the dataset using *Apache Lucene*. Since queries include target entity type, we wiped off all the entities except the ones we needed to answer. The remaining entities are *target_entities* and are ranked in coming steps.

4.4 Weighing sentences

The sentences are given a *relevance_score* which represents correlation of the sentences to the given query. The score of a sentence is "sum of the length of the *query_key_words* found in the sentence".

By the end of this step, we have following parameters for each sentence from all related documents:

doc_id , *html_tree_tag_id* , *sentence_id*
relevance_score , *target_entities*

4.5 Sentences to network translation

The processed sentences are translated to network. The vertices in the network are two types: entities and phrase called "*sentence_node*", formed by writing *doc_id*, *html_tree_tag_id* and sentence_id together i.e. *<doc_id>:<html_tree_tag_id>:<sentence_id>*. The *sentence_node*'s retain *relevance_score* values along with them in the network. Now an edge from an entity node and a sentence exists if the entity derived from the sentence. Two sentences are connected only if their *sentence_node* prefix matches up to *90%*. The limitation leads sentences, with common ancestor at very near to the sentence in a document html tree, are connected. In this way, we have links among sentences and sentences to entities.

4.6 Network processing

We applied network mining algorithm to find answer entity nodes in the network. It is obvious that the answer entities are descendants of the sentences or its descendants for which the relevance score is high. At the same time, we need to avoid sentences at very high level of the html tree to avoid more generic answers. Less number of *neighbor sentences* will give us the sentences at lowest level. The score that a sentence inherits to its descendants is equally distributed i.e. divided by *neighboring entities*. In summary, the score of a sentence node on the network is:

Score of a sentence = (*relevance_score* / (count{*neighbor entities*} * count{ *neighbor sentences* }

The entities might occur in more than one sentence and can inherit score from multiple sentences. The one score maximum of neighbor sentences is an entity's score.

Score of an entity = max{ *score of neighbor sentence* }

The sentence which inherits the score (max score) to an entity is supporting document for the entity in the query context.

4.7 Ranking network nodes i.e. entities

The entity nodes with their scores are sorted to rank. By this time, we have answer entities with their scores, rank and supporting document id's.

5. Variations of submitted runs

Thanks to super computing facility at our university, we were able to setup indri search engine. It made us to tune the framework for better results. We picked best four tunings from many. For the submission, we used the top four variations:

1. CARDHTMLA: In this run we leveraged entity identification method to identify phrases in given query narratives and then they are translated to indri query language to extract related documents.
2. CARDHTMLM: We manually identified phrases in the query narratives and then they are translated to indri query language to extract related documents.
3. CARDHTMLB: For this we took advantage of external web search engine *Bing*. The query narratives are directly sent to *Bing* which returns us set of result documents. They became related documents of the query in the next step.
4. CARDHTML: This run is very conservative. It does not go for query reformulation. The query narratives are queried to indri to get related documents. This run retrieved only top 100 document hits from the indri search engine whereas other runs retrieved 150 top documents.

6. Evaluations of the results

We observed promising results for various queries we tried with, but we are not able to quantify their accuracy. Since we received evaluation from TREC organizers, we evaluated our submissions over a ground truth. The ground truth is prepared from the evaluation of the task. Since our main task is to identify related entities, our evaluation is performed on entity names instead of URIs'. The evaluation metrics include nDCG (in Figure 5) and Average Precision (in Figure 6). The charts below depict the evaluation of the four submitted runs for the given 50 queries.

