Answering queries using Name Entity Recognition, Relation Extraction and Wikipedia Infoboxes

Rishitha Dega and Pranali Yawalkar

Indian Institute of Technology Madras,
{rishitha.dega,pranali.yawalkar}@gmail.com
http://www.iitm.ac.in

Abstract. The fundamental task of answering queries is Information Retrieval. Unlike search engine which gives a set of relavant documents, we try to answer the question precisely. In this project, we propose a 4-step model to deal with the problem of answering short queries using Wikipedia Infoboxes.

Keywords: Information Retrieval, Relation Extraction, Named Entity Recognition, Dependancy Parse, Wiki Infobox, Wordnet Similarity

1 Introduction

The project comes up with a novel technique of answering short queries using a model comprising of Named Entity Recognizer, Relation Extraction , Wikipedia Infoboxes and Wordnet Similarity.

NER is the task of identifying Named entities in Natual Language. According to previous studies 70% of the queries contain a Named Entity (Guo et al, 2009; Yin and Shah, 2010 [8]). Therefore NER is an important task in analysing the query. However, traditional Named Entity Recognisers fail in recogising named entities in short queries. This is because they have very little context available.

Relation extraction extracts semantic relations between entities in text and plays a key role in question-answering task. NER is a sub-task of relation extraction, that is relation extraction categorizes the entities into one of the predefined relations. Dependency parsers can be used to identify the relations.

Wikipedia Infoboxes is the summary of the article in concern. These infoboxes can be used to produce answers for the queries.

LESK wordnet Technique of identifying similarity between between the query and the Named entity is used.

2 Literature Survey

2.1 Named Entity Recognition

Several models have been proposed in the context of Named Entity Recognition. But there is not much work in recognising named entities in short web search

queries. Analysis of Named Entity Queries in Web Search Logs by Dayong, Yu and Ting LIU [9] is one among such works.

Studies in NER can be broadly classified into two groups: Handcrafted rules and Machine Leaning (to learn the rules automatically).

1. Handcrafted Rules: This is prefered over the ML techniques when there is not large training data.
2. Machine Learning: 3 learning techniques - Supervised, Semi-spervised and Unsupervised.
   (a) Supervised Learning: The main idea of supervised learning is to automatically build rules by learning features of positive and negative examples of Named Entities over a collection of annotated corpora. These include HMMs, Decision Trees, SVMs etc. Most of SL tasks involve the following tasks - reading from a large annotated corpus, memorize list of entities and create rules based on discriminative rules. However, the main drawback of SL approach is unavailability of large annotated corpus. Therefore semi-supervised and unsupervised methods were studied.
   (b) Semi-supervised Learning: Typical approach used is bootstrapping. Basic idea of bootstrapping is to provide a set of initial seeds to start the learning process. New pairs of seeds can be obtained from the existing seeds and imposing some constrains (Ex: all alphabets in a word are capitalized implies it is Organization). Several variations of bootstrapping have been proposed.
   (c) Unsupervised Learning: The following techniques are frequently used in unsupervised learning - Clustering (Ex: cluster based on the context similarity), techniques based on lexical resourses, patterns (Ex: Hearst Patterns) and large unannotated corpus.

In this project we used Stanford library for NER : `http://nlp.stanford.edu/software/CRF-NER.shtml`. It is a Java implementation of a Named Entity Recognizer that labels sequences of words in a text which are the names of things, such as person and company names, or gene and protein names. It includes good named entity recognizers for English, particularly for the 7 classes (PERSON, ORGANIZATION, LOCATION, TIME, MONEY, PERCENT, DATE).

2.2 Details about Stanford NER

Details about model : Conditional Random Fields

- Hidden Marov Models :
  - Generative Model
  - Assumes that features are independent
  - Aim while learning parameters : maximize P(X,Y)
  - When labeling Yi future observations are taken into account (forward-backward)
  - High speed model - faster implementation
- MaxEnt Markov Models

  - Discriminative Model
  - No assumption on independence of features
  - Aim while learning parameters : maximize P(Y | X)
  - Do not take future observations into account (no forward-backward)
  - Mid range speed model
- Conditional Random Fields
  - Discriminative Model
  - Doesn't assume that features are independent
  - When labeling Yi future observations are taken
  - Refer : `http://en.wikipedia.org/wiki/Conditional_random_field` into account
  - Low speed model
  - Best of both worlds.

