A retrieval-based chatbot
Natural Language - 1st Project
Instituto Superior Técnico - 2019/2020

João Tiago Aparício
Student number: 97155
joao.aparicio@tecnico.ulisboa.pt

Rostislav Andreev
Student number: 97040
rostislav.andreev@tecnico.ulisboa.pt

Joaquim Rocha
Student number: 97020
joaquim.rocha@tecnico.ulisboa.pt

1 Introduction

This project aims to implement a retrieval-based conversational agent. The knowledge base will be composed of the FAQs taken from "Balcão do Empreendedor" (document provided). Every group of similar questions and their respective answers from the FAQ's file (knowledge base) is identified by a unique ID. The goal of this project is to compare pre-processing techniques, as well as similarity metrics, to understand how to maximize accuracy on answers given by the agent. The implementation of the agent should output the answer (id) that best fits the input given by the user. To do so, each question input by the user will be compared to something in the knowledge base.

2 Proposal

To achieve the previously defined goal, we implemented and compared various pre-processing techniques and selected the one with a better performance. First and foremost, we start by describing all the pre-processing methods we used across all the implementations.

One of the pre-processing methods implemented was the removal of stop-words from the questions to be compared. We selected the 560 most common stop-words from the Portuguese language and grouped them in a .txt file. Another technique used was the lowercasing of all the questions. Punctuation and accents are also removed using regular expressions. A stemmer is tested individually in different analyses.

To identify what can be considered a word, we must do the tokenization process in our corpus. We compound terms to combine all the same context/application simple terms into a unique single one. The punctuation of the corpus also brings us problems. We need to make sure that our tokenization process doesn't ruin our text structure and logic. Lowercasing the whole corpus is essential to avoid data sparseness that could give us ambiguity. We also considered relevant to do the normalization of our questions to avoid potential redundancy as we go through our process. We also applied stemming to our tokens, using the RSLP Stemmer from the nltk library. Several metrics to obtaining the similarity measures between the questions in the corpora and the user input were chosen. When choosing the metrics, we based our decision on the lecture notes for this course, "Natural Language Processing a Nut(s)shell" [1]. The similarity measures chosen are tested individually as follows: Jaccard similarity, Dice similarity, TF-IDF, Cosine similarity (available on detail in Jurafsky's book [2]) and a weighted similarity analysis based on the method described in "Using subtitles to deal with Out-of-Domain interactions" [3]. These metrics were implemented using numpy and scikit-learn (Python libraries) to run faster.

$$Jaccard(s,t) = \frac{|s \cap t|}{|s \cup t|} \tag{1}$$

$$Dice(s,t) = 2 \times \frac{|s \cap t|}{|s| + |t|} \tag{2}$$

$$tf - idf(t,d,D) = tf(t,d) \times idf(t,D) = \tag{3}$$

$$= freq(t,d) \times \left(\log \left(\frac{|D|+1}{|d \in D : t \in d| + 1} \right) + 1 \right) \tag{4}$$

$$cosine\_similarity(x,y) = \frac{x \times y}{||x|| \times ||y||} = \tag{5}$$

$$= \frac{\sum_{i=0}^{n-1} x_i y_i}{\sqrt{\sum_{i=0}^{n-1} x_i^2} \times \sqrt{\sum_{i=0}^{n-1} y_i^2}} \tag{6}$$

It's important to emphasize that the formula calculates the tf-idf used differs from the one given in the lecture notes since it offered better results. Our version of the weighted analysis by Magarreiro et. al, does not use Lucene. Our implementation only uses 3 methods and does not take into account the "Time difference between trigger and answer" as this work will probably be graded automatically. Another reason for not using this metric is that, in the tests presented in the article, it proved to not be a relevant measure, as the percentage of plausible answers decreased with its use. We instead use response frequency similarity with input (defined here as the questions on the provided corpus, or triggers) and answer similarity with input. We used each of the previously referenced similarity measures individually and interchangeably in the same way as described by Magarreiro et. al.

Our implementation follows the formula below. Where M_1, M_2 and M_3 denote respectively similarity with input, response frequency, answer similarity with input. "Each inter-action $(T_i, A_i) \in U$", where U is the corpus given. And the final score of each answer A_i to the user input u. The $score(A_i, u)$ is calculated as a weighted average:

$$score(A_i, u) = \sum_{j=1}^{3} w_j M_j(U, T_i, A_i, u) \tag{7}$$

We also assigned the weights where $w_1 = 0.7$, $w_2 = 0.2$ and $w_3 = 0.1$ The goal was to give more importance to M_1 and least weight to M_3. The value of M_2 was normalized with all the values per each pair of sentences analysed. It was the configuration that provided the best results for our goal. More extensive scientific work can be written to study the distribution of these weights.

Implementation

3 Used corpora

The corpora used for this set of experiments is based on the second homework. We paraphrased each question to have at least 4 different questions per answer. We used the UTF-8 encoding because we want to keep the integrity of the corpus of the answers and questions. The knowledgebase format used is XML. The distribution of questions per answer in the knowledge-based given for the project is not uniform. Meaning, there is a highly variable number of questions for each answer. The corpus also exhibits the existence of indistinguishable questions with the same answers corresponding to different id's. To deal with this problem we created a method that keeps record ids of the duplicated questions. This method is also used to weight the questions that are equal but have different answers. The total number of answers is 615. And the total number of questions is 2641.