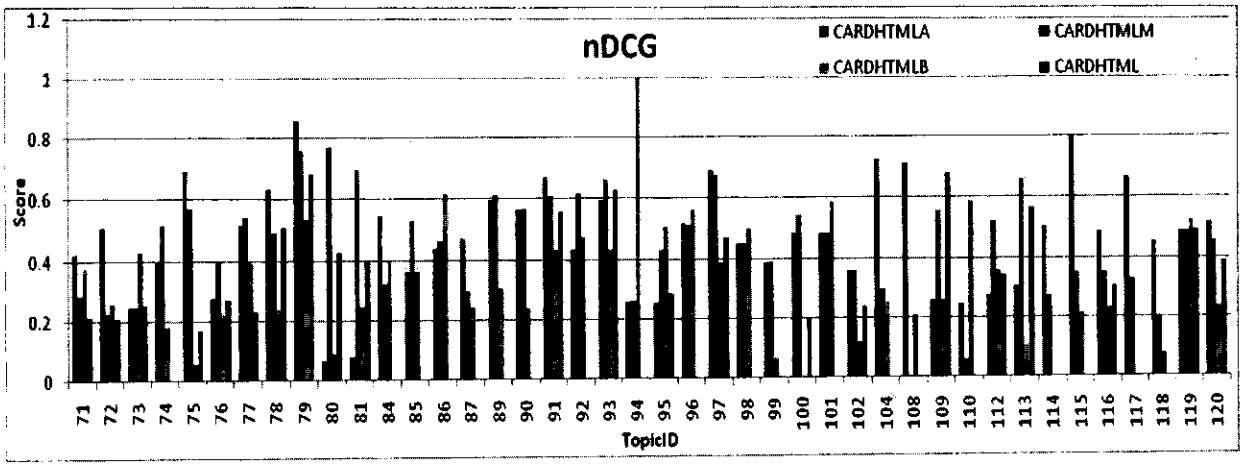

Figure 5 : nDCG scores of each topic for the four runs

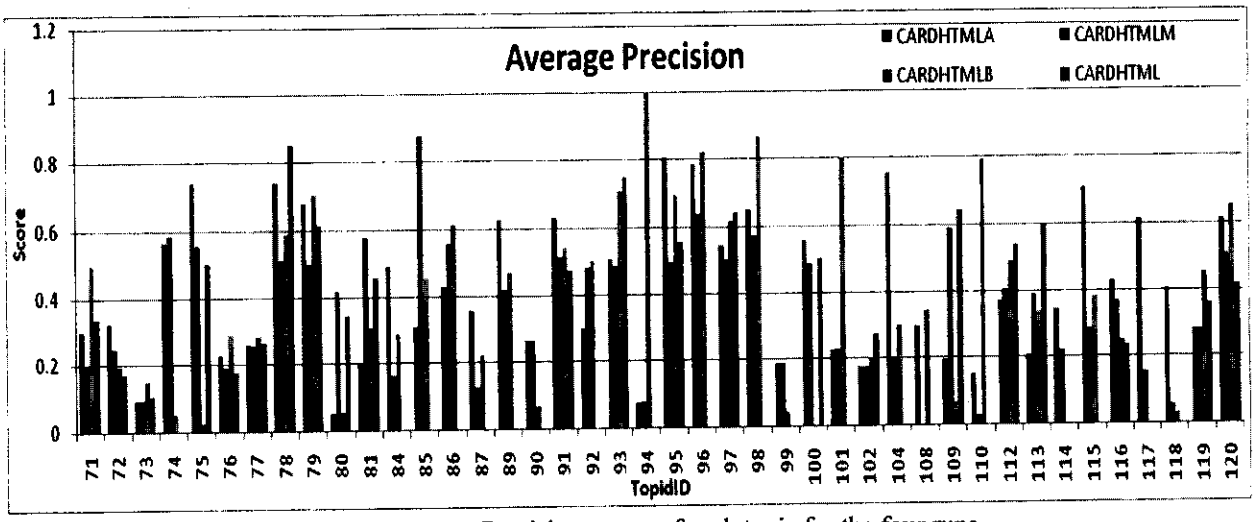

Figure 6 : Average Precision scores of each topic for the four runs

The averages of the scores of all the 50 queries are tabulated in **Table 1**

Table 1 Average of evaluation metrics

	nDCG	MAP
CARDHTMLA	0.47	0.41
CARDHTMLM	0.45	0.36
CARDHTMLB	0.32	0.41
CARDHTML	0.20	0.44

Automatic run employ high nDCG values while CARDHTML (no query processing) shows high Mean Average Precision (MAP) scores.

Besides the topic wise analysis of the results, we also summarized target entity type bases analysis. The following chart depicts the summary for run CARDHTMLA (automated query preprocessing).

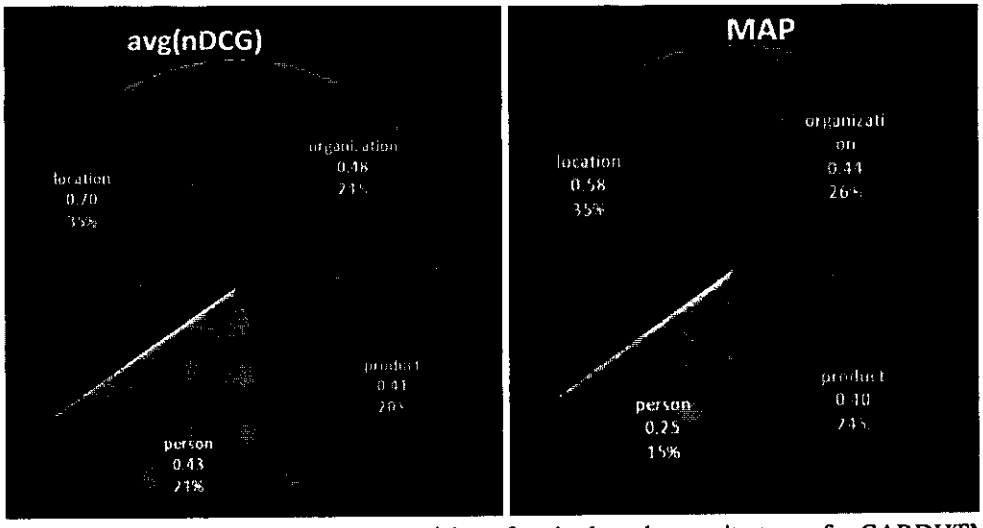

Figure 7 : average nDCG and Mean Average Precision of topics based on entity types for CARDHTMLA run

The charts in Figure 7 reveal that our approach performs better for location based queries.

7. Conclusions and Future work

The primary aim of the track is to answer (in specific, answer entities) a given question. This paper had shown a novel methodology to identify entities and to rank them in context to the given query. With the advantage of several third party parsers like html parser, sentence parser, we were able to extract semantic relationship among entities and sentences. Such relationship constituted to a network which further eased our task to rank the entities. The results shown are promising when we evaluated the queries with TREC organizers provided results. This work proves one more time that network mining algorithms can address complex problems with simple solutions.

8. References

[1] http://boston.lti.cs.cmu.edu/Data/clueweb09/

[2] http://nlp.stanford.edu/ner/index.shtml

[3] http://www.lemurproject.org/indri/

[4] http://www.lemurproject.org/

[5] S. Campinas, D. Ceccarelli, T. E. Perry, R. Delbru, K. Balog, and G. Tummarello. The Sindice-2011 Dataset for Entity-Oriented Search in the Web of Data. In *EOS, SIGIR 2011 workshop, July 28, Beijing, China.*

[6] http://htmlparser.sourceforge.net/javadoc/overview-summary.html

[7] http://incubator.apache.org/opennlp/documentation/manual/opennlp.html#tools.sentdetect

[8] http://incubator.apache.org/opennlp/documentation/manual/opennlp.html#tools.parser

[9] http://wiki.dbpedia.org/Downloads37

[10] http://data.sindice.com/trec2011/index.html