Simple Fact Checker

Pritilata Saha[1] and Abhirup Sinha[1]

Universität Paderborn, 33098 Paderborn, Germany
psaha@mail.uni-paderborn.de, abhirup@mail.uni-paderborn.de

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

1 Approach

In our approach, we used the training data to learn about resemblance between predicates in our data and Wikipedia predicate. At first, We created a fact triplet of subject, object and predicate from training data. Afterwards, we searched Wikipedia for a given subject. In the relevant Wikipedia page, we searched for our object and noted down the corresponding Wikipedia predicate. Thus, we made a mapping between our training predicate and corresponding Wikipedia predicate along with frequency.

For testing purpose, we used a probabilistic approach to predict a fact is true or false, based on the mapping between fact predicate and Wikipedia predicate.

The detailed discussion of our approach is as follows-

1.1 Making (Subject, Object, Predicate) triplet

At first, we transformed each fact sentence in training data to a fact triplet (Subject, Object, Predicate). For this task, we used Stanford CoreNLP library [2]. First, we found out the root verb in each sentence and then split the sentence based on that. Then we proceeded to make the triplet, by using rule-based approach to find subject, object and predicate.

Finding Root Verb in a Sentence:- We performed parts-of-speech tagging, using POS-Tagger from Stanford CoreNLP [4,5], to find the root verb of a sentence. For example, the root verb of the sentence "Gentry Lee is The Garden of Rama's generator." is "is" as well as the root verb of the sentence "Stars Jason Mewes has been clerked Clerks." is "has been". By using POS-tagger, we searched for these root verbs and then split that sentence according to the place where root verb occurred.

However, in some cases, ambiguity arises due to structure of a sentence. E.g. in "Om Shanti Om stars Uma Thurman", it is clear that "stars" is the root verb. However, "stars" is also plural form of noun "star". So, our tagger mislabeled it as plural noun. But judging by the structure of other nouns in our training data, we designated any lower-case plural noun as a verb.

Determining Subject, Object and Predicate:- After splitting the sentence into “first part” which is before root verb and “second part” which is after root verb we determine subject, object and predicate of a sentence. From training data we can see that there are some sentences having “ ’s or s’ ” with the subjects. For rest of the sentences there are no “ ’s or s’ ” with the subjects. We applied two different methods for these two types of sentences.

i. **Sentences with “ ’s or s’ ”:-** If the first part of the sentence contains the word with ” ’s or s’ ” we considered that part as the subject part and we took the part up-to the word with ” ’s or s’ as subject and after subject rest the words of first part was marked as object. The second part after the root verb was taken as the object.
   E.g. in this sentence “Tamara Brooks’ spouse is Thom Mathews.” the first part is “Tamara Brooks’ spouse” and the second part is “Thom Mathews”. The first part consists “ s’ ”, so our approach takes “Tamara Brook” as subject, the part after “ s’ ” till the end of first part as predicate which is “spouse” and the second part “Thom Mathews” is object.
   Similarly, if the second part contains ” ’s or s’ ” word then from the beginning of the second part till that word is the subject, the rest words of the second part is the predicate, and the first part is object in this case.
   E.g. this sentence ”London is Fred Zinnemann’s last place” contains “’s” in second part so we took the second part “Fred Zinnemann” up to “ ’s ” as subject the remaining of the second part “last place” as predicate and the first part “London” as object.
ii. **Sentences without “ ’s or s’ ”:-** For this type of sentences we marked the first part of the sentence as subject, the root verb part as object and second part as object.
   E.g. in this sentence “Last Action Hero stars Anthony Quinn”, first part is “Last Action Hero” which is subject, the root verb “stars” is the predicate and second part “Anthony Quinn” is object here.

1.2 Searching Wikipedia for a given fact

The MediaWiki Action API is a web service, which allows users to perform certain wiki-actions like page creation, authentication, parsing, searching, etc. The English Wikipedia API has endpoint as *“https://en.wikipedia.org/w/api.php”*. Our program sends requests to the API, using this endpoint, to get access to wiki features like search for a title, parse content etc., in XML format [3].

For each fact, using the API, we searched Wikipedia for a given subject and got the most relevant page-ID. Using that page-ID, we accessed the relevant page’s content to search for object and predicate.

