Utterances classifier for chatbots’ intents

AXEL JOIGNEAU
Utterances classifier for chatbots’ intents

AXEL JOIGNEAU

Master in Systems, Control & Robotics
Date: July 6, 2018
Supervisor: Hedvig Kjellström
Examiner: Olov Engwall
School of Electrical Engineering and Computer Science
Abstract

Chatbots are the next big improvement in the era of conversational services. A chatbot is a virtual person who can carry out a conversation with a human about a certain subject, using interactive textual skills. Currently, there are many cloud-based chatbots services that are being developed and improved such as IBM Watson, well known for winning the quiz show “Jeopardy!” in 2011.

Chatbots are based on a large amount of structured data. They contains many examples of questions that are associated to a specific intent which represents what the user wants to say. Those associations are currently being done by hand, and this project focuses on improving this data structuring using both supervised and unsupervised algorithms.

A supervised reclassification using an improved Barycenter method reached 85% in precision and 75% in recall for a data set containing 2005 questions. Questions that did not match any intent were then clustered in an unsupervised way using a K-means algorithm that reached a purity of 0.5 for the optimal K chosen.
Sammanfattning


Chatbots baseras på en stor mängd strukturerade data. De innehåller många exempel på frågor som är kopplade till en specifik avsikt som representerar vad användaren vill säga. Dessa föreningar görs för närvarande för hand, och detta projekt fokuserar på att förbättra denna datastrukturing med hjälp av både övervakade och oövervakade algoritmer.

En övervakad omklassificering med hjälp av en förbättrad Barycentermetod uppnådde 85 % i precision och 75 % i recall för en dataset innehållande 2005 frågorna. Frågorna som inte matchade någon avsikt blev sedan grupperade på ett oövervakad sätt med en K-medelalgoritm som nådde en renhet på 0,5 för den optimala K som valts.
# Contents

1 Introduction .................................................. 1
   1.1 Chatbots ....................................................... 1
      1.1.1 History of chatbots ................................. 3
      1.1.2 Intents and utterances ............................ 4
   1.2 Research Focus ............................................ 5
   1.3 IBM ............................................................ 5
   1.4 Sustainability and ethics ............................... 6
      1.4.1 Social impacts ......................................... 6
      1.4.2 Economic aspects ..................................... 6
      1.4.3 Biased responses ...................................... 7
      1.4.4 Privacy, identity, and other ethical concerns .... 8
   1.5 Outline ....................................................... 8

2 Background .................................................... 10
   2.1 The IBM chatbot ........................................... 10
      2.1.1 Intent detection and classification ............... 12
   2.2 Word embeddings .......................................... 13
      2.2.1 The TF-IDF algorithm ................................. 14
      2.2.2 Cosine similarity ....................................... 15
      2.2.3 Considering synonyms, stop-words and stemming 16

3 Related Work .................................................. 17
   3.1 Supervised methods ....................................... 17
      3.1.1 Traditional methods .................................. 17
      3.1.2 Neural Network approaches ........................ 19
   3.2 Unsupervised methods .................................... 20

4 Classification methods and data set .......................... 22
   4.1 Classification methods ................................... 22
      4.1.1 Supervised ............................................. 22
CONTENTS

4.1.2 Unsupervised ........................................... 24
4.2 Data set ................................................. 26

5 Reclassification ........................................... 29
  5.1 Preprocessing ........................................... 29
  5.2 Barycenter method ...................................... 31
    5.2.1 First attempt ...................................... 31
    5.2.2 Second attempt .................................... 32
  5.3 Neural network ........................................ 34
    5.3.1 First attempt ...................................... 34
    5.3.2 Second attempt .................................... 35
  5.4 Discussion ............................................ 37

6 Creation of new clusters .................................. 39
  6.1 Synonyms and stop words ................................ 39
  6.2 Method and results ..................................... 40
  6.3 Label suggestion ........................................ 40

7 Conclusion .................................................. 43
  7.1 Future Work ............................................ 44

Bibliography .................................................. 46
Chapter 1

Introduction

1.1 Chatbots

Modern search engines have become very good at understanding typical and most popular user intents, recognizing topic of a question and providing relevant links. However, search engines are not necessarily capable of providing an answer that would match personal circumstances, knowing a specific state and an attitude of a user who formulated a query. This is particularly true for long, complex queries, and for a dialogue-based type of interactions.

Figure 1.2 shows the results obtained by Boris Galitsky and Dmitry Ilvovsky [13], who decided to build a chatbot and to compare its performances with search engines. Their chatbot’s time of knowledge exploration session is longer than for search engines. Although it might seem to be less beneficial for users, businesses prefer users to stay longer on their websites, as the chance of user acquisition grows. Spending 7% more time on reading the chatbot’s answers is expected to allow a user to better familiarize himself with a domain. But the most interesting is that the number of steps of an exploration session for the chatbot is 25% lower than for Google search. Chatbots represent much more application domains, for both entertainment purposes or in companies. In a help desk for example, their common application domain is to be complementary to a human assistance by answering the basic questions and saving human time for more complicated problems. Many chatbots are popular nowadays, such as IBM Watson. This kind a chatbot results from a long improvement.
Figure 1.1: Comparing conventional search engine with chatbots in terms of a number of iterations [13]

<table>
<thead>
<tr>
<th>Parameter / search engine</th>
<th>Web search</th>
<th>Chat bot</th>
</tr>
</thead>
<tbody>
<tr>
<td>Av. time to satisfactory search result, sec</td>
<td>45.3</td>
<td>58.1</td>
</tr>
<tr>
<td>Av. time of unsatisfactory search session (giving up and starting a new search), sec</td>
<td>65.2</td>
<td>60.5</td>
</tr>
<tr>
<td>Av. # of iter. to satisfactory search result</td>
<td>5.2</td>
<td>4.4</td>
</tr>
<tr>
<td>Av. # of iter. to unsatisfactory search result</td>
<td>7.2</td>
<td>5.6</td>
</tr>
</tbody>
</table>

Figure 1.2: Comparison of the time spent and a number of iterations for the chatbot [13] and Google search in the domain of personal finance
1.1.1 History of chatbots

In 1950, Alan Turing’s famous article "Computing Machinery and Intelligence" was published, which proposed what is now called the Turing test as a criterion of intelligence. The work of Alan Turing was taken with great interest by Joseph Weizenbaum, a professor at Massachusetts Institute of Technology. In 1966, he developed the program ELIZA, which aimed at tricking its users by making them believe that they were having a conversation with a real human being [33].

In 1995, inspired by ELIZA, another chatbot program named A.L.I.C.E. (Artificial Linguistic Internet Computer Entity) was created. It applies heuristical pattern matching rules to the human’s input. It is one of the strongest programs of its type and has won three times the Loebner Prize Competition in Artificial Intelligence, which rewards the most human-like bot. ALICE knowledge is stored in AIML files [34], where AIML is an abbreviation of The Artificial Intelligent Mark up Language.

In 2011, Cleverbot fooled 59 percent of its human interlocutors who thought they were talking to a human [40]. This chat software, created in 1988 by Rollo Carpenter, is based on a crowd-sourcing method, unlike ELIZA. Since its apparition online in 1997, Cleverbot has had millions of conversations with internet users around the world, who chat with it for fun via the Cleverbot website. Like a human learning appropriate behavior by studying the others, Cleverbot "learns" from these conversations. It stores them all in a huge database, and in every future conversation, it will mimic the human responses from the past conversations. The key to Cleverbot’s success is that it does not only take into account the last thing typed, but it keeps track of words and phrases that appeared in the conversation already. As Carpenter explained, "It looks back to the whole conversation and says, there are maybe tens of thousands of people who have maybe responded to ‘how are you?’ in the past; which of those tens of thousands of responses is most suitable to use this time?"

