Truthfulness Verification System

Tathagata Dasgupta, ABM Musa
Email: {tdasgu2, amusa2}@uic.edu
URL: http://code.google.com/p/tverifier

1 Introduction

Web has became the most prevalent source of information now a days. However, many information on the web is untruthful. Also because of the widespread reach of web, sometimes web is used to propagate untruthful facts for social and political reasons. Hence with the increasingly use of web as information source, verification of information truthfulness became an important facet to consider.

The popular search engines extract information from web based on keywords and metadata without considering the truthfulness of the facts. Also, there have been not much research in this area. To our best knowledge, the most recent work on this area is T-verifier [5], which uses results from search engines to verify truthfulness of statements. T-verifier performs very well on the test-dataset. However, T-verifier has some problems with the overall approach to verify the truthfulness of statements and there is room for improvement. In this work, we will extend the T-verifier system so that it's weaknesses can be resolved to get a more robust truthfulness verification system.

2 Current System

Current system for truthfulness verification is called T-verifier [5], which uses two phase methods for truthfulness verification of statements. Each of these two phases rely heavily on search results returned by popular search engines. T-verifier takes the doubtful statements (DS) as input from the user along with the doubtful unit (DU). Phrase after removing DU from DS is called topic unit (TU).

At the first phase, T-verifier generates alternative statements by supplying TU to search engine and collecting the relevant alternate DU-s. However, from basic web search may result in lot of alternate DUs that are not semantically or logically relevant to the original DS. Hence, T-verifier uses combination of seven features to rank alternate DUs. These features primarily exploit the facts that relevant alternative units frequently co-occur, people often mention both misconception and truthful conception together, data-type matching, and sense-closeness. T-verifier chooses top 5 alternative statements based on top 5 alternate DUs obtained in this phase.

At the second phase, top 5 alternative statements from phase 1 is supplied to the search engine again. Then the returned searched result is ranked by multiple rankers such as Alternative Unit Ranker, Hits ranker, Text-feature Ranker, Domain Authority Ranker etc. Then all those ranks are

merged to form an overall ranking among the alternate statements and top statement in this final merged ranking is considered as truthful statement.

3 Problems with Current System

Although results from the tested dataset achieves good performance (90% accuracy), failing of T-verifier for some statements shows that it has some inherent problems and there is room for improvement.

First, T-verifier assumes that truthful statements will be more propagated in the web compared to the untruthful statement. However, this may be not true in general because of intended and planned propaganda for establishing some untruthful statements. This kind of propagandas are even becoming more common now a days due to widespread reach of Internet. T-verifier also showed that *"Hillary Clinton is the President of United States"* has more hits than *"Hillary Clinton is the Secretary of State"*. Although T-verifier was able to find the correct statement in this case using multiple ranks together, in general the untruthful statement can be prevalent in the web compared to truthful statements.

Second, T-verifier do not use the reputation of the information source. T-verifier uses only search results returned by the search engine irrespective of the origin and believability of the information origin. Hence there is room for improvement here to give more weight to the information obtained from trustworthy sources such as Wikipedia.