When we manipulate our project, we must follow some good practices in our process. One of them is to divide the corpus into different sets: training set, testing set, and the development set. In our project, we used a Python dictionary that extracts and groups the questions in an array by IDs.

$$dict = \{id_1 : [q_1, q_2, q_3, ..., q_n], ...\} \tag{8}$$

When all the questions related to a specific ID are in their respective array, the one in the last position of the array is picked and used for the test and is then removed from the array.

4 Experimental results

In this section, we discuss the tests we used to choose the algorithm we submitted. For simplifying notation, the implementation based on the one by Magarreiro et. al[3] will be called WAM (for weighted average metric). Following WAM a letter is added to specify the similarity metric used for the algorithm: J for Jaccard, D for Dice, C for Cosine. In each cell are the rounded accuracy (two decimal points) and the time each test took in seconds rounded (two decimal points).

The configurations for the tests are the following:

The data is always lowercased;

Test 1 - Data with stop words;

Test 2 - Data with no stop words;

Test 3 - Data with no stop words, and is stemmed;

Test 4 - Data with no stop words nor punctuation;

Test 5 - Data with no stop words nor punctuation, and is stemmed;

The tested results for our implementation are in the annex.

As it can be seen on the tables, the tests that depend on stemming have lower accuracy. The stemmer used was from nltk and was for the Portuguese language. In this case, the lower values of accuracy can be related to stemming in the sense that different important words can be stemmed to lemmas that are equal. This can generate more ambiguity. Removing stopwords, on the other hand, does not have a clear pattern in terms of its impact on accuracy. The algorithms based on cosine similarity tend to benefit from this pre-processing method. But others such as TF-IDF and Jaccard did not benefit from it. The algorithms based on Dice may either benefit or not

from stopword removal. In this case, the WAM D analyses the similarity of many more sentences than the one that only compares questions with other questions using Dice. That may be indicative that using this method is more beneficial if the metric is more dependent on the size of the corpora being analyzed. Removing punctuation seems to be always beneficial for the accuracy independently of the metric used.

The results refer to tests using 615 questions form the corpus (and deleted from training data before training). Based on our intrinsic evaluation, and our results, we decided to choose the method WAM D with the following pre-processing techniques: remove stop words, remove punctuation, lowercasing. This option was based on the fact that it had the best accuracy in the totality of the tests done. And the time needed to process the output was quite low for the accuracy given. On one hand, these results are representative of a relatively small corpus as a dataset, so, these values may be prone to have a completely different accuracy in different corpora. The size of the test set was quite big in comparison to the training set because an ample test set is more representative than a smaller one.

5 Conclusions and future work

In this project, we described the proposed solution for a set of solutions to solve the similarity evaluation, in the context of a retrieval-based conversational agent. We also proposed the use of standard metrics for this kind of problems such as Jaccard, Dice and Cosine similarity. In addition to that, we also proposed a state of the art metric based on the work of Magarreiro et. al. We also implemented all the metrics described bellow and ran several experiments with different pre-processing configurations. These experiments proved empirically that the measure proposed was better as it is more comprehensive.
According to our intrinsic evaluation, we can conclude that the best approach to achieve our goal is to use the implementation WAM using Dice similarity. Using as pre-processing: elimination, eliminate punctuation, lowercasing and stop-words removal. Our work showed that even using simple metrics we can empirically get significant results in sentence similarity (in this particular context). The size of the corpus may be a possible problem for the results obtained as the implemented algorithms may behave quite differently in larger datasets. Even though a further study of the weights for the WAM was needed. In a future project, automatic learning techniques based on optimization could be a possible solution to this problem. We might also propose an extrinsic evaluation of the proposed solution, for more generalizable results. Other works on this context would include solving the maximisation of the accuracy using several methods for each type of comparison on the WAM, understanding the exact problems that lead to the uselessness of the stemming in the context of our implementation.

References

[1] L. Coheur. *Natural Language Processing a Nut(s)shell.* 2019.

[2] D. Jurafsky and J. H. Martin. *Speech and language processing: an introduction to natural language processing, computational linguistics, and speech recognition.* Prentice Hall, Pearson Education International, 2009.

[3] D. Magarreiro, L. Coheur, and F. S. Melo. Using subtitles to deal with out-of-domain interactions. In *Proceedings of 18th Workshop on the Semantics and Pragmatics of Dialogue (SemDial)*, pages 98–106, 2014.

6 Annex

Table 1: Test results comparison

	Test 1	Test 2	Test 3	Test 4	Test 5
Jaccard	62.11% | 6.47	59.51% | 5.93 s	59.51% | 8.37 s	73.33% | 5.91 s	63.25% | 8.10 s
Dice	62.11% | 6.52 s	59.51% | 5.76 s	59.51% | 8.46 s	73.33% | 5.78 s	63.25% | 8.25 s
Cosine	82.43% | 266.38 s	86.17% | 251.49 s	85.69% | 244.95 s	85.04% | 227.82 s	78.69% | 224.31 s
TF-IDF	75.93% | 2089.75 s	75.60% | 1768.05 s	75.93% | 1936.77 s	75.93% | 1149.32 s	78.21% | 1300.43 s
WAM J	76.91% | 10.66 s	74.30% | 8.88 s	74.30% | 11.19 s	85.69% | 8.51 s	77.72% | 11.37 s
WAM D	77.88% | 10.54 s	80.97% | 8.90 s	81.13% | 11.22 s	87.80% | 8.46 s	82.76% | 10.63 s
WAM C	84.71& | 471.70	87.80% | 422.83 s	87.31% | 420.49 s	87.15% | 353.73 s	82.76% | 346.45 s