The last winner of the Loebner Prize is the chatbot Mitsuku, which was also created from AIML technology [25]. It is currently the most human-like chatbot available online. This kind of bot is made to chat about anything, which makes it more human than other bots made for specific cases, such as LUIS from Microsoft or Watson from IBM. Those two specific chatbots are based on neural networks. Both un-
The Natural Language Process of Watson framework analyzes text to extract meta-data from content such as concepts, entities, keywords, categories, relations and semantic roles, and it understands 9 languages. For its part, Microsoft Bot framework understands users’ intents dynamically, can use pre-built models and supports automatic translation of more than 30 languages [37].

There are many other chatbots on the market based on neural networks, but they cannot always be compared to each other. On the one hand, smart assistants such as Amazon Alexa or Google Assistant have conquered the smartphones, tablets, and corners of the smart home, but not so the professional world [28]. On the other hand, Watson assistant was designed expressly for business cases, can easily be integrated to the workflow of other applications, and each company can have its own instance of Watson to keep the intellectual property. The other main difference is the size of data sets: Alexa needs to learn from millions of consumer interactions, whereas Watson learns from less, and very specific business-centric data [29]. In the framework of this project, we will only work with a business-oriented chatbot which is based on neural networks.

### 1.1.2 Intents and utterances

Natural language processing (NLP), that includes natural language understanding and natural language generation, represents a real challenge in the chatbots area. For the moment, most chatbots search for keywords in the input and then reply with the most matching answer from a database, but some of them use sophisticated natural language processing systems and are focused by companies like Facebook with Deep Text [1], Google with Syntax Net [30], Microsoft with [14] or IBM with Watson [16].

An important thing is to understand a chatbot as a database-dependent system. It stores in its database many possible utterances, which are the questions asked by the user, and clusters them in different intents, which represent what the user wanted to say. The goal is to be able, for a new question, to associate it with the right intent, in order to answer correctly or at least to return what the user expected. For example, for the question “what’s the weather?”, a correct answer would be “The weather is the state of the atmosphere at a particular place and
time", but the answer expected by the user would be something like "It is raining in New York today". Some queries are harder than others, especially if the database was not trained with them. A chatbot that gets the same question “what’s the weather?” will understand what to answer but the question “Could you check the weather?” might not give the proper answer, depending on the utterances used to train the database. More details about the specific chatbot used in this project are provided in Chapter 2.

1.2 Research Focus

Since the accuracy of a chatbot is highly dependent on the database of examples on which it is trained, and the classification of the different utterances, the problem is: what if there are utterances from the database that are misclassified?

The focus of my project is finding solutions to automate a reclassification of the misclassified utterances. This task is currently done by hand, which can be time consuming if you consider more than thousands of questions. The goal is here to propose new intents that are judged as more relevant for the questions detected as misclassified. The evaluation will mainly be done on the accuracy of the misclassification detection, but also on the relevance of the new intents proposed. In the case of new intents creation, the quality of the clustering will be assessed.

1.3 IBM

This thesis work was performed for IBM Watson from September 2017 to January 2018. The result is intended to be sold to IBM’s business clients as an additional feature for their chatbot’s continuous improvement. The data set used for tests are confidential but generic examples are given instead.
1.4 Sustainability and ethics

1.4.1 Social impacts

A common fear that appeared with the first robots is about machines taking jobs away from humans. Today, with the emergence of new technologies like artificial intelligence (AI) and chatbots, that fear seems to have increased. Many sectors such as finance, health, retail, and law are adopting AI and chatbots into their everyday functions. A study by Forrester [12] shows that 25 percent of today’s jobs will be impacted by AI technologies (which includes intelligent chatbots) by 2019. But the point is that chatbots are not here to replace humans, but rather to assist them. As an example, a chatbot can help a client center avoid being overwhelmed by too many questions, and humans will be able to focus on requests that require more attention. Chatbots are very effective and can treat thousands of demands at once, so they are here to be complementary with humans’ creativity and adaptation.

Some jobs will disappear with chatbots, but it will create others to supervise, maintain, and work with chatbot tools. Gartner [11] estimates that by 2020, customers will manage 85 percent of their relationships with enterprises without interacting with a human. Companies need to evolve and adapt to this new era where humans and technology work together.

1.4.2 Economic aspects

According to a study by BI Intelligence [15], 80% of businesses want chatbots by 2020. Additionally, the survey shows that business leaders and decision makers are turning to the broader umbrella of automation technologies, which includes chatbots, for things like sales, marketing, and customer service. Forty-two percent of participants believe automation technologies in these areas will most improve the customer experience. And 48% said that they already use automation technology for these business functions, with 40% planning to implement some form of automated technology by 2020.
1.4.3 Biased responses

It is important to consider the effects of delivering information to the public via a bot, which necessarily has a limited range of responses. Those responses are also pre-programmed by individuals who have biases and tendencies of their own, which could lead to additional concerns about impartiality, fairness, and manipulation if the call and response databases are not closely monitored.

Amir Shevat [35] reminds us that it’s important to ask the question, “Does this bot serve me, or the service provider?” For example, Shevat continues, “will the food-ordering bot recommend the pricey/low-quality items or the best-priced and quality food?” Does the limited nature of the bot’s responses lead to a reduction in the nuance and sensitivity contained in each response? It’s also important to consider where the bot sources it’s information from and how it makes sure that those sources are themselves free of their own undue bias or corruption.

These questions, and many more, are perfect examples of why it is important to maintain diverse human oversight and supervision of bots and their library of inputs and outputs.

Example of Tay Twitter Bot

Even if we did not work on self-taught chatbots, it is interesting to understand the more global context with this other kind of chatbots that can easily raise issues since they can make their own decisions with uncontrolled consequences.

A good example is the Microsoft’s NLP Twitter chatbot Tay, which was designed to learn from its conversations with the users. In less than 24 hours after its deployment, it learned from users to tweet in ways that were anti-semitic and racist [22]. This unethical behaviour was actually unpredictable because what it learned only depended on interactions with an unknown environment.

This example illustrates the more general issue of wondering who is responsible if a robot makes a mistake and what a self-learning robot is doing to improve its judgments from experience.
1.4.4 Privacy, identity, and other ethical concerns

Amir Shevat [35] also tackles the issue of privacy. One can wonder if a bot can share information or not with other bots or human overseers, and if information should be anonymized. It is important to maintain the security of the bot’s input and output databases in order to avoid the loss of sensitive corporate information or private user information. Users need to know that the questions they ask and the interactions they have with your bots will remain private and secure. Chatbot responses, and all other communications, should also include some level of empathy and sensitivity when it comes to interacting with users. Amir Shevat [35] even questions whether or not humans should be allowed to abuse bots, as well as whether or not bots should be able to abuse humans.