We used regular-expression-based pattern-matching to find predicate for a given object. Usually, Wikipedia page infobox follows some generic patterns for its contents. Some of these patterns are -

i. `| WORDS = [[WORDS]] |`

```
ii. | WORDS = [[WORDS]], [[WORDS]] |
iii. |WORDS = WORDS [[WORDS | WORDS]]|
iv. | WORDS = {{WORDS | [[WORDS]]}} |
v. |WORDS = {{WORDS | [[WORDS]] | NUMBERS}}|
vi. |WORDS = {{WORDS|[[WORDS]]|NUMBERS}} {{WORDS|[[WORDS]]|NUMBERS}}|
vii. | WORDS = {{WORDS | * [[WORDS]] * [[WORDS]] * [[WORDS]] }}|
```

On closer inspection, we found that on the right-hand side of "=" symbol, we might find our desired object; and if we find our object, left hand side of "=" symbol serves as relation to object, which we can consider as "wiki predicate". In our case, the "wiki predicate" can be an inflected form of our predicate in consideration, or it might be a different one. So, we had to build a probabilistic model to determine relation between "predicate", "wiki predicate" and "fact truth value".

1.3 Building Probabilistic Model

After finding the Wikipedia predicate we compared that predicate with the predicate we found form our sentence triplet. There were some cases where the Wikipedia's predicate didn't match with the sentence predicate exactly. E.g. for "last place" Wikipedia XML uses keyword "death_place", for "honour" it uses "awards", for "subsidiary" it uses "subsid". To compare these type of synonyms, we used a probabilistic model which will determine if the predicates are similar or not. Before making any prediction, we preprocessed our predicates. We used common english stopwords filtering to removes words from predicate, that are not useful for prediction. Then for every predicate, we used the first word to compare the predicates. We used our training data to learn such similar words from both the predicates, then predicted on test data according to that. The following equation 1, serves as the base equation of our model.

$$\mathcal{P}(Fact\,Value|Predicate\,,Wiki\_Predicate) = \frac{Count(Predicate \cap Wiki\_Predicate) + 1}{Count(Predicate \cap Wiki\_Predicate \cap Fact\,Value) + V} \tag{1}$$

For each fact value we calculated the probability using using the above mentioned formula. Then with the highest probability we took that as expected fact value and predicted that value.

2 Diagrams

A class diagram shows a set of classes, interfaces, and collaborations and their relationships, and is used to model the static design view of a system. A sequence diagram is an interaction diagram, which shows interactions, consisting of a set of objects and their relationships, including the messages that may be dispatched among them. It addresses the dynamic view of a system and emphasizes the time-ordering of messages [1]. Class diagram and sequence diagram of our application is shown in Fig. 1 and Fig. 2 respectively.

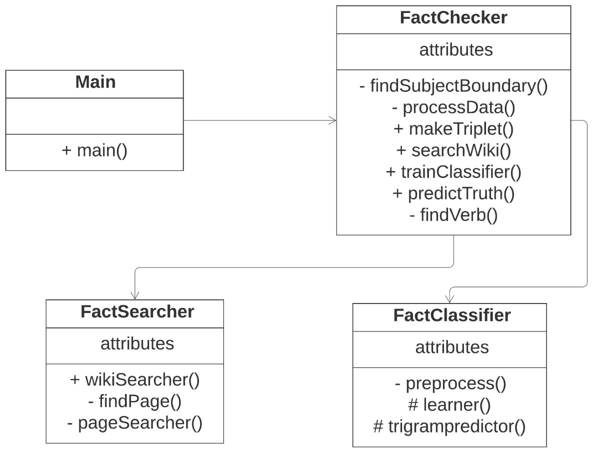


Fig. 1. Class Diagram for Our Application

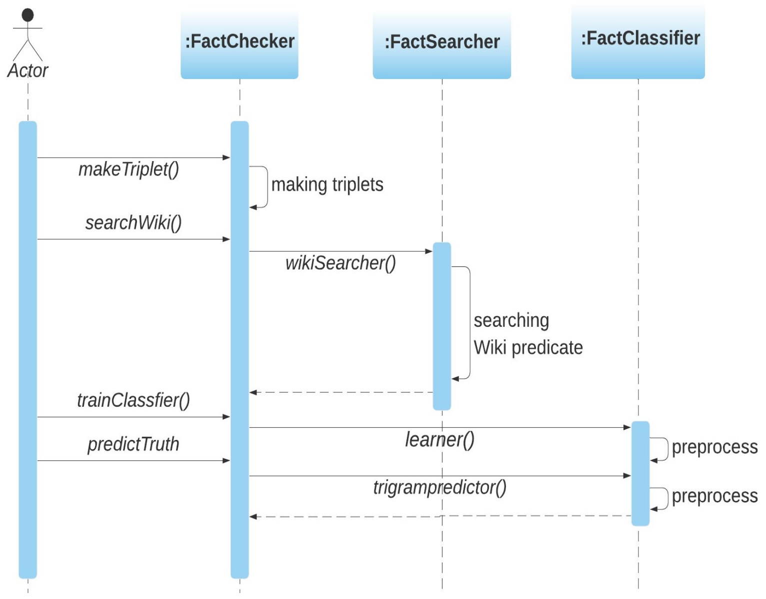
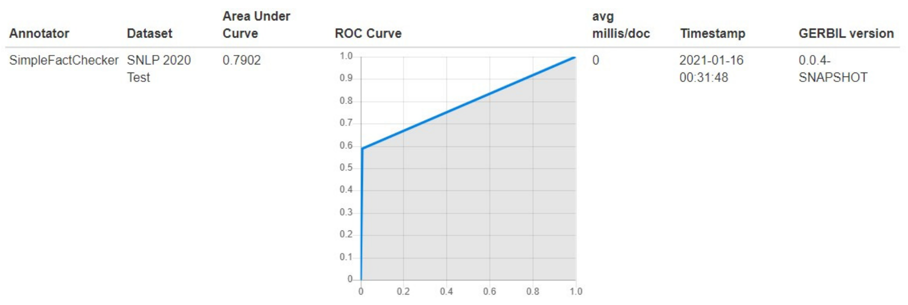


