

WikiKB – Relation Extraction From Wikipedia

Vikrant Sagar and Prachi Maheshwari

{vikrant, prachi}@cise.ufl.edu

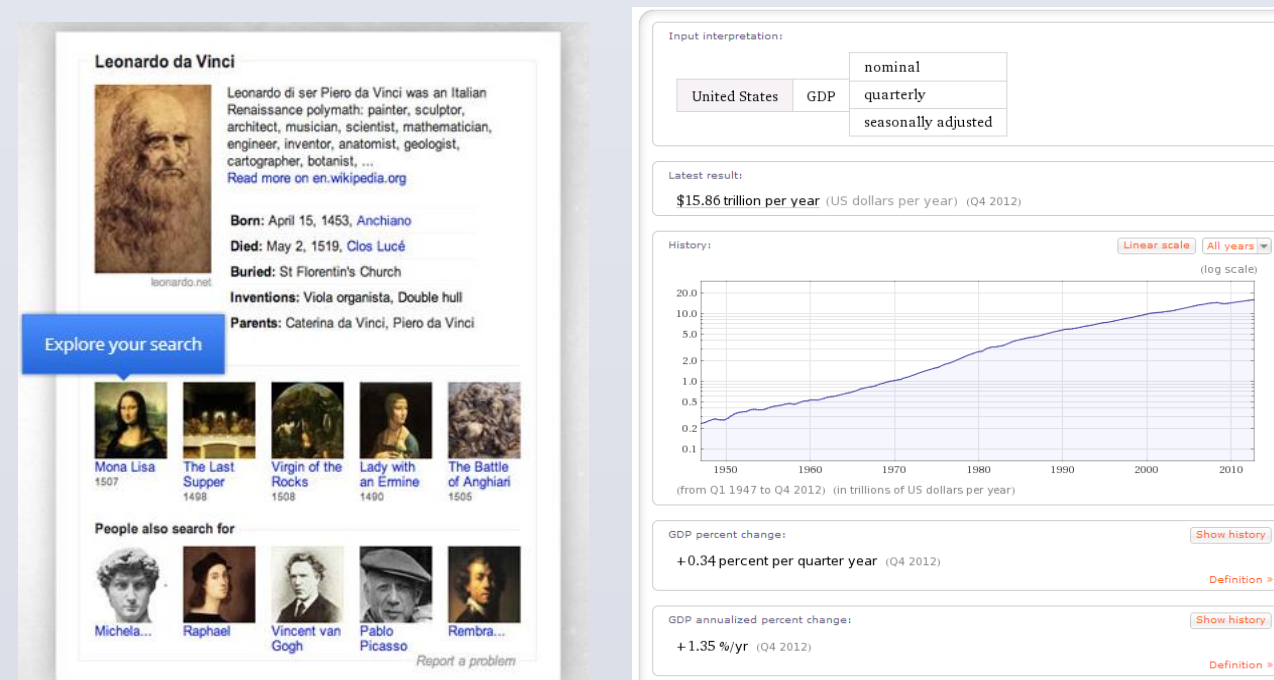


ABSTRACT

Wikipedia contains millions of articles created using collaborative editing and the only way to access them is to search for an article. This way users manually navigate Wikipedia to get the needed information. Another way is use search engine and write free form query and get thousands of search results sorted by their relevance to the free form query. The third way is to go directly from one article to another using the internal links present inside an article. All these ways require users to navigate manually to get to their information. **Our project aim is to find and implement Natural Language Processing algorithms to extract relation triplets (Subject, Relation, Object) from each and every article and store them in a graph format/database and then provide a Query Interface to get the answers, not search results.** Our methods are based on the finding readily used patterns when writing something in Natural Language. We are using manual and automatic approaches to find patterns in Wikipedia articles. These patterns are then used to extract similar information from all other similar articles. Pattern verification is done manually to get more accurate relations rather than any Statistical approach.