Gender and identity are two additional and important concerns for chatbot owners and operators. Should a chatbot be male, female, gender neutral, or perhaps entirely genderless? Should the bot have an identifiable race, ethnicity, or nationality? Is it possible to create a bot that is devoid of national, ethnic, or racial identity without inevitably reinforcing the dominant narratives about race and ethnicity that are already at play in the country or area where your users live? These are important questions for companies need to answer before incorporating chatbots into their day-to-day operations and user interactions.

1.5 Outline

We will start by explaining the different concepts and backgrounds required for the project in Chapter 2. Chapter 3 will present a study of relevant literature around state of the art text classification methods. We will describe the different methods selected for the project and give specifications about the data set used in the project in Chapter 4. We present then the two main parts, containing the methods and the results obtained:

- Chapter 5: Reclassification. We aim at detecting misclassified questions, to reclassify them in the right intent, and isolated questions that do not match with any intent. This part uses supervised methods.
• Chapter 6: Creation of new clusters. The goal is here to regroup unclassified questions into clusters and create new intents with label suggestions for each. This part is unsupervised.

A global conclusion is presented in Chapter 7, as well as limitations about the methods used in this project, and suggestions of improvements and future work.
Chapter 2

Background

In this chapter, after a short presentation of the IBM chatbot, we will explain the different methods and algorithms that will be used for text classification in this project. We will focus on the way we can transform words and sentences to manipulate them, and we will finally deal with the classification methods, both supervised and unsupervised.

2.1 The IBM chatbot

The main thing to understand with chatbots is the intent detection. It is here to categorize the user’s utterance into predefined intents. The intent reflects what the user is trying to say or achieve [10], and hence prescribes an action that defines the desired outcome. In order to make the chatbot understand the meaning of an utterance, it needs to be trained before on this specific intent, by learning several utterances called "variations". Entities are then used to precise the intent, and they are usually single words. As an example, if we consider the utterance “I would like you indicate me the nearest restaurant”, the detected intent could be “find a location”, and the entity “restaurant” would precise this intent. The more precise the classification is, the more relevant will be the answer. The chatbot then acts like a states machine and each utterance you say will bring you in a different dialogue configuration, depending on the intent detected. The point is: how is this intent detected?

This figure 2.1 presents the global anatomy of the IBM chatbot, from the user’s text input to the chatbot’s output.
Figure 2.1: IBM chatbot architecture [31]
2.1.1 Intent detection and classification

In this dialogue (Figure 1), we can see that the chatbot aims at detecting the user’s intent, and since the question does not correspond exactly to an existing intent, the chatbot tries to get more information by asking the user to choose between several propositions.

In fact, the intent detection is an utterance classification task that can be formulated as:

\[ y' = \arg \max_y p(y | w_1, w_2, ..., w_n) \]  \hspace{1cm} (2.1)

where \( w_i \) is the i-th word of a sentence and \( y \) is the intent. [19]

An approach can be to consider that intents are first defined by the utterances they contain. The classification problem can now be seen as a similarity problem between the utterance we want to classify and the utterances of the different intents. The issue consists now in determining a similarity between two sentences. One method to compare semantically two sentences is to transform them into vectors, hence the use of word embeddings.
2.2 Word embeddings

Word Embeddings are the texts converted into numbers and there may be different numerical representations of the same text. A vector representation of a word may be a one-hot encoded vector where 1 stands for the position where the word exists and 0 everywhere else. If we consider the sentence “I want to buy a new phone”, the vector representation of “want” in this format is [0,1,0,0,0,0,0] and for “phone” [0,0,0,0,0,0,1]. There are different kinds of Word embeddings, but we will focus on the frequency-based embeddings, and especially on the Count Vector and the TF-IDF Vector. The Count Vector is the basic way to convert a document of words into numbers. It only consists in counting the number of times a word occurs in a document and then build a giant vector containing all the numbers of apparition of each word, like in the figure 2.3

Another possibility more often used is to divide by the number of words in the document in order to obtain a vector of words frequencies.
2.2.1 The TF-IDF algorithm

Term frequency

Suppose we have a set of English text documents and wish to determine which document is most relevant to the query "the yellow car". A simple way to start out is by eliminating documents that do not contain all three words "the", "yellow", and "car", but there will be still many documents left. To discriminate them further, we might count the number of times each term occurs in each document: the number of times a term occurs in a document is called term frequency.

However, in the case where the length of documents varies greatly, adjustments are often made. The first form of term weighting is due to Hans Peter Luhn (1957) and is based on the Luhn Assumption [27]: "the more often a term occurs in the text of the document, the higher its weight".

In the case of the term frequency $tf(t,d)$, the simplest choice is to use the raw count of a term in a document, i.e. the number of times that term $t$ occurs in document $d$. If we denote the raw count by $f(t,d)$, then the simplest $tf$ scheme is $tf(t,d) = f(t,d)$.

Inverse document frequency

The first problem we meet is that the term "the" is so common that its frequency will be very high even in documents that don’t correspond to what we are looking for, unlike the words “yellow” and “car” which are less common and then more meaningful. Then, since words like “the” are not as relevant as less common words, an inverse document frequency factor was incorporated to reduce the weight of terms that occur very frequently in all documents and increases the weight of terms that occur rarely.

Karen Spärck Jones [17] conceived a statistical interpretation of term specificity called Inverse Document Frequency (IDF), which became a cornerstone of term weighting: “The specificity of a term can be quantified as an inverse function of the number of documents in which it occurs.”

The inverse document frequency is a measure of how much information the word provides, that is, whether the term is common or rare across all documents. It is the logarithmically scaled inverse fraction of the documents that contain the word, obtained by dividing the to-
tal number of documents by the number of documents containing the term, and then taking the logarithm of that quotient.

**TF-IDF**

For a given term $t$, and a given document $d$ that is part of a collection of documents $D$, the TF-IDF is finally calculated as:

$$TF - IDF(t, d, D) = TF(t, d) \times IDF(t, D)$$ (2.2)

A high weight in TF-IDF is reached by a high term frequency (in the given document) and a low document-frequency of the term in the whole collection of documents: the weights will then filter out common terms. Since the ratio inside the IDF’s log function is always greater than or equal to 1, the value of IDF (and TF-IDF) is greater than or equal to 0. If a term appears in many documents, the ratio inside the logarithm approaches 1, bringing the IDF and TF-IDF closer to 0. The complete TF-IDF method and many variants are explained in [6], such as methods without logarithms and with specific normalizations, but we will here only focus on the most used version that implies logarithm.

### 2.2.2 Cosine similarity

Cosine similarity is a measure of similarity between two vectors. It gives a value between -1 and 1, and between 0 and 1 for positive vectors. It is only a measurement of orientation and not magnitude. If two vectors are similar, their cosine similarity should be near 1, and if they are significantly different it should tend to 0.

$$\cos(\theta) = \frac{\sum_{i=1}^{n} (A_i \times B_i)}{\sqrt{\sum_{i=1}^{n} (A_i^2)} \times \sqrt{\sum_{i=1}^{n} (B_i^2)}}$$ (2.3)

$$= \frac{A \cdot B}{||A||_2 ||B||_2}$$ (2.4)

where $A_i$ and $B_i$ are the components of the vectors A and B, and $\theta$ is the angle between these two vectors.
2.2.3 Considering synonyms, stop-words and stemming

There are many ways to improve this TF-IDF algorithm to get better results for utterances classification. Imagine an intent containing a lot of occurrences of the word “music”, and you say “Please turn on the radio.” This sentence would possibly be badly classified if the words “turn on” and “radio” never appear in the other utterances of the intent. That’s why considering synonyms, or at least words belonging to the same context, could be a considerable source of improvement. For this achievement several ways can be explored, such as generating synonyms by using dictionaries based on word embeddings, such as Word2vec [24].