Fig. 2. Sequence Diagram for Our Application

3 Results

Following the approach as described above, our model has done a pretty good job in checking facts in test data. It incorporates a intelligent way to split each fact sentence into subject, object, predicate; and provides a way to search Wikipedia using MediaWiki API. Based on test data, our AUC score has been near 80%. Fig. 3 is a snapshot of our model's AUC score and ROC curve, which can also be seen at the below mentioned URL -

URL : http://gerbil-kbc.aksw.org/gerbil/experiment?id=202101160001



Annotator	Dataset	Area Under Curve	ROC Curve	avg millis/doc	Timestamp	GERBIL version
SimpleFactChecker	SNLP 2020 Test	0.7902		0	2021-01-16 00:31:48	0.0.4-SNAPSHOT

Fig. 3. AUC Score (0.79) and ROC Curve of Our Model

Data and code used in our work is also available on GitHub (URL : https://github.com/Pritilata95/SimpleFactChecker).

4 Discussion

Below, we have discussed about some of the pros of our approach, and some the shortcomings we noticed. For the shortcomings, we used data which are not in our test dataset. We have also discussed about our thoughts on tackling the drawbacks in our model.

4.1 Pros of our approach -

- Most common method of information collection is web-scraping, in which one can fetch data available in a web page's HTML tags. But sometimes, some data might not be available in HTML tags. Our approach comes handy in such cases. As we parse XML schema of a page, we can scrape nearly all possible information a page has in it's infobox, thus overcoming the problem of data non-availability in plain sight.

- Our pre-processing methods even work with unicode characters. So, we could preserve subject and/or object words with unicode characters like German umlauts, accented characters in Spanish or French etc. Also, the pattern-matching approach referred in Section 1.2 works with unicode characters, so we suffer from no loss of data due to unicode chracters.
- As mentioned in Section 1.2, we have used the most relevant page's data for a given subject. So, even if our subject in consideration and Wikipedia page title differs a bit, we can still perform fact-checking for that fact.

Table 1 shows some of the facts from test dataset, which were classified properly by our model.

Table 1. Some Correctly Classified Facts from Test Dataset

Fact_ID	Fact_Statement	Predicted_Value
3603476	Mad Max stars Anna Maria Perez de Taglé.	0
3377893	Frédéric Passy's award is Nobel Prize in Literature.	0
3857330	Albert VII, Archduke of Austria's birth place is Wiener Neustadt.	1
3479689	Christel Khalil is Susannah Harker's better half.	0
3876934	Seville is Amerigo Vespucci's last place.	1
3861976	Oracle Corporation's subsidiary is List of acquisitions by Oracle.	1
3216781	Shake Milton's team is Philadelphia 76ers	1

4.2 Cons of our approach -

- Running time of our application is significantly high (around 25-30 minutes), as we had to search Wikipedia for each and every facts.
  Future Improvement- There is a scope of improvement here. As a future improvement, we can think of incorporating cache-like concept to reduce run-time while searching.
- For a given fact, sometimes the most relevant page returned by MediaWiki API is not the one referred in the fact. E.g., If we search for the fact "After Worlds Collide is When Worlds Collide's sequel.", our subject in consideration will be "When Worlds Collide". When we perform our search using that, most relevant page as returned by Wikipedia is the 1951 film adaption of our subject in cosideration i.e. "When Worlds Collide (1951 film)". The situation is depicted in Table 2.

