IntelligentFactChecker

Akshit Bhatia, Nalanda Chakrabarti and Muhammad Faizan Batra

Universität Paderborn

1 Problem Definition

Build a corpus-driven fact-checking engine, which returns a confidence value between -1 (fact is false) and +1 (fact is true) given a fact from DBpedia.

2 Approach

1. **Finding Subject, object and Predicate in the sentence**

   (a) **Find Predicates**: Based on the training data, we selected certain predicates which are used in majority. As per the infobox on wikipedia, we know that all the data are arranged based on some keywords. For example, for wife, the infobox uses keyword: 'spouse'. So, we look out for those words in a sentence and remove it from that sentence. Thus, our new sentence would contains a subject and an object along with certain stopping words.
   Different Predicates: We can have predicates of two types. A one word predicate or two words predicates. For example: spouse, author, starring etc are one word predicate. While birth place, death place, prime minister etc are two words predicates. While searching two words predicates, to do it efficiently, we searched only for place in the sentence and then selected the word lying before place in that sentence. That word would tell us the predicate that has to be used. i.e if it is birth place, innovation place or death place that has to be searched.

   (b) **Separate Subject from Object**: On observing the training data, we could see that majority of sentences has 'is' in the sentence. Then there were some sentences with "'s" in the sentence. Using this we were able to determine how to separate subject from object in the sentence. We didn't remove any stop words because all of those stop words are important too, as you will see later. The word with "'s" is the subject that we searched on wikipedia. We replaced 's with zzz for those words. At the last, there were sentences that have neither "is" nor "'s" for example, Camp Rock stars Nick Jonas. So after removing stars, we divided the sentence into 2 parts.

(c) **Remove unnecessary symbols** : Now that we have replace "'s" with zzz, we remove all the other unnecessary symbols like fullstop or comma from the sentence.

(d) **Searching through Internet** : After performing the above three steps, we are left with a tuple with Subject, Object and a predicate. For searching, we use Wikipedia infobox. The class FetchOnline.java is responsible for doing this. The next section describes this approach in details.

(e) **comparing the information from internet against the fact** : We used Levenstein Distance to compare two objects. We used this, as there is a possibility that on internet, the word is not in its stemmed form but with us, we may have the root word. The Levenstein distance, will calculate the number of changes to make both words similar. So it was the best approach.
We also thought of using the Jaccard Similarity, but that will require Shingling of tokens, and that will be waste of efficiency for the program. As we know that amount of data will involve maximum of 5 words on each side(usually), so using tfidf and cosine metrics would have also been a misadventure. Thus Levenstein distance was best choice.

2. **Searching through Wikipedia**

(a) **Library used** : To access wikipedia, we sticked with JSOUP. To access information, Wikipedia has provided several endpoints that can be used by the developers.
URL: https://en.wikipedia.org/w/index.php?search=<XXX> with XXX being the keyword, we can search information related to this keyword. From this information, we searched the elements that contain the word "infobox".

(b) **Differentiating between Subject and Object** : Earlier we have marked 'zzz' to the word that had "'s". So the word with this information, we used it as the Subject and this we search from the Wikipedia.

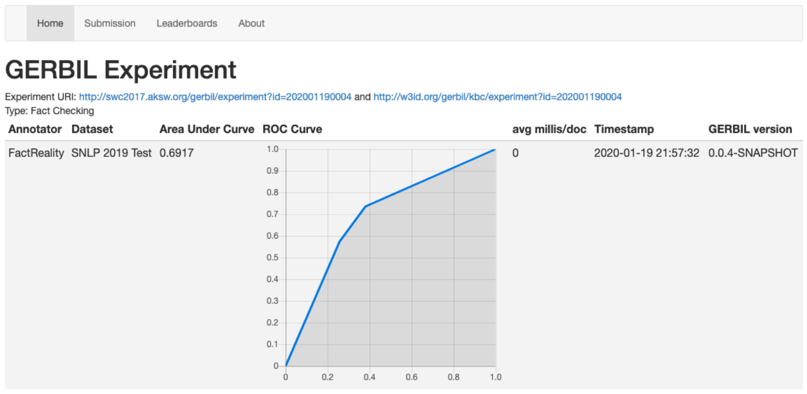
(c) **Calculate the edit distance** : Now that we have the Wikipedia result for our predicate, we compare that information against the object that we have. We determine the length of object, and then for every sentence searched from Wikipedia, we compare as many words from that sentence as there are in the object. As mentioned earlier, Levenstein Distance metric is used here.

(d) **Normalising the edit distance** : The Levenstein Distance only gives the number of changes required. We used the normalising formula to convert that distance into the error measure in percentage. Thus 1 minus this measure will tell us our confidence that the information is similar to the object and hence we can tell if the fact is true or false.

3 Result

Taking the approach defined above, our model has done a pretty good job. It wasn't a tough program to code and it incorporates an intelligent way to divide the sentence into predicates, object and subject. Based on the test data, our result have been around 70%. Below is the result AUC.

URL: http://swc2017.aksw.org/gerbil/experiment?id=202001190004



GERBIL Experiment

Experiment URI: http://swc2017.aksw.org/gerbil/experiment?id=202001190004 and http://w3id.org/gerbil/kbc/experiment?id=202001190004
Type: Fact Checking

Annotator	Dataset	Area Under Curve	ROC Curve	avg millis/doc	Timestamp	GERBIL version
FactReality	SNLP 2019 Test	0.6917		0	2020-01-19 21:57:32	0.0.4-SNAPSHOT

Fig. 1. Area Under Curve - 0.69

4 Examples - Working and Non Working

FactId	Fact_Statement
3407450	Rolandas Paksas' office is Lithuania.
3857435	London is Hawley Harvey Crippen's nascence place.
4304857	Bochum is G Data's innovation place.
3610986	Kill Bill stars Seth Gree.

Table 1. Working Example.

FactId	Fact_Statement
3316015	Nobel Prize in Literature is John Strutt, 3rd Baron Rayleigh's honour.
3212682	Sacramento Kings is Shane Battier's squade.
3339820	Nobel Peace Prize is Doris Lessing's honour.

Table 2. Non Working Example.

This model works only with certain number of predicates. Below is the list of predicates involved in the model.

(a) **spouse:**
  - spouse
  - wife
  - better half

(b) **author:**
  - spouse
  - wife
  - better half

(c) **starring:**
  - star

(d) **died:**
  - death place
  - last place

(e) **born:**
  - birth place
  - nascence place

(f) **innovation:**
  - innovation place

(g) **award:**
  - award

(h) **prime minister:**
  - role

(i) **team:**
  - team