But we can also try algorithms to detect synonyms in the corpus of the chatbot. If two words have a similar meaning, it is often possible to find them in similar sentences. For example, there can be utterances such as “I will have lunch in a restaurant” and “I will have diner in a restaurant”, where similar words like “diner and lunch” are surrounded by the same words, and they appear in the same context. So if we implement an algorithm that searches for words used in similar context, it could be easy to get synonyms, or at least similar words. Another way to improve again this classification could be to use “stop-words”, which means deleting words that are not relevant and that would not help in the classification. These are the very common words such as “the,” “a”, which are already less important since we use a TF-IDF, but removing them totally could be a solution since utterances are usually very short.

Another methods, called stemming, consists in replacing each word by its stem. For example, the words "fishing", "fished", and "fisher" would be replaced by the root word "fish". This process makes it easier to detect similarities between sentences. But do stemming help for text classification? According to [7], stemming necessarily delivers no value for text classification. Techniques like stemming help only in compensating for data sparseness. This can be a useful role, but often different forms of a word can convey significantly different cues about the correct document classification. Overly aggressive stemming can easily degrade classification performance.
Chapter 3

Related Work

This chapter will present different interesting approaches that have been done concerning text classification in general. It will be divided in two parts, dealing first with supervised methods that train on data sets, and finally unsupervised methods.

3.1 Supervised methods

3.1.1 Traditional methods

In this paper by Mubashir Ali et al. [2], a probabilistic approach for short text classification is proposed. The paper presented basically focuses at news headlines classification where each of the news headline is categorized into its defined class respectively. The system reads the news headlines and it categorizes it into suitable category such as sports, entertainment, fashion, and others. The classes are self-defined in the training data set and the two data sets have been prepared explicitly for this purpose including training and test data set. The proposed approach is generic for short text classification and can be applied on any kind of short text. Classification accuracy as well as efficiency of proposed approach demonstrates the acceptability of this approach for various short text classification problems. This study is relevant for our project in the sense that it is also about short-text classification, but categories are pre-determined and their number is low and fixed.

Most papers use classic machine learning methods such as Naive Bayes, e.g. [8] and [18], but also unsupervised approaches as [42]
Christopher D. Manning, Prabhakar Raghavan and Hinrich Schütze deal with documents in high-dimensional spaces [8]. To improve the efficiency of the algorithm, they use a technique known as feature selection in order to reduce the dimensionality by reducing the vocabulary size. For a given class $c$, a utility measure is computed for each term of the vocabulary and they then select the $k$ terms that have the highest "utility". All other terms are discarded and not used in classification. The paper focuses on improving a single algorithm whereas [18] proposes a more transversal approach by comparing Naive Bayes to SVM and k-nearest neighbors, and using also different distance measures. The kNN method got the most interesting results in this study. This type of classification can be applied to classify any short texts such as headlines of news articles, subjects of emails or chatter messages, which is the point that most interests us. Still SVM are used in many papers about short text classification, such as [9] that deals with Twitter news classification. To create the training data and testing data, each short message was classified to a group manually, within 12 different groups. One short message might belong into several groups, contrary to our case where one utterance is supposed to belong to a unique intent. Unlike other papers such as [2], each category was here [9] considered as a separate binary classification problem. The training process was developed in order to recognize whether the selected short message belong to the group A, short messages will be classified manually as “Group A” or “other”. 90% data was used to train the system and 10% were used to test the system.

An interesting discussion is proposed in [7] where methods advantages are pointed out depending on the size of the data set used. For example, high bias classifiers like Naive Bayes are well indicated for a supervised classification on little data. On the contrary, a very low bias model like a nearest neighbor model will not be advised for any data set. With a reasonable amount of data SVM become adapted for text classification. An interesting point is that the choice of the classification method is no more important if a huge amount of data is available, and emphasis should be given on scalability and runtime efficiency for example.

Okapi BM25, in which BM stands for Best Matching, is a function that allows to select documents based on a search query. This algorithm, which is specially used by search engines, works with bag of words and their relationship in the document. It is not a single func-
tion, but actually a whole family of scoring functions, with slightly different components and parameters [5]. The algorithm BM25 is one of the competitors to the TF-IDF algorithm, it was developed later and has better results than the TF-IDF but it is much more complex to implement.

3.1.2 Neural Network approaches

There are many libraries for supervised neural networks, such as FastText which was created by the Facebook Research Team for efficient learning of word representations and sentence classification. This library is based on supervised machine learning [4]. As an input you select different words (called “label”) that represent each sentence, and then for a new sentence the library will give the corresponding labels as an output [39]. It is a different way to proceed since it clusters sentences with several “labels” which can be compared as “entities”. The goal here is not to choose the most relevant label, but rather to choose many of them that match with the utterance. FastText works with keywords rather than global understanding of sentences. On the one hand libraries like FastText can be very efficient but on the other hand it is more difficult to adapt them for specific problems since we do not have access to the parameters.

Hence the study of other papers such as [43], where convolutional networks are used as a method for text classification. The issue consists in classifying Chinese texts, and Chinese language is not only composed of words and characters, but also of strokes. The network used in this paper is more general and easier to adapt, but the problem comes from Chinese language that is too different from English or French. Indeed since there is no obvious division between words in Chinese, common comparison methods using TF-IDF for instance are no longer viable. In another article, that does not deal with text classification, Sunil Ray presents a simple and interesting neural network with only one hidden layer [32]. This network is a fast method compared to deep convolutional networks, and it can solve non linear problems such as the famous XOR problem. In the article the network is only used for a binary classification, but it can be adapted for a natural language problem with a classification in several intents. This article is interesting in the sense that the network can be adapted easily with TF-IDF vectors for example.
3.2 Unsupervised methods

Supervised methods are often the most efficient, but they require labeled data which is not always available. Here are presented different papers that deal with unsupervised machine learning methods applied to text classification.

Jiaming Xua et al. propose a short text clustering [41] by implementing a Convolutional Neural Network combined with traditional unsupervised methods such as K-means. The pre-processing is done using Word2Vec [24], a tool to train word embeddings, then many clustering methods are used and compared, such as K-mean with TF-IDF, Skip-thought Vectors, Recursive Neural Network and others. The well-known K-means algorithm is also used in [36] where it is compared with heuristic K-means and fuzzy C-means algorithms to cluster longer text documents. They have experimented with different representations (TF, TF-IDF & Boolean) and different feature selection schemes (with or without stop word removal & with or without stemming). They ran the implementations on some standard data sets and computed various performance measures for these algorithms. The results indicate that TF-IDF representation, and use of stemming obtains better clustering. Moreover, fuzzy clustering produces better results than both K-means and heuristic K-means on almost all data sets, and is a more stable method. The combination of K-means and TF-IDF is also proposed by Suresh Yaram [42]. It focuses on the implementation of both document clustering algorithm, by combining a TF-IDF preprocessing with a K-means clustering, and a set of classification algorithms (Decision Tree, Random Forest and Naive Bayes). For the K-means algorithm, the Elbow method has been used to decide an optimum value of ‘K’. Their analysis reveals that Decision Tree and Random Forest algorithms perform better than Naïve Bayes algorithm.

