COMP 3074 Human-AI Interaction

Coursework 1: An interactive NLP-based AI system

Student name: Grigoriy Kirpa

Student ID: 20113717

Introduction

This report concludes information about design, implementation and testing of one of the Natural Language Processing (NLP) based systems – chat user interface (CUI). According to Liddy (2001), Natural Language Processing refers to a set of theories and techniques focused on analyzing, representing, and generating human-like natural language. NLP became a rapidly growing field in the past decades, due to the vast range of possible applications, where replacing or at least pairing a human with an NLP based system brings lots of benefit from economical and user experience perspective (the last one comes from reduced waiting time and 24/7 availability).

One of the simpler yet most popular NLP applications is CUI, also known as chatbot. Chatbots became an essential part of every private and government organizations, small or large. However, despite lots of research and investment, there are still quite many issues, e. g. turn based conversations, stigma of some users towards bots and most importantly biases and racism, that were inherited from training data or/and their developers.

Background

First step of implementing NLP based system is related to modeling and representation of the language. There are many existing ways to do it, and each has its own pros and cons. For this project the Bag of Word (BoW) model was chosen for data representation. In the BoW, each document is treated as a collection of unordered n-grams (tokens) and represented by a fixed length sparce vector of frequencies of each token (Sethy & Ramabhadran, 2008). The BoW model is very simple yet proved to be quite effective in NLP. The main issues are large sparce vectors which slow down the processing and ignorance of order, however for chatbot implementation, the BoW is sufficient and the simplest, hence the fastest to implement.

Each NLP system, regardless of the components, utilizes same techniques. In particular, text classification and information retrieval. Additionally, some preprocessing is required for every task.

According to Kannan et al. (2014), preprocessing for NLP consists of tokenization, stop word removal and stemming/lemmatization. Tokenization is a process of converting a document into a sequence of n-grams (tokens). Stop word removal is a process of removing some so-called *useless* words, in order to reduce size of

vocabulary, which speeds up the performance. Stemming/lemmatization is a process of reducing words to their basic form, which also meant for reducing vocabulary size.

The figure 1 describes the process of text classification using machine learning algorithms (regards to Ikonomakis et al. for the figure).

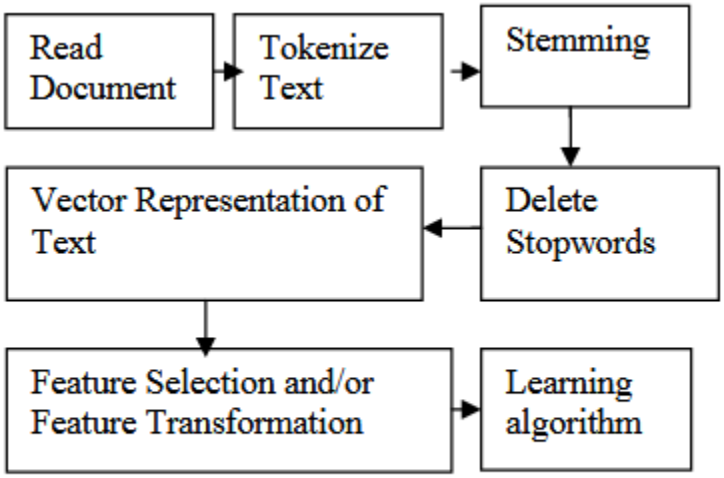

Fig. 1. Text Classification Process

There are other text classification techniques, however for this project, machine learning algorithm approach was chosen to be the most suitable, due to high speed and yet quite high accuracy.

Information retrieval is described in Figure 2. Preprocessing mentioned in that figure follows same steps as for text classification (refer to Figure 1). For computing relevance, cosine similarity will be used. According to Singhal (2001), it depends only on the angle between two vectors, requiring vector representation to be used by definition. As mentioned earlier, BoW model is used for this project, hence the need is fulfilled.

Proposed system.

