Just.Chat - a platform for processing information to be used in chatbots

Maria João Pereira
m.joao.madeiras@ist.utl.pt
Instituto Superior Técnico, Taguspark
Lisboa, Portugal

Luísa Coheur
luisa.coheur@ist.utl.pt
Instituto Superior Técnico, Taguspark
Lisboa, Portugal

May 2013

Abstract

The number of chatbots that can be found in the web increases everyday and, alongside these, the amount of resources which can be used to build new chatbots. In this paper we target to create a platform, Just.Chat, that helps in the creation of chatbot's knowledge bases by using available chatbot's interactions, which are automatically processed by three filters. The first filter discards interactions overlapping others previously scripted; the second filter is responsible for identifying personal questions; finally, the third filter deals with interactions containing unwanted terms or topics. All these filters are individually evaluated, as well as Just.Chat, which is used to enrich the knowledge base of Edgar, a butler that has Monserrate's palace as its field of expertise.

Keywords: chatbot systems, chatbot's knowledge base creation, filtering interactions, personal questions identification, topic expansion

1. Introduction

Chatbot, a term coined by Mauldin [13], is a system that "seek to mimic conversation rather than understanding it" [18]. Since Eliza, the first chatbot, the number of this type of systems has been increasing at a dizzying pace, being applied to a panoply of applications like e-commerce or learning environments. Through the years of developing and perfecting chatbots, different ideas and technologies were explored in the chatbots' community. However, despite these advances, the basis for the creation of chatbots rests in pre-written pattern-matching templates, and in the exploration of large stores of prepared small talk responses, although more complex architectures based on learning were also proposed.

As chatbots developers generally want their systems with certain characteristics (like a name, personality or attitude[1]) in order to develop a chatbot the most common is to start by scripting its knowledge base, a database with the desired matches for inputs and corresponding answers. So, one "only" needs to think about a character, and enrich its knowledge base with possible interactions. Even better, many platforms already provide predefined interactions, which can be adapted according to the chatbot character.

[1]`http://www.chatbots.org/ai_zone/viewthread/492/`

Similar with what is done with Iris [3], which uses Movie-DiC (a dialog corpus extract from movies scripts [2]) as its knowledge base, we found that it would be equally viable to use all these interactions provided by the chatbot's knowledge base to create a chatbot's corpus, instead of having to write it from scratch. However, even if these interactions are used, there is the need to go through the tiresome process of analyzing and rewriting them; such is a necessity as we could observe that roughly including these interactions could pose some problems. The first problem we have identified was having overrides of information that could have already been scripted, that is, encountering conflicts between the gathered interactions and an already created chatbot knowledge base. The second problem is the possible existence of personal questions, as these should be customized accordingly to the chatbot's character. Finally, it is also likely that these interactions may contain undesired words or topics. This way, the following question arises: "how to automatically use all this information to build a chatbot knowledge base, but minimizing these problems?". Our researches led to no answers: we did not find any approach to cover this problem. The lack of an automatic processing tool to address this issue led to the creation of our platform Just.Chat. It aims at identifying interactions containing the previously mentioned problems and having them filtered so that they can be manually treated, while the remaining interactions can be used for constructing or improving a chatbot's knowledge base.

This document is divided as follows: in Section 2 we present an overview of chatbots alongside their main resources and ideas; in Section 3 we describe how we built the *Chat corpus*; Just.Chat's architecture and main modules are exposed, respectively, in Section 4 and 5. We evaluate this platform and Edgar's improvements in Section 6, ending with the conclusions and future work in Section 7.

2. Related Work

In this section we review the main works done in the chatbot's community. We also present Edgar, a chatbot-like system developed by L2f under the project FalaComigo.

2.1. Historical Overview

It all began in 1950, with the British mathematician Alan Turing question "can machines think?" [21] and the proposal of a way of testing it: the imitation game (now known as the Turing Test).

In 1966, the first approach to this problem came to public under the appearance of a Rogerian psychotherapist called ELIZA [22], a program developed by Joseph Weizenbaum that was able to establish a conversation with human beings, simulating it was one too. Eliza's conversational model was based in the rephrasing of input sentences, when these matched a set of pre-defined rules. ELIZA completely exceeded the expectations, given that many people, when using the program, believed they were talking with another human and that it understood their problems [10].