Details about Features :

- Word features: current word, previous word, next word, all words within a window
- Prefixes and Suffixes: Eg, Jenny = { <J, <Je, <Jen,..., nny>, ny>, y> }
- Label sequences

Details about Training :

- Large, unannotated corpus
- Each word will appear in contexts - induce a distribution over contexts
- Cluster words based on how similar their distributions are
- Use cluster IDs as features
- Great way to combat sparsity
- 200 clusters, used 100 million words from English gigaword corpus

2.3 Details about Stanford Dependency Parser

- Uses probabilistic natural language parsers, both highly optimized PCFG and lexicalized dependency parsers, and a lexicalized PCFG parser
- The lexicalized probabilistic parser implements a factored product model, with separate PCFG phrase structure and lexical dependency experts, whose preferences are combined by efficient exact inference, using an A* algorithm
- It uses a transition-based parser which produces typed dependency parses of natural language sentences using POS tags and label embeddings. The parser is powered by a neural network.
- Very fast implementation.
- Refer : A Fast and Accurate Dependency Parser using Neural Networks by Danqi Chen and Christopher D. Manning
- The neural network learns compact dense vector representations of words, part-of-speech (POS) tags, and dependency labels. This results in a fast, com- pact classier, which uses only 200 learned dense features while yielding good gains in parsing ac- curacy and speed

2.4 Related Entity Recognition

Razvan C. Bunescu and Raymond J. Mooney [6] studied the problem of relation extraction which is briefly described below.

- Designing a kernel model that uses the shortest path between two entities in the dependency graph to establish the relation between them
- The set of relations are : Predefined. Like (Person-Social, Organisation-affiliation etc)
- A subsidiary NLP model to generate the dependency graph would be needed to form the dependency graph for each sentence. A dependency path is represented as a sequence of words interspersed with arrows that indicate the orientation of each dependency. Undirected dependency graph is a connected graph.
- Keeping in mind our Named entity, and the relation under consideration, like LOCATED_AT (Person, Location) and if our Named entity (**Primary**) is India, we would find a word in the query which is classified as a Person and will find the shortest path between them in the undirected graph. If the length of the shortest path is lesser than some threshold, we can say that the two entities are related by the relation LOCATED_AT. This would give us a set of plausible relations from the initial set of relations of the model.

In this project we used the Stanford library for Lexical parser : `http://nlp.stanford.edu/software/lex-parser.shtml` After obtaining the dependency graph, we can find the entity related to the **Primary** entity using reference paper 6 which uses the shortest path hypothesis and wordnet synsets to establish relations.

2.5 Relation Extraction

We would like to discuss two domains of related work here :

1. Supervised techniques : Relation extraction as a classification task. Precision, Recall and F1 become the measures of evaluation.
   (a) Feature Based : Formulating the problem as a Binary Classification problem which classifies the entities under relation R if they are related by it. Any discriminative classifier like SVM, Perceptron, Log linear model, can be used for this. The features can be extracted using syntactic and semantic analysis of the sentence like POS tagging, parse trees, NER. Feature extraction technique can be used based on cosine similarities of the features to identify distinct and maximum encompassing features. Finally the set of features can comprise of "Words between and including entities", "Types of entities (person, location, etc)", or "Path between the two entities in a parse tree"
   (b) Kernel Based : String kernels base the intuition behind kernel methods. String kernels define similarity between two strings x and y based on their common substrings. Extending this intuition to work on metric based on

word sequences and parse trees. Word context can be an indicator for a relation by finding the match of the context(POS tagged context) with the training data and finding the corresponding relation.

(c) Concerns with Supervised techniques :
   i. Can't extend to new set of relations for every test sample
   ii. Textual analysis like POS tagging, shallow parsing, dependency parsing is a pre-requisite. This stage is prone to errors.
   iii. Annotated data is not widely available

2. Semi Supervised techniques : Bootstrapping based approaches result in the discovery of large number of patterns and relations. Approximate value of precision computed by drawing a random sample and manually checking for actual relations will give the evaluation metric for this task.
   (a) Bootstrapping with EM : Given a set of relations and the task to allot each entity pair to a relation, the unknown data to be estimated in the E stage is the relation allocation and the parameters to be learnt in the M stage are the weights given to the features in the log linear model.
   (b) DIPRE ((Dual Iterative Pattern Relation Expansion)
      i. Given a small seed set of (author, book) pairs.
      ii. Use the seed examples to label some data.
      iii. Induces patterns from the labeled data.
      iv. Apply the patterns to unlabeled data to get new set of (author,book) pairs, and add to the seed set.
      v. Return to step 1, and iterate until convergence criteria is reached

   The initial seed is obtained by crawling the web the finds all documents containing the entities pairs.