The CUI implemented for this project, has 3 main functions. Firstly, chatbot is able to handle small talk, e. g. “Hi”, “How are you?” etc. It supports 9 different categories of small talk and has a set of answers for each category, to ensure variety in answers given. Secondly, chatbot is able to answer some questions and requests. The knowledge base at the moment is insufficient, resulting in limitation to the questions that can be asked. Regarding the requests, chatbot is able to

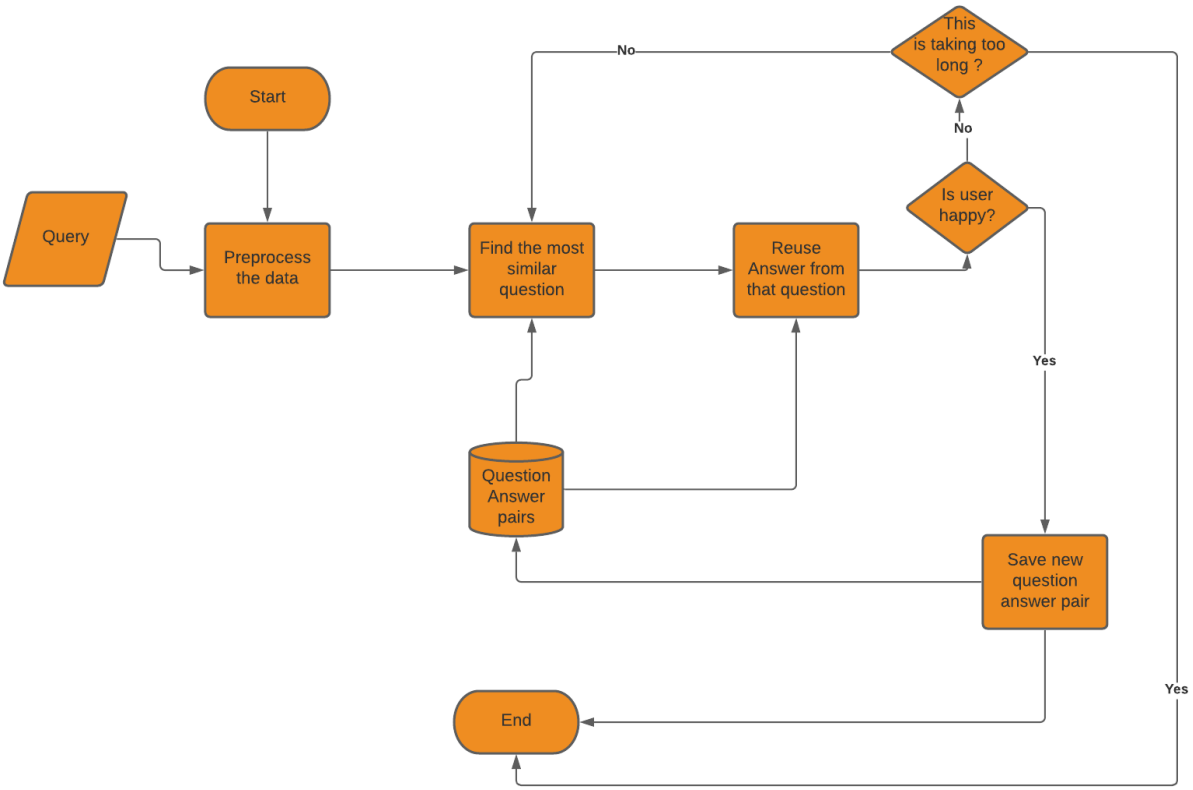

Figure 2. *Information retrieval Flowchart.* This Figure briefly describes information retrieval technique.

answer properly to requests of time, username and chatbots name. Last but not least, CUI has inbuilt support for identity management. It is able to request user for the name when *it is appropriate* and use it during small talk and question answering. Additionally, a multilevel intent matching was implemented to route between and within the functions. Thus, allowing a more customizable answers to different small talk queries and extensible question answering system, where more requests can be added later during system upgrades.

