# Final Project report – Spanish chatbox

The idea for this project came from Prof. Malamud's ongoing research in linguistics, which I joined at the beginning of the semester. In it, she is investigating the compositional nature of utterance modifiers (such as tag questions) across clause types. Part of my role in the project has been to look at instances of such utterance modifiers in CORLEC (an oral corpus of contemporary Spanish), because tags are most commonly found in spoken language. Given that I was working with this oral material, I began to think about possible ways in which I could make use of these files for the present class. Originally, I thought about creating some sort of automatic text generator, but after meeting with Prof. Verhagen my idea shifted towards a Spanish chatbot.

Thus, I began my project by going over the different kinds of files that CORLEC contained. I found their classification system very useful, since each of their files is labeled as belonging to a specific category depending on its topic and the place where the audio was recorded. Eventually, I decided to select only the files whose utterances I thought would fit best the kind of output that could be expected from a chatbot. This included the conversation files grouped under the labels "conversation", "informal setting", "debate", and "interview". Among the ones left out, one could find news reports, oral speech from educational settings, or instructions over PA systems. Some reasons for me to exclude these other files are that sometimes their content was either too technical, or that the utterances were too long. This was particularly the case of the content recorded in an educational setting, where an "utterance" could be the entire speech of a presenter at a conference, or where the nature of the conversation could be inherently unbalanced (e.g. teacher-student exchanges where the former will typically utter longer and more frequent sentences).

Once I had selected the information that would constitute my training data, the first piece of code I wrote was that included in `preprocessing.py`. This file contains the functions I created to process and extract information from the corpus files. Using the function `combine_files`, I put together all the selected files into a mega-document. From it, I extracted only the utterances proper, which I tokenized and added to a list. Then, I used this list to create the vocabulary of the mega-document, which amounts to some 26,000 words (as will be discussed below, I believe that part of the limitations of this project lie on the limited number of words and utterances used for training). Finally, by means of `make_freq_dict`, I created a dictionary of the words in the vocabulary, which I mapped to lists of the sentences in which they appear. This constituted the main component of the training process and, given that this dictionary took some time to create, I decided to pickle it into a file for easier on-line access once the chatbot is running.

Next, I focused on creating the implementations of the selection algorithms that the chatbot uses when outputting a response to the user. I decided to implement three different algorithms, which are included in the file `selection_methods.py`. The first of them, `basic_selection`, is the simplest one: for each word in the user's input, this function collects all the training sentences where that word appears, and selects a random one as the reply. Should an input word not be found in the training data, a generic *no sé qué es <word>* (Spanish for "I don't know what <word> is") would be added to the collection so that the chatbot can return it.

The second, more elaborated selection method is based on the Jaccard coefficient. The idea behind this algorithm is that the sentence selected will be the one that maximizes the Jaccard coefficient between itself and the user's input. In order to accomplish that, the algorithm implemented in the `jaccard_selection` function builds up from the previous, random one. After collecting all the training sentences where each of the input words appear, a dictionary of Jaccard coefficients is created, each of which is mapped to a list containing the sentences that have that coefficient with respect to the input sentence. In a nutshell, the way I obtained this coefficient is as follows. For each sentence in the list mentioned above, I calculated both the intersection between that sentence and the user input (i.e. the number of words that appear in both sentences), and their union (i.e. the sum of all the words in each sentence – minus any potential duplicate). The Jaccard score is then the result of dividing the intersection by the union. The rationale behind this calculation is that the greater the overlap between the two sentences, the higher this score will be.

Third, I wrote the function `tf_idf_selection`, which is rather similar to `jaccard_selection`, but uses the TF-IDF measure instead. As with the previous two algorithms, first I collected all the sentences in which the words in the input appear. Then, for each of those words, I calculated its frequency in each sentence and multiplied it by the IDF of each word to obtain the TF-IDF measure for that word relative to each sentence. Since the IDF score remains constant for each word in the vocabulary given the training data, I created a helper function `idf_calculator` that builds a dictionary of words and their IDF score, which I later pickled for the `tf_idf_selection` algorithm to access when necessary. Finally, I decided to include two possible selection outputs for `tf_idf_selection`. The first one is the most intuitive one: simply return a random sentence from among the ones that got the highest TF-IDF score. Given that this will virtually always result in the same sentence being returned when two inputs are identical, I made a second return option available. In this second case, the algorithm returns a random sentence that contains *the word* from the input that got the highest TF-IDF score (as opposed to *the sentence* that maximizes such coefficient). A consequence of this is that it makes the TF-IDF score have less importance because a sentence might be selected where the word at stake does appear, but which otherwise would result in a rather low TF-IDF

score. By default, the standard first option is selected, but I decided to include this second one for experimentation.

The final elements of the chatbot are contained in `main.py`, which is the module to be executed in order to run the program. It contains the interface with the user and the calls to either of the three selection algorithms described in the previous paragraph. This can be commented/uncommented to experiment with the different outputs that can be obtained.

Upon interacting with the chatbot, it quickly becomes apparent that it is rather limited and that obtaining some kind of coherent and cohesive interaction becomes very difficult, even when the more elaborate algorithms (the ones in `jaccard_selection` and `tf_idf_selection`) are used. Actually, the most noticeable improvement came once I eliminated the Spanish *stopwords* from the user input, so that they would not be used in the computation of the Jaccard or TF-IDF scores.

As possible avenues for improvement, I believe that enlarging the training data would go a long way in increasing the capabilities of the chatbot because it would give it a much wider range of possible outputs. Another way to increase the impression that the user is interacting with an *intelligent* agent could be to incorporate a number of default or pre-set responses that request the user to expand on their previous input (similar to the way that the chatbot ELIZA does). This would shift the conversational load to the user in that the chatbot would behave more like a wall off of which the user's utterances bounce. This does become tiresome after a while, but it does initially give the impression that the chatbot is coherent.

Finally, it is my intention to continue developing this project after I learn more about machine learning techniques in my future classes, so I hope to be able to take it from its current crude form to a more polished one.