Another study [23], that also uses the TF-IDF algorithm, presents a more global overview on chatbots and aims at analyzing their intelligence by testing different methods. The database consists in thousands of tweets that result in interactions people had with the Microsoft’s chatbot Tay. This database can be compared to ours, which is composed of questions that human users ask to a chatbot, since
both of them consist in short sentences. A pre-processing was carried out, including the removal of common stop-words, and a frequency analysis to keep only words that appear the most frequently in those tweets. The main idea was then to replace words by vectors using either Word2Vec or the TF-IDF algorithm with a cosine similarity. The principle of Word2Vec is to get closer vectors when words are similar, i.e. appear in same contexts. This method is interesting to establish a word’s association with other words, like synonyms. Nevertheless, the goal of this study was to find the topic the chatbot was talking about.

Finally, this paper [26] shows that the accuracy of learned text classifiers can be improved by augmenting a small number of labeled training documents with a large pool of unlabeled documents. This is important because in many text classification problems obtaining training labels is expensive, while large quantities of unlabeled documents are readily available. They introduce an algorithm for learning from labeled and unlabeled documents based on the combination of Expectation-Maximization (EM) and a Naive Bayes classifier. The algorithm first trains a classifier using the available labeled documents, and probabilistically labels the unlabeled documents. It then trains a new classifier using the labels for all the documents, and iterates to convergence.
Chapter 4

Classification methods and data set

4.1 Classification methods

In this section we will present some relevant classification methods that will be used in this project either for a supervised reclassification purpose, or for an unsupervised creation of new clusters (see Chapters 5 and 6 below). Then we will give precisions about the data set used.

4.1.1 Supervised

Barycenter Method

One basic way to classify vectors with supervised learning is the barycenter method. For each cluster of sentences, which are converted into vectors, a barycenter vector is calculated, which is the mean of all vectors in this cluster. Once every cluster is represented by a barycenter vector, a cosine distance can be used to compare the new sentence to these vectors.

\[
\tilde{\mu}_l = \frac{1}{C_l} \sum_{i=1}^{n} (\tilde{x}_i)
\]  

(4.1)

where \(C_l\) is the number of questions in the intent \(l\), and the \(\tilde{x}_i\) the TF-IDF vectors of each question. Once the barycenters are computed, the nearest centroid classifier can be used. The new intent can be given by:
Neural networks are machine learning systems inspired from the way human brains work. They are composed of layers containing neurons, and each neuron from one layer is connected to all the neurons from the previous layer. Each neuron takes a vector $x = (x_1, x_2, ..., x_n)$ as input, and then calculates an output using a weight vector $w = (w_1, w_2, ..., w_n)$ and a bias vector $b = (b_1, b_2, ..., b_n)$ that are both characteristics of the neuron.

\[
output = f \left( \sum_{i=1}^{n} x_i w_i + b_i \right)
\]  

(4.3)

where $f$ is an activation function.

Activation functions are used to transform the linear combination into the desired output. There are many activation functions possible. For example, the simplest one is a threshold at 0, which gives a binary output:

\[
f(x) = \begin{cases} 
0 & x \leq 0 \\
1 & x > 0
\end{cases}
\]

(4.4)
4.1.2 Unsupervised

Unsupervised classification is a machine learning method aiming at establishing a structure from unlabeled data.

K-means

The K-means clustering algorithm is known to be efficient in clustering large data sets. This clustering algorithm is one of the simplest and the best known unsupervised learning algorithms that solve the well-known clustering problem. The K-Means algorithm aims to partition a set of n vectors into K clusters, where k is a predefined constant. The main idea is to define K centroids, one for each cluster. The centroid of a cluster is chosen in a way that it is the close to all objects in that cluster, according to a specific distance measure. This is obtained by iterations, as is shown in figure 4.2:

- Centroids are chosen randomly among the different objects
- objects are assigned to the nearest centroid
- centroids values are recomputed to be the mean of all the values of objects that are assigned to this centroid
- And then objects are re-assigned and centroids values are re-computed until convergence

Purity measure

Purity is a common measure associated to the K-means algorithm and used to evaluate if clusters are homogeneous or not. The exact formula is presented in equation 4.5: for each cluster $c_k$, you determine the dominant intent $i_j$, and then count the number of questions in this cluster that belongs to this intent $i_j$. The purity will be the sum of those questions over all K clusters. Its value is between 0 and 1 where 1 means that each element is clustered with similar elements, as you can see in Figure 4.3.

$$Purity(C, I) = \frac{1}{N} \sum_c (max|c_k \cap i_j|)$$ (4.5)

where $c_k$ is the k-th cluster and $i_j$ is the j-th intent [8].
CHAPTER 4. CLASSIFICATION METHODS AND DATA SET

Figure 4.2: K-means steps [21]

Figure 4.3: Purity applied to a simple example [8]
The Elbow method

The Elbow method is a common method used to determine an optimal number of clusters when using the K-means algorithm for example. The principle is first to calculate a total distance between the elements that belong to the same cluster, then take the average of it for all the clusters. This measure called "variance" should decrease as long as K increases since clusters containing few elements are often uniform, and varies from 0 to 1. Note that the best variance v=1 is obtained for a number of clusters that equals the total number of elements, which is clearly not relevant. How to choose the right number of clusters then? The method consists here in plotting the variance and then looking for the critical point where the variance stops decreasing drastically, like it is showed in the figure 4.4. In this way, the number of clusters selected should both correspond to an acceptable variance and acceptable clusters' sizes.

4.2 Data set

The data used in this project consists in utterances that human users could use to talk to a chatbot. Several sets will be used, either labeled or not. The main set we will use contains 2005 questions in French language that have already been classified by hand in 79 different clusters called "intents". The text is encoded in UTF-8 and does not contain accents for simplification. Those algorithms were mainly tested on French language, but there are other data set that were used for experimentations, but their size is much smaller and the results were not significant enough to be reported here. Nevertheless the goal was to have algorithms that could work for any language that have a similar structure to French (spaces between words, etc).

The data set also included "entities" which are groups of words that belong to the same context, including synonyms, that will be used in the second part "Unsupervised classification" to improve the performance of the clustering.

The main data set used is specific to telecommunications and contains mainly languages related to it, which can make clustering a little more difficult (even for a human being) since there can be similarities between two or more existing intents. Among the existing intents, some of them concern "chit chat" which means common questions that
Figure 4.4: This graph represents the average distance intra clusters, which reduces when $K$ increases. The optimal $K$ to choose is the critical point (here $K=3$) where the variation is no more significant. [3]
anyone could ask to a chatbot such as "What is your name", "How are you today" or "Marry me". Since these categories are less important for IBM’s clients, the reclassification improvements were done without having to worry about chit chat results.
Chapter 5
Reclassification

The first goal here is to consider a set of questions already classified by hand in different intents and to create an algorithm to find misclassified questions before reclassifying them in the right intent. The problem is to find an objective way to evaluate the similarity between an utterance and an intent.