CUI implemented, when compared to other related projects, will be strongly standing out due to several reasons. The first one is, the ability to request for a username when appropriate. For example, during greeting, but not during courtesy greeting, e. g. “Hi, how are you?”. The reasoning behind this specific case is that during courtesy greeting there is already a question pending. Asking too many questions in one turn is not considered a good behavior. However, after receiving an answer to a courtesy greeting from a user, something like, “I’m great, thanks” etc., bot will use its turn to ask for a name. Similar reasoning applies to asking for

a name during question answering (in case this feature is used first after startup). Secondly, the ability to customize and create more complex answers based on category of the query within the function used, which is achieved with the help of multi-level intent matching. Partially, the previous feature of asking name when appropriate was implemented with the help of this achievement. Thirdly, the name and time queries are handled by question answerer, instead of small talk handler, despite the common practices. This is done to ensure greater accuracy with intent matcher that is based on machine learning approach. In particular, queries of type "What …", "Could you tell me…" and "Do you know…" are naturally considered questions or/and requests, and due to the fact that name and time queries often start with the same pattern, it results in higher accuracy and f1-score when training classifier with those queries in the question-and-answer knowledge base. The last and the most questionable difference, is the fact that question answerer module doesn't learn on new associations and in general doesn't ask the user for their feedback if the answer to the query was helpful. This is done, to a) ensure better user experience (constant requests of feedback are annoying) and b) avoid learning misleading, bias, and possible racist associations. More on the last one in discussion part.

When building a system, modularized approach was used. Every function is a separate module, that loads their data and trains separately. This approach was chosen, to enable singular module retraining and hence make implementation and deploy processes more convenient. All of the modules are controlled by Main file. At first Main initializes instances of intent matcher, question answerer and small talk handler. Then Main loop starts (refer to Figure 3). When training intent matcher, no stop word removal was performed, due to the fact that small talk is mostly consists of stop words. Only a selection of questions was used for training of the classifier, to ensure that distribution of questions and small talk examples are roughly same. Small Talk handler contains queries and responses for multiple categories in a dictionary, however when building a vocabulary and computing BoW model all of the queries are combined. To handle a new query, the process described by Figure 4 is used.

For small talk handler preprocessing excludes stop word removal as well.

Question answerer handles queries same way as Small talk handler, except for removing stop words during preprocessing. However, words "me, my, you, your"

are whitelisted, in order to distinguish when chatbots and when users name is requested.

Identity manager is a singleton class, whose instance is shared between all other modules. It is able to extract name from a query, given that it is a query of user providing its name. The algorithm for this process is simple, remove all the words that are used in name intros from the query. If anything is left, consider it a name. Identity manager also generates so called name outros (refer to Figure 4). These are: a) some outro with the username in it or b) if name is not defined, the outro is query requesting a name. If b) is executed, then during next loop iteration, main will handle the control to Identity manager for name extraction (refer to Figure 3).