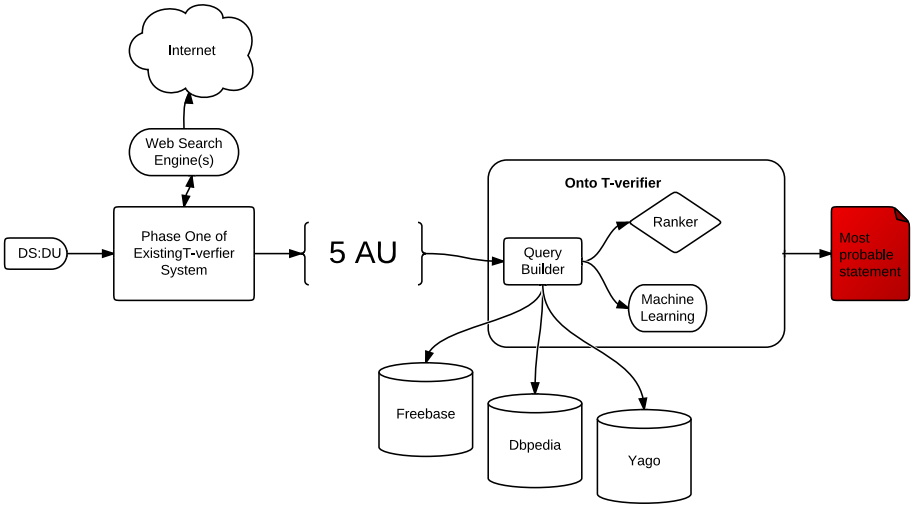
4 Propsed System

4.1 Description of extraction algorithm

For disambiguation among the alternative statements, Wikipedia is generally used as an authoritative source. However, the contents of Wikipedia is not available in the form that is consumable in a programmatic format. To address such difficulties a number of projects Yago, DBpedia, Freebase have organized the massive amount of data in a searchable fashion e.g DBpedia uses SPARQL endpoint, Freebase uses MQL api. Open source implementation of Python wrappers exist for both the interfaces exist and appear to be mature enough for our needs.

4.1.1 Freebase

Freebase has information about approximately 20 million Topics, each one having a unique Id, which can help distinguish multiple entities which have similar names, such as Henry Ford the industrialist vs Henry Ford the footballer. Most of the topics are associated with one or more types[1] (such as people, places, books, films, etc) and may have additional properties like "date of birth" for a person or latitude and longitude for a location. Freebase not only contains data from the Wikipedia but also other sources; users can submit data to the Freebase datastore and expand it in richness. We tinkered with the api[2] and it appeared to be the most viable starting point for the project.



(a) System overview

Figure 1: System overview

Listing 1: Minimal code to Freebase

```
import freebase
import pprint

query = [{
  "a:starring": [{
    "actor": "Meg Ryan"
  }],
  "b:starring": [{
    "actor": "Tom Hanks"
  }],
  "type": "/film/film",
  "*": [],
}]

pp = pprint.PrettyPrinter(indent=4)
result = freebase.mqlread(query)

print "Movie names & their various forms"
```

```
for i in result:
        pp.pprint(i["key"])
```

Listing 2: Cleaned Output

```
Movie names & their various forms
[    '158982',
     'You$0027ve_Got_Mail',
     '18171032',

      ...
     'E-m$0040il_f$00FCr_Dich',
     'youve-got-mail']
[    '176489',
     'Joe_Versus_the_Volcano',
     'Joe_Vs$002E_The_Volcano',
     'Brain_Cloud',
      ...
     '2327353',
     'joe-versus-the-volcano']
[    '226198',
     'Sleepless_in_Seattle',
     'Sleepless_In_Seattle',
      ....
     '169146',
     '106482',
     'Insonnia_d$0027amore',
     '62812',
     'Schlaflos_in_Seattle',
     'sleepless-in-seattle'\]
```

The above results show how the three movies starring Tom Hanks and Meg Ryan. When we query Google with Tom Hanks and Meg Ryan, the top result is a page from Answers.com where a user has asked which are the movies where the two actors appear together, and the answer lists these three movies namely - "Joe versus the Volcano", " Sleepless in Seatle" and "Youv́e got Mail". A quick lookup of the Wikipedia and IMDB pages also confirm the same.

4.1.2 Dbpedia

DBpedia is a similar project to Freebase, but it focuses mainly on the content available from Wikipedia. It scores in being precisely importing the data from the info boxes in Wikipedia pages, but at this stage it does not seem to be offering anything additional over Freebase [4]. We are yet to explore its programmatic interface [3].

4.1.3 Yago

YAGO is a semantic knowledge base with over 900,000 entities (like persons, organizations, cities, etc.) and uses Wikipedia and Wordnet as its main source of information. We are yet to explore the programmatic interfaces it provides and how we can use it for the project.

4.2 Step-by-step Algorithm for Truthfulness Verification

4.3 Examples of Truthfulness Verification

4.3.1 Use of Freebase

Consider the following sentence:

"Tom Hanks was the lead actress in the movie Sleepless in Seattle"

Content words for this sentence are "Tom Hanks", "lead", "actress", "movie", "Sleepless in Seattle".