Table 2. Example of Wrongly Returned Wikipedia Page

Fact_Statement	Subject	Most Relevant Wikipedia_Title
After Worlds Collide is When Worlds Collide's sequel.	When Worlds Collide	When Worlds Collide (1951 film)

Future Improvement- If we search for exact match of our subject in consideration with Wikipedia titles of most relevant pages, then we can avoid this problem. However, this kind of search will increase run-time further as we have to go through titles of a list of pages. Also, this can affect our application's flexibility as mentioned in 3rd point of Section 4.1, as exact match cannot be found in some cases.

- In some cases, there is no infobox present in Wikipedia. Our application fails to fetch any data in such scenarios. Such an example is shown in Table 3.

Table 3. Example of Page having No Infobox

Fact_Statement	Subject	Wikipedia_Title
Cole Porter is You'd Be So Easy to Love's generator.	You'd Be So Easy to Love	You'd Be So Easy to Love

Future Improvement- Instead of our regular expression-based infobox search, a search for relevant information can be performed on all sections of the Wikipedia page in consideration. This will improve the model's accuracy, but run-time might also increase.

- In cases like a company or several companies are subsidiary of a company or a company's subsidiary is the list of acquisitions by the company, our model fails to return truth value if the information is not explicitly present in Wikipedia infobox. Some examples are provided in Table 4.

Table 4. Example of Pages having No Explicit Information on Subsidiary

Fact_Statement	Subject	Object	Wikipedia_Title
Facebook's subsidiary is List of acquisitions by Facebook.	Facebook	List of acquisitions by Facebook	Facebook
Carbonite, Inc.'s subsidiary is Mozy.	Carbonite, Inc.	Mozy	Carbonite, Inc.

Future Improvement- We can tackle this problem in a two-pronged approach. Firstly, if a company's name occur in both subject and object part, we can consider the fact as true and skip Wikipedia checking for that fact. Secondly, if name of subsidiaries of a company is not mentioned in infobox, we can search all the sections of the page to find our desired name.

- Although the regular-expression pattern we works for most of the pages in Wikipedia, not all pages follow the same regex pattern in infobox. In such scenarios, it is not being possible to fetch any information. The following Table 5 provides some facts for which our application failed to fetch desired information.

Table 5. Example of Pages having Different Regex Patterns

Fact_Statement	Subject	Object	Predicate	Wikipedia_Title
Paul Pogba's team is Manchester United.	Paul Pogba	Manchester United	team	Paul Pogba
Shivaji's birth place is Shivneri	Shivaji	Shivneri	birth place	Shivaji
Nobel Prize in Physics is C.V. Raman's honour.	C.V. Raman	Nobel Prize in Physics	honour	C. V. Raman
Neils Bohr's last place is Copenhagen, Denmark.	Niels Bohr	Copenhagen, Denmark	last place	Niels Bohr

Future Improvement- It is required to improve the existing regular-expression based pattern-matching used in our application. Or one can also use other APIs for collecting data from Wikipedia. HTML scraping can also be used; but, as outlined in Section 4.1, we refrained from using HTML based scraper as it cannot fetch data not occurring in the webpage's HTML tags. However, HTML tags may come in handy for the cases mentioned above, as it would not require learning every types of data pattern in Wikipedia or forming very complicated regex expressions. We believe, by using HTML tag based scraping, along with regex-based scraping, we can collect a lot of information from a given page.

References

1. Booch, G., Rumbaugh, J., Jacobson, I.: Unified Modeling Language User Guide, The (2nd Edition) (Addison-Wesley Object Technology Series). Addison-Wesley Professional (2005)
2. Manning, C., Surdeanu, M., Bauer, J., Finkel, J., Bethard, S., McClosky, D.: The Stanford CoreNLP natural language processing toolkit. In: Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations. pp. 55–60. Association for Computational Linguistics, Baltimore, Maryland (Jun 2014). https://doi.org/10.3115/v1/P14-5010, https://www.aclweb.org/anthology/P14-5010
3. MediaWiki: Api:main page — mediawiki, (2020), https://www.mediawiki.org/w/index.php?title=API:Main_page&oldid=4244097, [Online; accessed 22-January-2021]
4. Toutanova, K., Klein, D., Manning, C.D., Singer, Y.: Feature-rich part-of-speech tagging with a cyclic dependency network. In: Proceedings of the 2003 Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics. pp. 252–259 (2003), https://www.aclweb.org/anthology/N03-1033
5. Toutanvoa, K., Manning, C.D.: Enriching the knowledge sources used in a maximum entropy part-of-speech tagger. In: 2000 Joint SIGDAT Conference on Empirical Methods in Natural Language Processing and Very Large Corpora. pp. 63–70. Association for Computational Linguistics, Hong Kong, China (Oct 2000). https://doi.org/10.3115/1117794.1117802, https://www.aclweb.org/anthology/W00-1308