MOTIVATION and CURRUNT TECHNOLOGIES

We all use one or the other search engine hundreds of time each day to find the information we are looking for. But all search engines give top results not answers. Although the problem to get accurate answers is very hard but some technologies are making progress in providing good answers to a query. Some of these technologies are:



Google Knowledge Graph

Wolfram Alpha

The above two technologies are using a very large related dataset to execute queries on and get back with answers. But how this related information is extracted and stored is what the first part our project. Some technologies that are used to extract relations and store them are:

Instance	Iteration	date learned	confidence
benjamin is a European person	719	31-mar-2013	96.5
phone_dessa is a item found on a table	721	04-apr-2013	99.8
northwest_florida is a state or a province	724	12-apr-2013	96.9
dune_dunlop is a hobby	719	31-mar-2013	95.0
john_a_katites is a male	719	31-mar-2013	94.8
diary is an agricultural product coming from animals001	724	12-apr-2013	100.0
jim_rangers works for dake_energy	719	31-mar-2013	96.9
jan_wilbos coaches in the league 01	724	12-apr-2013	100.0
charles03 has citizenship in the country (country)	719	31-mar-2013	99.6
staples is a male person who moved to the state (or province) new_york_city	722	08-apr-2013	100.0

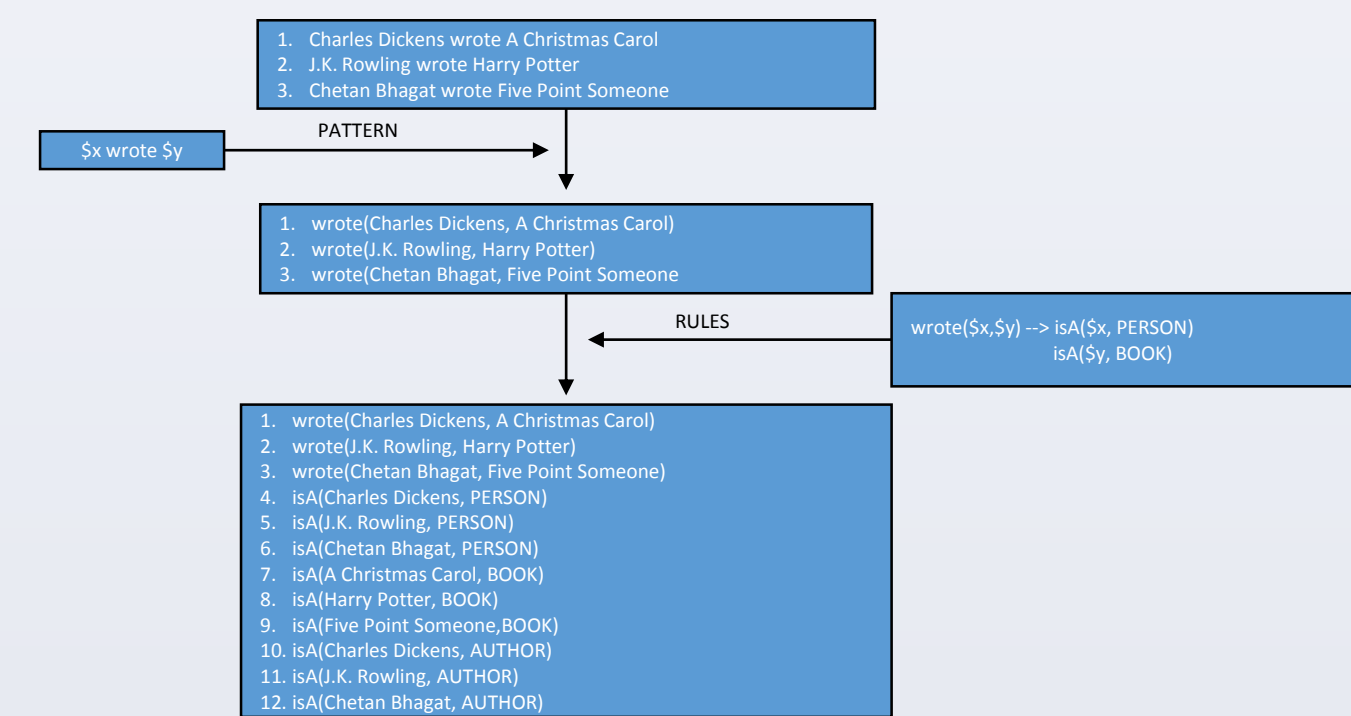
NELL

To above shown project extract relations from text but cannot link them as they not have any information on the whole context of the text. Our approach is to extract relations from an article which talks about a concept so we can build relations without having any disambiguity.