Now for actress we will not find any result from Freebase query. So we will get synset of actress from wordnet. Wordnet synset for actress content for actress is *"female actor"*

Now for the noun phrase *"sleepless in Seattle"*, we can generate "id" for the query as "sleepless in seattle". But this id will be associated with many properties. To select the relevant property we can use synset obtained from the actress. And this synset has actor, which is one of the property for id "Sleepless in Seattle". Hence the Freebase query can be following:

```
[{
  "id" : "/en/sleepless_in_seattle"
  "/film/film/starring" : [{ "actor" : null }]
}]
```

Now the result of this query returns following output:

```
{
  "code":           "/api/status/ok",
  "result": [{
    "/film/film/starring": [
      {
        "actor": "Tom Hanks"
      },
      {
        "actor": "Meg Ryan"
```

```
      },
      {
        "actor": "Bill Pullman"
      },
      {
        "actor": "Rosie O'Donnell"
      },
      {
        "actor": "Rob Reiner"
      },
      {
        "actor": "Victor Garber"
      },
      {
        "actor": "Gaby Hoffmann"
      },
      {
        "actor": "Carey Lowell"
      },
      {
        "actor": "David Hyde Pierce"
      },
      {
        "actor": "Ross Malinger"
      },
      {
        "actor": "Frances Conroy"
      },
      {
        "actor": "Rita Wilson"
      }
    ],
    "id": "/en/sleepless_in_seattle"
  }],
  "status":          "200 OK",
  "transaction_id": "cache;cache03.p01.sjc1:8101;2011-03-09T05:09:44Z;0032"
}
```


One of the actors in this result set is "Tom Hanks" that matches with our content word "Tom Hanks" in the given sentence. Now as we know earlier that actress means female actor, we can use the keyword female to find the fact that female is type of gender and Freebase id of "tom hanks" has a property gender associated with it. So we can formulate following query.


```
[{
  "id" : "/en/tom_hanks"
  "/people/person/gender" : {}
}]
```

The result of this query is following:


```
{
  "code":             "/api/status/ok",
  "result": [{
    "/people/person/gender": {
      "id":     "/en/male",
      "name":  "Male",
          .
          .
          .
}
```


Here the gender is Male, which contradicts with our gender female. Hence we can decide that this statement is false.

For finding the true statement i.e. the actress we can use all actors obtained in the first query result and form second query with their names and output the truthful sentence if we get female as the gender.

4.3.2 Use of Wikipedia

Although structured databases such as Freebase, DBPedia extract the information from Wikipedia and format them into a structured and query-friendly way, it does not contain all the relevant information all the time. For example, consider the following sentence:

Less Paul invented the electric guitar

The wikipedia page "http://en.wikipedia.org/wiki/Electric_guitar" contains the inventor of the electric guiter in one of the sentences in the body. But this information is not available in the Freebase.

We can use a very simple and elegant method to find the relevant information from Wikipedia by using simple web-scrapping and pattern matching using regular expression. The process is described below.

After getting the wikipedia page we can convert the html file into text file and then we can separate all the sentences in different lines. The sentence is the part of the whole text that exist between two full-stops. Another important observation is that the relevant information will be within the same sentence in the question-answer type sentences almost all the time. So we can match the content words to the sentences extracted from Wikipedia page and verify the truthfulness of the sentence.

For the example sentence above, after processing the Wikipedia page and matching by inventor we get following two sentences:

"Commercial production began in late summer of 1932 by Electro-Patent-Instrument Company Los_Angeles, a partnership of Adolph_Rickenbacker, PaulBarth and George_Beauchamp, the inven-

tor"

In a morecontemporary style, Little Willie Joe, the inventor of the Unitar, had a rhythmand_blues instrumental hit in the 1950s with "Twitchy", recorded with the ReneHall Orchestra

Now we can match the doubt unit Les Paul with these two sentences and we will not have any match. Hence we can conclude that the statement is not true.

Moreover, we can also look for truthful statement by using those above extracted sentences. We can POS (Parts-of-Speech) tagger to get the noun phrases and create the list of possible persons that invented the electric guitar and match them with the five alternative sentences given by T-verifier.

5 Conclusion

It this paper, we described current system for truthfulness verification of statements found in the web called T-Verifier and described the possible improvements to the current system. We will primarily use authentic information sources such as Wikipedia and Ontologies extracted from Wikipedia such as FreeBase, DBPedia, and YOGA to found the truthfulness of doubtful statements.

References

[1] http://blog.freebase.com/2007/03/19/the-type-system-in-freebase/.

[2] http://code.google.com/p/freebase-python/.

[3] http://sourceforge.net/projects/sparql-wrapper/files/sparql-wrapper-python/1.4.2/.

[4] http://wiki.freebase.com/wiki/dbpedia.

[5] Clement Yu Xian Li, Weiyi Meng. T-verifier: Verifying truthfulness of fact statements. In *ICDE*, 2011.