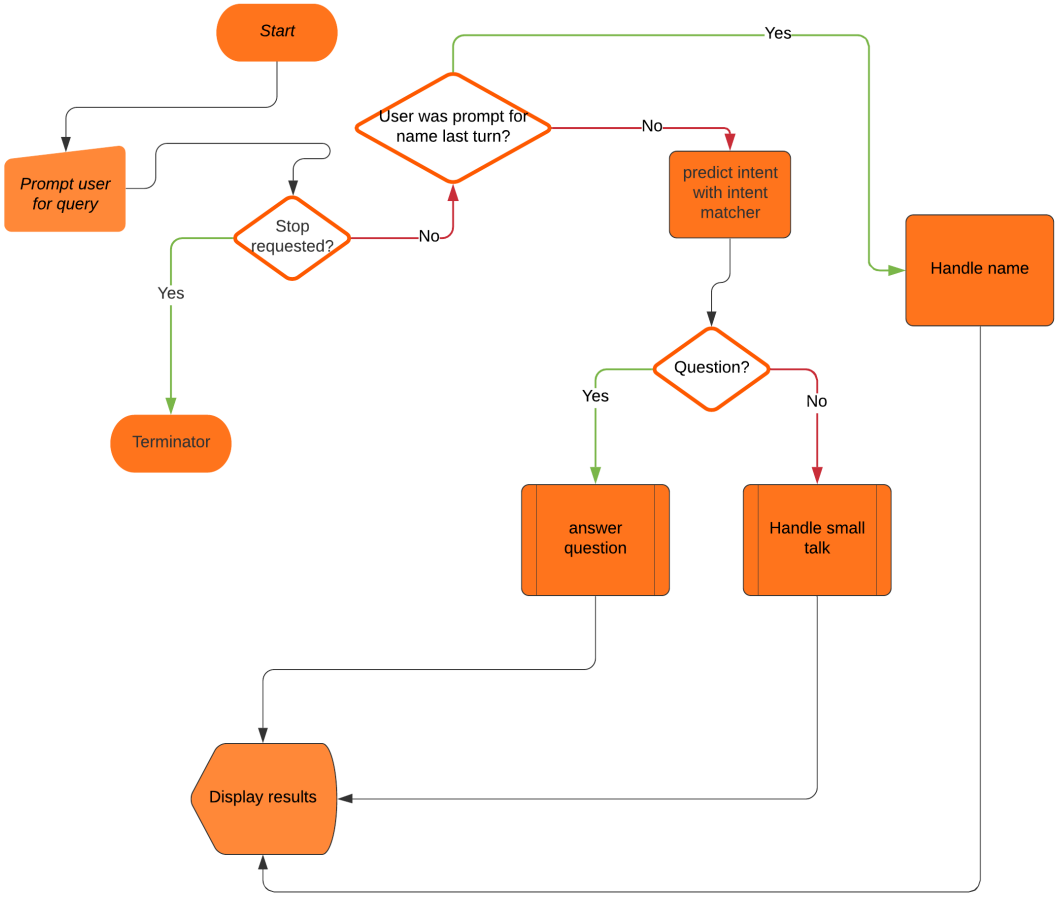

Figure 3. *Main loop*. This figure briefly describes the main loop of the system.