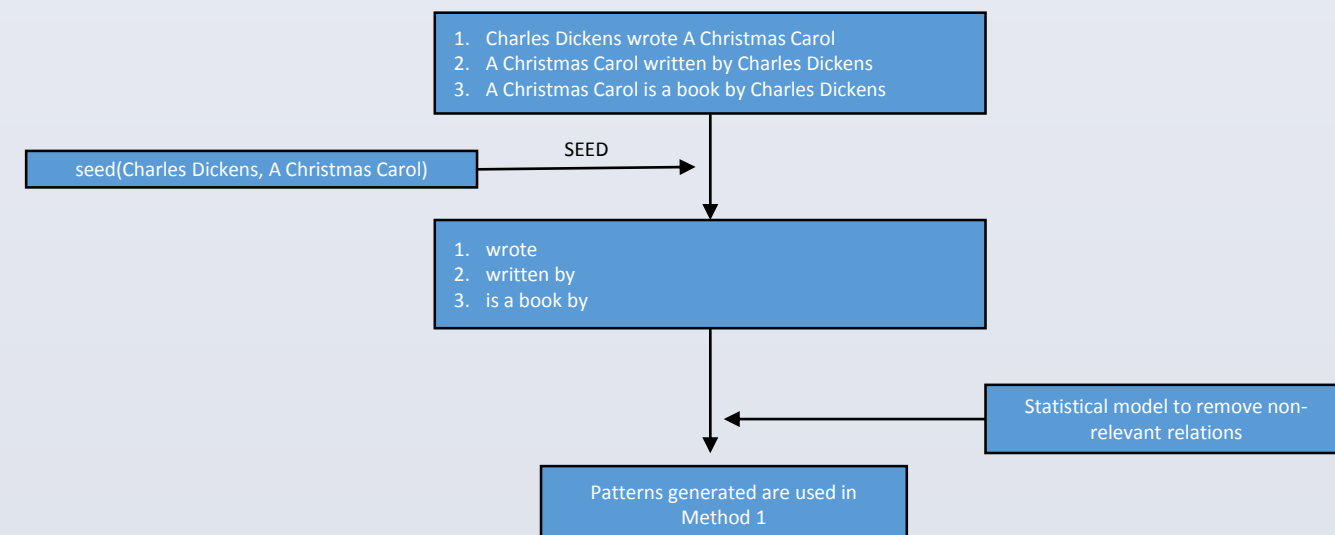
ALGORITHMS

We have implemented some pre-existing algorithm for relation extraction and also modified them to perform better. The most important part in our algorithms is that they all need pre-annotated input before the extraction process starts. We have listed the algorithms used in flow diagrams.