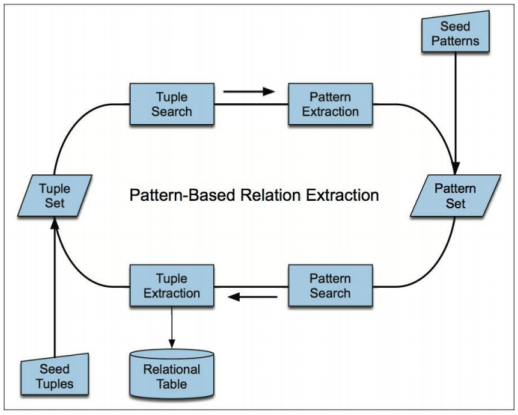

Fig 1: Relation Extraction - Bootstrapping

   (c) Snowball : Similar to DIPRE but with relaxed pattern matching. DIPRE has exact pattern matching constraint.
   (d) KnowItAll : An autonomous, domain-independent system that extracts facts from the Web. The input to KnowItAll is a set of entity classes to be extracted, such as "city", "scientist", "movie", etc., and the output is a list of entities extracted from the Web.

(e) TextRunner : DIPRE, Snowball, KnowItAll: relation types are predefined. TextRunner discovers relations automatically. It is a self supervised binary classifier.

2.6 Extracting Wiki Infoboxes

In this project we used the MediaWiki web API: Base URL forEnglish Wikipedia's API is `http://en.wikipedia.org/w/api.php`. The MediaWiki web API is a Web service that provides convenient access to wiki features, data, and metadata over HTTP.

Another way to parse Wikipedia infobox html is to use JSOUP, which gives infobox in HTML format. However, we have preferred MediaWiki because we have obtained the result in txt format and it got easier to parse. Also, the output of MediaWiki was more precise while JSOUP's was very generic. Example, while the attributes produced by the JSOUP related to "Birth" was just "Born" valued for Place, Date, Age, Mediawiki output gave separate attributes for "Birth Place", "Birth Date" and "Age".

2.7 Wordnet Similarity - LESK

We have used the Adapted LESK algorithm for finding the wordnet similarity between two entities. The application of this measure in the project is discussed below with an example. In the original Lesk Algorithm, the similarity of two words is determined by the largest number of words that overlap in their glosses or definitions. In the Adapted Lesk Algorithm, a word vector w is created corresponding to every content word in the WordNet glosses. Concatenating glosses of related concepts in WordNet can be used to augment this vector. The vector contains the co-occurrence counts of words co-occurring with w in a large corpus. Adding all the word vectors for all the content words in its gloss creates the Gloss vector g for a concept. Relatedness is determined by comparing the gloss vector using the Cosine similarity measure.

3 Methodology

The proposed problem concerns with answering queries using information from wikipedia infoboxes. There are four components associated with finding the correct answer for a given query namely :

1. Named entity recognition: To find the Primary entity concerned to which the query is related
2. Parsing: Finding the Property entity which speaks about the Primary entity.
3. Finding the relation between these two entities and lastly.
4. To query the wikipedia infobox for the Primary entity to find the value of the Property entity being asked.

Consider this example query : President of USA. Here the Primary entity is USA and Property entity is President. The relation between these two entities is that of President_Country(Primary, Property) and the answer for this relation can be extracted from the infobox of USA giving us the answer as Barack Obama. In this case the Property entity overlapped with the relation being established. Consider another example query : Head of USA. Here the Primary entity is USA and Property entity is Head. The relation between these two entities is that of President_Country(Primary, Property) and the answer for this relation can be extracted from the infobox of USA giving us the answer as Barack Obama. Establishing the President_Country(Primary, Property) relation is done by the relation extraction job. The possible relations that a Property entity can have with the Primary entity is predetermined in the model. Then the wordnet similarity is computed between the Property entity and all the properties listed in the infobox of the Primary entity. Also the wordnet similarity between the relationship extracted and the properties listed in the infobox of the Primary entity is found. The best match with highest wordnet similarity is taken to be the answer and the corresponding value in the infobox is returned. For example, consider the second query above, all possible properties of USA from its infobox are : Capital, Largest city, Official languages..President, Area, Population, etc. The aim of the relation extraction task is to identify which relations (here : President_Country) among the predefined set of relations follow the Primary entity , Property entity pair. The wordnet similarity will return relation President_Country to be closest to property President (from the infobox) and the answer Barack Obama will be returned.