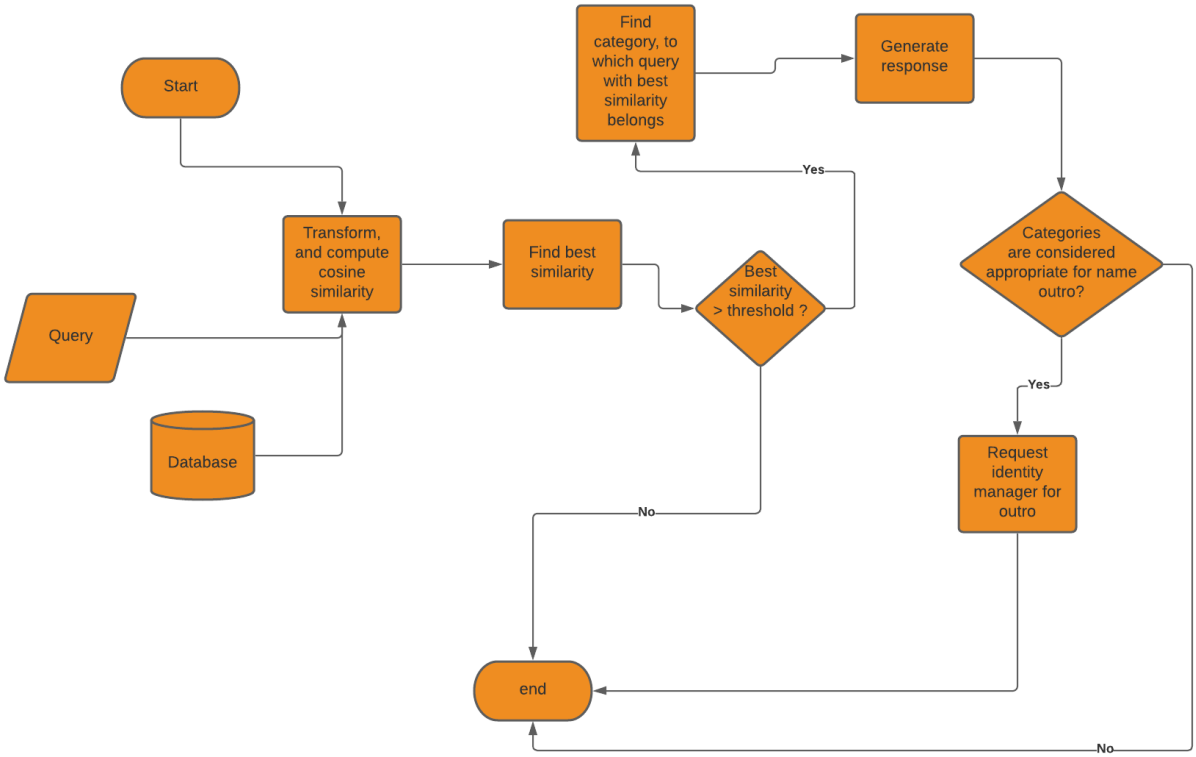

Figure 4. *Small talk handling*. This figure describes the process of handling small talk query.

Evaluation.

Refer to Table 1 and 2 for test cases and test results. Overall, most of the test cases were passed.

Discussion.

Though most of the tests were passed, those that didn't highlights some technical issues of the chatbot. Tests 4 and 6 were checking the robustness of the system to grammatical errors. The intended behavior of the CUI was that it shouldn't try *too hard* to answer the given query. Sometimes it is better to just return an error, instead of a nonsense answer. For that reason, a threshold of similarity was introduced to ignore barely relative queries for both small talk handler and question answerer (refer to Figure 3). Yet, the system still doesn't respond how intended. This is due to the issue with sparse vectors. Most of the queries have no words in common and each of them introduce one or more new tokens to a vocabulary, hence their frequency vectors will look like long vector of 0's with one

or more 1’s somewhere along the way. And if the new query consists only of words that have been unseen before, then it’s vector will be full of 0’s which will make it very similar to most of the queries in the data set, especially those that introduced only one new word to the vocabulary, while all other words are stop words. This conclusion was confirmed by trying inputs that consisted only of one character, which returned exactly same answer as result of Test 4, option 5 and 6. One of the ways to fix this issue is the original usage of question answer system, which involves feedback and learning from success. That might potentially result in finding correct answer for the fully unknow query, which then will lead to addition of its words to the vocabulary. However, learning from users might introduce another problem. Not only multiple attempts followed by asking for feedback all the time is leading to poor user experience but might also result in user giving positive feedback for a wrong pair, just to *get it over with*, or what is even worse, learning from user might result in racial or sexist biases. In this project, it was attempted to omit learning from the user, which enables to have more control over what data is getting stored to the database, and hence helps again biases. However, this approach will result in either the chatbot getting outdated significantly faster or consume much more money for constant data labeling requests and since CUIs are used mainly for economic reasons, both of those outcomes are undesirable.

Conclusion.

Chatbots seem to be quite simple in terms of implementation, however lot’s of side details must be considered, during designing and implementation.

Table 1.
Test cases.

ID	Description	Steps
1	Check if the chatbot can run	1. Run the main.py file
2	Check if the chatbot can be stopped when necessary	1. Run the chatbot 2. Wait for instructions of how to stop the chatbot appear 3. Type stop word from instructions
3	Check response when valid small talk query entered	1. Run the chatbot 2. Type 2 query for each category of small talk (18 queries in total). P.S. provide the name when prompted. 3. Terminate the bot
4	Check response when small talk query with grammatical errors entered	1. Run the chatbot 2. Type 1 query for each category of small talk (9 queries in total). P.S. provide the name when prompted. 3. Terminate the bot
5	Check response when valid question is entered	1. Run the chatbot 2. Type 2 query for each category of questions, excluding username query (6 queries in total). P.S. provide the name when prompted. 3. Terminate the bot
6	Check response when question with grammatical errors entered	1. Run the chatbot 2. Type 2 query for each category of questions, excluding username query (6 queries in total). P.S. provide the name when prompted. 3. Terminate the bot
7	Check if name is recognized correctly	1. Run the chatbot 2. Type “Hi” 3. Introduce your name. 4. Terminate the bot and go to step 1. Repeat 5 times.
8	Checking the case: Request for a name, immediately after starting the bot	1. Run the chatbot 2. Type “What is my name?” 3. Terminate the bot
9	Checking the case: Request for a name, after you introduced yourself	1. Run the chatbot 2. Type “Hi” 3. Introduce your name 4. Type “What is my name?” 5. Terminate the bot
10	Check what happens when typing nonsense	1. Run the chatbot 2. Type random combination of words 3. Terminate the bot

Table 2.