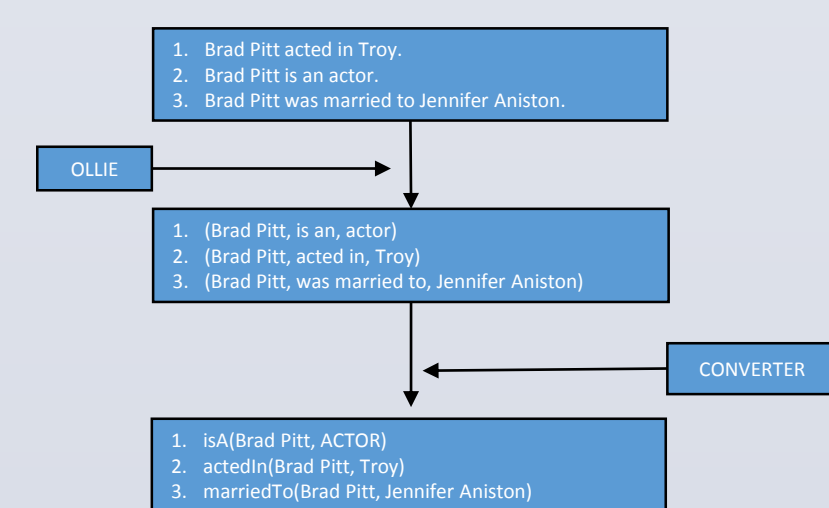
1. Manual Pattern Extractor



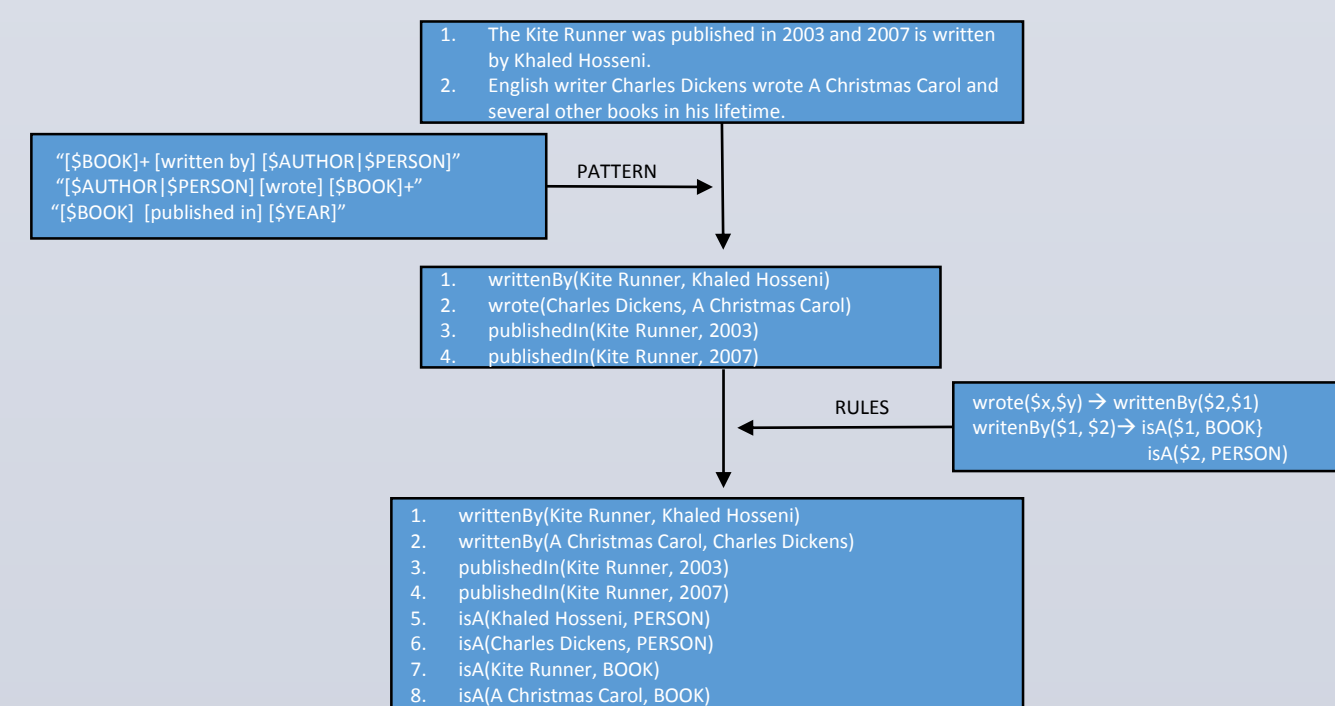
2. DIPRE



3. Dependency Parsing



4. Modified DIPRE

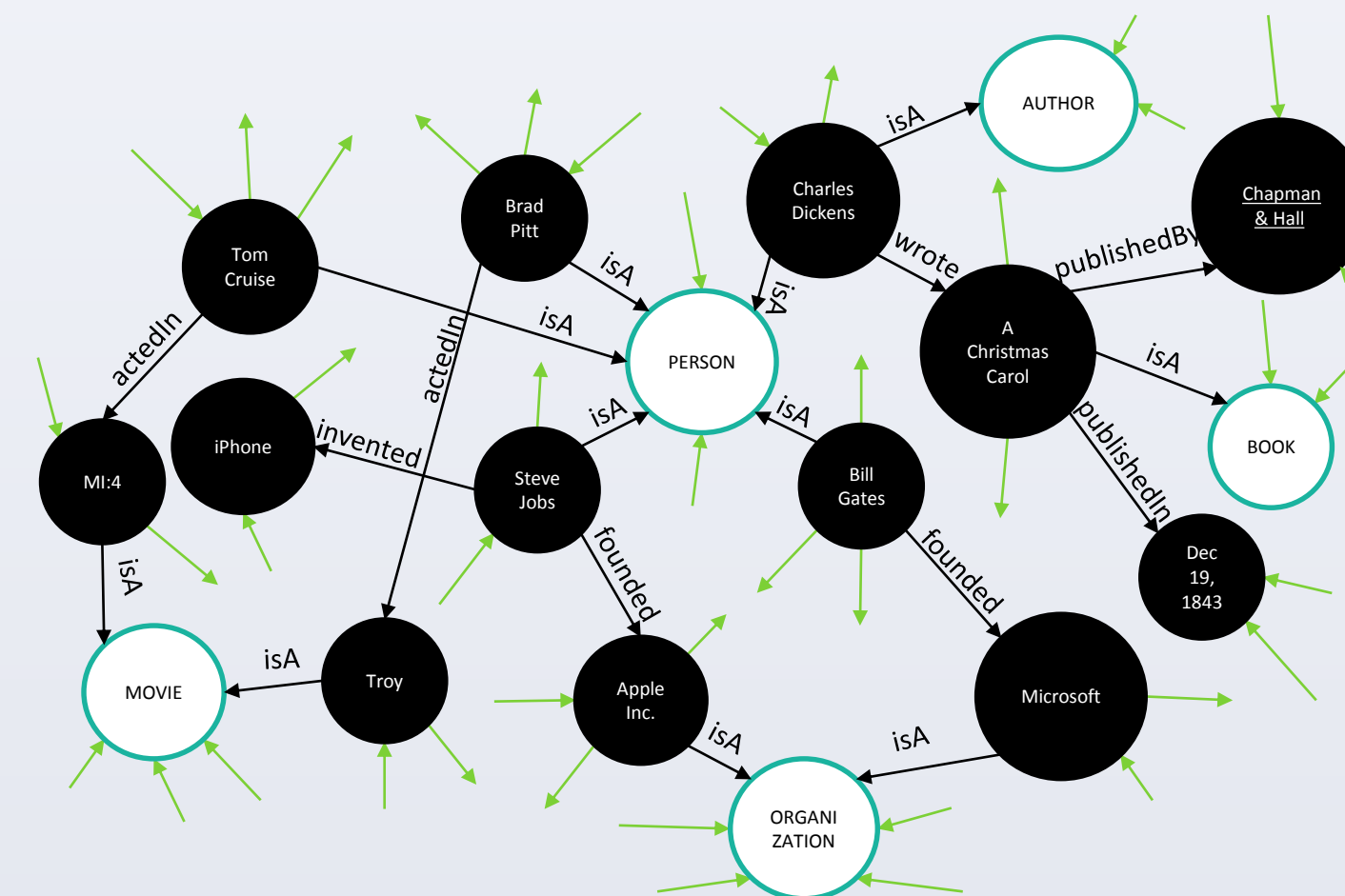


Two more algorithm are also implemented. Search and Extract algorithm finds answer from unstructured text. Recursive Descent Parser algorithm uses LALR grammar to form dependency tree.

All these methods require their input text to be annotated with type of entity information. This is achieved using Trie data structure. We have used bi-gram sliding window to annotate our input text.

DATABASE and QUERY INTERFACE

We will be using Graph Database to store the extracted relations. We have used Neo4j database to store and query our relations. Neo4j provides a good query language to traverse the graph data structure. Below is an example relation cloud which is stored in the database.



Query Interface:

Since the extracted relations are stored in graph database, we can easily write complex queries just selecting a starting point and traversing through its connections. Below are some sample queries using Graph Query Language Cypher(Neo4j):

QUERY:

```
START n=node(name: "Charles Dickens")
MATCH n->[r:wrote]->a->[r1:publishedIn]->b
RETURN a.name, b.name;
```

QUERY:

```
START n=node(name: "MOVIE")
MATCH n<-[r:isA]-a<-[r1:actedIn]-b
RETURN b.name, a.name;
```

QUERY:

```
START n=node(name: "PERSON")
MATCH n<-[r:isA]-a-[r1:isA]->b
WHERE b.name = "AUTHOR"
RETURN a.name;
```