The work-flow of the algorithm is shown in the figure below:

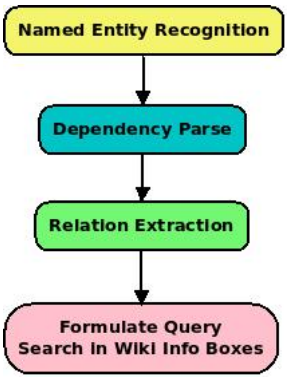

Fig 2: Steps involved in Answering queries

Example: Wife of ShahRukh
Step 1: Identifying Named Entity. Recognizes ShahRukh as a PERSON. The output is: Wife/O of/O ShahRukh/PERSON. O - Not a named Entity

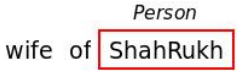

Fig 3: Recognising the named entity

Step 2: Dependency Parse: Identifies the dependency of Wife on the Named Entity. The output of the dependency parse is: root(ROOT-0, Wife-1) prep_of(Wife-1, ShahRukh-3)

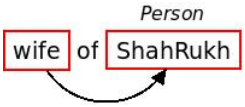

Fig 4: Entity depending on the Named Entity

Step 3: Relation Extraction: Returns the entry in the infobox of the named entity which has highest WordNet similarity with the entity found using dependency parse. WordNet similarity is found between Wife and {birth_name, birth_place, occupation, ..., Spouse, Children}. The output is Spouse.

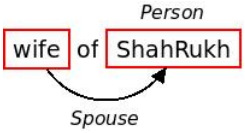

Fig 5: Finding the relation between entities

Step 4: Query using WikiInfo Boxes. Returns the value corresponding to the entry which had highest WordNet similarity. Output : Gauri Khan.

Fig 6: Wiki Infobox of Shah Rukh Khan

Query URL is composed of the following

1. Base URL: This is the base URL for English Wikipedia's API.
2. Action: API supports over 50 actions. We use "action=query" to tell the API that we need to get some data.
3. Format: Again the API supports many formats including text, json, xml. We use "format=txt" to tell Wikimedia API that we want the output in text format.
4. Action-specific parameters:
   (a) titles: This parameter contains the title wikipedia page from which we want to extract the information. Ex: title=Sachi%20Tendulkar says that we want to access the information in `en.wikipedia.org/wiki/Sachin_Tendulkar`. %20 is percent-encoding of space.
   (b) prop: This parameter tells that we are interested in particular revision of the page mentioned in titles parameter. We use "prop=revisions", since no revision information is provided the API returns information of the latest revision.
   (c) rvprop: This parameter indicates that we need the content of the latest revision of the page. We use "rvprop=content" to tell API that we need the contents of the page.
   (d) rvsection: This parameter specifies from which section we want to retrieve the information. We use "rvsection=0" to access information in the infoxbox of a page.

The following URL, says to return the Infobox content of Shah Rukh Khan.

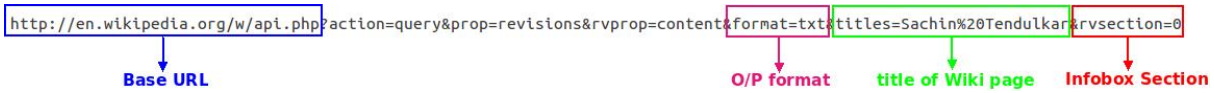

Fig 7: Wiki Infobox of Shah Rukh Khan

4 Experiments and Results

In this section, we report the experimental results that validate the efficiency of the proposed technique.
Experiments were carried on synthetic dataset containing roughly 100 queries. All the experiments are implemented on a 1.1GHz, 4GB memory PC running Ubuntu 11.04
Note: All the queries considered contain a Named Entity.