5.1 Preprocessing

Comparing sentences, that are bag of words, is way more complicated than comparing numbers. Hence the first step that consisted in transforming sentences into vectors. An appropriate algorithm for this transformation is the Term Frequency - Inverse Document Frequency (TF-IDF). A dictionary containing every single word that appear at least once in the set of questions must be created. Here is an example with three questions:

| question 1 | Do you need advice? |
| question 2 | What do you need?   |
| question 3 | I need help.        |

Table 5.1: Example of set of questions

Then the Term-Frequency matrix is calculated on table 5.3. Each value represents the number of occurrences of the word w in the question q.
Table 5.2: The dictionary with the number of occurrences of each word

<table>
<thead>
<tr>
<th>words</th>
<th>do</th>
<th>you</th>
<th>need</th>
<th>what</th>
<th>advice</th>
<th>I</th>
<th>help</th>
</tr>
</thead>
<tbody>
<tr>
<td>occurrences</td>
<td>2</td>
<td>2</td>
<td>3</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

Table 5.3: Term Frequency

<table>
<thead>
<tr>
<th>question</th>
<th>do</th>
<th>you</th>
<th>need</th>
<th>what</th>
<th>advice</th>
<th>I</th>
<th>help</th>
</tr>
</thead>
<tbody>
<tr>
<td>Do you like advice?</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>What do you like?</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>I like help.</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

On the table 5.4 is calculated the number \( \text{appear} \) which represents the number of questions where the word \( w \) appears at least once. This value makes it possible to get the Inverse Document Frequency with:

\[
IDF = \log\left(\frac{D}{\text{appear}}\right)
\]  

(5.1)

where \( D \) is the total number of questions (in the example \( D = 3 \)).

Table 5.4: Inverse Document Frequency

<table>
<thead>
<tr>
<th>words</th>
<th>do</th>
<th>you</th>
<th>need</th>
<th>what</th>
<th>advice</th>
<th>I</th>
<th>help</th>
</tr>
</thead>
<tbody>
<tr>
<td>appear</td>
<td>2</td>
<td>2</td>
<td>3</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>IDF</td>
<td>0.18</td>
<td>0.18</td>
<td>0</td>
<td>0.48</td>
<td>0.48</td>
<td>0.48</td>
<td>0.48</td>
</tr>
</tbody>
</table>

Table 5.5: TF-IDF

<table>
<thead>
<tr>
<th>question</th>
<th>do</th>
<th>you</th>
<th>need</th>
<th>what</th>
<th>advice</th>
<th>I</th>
<th>help</th>
</tr>
</thead>
<tbody>
<tr>
<td>Do you need advice?</td>
<td>0.18</td>
<td>0.18</td>
<td>0</td>
<td>0</td>
<td>0.48</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>What do you need?</td>
<td>0.18</td>
<td>0.18</td>
<td>0</td>
<td>0.48</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>I need help.</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0.48</td>
<td>0.48</td>
</tr>
</tbody>
</table>

Precision-Recall

The table 5.5 shows that common words that are not discriminant are given low TF-IDF weights. For example, the word "need" that appears in every question is given a weight of zero for each question, whereas words that appear only once such as "help" are given a higher
weight. In the table 5.5, the rows can be interpreted as vectors that re-
represent the questions. For example the question “Do you need advice?”
becomes the vector \([0.18 \ 0.18 \ 0 \ 0 \ 0.48 \ 0 \ 0]\).

In the data set used, the 2005 questions are turned into 2209 dimen-
sions vectors, where each dimension represent a unique word.

### 5.2 Barycenter method

In this section we describe the different attempts that used the barycen-
ter method.

#### 5.2.1 First attempt

The first step consisted in calculating a barycenter vector for every in-
tent which is the mean of all TF-IDF vectors of questions that belong

to this intent. Once we have the barycenter vectors, the basic way to
reclassify the question in the right intent is to compare the Manhattan
distance between the TF-IDF vectors and to select the intent with the
lowest distance.

\[
y = \arg \min_{j \in Y} \| \mu_j - x \|_1
\]

where \(Y\) is the group of intents, and \(\mu_j\) is the TF-IDF barycenter
vector for the intent \(j\), \(x\) is the TF-IDF vector of the current question we
want to reclassify, and \(y\) the final intent.

The results were not measured here since small tests were enough
to show that the method had to be improved. The problem with this
method is that a high difference in questions length can interfere with
the result. As an example, let’s consider 2 intents represented by one
question each:

Intent 1: "I want to change my phone."

Intent 2: "I would like to know the different types of payment pro-
posed."

And here is the new query that we want to classify either in Intent 1 or
Intent 2:

Query : "How can I pay?"

This query should rather be classified in Intent 2 because it deals
with means of payment. However, the distance from Query to Intent
2 (in terms of TF-IDF vectors) is higher than the distance to Intent 1. This is due to the high length of Intent 2 which contains many words that Query does not contain.

The best way to tackle this issue is to compare the directions of the vectors (which can be interpreted as the meaning of the sentence) instead of just comparing the distance.

5.2.2 Second attempt

The idea here is to keep the barycenter methods, but using the cosine similarity as the similarity measure instead of the Manhattan distance. It is hard to evaluate our re-classifier since we expect it to find new unexpected classifications. There are misclassified questions that must be detected to reclassify them, we can call them "positive", and questions that are already well classified and that do not interest us, that can be called "negative". Deciding if questions are "positive" (misclassified) or "negative" (well classified) was done by a NLP expert that labeled the questions. Then we assign the different labels TP, FP, FN and TN according to Figure 5.1

Over 2005 questions, we have:

- Misclassified questions correctly detected (TP): 9
- Well classified questions but detected as misclassified (FP): 38
- Well classified questions correctly detected (TN): 1958
- Misclassified Questions not detected (FN): 3

\[
Precision = \frac{TP}{TP + FP}
\]  
(5.3)
Recall = \frac{TP}{TP + FN} \quad (5.4)

Here we get \textit{precision} = 19\% and \textit{recall} = 75\%.

The recall seems acceptable but the real problem is for the precision: there are too many utterances that are wrongly detected as misclassified. Actually, the result is not as bad as it seems because there are some questions labeled as "well classified" but that could also be reclassified (they correspond to several intents that are similar). If we take into account the 10 concerned questions, the resulting precision becomes 40\%, which is still not as good as expected.

A solution that was implemented is to select only the utterances with the highest confidence level, which means the utterances that got the highest percentages for another intent. For the data set we use, if we consider that we want to reclassify at most 1\% of the utterances in the data set (20 utterances over 2005), and we then select 20 questions by the Barycenter method, we obtain a precision of 85\% instead. Indeed, most of questions that got high confidence levels were the True Positive questions.

Concerning the recall, one could argue that the good result of 75\% was only obtained observing 12 misclassified utterances, among which 9 were detected. In order to confirm this result, a test was performed consisting in misclassifying on purpose 79 utterances (1 in each intent). With this test, 53 utterances were detected over those 79, and added with the 9 detected over 12, we obtained a result of 68\% in recall, which is worse than 75\% but still confirms the previous result.

But here is the difficult point to evaluate: the questions that get good scores for several intents. For example, if a questions Q gets 40\% for intent 1 and 39\% for intent 2, the difference of percent is too low to make a decision, and the question Q should be reformulated or removed. It also happens that a question Q has low scores for all existing intents, for example 10\% for intent 1, and 8\% for intent 2, etc. An error could be to classify this question in intent 1 since it obtained the highest score. Yet this low score means that the question does not belong to any intent, and should then be removed too. Some rules were implemented to fix this issue. A question is removed if:

- the first intent score is < 25\%
- the difference between the 1st intent score and the 2nd intent score is < 5\%
Different thresholds were tried, and the two presented allowed to select the most relevant utterances to remove among the data set used. These "removed questions" can be used later for an unsupervised clustering in order to create new intents (cf chapter 6: Creation of new clusters).