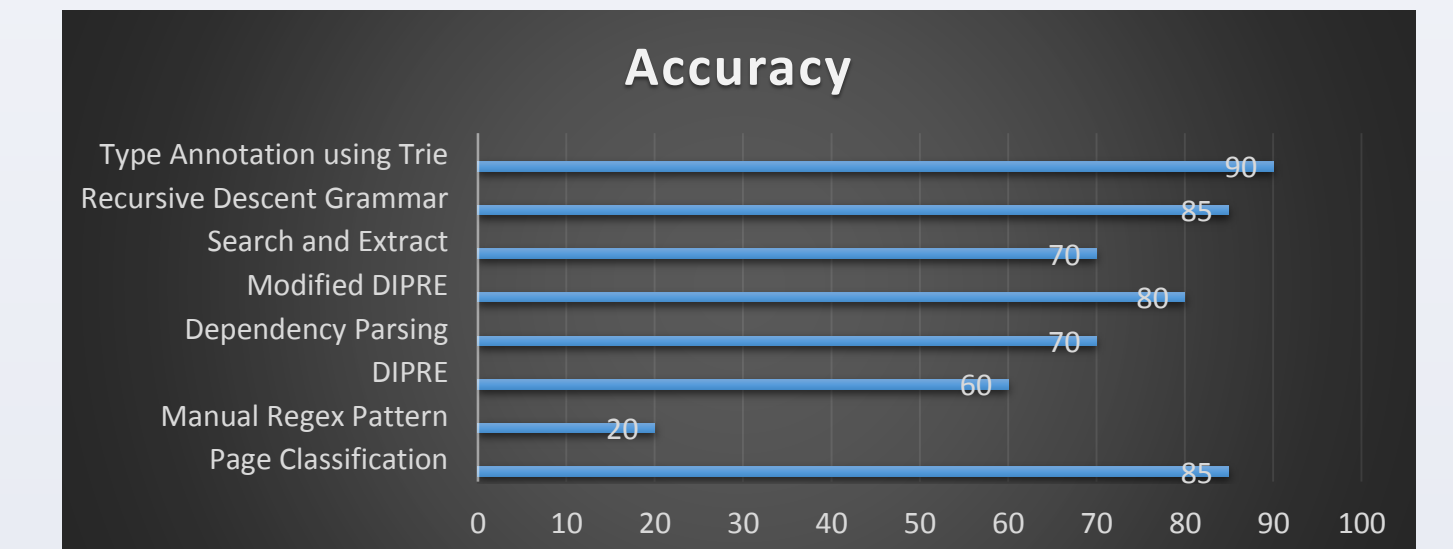
QUERY:

```
START n=node(name: "ORGANIZATION")
MATCH n<-[r:isA]-a<-[r1:founded]-b
WHERE b.name = "Bill Gates"
OR b.name = "Steve Jobs"
RETURN a.name, b.name;
```

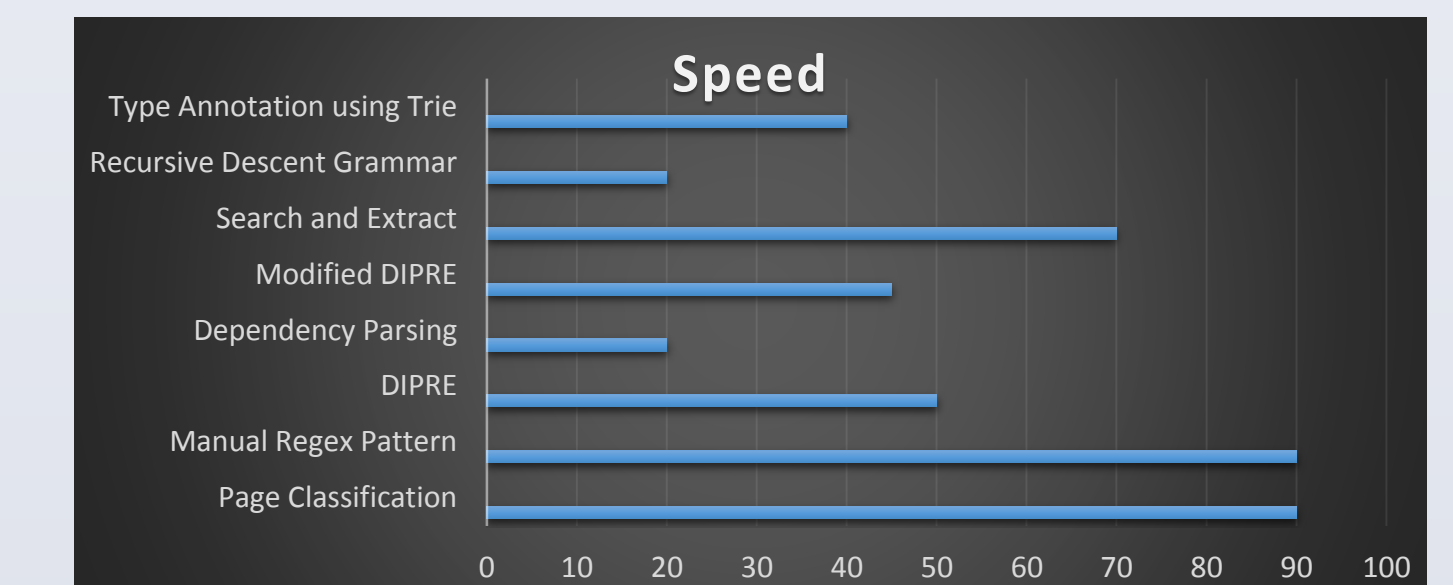
Free Form queries is the next step in Query Interface.