The following table gives the time taken to run various tasks.

#Queries	Avg NER time	Avg Parse time	Avg Best match time	Avg total time
100	2250.36ms	1944.21ms	23540.36ms	27734.93ms

The following table shows results of the algorithm on 100 queries (synthetic dataset).

#Queries	Failed at NER stage	Failed in finding answer	Correct Answers
100	4	7	89

$$Precision = \frac{\#Correctly\ answered\ Queries}{\#Identified\ Queries}$$

$$Precision = \frac{89}{100}$$

$$Precision = 0.89$$

Some queries are unidentified in the sense that the named entity is not recognized. Example, for the query "When was CERN founded?", CERN is not identified as the NE but for "Who were the founders of CERN", CERN is identified as NE.
Dependency parser may fail for longer queries where long ranged dependencies are unidentifies.
Example where MediaWiki api may fail is : Consider example query as "Profit of Nestle" where Nestle is identified as the Named entity. However, the wikipedia article on Nestle `http://en.wikipedia.org/wiki/Nestle‘` where the entity named being Nestlè. Since they don't match, Wiki API doesn't return the page on Nestlè.
Example where Wordnet similarity measure may fail is - "What is Jennifer Aniston doing in her life?" and the relation extracted is "doing life" which matches most with attribute "residence" than "occupation" if considered without context.
Example templates of Queries used are given below: On an average each query has 4-5 words.

1. [Entity] in [PLACE] .
   (a) Largest city in India.
   (b) Countries in South Africa.
   (c) Total members in Lok Sabha.
2. [Entity] of [PLACE].
   (a) Poupulation of Tamil Nadu.
   (b) Capital of Chennai.
   (c) Emblem of Andhra Pradesh.
3. [Entity] of [PERSON]
   (a) Who is the CEO of Google?
   (b) Number of satellites of Neptune.
   (c) Political party of Sonia Gandhi.
4. [Entity] of [ORGANIZATION]
   (a) Coach of FC Barcelona.
   (b) head of World Health Organization.
5. Others
   (a) Who preceded Narendra Modi?
   (b) Awards won by Alan Turing?
   (c) Harry Potter was written by whom?
   (d) Sonia Gandhi belongs to which organization?

5 Conclusions and Future Work

In this project, we proposed a 4-step approach to solve the problem of answering short queries using Wiki infoboxes. We have focussed in providing quick answers to short queries with the help of information in wikipedia infoboxes. Some of the applications of this could be on the levels of google knowledge graph result which comes up for short queries. If the model is tuned further for finess, robustness and wider incorporatability, it can be an efficient search engine for short queries.

- Remembering the past history in order to answer future queries. For example:
  Query1: Wife of ShahRukh. Answer: Gauri Khan.
  Query2: Her age. Answer: 44.
- Making the model work for search engine queries instead of just phrases. The major problem with real world queries is that they are not properly structured, therefore the traditional named entity recogniser fails. For example:
  Query: president of usa. Here USA is not written in capital letters, hence traditional NERs fail to recognise USA as a LOCATION.

References

[1] Ang Sun, Ralph Grishman, Satoshi Sekine - "Semi-supervised Relation Extraction with Large-scale Word Clustering "
[2] Girish Keshav Palshikar, Sachin Pawar, Pushpak Bhattacharyya - "Semi-supervised Relation Extraction using EM Algorithm"
[3] Nguyen Bach, Sameer Badaskar - "A Review of Relation Extraction "
[4] ZHOU GuoDong, LI JunHui, QIAN LongHua, ZHU Qiaoming - "Semi-Supervised Learning for Relation Extraction "
[5] Katrin Fundel, Robert Kuffner and Ralf Zimmer - "RelEx-Relation extraction using dependency parse trees"
[6] Razvan C. Bunescu and Raymond J. Mooney - "A Shortest Path Dependency Kernel for Relation Extraction"
[7] Smith, T.F., Waterman, M.S.: Identification of Common Molecular Subsequences. J. Mol. Biol. 147, 195–197 (1981)
[8] Xiaoxin Yin and Sarthak Shah. 2010. Building Taxonomy of Web Search Intents for Name Entity Queries. In Proceedings of WWW-2010
[9] Dayong WU, Yu ZHANG, Ting LIU. Analysis of Named Entity Queries in Web Search Logs.