5.3 Neural network

Even if the barycenter method gave acceptable results with adjustments, it can be interesting using another approach such as a neural network since it can solve more complex problems, notably nonlinear ones such as the XOR problem illustrated in Figure 5.2. The network used is fully-connected and contains 3 layers (including 1 hidden layer). The main activation function used will be a sigmoid. Each input $X$ is the TF-IDF vector corresponding to the current utterance and the output is the associated intent. The network is trained with 90% of the data set and the remaining 10% is used for validation.

5.3.1 First attempt

The first attempt consisted in having a one-dimensional output which shall be an integer representing an intent: an ID (or identifier). This value was adapted in the network to be between 0 and 1, and then re-adapted to fit the intent IDs. Many utterances that were supposed to be Positive, which means well classified according to the NLP expert, were detected here as Negative, which means that a better intent was
found for reclassification. Since the results obtained were not satisfying at all, it would not have been relevant to detail them here, but the point is that the resulting precision never exceeded 50%, no matter the parameters chosen. Results were not as good as expected because the activation function does not have the same derivative everywhere so there were many mistakes concerning intents associated with a high or low ID.

### 5.3.2 Second attempt

Hence the second experiment with an output containing as much dimensions as the number of intents $J$. The expected outputs used for training are vectors containing only zeros and with a one at the intent ID position.

<table>
<thead>
<tr>
<th>question</th>
<th>intent ID</th>
<th>expected output $y$</th>
</tr>
</thead>
<tbody>
<tr>
<td>question 1</td>
<td>2</td>
<td>[0 0 1 0 0]</td>
</tr>
<tr>
<td>question 2</td>
<td>4</td>
<td>[0 0 0 0 1]</td>
</tr>
</tbody>
</table>

Table 5.6: Example of expected outputs for a data set containing 5 intents

The first experiments were done using a small number of intents (for low execution times), with the number of utterances questions as-
Table 5.7: First results for the neural network using 7 intents and 126 questions to choose the learning rate and the number of iterations (epoch)

<table>
<thead>
<tr>
<th>hidden nodes</th>
<th>epoch</th>
<th>$\mu$</th>
<th>accuracy train</th>
<th>accuracy test</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>100</td>
<td>0.1</td>
<td>50%</td>
<td>46%</td>
</tr>
<tr>
<td>7</td>
<td>100</td>
<td>0.4</td>
<td>27%</td>
<td>31%</td>
</tr>
<tr>
<td>7</td>
<td>1000</td>
<td>0.1</td>
<td>100%</td>
<td>92%</td>
</tr>
<tr>
<td>7</td>
<td>1000</td>
<td>0.4</td>
<td>38%</td>
<td>46%</td>
</tr>
</tbody>
</table>

The second experiments aimed at finding the appropriate number of hidden nodes, using the learning rate and epoch previously found. The number of intents was chosen higher from the previous experiments to start evaluating the scalability of the method for the accuracy. According to Figure 5.8, the best results were obtained for 5 nodes in the hidden layer.

Table 5.8: Second results for the neural network using 10 intents and 196 questions to choose the number of hidden nodes

<table>
<thead>
<tr>
<th>hidden nodes</th>
<th>epoch</th>
<th>$\mu$</th>
<th>accuracy train</th>
<th>accuracy test</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>1000</td>
<td>0.1</td>
<td>99%</td>
<td>100%</td>
</tr>
<tr>
<td>10</td>
<td>1000</td>
<td>0.1</td>
<td>92%</td>
<td>60%</td>
</tr>
<tr>
<td>15</td>
<td>1000</td>
<td>0.1</td>
<td>82%</td>
<td>10%</td>
</tr>
<tr>
<td>20</td>
<td>1000</td>
<td>0.1</td>
<td>7%</td>
<td>5%</td>
</tr>
</tbody>
</table>

In the last experiments, we chose 15 intents to confirm the number of hidden nodes and to test the other kind of scalability that is the computation time.

The main issue we get here is the difficulty to identify the questions that should not be classified at all, called the "questions to remove", as is explained in the second attempt of Barycenter method. The play on percent that was performed in the Barycenter method can hardly be done here, and the implementation would lengthen the execution.
Table 5.9: Last results for the neural network using 15 intents and 326 questions

time even more.

5.4 Discussion

First the Barycenter method worked well and found interesting reclassifications with a precision of 85% and a recall of 75%. However this precision can depend a lot on the data set used. Moreover, there was many intents (79 intents for 2005 questions), most of which concerned the same topic (telecommunication here), which make borders between intents harder to fix. Indeed intents decisions made by hand can depend on the human person. This is why results are hard to evaluate. The other interesting point is the "removed questions" that are easy to catch with this barycenter method by introducing a threshold on the level of confidence.

However this classifier considered that intents can be separated linearly, which can make sense, but another approach could be relevant, hence the neural network method. For the neural network, the results helped to choose the right parameters such as the number of hidden nodes, the number of iterations (epoch) and the best learning rate $\mu$ in order to reach the highest accuracy for both training and testing sets. This is true for a small number of intents (< 15 intents and 326 questions). A huge issue that occurred for a larger number of intents is that the accuracy for the training set started plummeting. Actually, some intents were not taken into account as if they did not exist. This problem might be due to an unadapted activation function, or to the network itself. A network with more hidden layers might solve this issue, but it would result in another problem: the time necessary to run the algorithm. In the table 5.8, we can see that a couple of minutes is already necessary to run a simple network with only 15 intents. Un-
like the barycenter method, the neural network might not scale that well with an interesting number of intents.
Chapter 6

Creation of new clusters

As a continuation of the detection of "question that cannot be classified", the next step is to create new intents to contain them. The goal here is to perform unsupervised clustering in order to create groups of utterances. These groups will then be proposed as suggestions before being approved or not during a human validation.

6.1 Synonyms and stop words

The main problem here is that the classification was very impacted by common words such as "the", "and", "what" and most questions were clustered together wrongly. The TF-IDF pre-processing should have reduced the impact of those "stop-words", but it was still disturbing. Another problem is that, according to the TF-IDF vector, there is no link between two words that are synonyms or that often appear in same contexts. The idea was then to change the creation of the TF-IDF vectors to take into account the possible synonyms. Here comes another kind of data, named "entities", which contains many synonyms and words that belong to the same lexical field. Each data of this kind was made by hand and is specific to a special customer in order to be more relevant.

The first step was to use synonyms by claiming that there is no difference between a word and any of its synonyms, and vice-versa.
6.2 Method and results

The method used in this section is the K-means algorithm, which is appropriate to cluster large data sets without restrictions on the dimensions. Since the goal is rather to create small and pertinent clusters instead of huge clusters that could contain noise, the selection of the number K of clusters will certainly be chosen adequately.

\[
\text{variance} = \frac{1}{N_c} \sum_{j \in J} \sum_{i \in I} ||\mu_j - x_i||_2 \tag{6.1}
\]

where \(N_c\) is the total number of clusters, \(J\) represents the sets of intents, \(I\) represents the set of questions contained in the current intent \(j\), \(\mu_j\) is the TF-IDF barycenter vector for the intent \(j\) and \(x_i\) is the TF-IDF vector of the question \(i\).