RESULTS

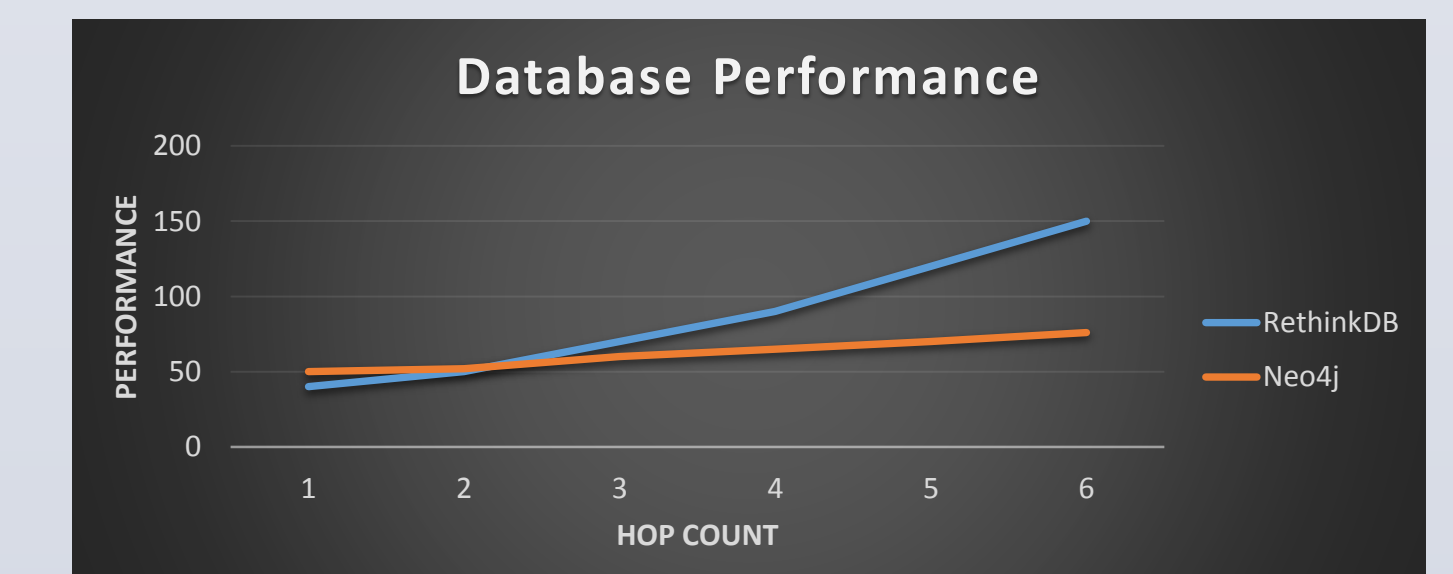
Results are divided into two parts, Accuracy and Speed. Accuracy is calculated based on the percentage of correct extracted relations.



Speed is calculated based on the run time of the algorithms on the same dataset where baseline is just reading of the dataset without any processing.



We have also gathered around 1 million relations and stored them in both key/value database RethinkDB and also in graph database Neo4j. We tried to compare the performance of both database based on the no. of hops we are doing in our query i.e. no. of nodes traversed.



REFERENCES

1. Wikipedia Dataset – <http://wikipedia.org>
2. Text Runner - <http://openie.cs.washington.edu/>
3. Reverb - <http://reverb.cs.washington.edu/>
4. Freebase – <http://freebase.com>
5. DBpedia - <http://dbpedia.org/>
6. DIPRE - <http://www.alexmayers.com/projects/DIPRE/>
7. Snowball: Extracting Relations from Large Plain-Text Collections
8. Yago - www.mpi-inf.mpg.de/yago-naga/yago/
9. NELL - <http://rtw.ml.cmu.edu/rtw/>
10. SOFIE: A Self-Organizing Framework for Information Extraction
11. Ollie - <https://github.com/knowitall/ollie>