Test results.

ID	Data	Expected result	Actual result	Pass/ Fail	Notes
1	-	Program must run without any errors	Program runs without any errors	Pass	
2	-	Instructions must be provided on how to stop the bot. Execution of instructions must result in termination	After the program started, a stop word was provided. After typing the stop word, the program terminated	Pass	
3	"Hello" "Howdy" "How are you" "How is life" "I'm good" "So far so good" "Hi how are you" "Hey yo, how is life" "That's enough, thank you", "Thx" "Shut up" "Stop talking" "Bye" "Adios" "Thanks bye" "Cheer's bye" "You are very clever boy" "You are clever girl"	Answers from respective categories.	Answers were looking natural to the query typed.	Pass	
4	"Helo" "Hwo are yoo" "Im good" "Hai how aer you" "Enagh, thenk you" "Shoot up" "Godbye" "Thanks byebye" "You is clever"	Bot should return either proper answer or an error.	Number 1,2,3,5,9 returned proper answers. Number 4 and 8 returned answers from small talk but wrong category. Number 6 and 7 were misidentified as questions and the response was "A will or testament is…"	5/9 Fail	The result that is returned for 6 and 7 is the same for just empty string input

5	“Let me know your name” “Who am I talking to” “What is the time?” “Could you tell me what time it is” “how long was richard nixon a president” “How long was Mickie James with WWE?”	For option 1 and 2, the bot should state their name. For option 3 and 4, the bot should state the time. For option 5 and 6, the bot should answer appropriate answer from the database	For option 1 and 2, the bot answered that their name is Elfie. For option 3 and 4, the bot stated the time. For option 5 and 6, the bot answered something that looked like truth.	Pass	
6	“Leet me konw you name” “Let me know yur naem” “Wht iss teh time?” “What is the tmie?” “hwo long wos richard nixon a presdent” “how long was richrd nxon a presdent”	Bot should return either proper answer or an error.	For option 1, the bot stated my name instead of theirs. For option 2 and 3, bot provided me with the time. For option 4, bot gave an error. For option 5, bot gave a correct answer. Option 6. Bot gave an error	4/6 Pass	
7	“My name is -” “Call me -” “My first name is -” “I’m -” “I’m”	For options 1-4, the name must be properly recognized. For option 5 chatbot must inform about failure and ask again	Options 1-4 the name was recognized. Option 5, Bot gave an error and prompted again	Pass	
8	“What is my name”	Bot will inform the user that it doesn’t know it yet and then, prompt the user for the name	Bot informs that the username is not yet known, and then typed what is your name twice.	Fail	Fixed.
9	“Hi”, then introduce the name then “What is my name”	Bot should state the name of the user	Bot answered “You are <my name here>”	Pass	

10	“Why do me bird yet” “how can park fly to the beach”	Bot should inform the user about the error.	For option one the bot returned some answer involving yellow bird. For option 2 there was an error	1/2 Fail	

References

Liddy, E. D. (2001). Natural language processing.

Sethy, A., & Ramabhadran, B. (2008). Bag-of-word normalized n-gram models. *In Ninth Annual Conference of the International Speech Communication Association.*

Kannan, S., Gurusamy, V., Vijayarani, S., Ilamathi, J., Nithya, M., Kannan, S., & Gurusamy, V. (2014). Preprocessing techniques for text mining. *International Journal of Computer Science & Communication Networks*, 5(1), 7-16.

Ikonomakis, M., Kotsiantis, S., & Tampakas, V. (2005). Text classification using machine learning techniques. *WSEAS transactions on computers*, 4(8), 966-974.

Singhal, A. (2001). Modern Information Retrieval: A Brief Overview. *Bulletin of the IEEE Computer Society Technical Committee on Data Engineering*, 24 (4), 35–43.