The graph 6.1 represents the Elbow method [42] used to determine the optimal K for the K-means clustering.

<table>
<thead>
<tr>
<th>K</th>
<th>purity</th>
</tr>
</thead>
<tbody>
<tr>
<td>50</td>
<td>0.213</td>
</tr>
<tr>
<td>100</td>
<td>0.278</td>
</tr>
<tr>
<td>200</td>
<td>0.415</td>
</tr>
<tr>
<td>300</td>
<td>0.489</td>
</tr>
<tr>
<td>400</td>
<td>0.544</td>
</tr>
<tr>
<td>500</td>
<td>0.568</td>
</tr>
<tr>
<td>700</td>
<td>0.638</td>
</tr>
<tr>
<td>1000</td>
<td>0.709</td>
</tr>
</tbody>
</table>

Table 6.1: Table that represents the purity of the clusters depending on the number of clusters K for 2005 questions.

6.3 Label suggestion

Once the clustering was done, since the goal here was to "create new intents", a label had to be found for each new cluster. Finding a relevant label just by using the questions contained in the cluster is a hard task, so we had to make it simpler. Suggesting key words can be enough since a human validation is mandatory for each cluster: the
person in charge can then find the appropriate label for the cluster with the help of key words that represent in the most relevant way the cluster.

In a certain cluster, the 5 words that get the highest TF-IDF weights would be selected to represent this new intent. It seems fair since those words combine a frequent appearance with a relevant meaning (remember that the TF-IDF weight of frequent meaningless words such as "the" or "and" is largely reduced).

For example, a cluster containing many sentences such as "I would like to go to the restaurant", "Where can I find a pizzeria" would probably get a key words list like: [restaurant, go, to, where, I], which will clearly help the person in charge of finding an appropriate label. But just in case key words are enough to find the label, a "representative sentence" could be added, which means a question from the cluster that most represents it. There are multiple ways to perform it, but it was decided to select the question that is closest to the intent’s barycenter (in term of euclidean distance with the TF-IDF vectors).

\[
\text{representative question} = \arg \min_{i \in I} \| \mu - x_i \|_1
\]  

(6.2)

where \( I \) is the group of questions contained in the current intent,
\( \mu \) is the TF-IDF barycenter vector for the intent and \( x_i \) is the TF-IDF vector of the question \( i \). This label suggestion is hard to evaluate quantitatively since the relevance of the words proposed is very subjective. Tale 6.2 shows a result for one of the clusters that contains 4 questions Q1-Q4. "Label" is the question that most represents the clusters (here it is Q1), which means the question that is closest to the barycenter of the cluster in terms of cosine similarity, and "Words" are the most representative words of the cluster, that obtained the highest TF-IDF in the barycenter vector of the cluster. Both the "Label" and the "Words" will be used to make suggestions for a human that will finally decides how to name the new intent.

<table>
<thead>
<tr>
<th>Q1</th>
<th>The operator takes over my termination costs?</th>
</tr>
</thead>
<tbody>
<tr>
<td>Q2</td>
<td>Can you do a commercial gesture about the termination costs?</td>
</tr>
<tr>
<td>Q3</td>
<td>Do you reimburse the termination fees if I change my operator?</td>
</tr>
<tr>
<td>Q4</td>
<td>Do you take over the termination costs for my previous operator?</td>
</tr>
<tr>
<td>Label</td>
<td>The operator takes over my termination costs?</td>
</tr>
<tr>
<td>Words</td>
<td>termination, over, costs, operator, take</td>
</tr>
</tbody>
</table>

Table 6.2: Example of label suggestion obtained for a cluster.
Chapter 7
Conclusion

The goal of this project was to improve chatbots data sets in multiple ways. The first improvement, called "Reclassification", consisted in reorganizing the utterances classification into different intents. A pre-processing was here needed to turn the utterances into vectors of numbers that are easier to handle, and the TF-IDF method was relatively appropriate. This method is more suitable for long documents classification since it lowers the common words importances, such as "the", "what", etc, but it was still discriminant enough in our case.

Once pre-processing done, the reclassification was tackled by the Barycenter method, which is quite naive and simple, though interesting results were achieved in the second attempt when combined with the cosine similarity: 75% in precision and 85% in precision. The precision is here more interesting than recall because it is preferable not to detect all the misclassified questions than to detect too many questions. Indeed, since a human validation is required after the automatic detection, it would be time consuming to propose a reclassification for questions that are already well classified. Then, another approach using neural networks was performed in order to compare the results with the Barycenter method and try to improve our reclassification. Many parameters were taken in to account, but the accuracy has proved to be hardly scalable with too many intents. Another scalability problem was the execution time which was much higher than for the Barycenter method. Even if the Barycenter method was more "naive" than the neural networks, it obtained more interesting results for this project. Indeed, the more complicated a method is, the less flexible it can be. Furthermore, since the computation time was signifi-
The purpose of the second improvement, called "Creation of new clusters", was to use the questions that did not match any existing intent in order to create new intents. This unsupervised classification consisted in combining a classical K-means algorithm with a process that takes synonyms and entities into account. The Elbow method allowed us to find interesting results with a number of clusters $K = 300$ for 2005 questions (about 6 utterances per cluster), with an acceptable purity of 0.489. A label was then suggested for each new cluster, giving keywords and the most representative question from the cluster, before being submitted to a human validation.

As a consequence, those two improvements first make it possible to readjust data sets in order to have more homogeneous intents in a more faster way than a human. If too many questions remained misclassified, the chatbot could have detected a wrong user’s intent by assimilating the user’s query with those misclassified utterances, and it could have thus provided a wrong answer. Secondly, it can help the human in charge of the validation to realize that there are very similar intents, so he or she can merge them. Another point is that it can detect inappropriate utterances in case they obtain low confidence level for all existing intents for example. Finally, it saves once again human time with the creation of new clusters that are pre-labeled.

### 7.1 Future Work

The goal of this project was first to make actual improvements to save human time and that can be used in any chatbot that relies on a structure with utterances, intents, and, to a slightly lesser extent, entities. This structure is used in most business-oriented chatbots as seen in the Chapter 3: Related Works. The second main purpose was to explore, experiment and compare as far as possible different methods for this specific case of short-text reclassification and clustering. The methods used in this project are neither the most efficient nor the ones that get best accuracy, but they are functional, easily scalable and do not require much computational power or execution time. Other methods could be explored for those specific uses, especially neural networks with more than one hidden layer for the reclassification algorithm. For the "creation of new clusters", comparing the K-means re-
sults with another unsupervised method could have been interesting, especially with Support Vector Machines or an adapted neural network. It could also be interesting to implement other pre-processing methods and compare the results with the combination TF-IDF and cosine similarity, especially for very short texts that are commonly used with chatbots.
Bibliography


[16] IBM Cognitive Insight. “IBM Watson is now fluent in nine languages”. In: (2016).


Adam Milton-Barker. “Mitsuku chatbot wins Loebner Prize for most humanlike A.I., yet again”. In: (2016).


Joe Osborne. “IBM Watson smart assistant rises to answer Alexa and Google Assistant”. In: (2018).


Manisha Salecha. “Story of ELIZA, the first chatbot developed in 1966”. In: (2016).

BA Shawar and E Atwell. “A comparison between Alice and Elizabeth chatbot systems”. In: (2002).

Amir Shevat. “Hard questions about bot ethics”. In: (2016).