ELIZA is still one of the most widely known applications in AI and it is at the base of a great number of chatbots, including PARRY, its "successor". Following a very similar architecture to that of ELIZA, PARRY appeared in 1971 by the hands of Kenneth Colby, simulating a paranoid mental patient [16]. Through the usage of different tricks [13], PARRY not only could not be told appart from a real patient pointing to the fact that the emotional side can be easier to imitate than the intellectual one [11].

Many years after, Loebner understood the importance of Turing's ideas and stipulated a reward for the first person whose program could pass the Turing test. Therefore, in 1991, the first Loebner Prize Contest took place at Bostons Computer Museum [8] and, since then, the competition has been held annually in the quest of finding the "thinking computer". Besides the Loebner prize, several other competitions emerged, such as the Chatterbox Challenge[2], which started in 2001, or, more recently, the Chatbot Battles[3].

Moving back to the Loebner contests, its first winner, in 1991, was Joseph Weintraub's PC-Therapist program, based on ELIZA. Since then, many chatbots, with different goals, emerged from the competing systems, including JABBERWACKY[4], created by Rollo Carpenter and released to public in 1997 [1], targeted not to pass as a human but as an individual, a specific person with a specific personality [5]. The idea of having a chatbot that is a result of the knowledge gathered from the conversations with it is the key beyond the way JABBERWACKY works. JABBERWACKY is in the base of the popular chatbot CLEVERBOT[5].

Another competing system in the Loebner contests that needs to be highlighted, as it was responsible for boosting the chatbots field, is the ALICE. It was invented in 1995 by Dr. Richard Wallace, as a result of gathering default responses by its creator [19]. Even though it is a modern ELIZA, ALICE differs from it by not playing a specific role, but by trying to reflect a human in general. Associated with ALICE there is a panoply of resources that have been widely used by the chatbots community like, for instance, the chatbot hosting service Pandorabots[6].

As many different resources are available today, chatbots become a field in large expansion, as their technology can be used by anyone, as the most important requirement is to be creative. Due to this, chatbots can be found in a huge diversity of services, including e-commerce.

2.2. Building Chatbots

Behind each chatbot there is a development platform. These are typically based on a language that allows to define the chatbot knowledge base and an engine capable of mapping a user utterance into the most appropriate answer. This knowledge bases are usually hand crafted by the bot developer, but some platforms provide some sort of learning environment.

The writing of the chatbots knowledge sources is an extremely difficult task, since making a chatbot involves preparing it to the impossible mission of giving a plausible answer to all possible interactions. Such led to the other approach for the creation of a chatbot knowledge base by learning: by keeping new phrases and posing them later so they can be taught suitable answers for those same phrases; this approach can be found in systems like

[2]http://www.chatterboxchallenge.com
[3]http://www.chatbotbattles.com
[4]http://www.jabberwacky.com/
[5]http://www.cleverbot.com
[6]http://www.pandorabots.com

the previously mentioned JABBERWACKY or FRED. Though such an unsupervised learning may lead to unexpected and undesirable results, with the internet growth and the possibility of having many people talking with the chatbots, one may foresee that these will quickly learn and evolve.

2.3. Towards the illusion of intelligence

As not all interactions can be predicted, different strategies need to be implemented in order to allow chatbots to give a plausible answer and, thus, simulate understanding/intelligence.

A possible approach is the association of an a priori **personality** to a chatbot, as such can justify some answers that otherwise would be considered inappropriate. For instance, Rogerian mode of ELIZA never makes affirmations, Colby's PARRY being a paranoid mental patient makes acceptable its incongruous answers or even Thomas Whalen' JOE[7] which due to its working hours and being "only marginally literate" had a "fairly narrow worldview".

Another trick, is to **direct the conversation**, by using questions inciting the user participation and made him/her keep the conversation with little contribution from the program. David Levy's CONVERSE [6] used so well this trick that in the 1997 Loebner competition it convinced a judge for the first five minutes that he was really human.

Small talk, also known as phatic communication [12], is another hot topic in chatbots advances. It can be viewed as a "neutral, non-task-oriented conversation about safe topics, where no specific goals need to be achieved" [7]. Small talk can be used for two main purposes [17]: establish a social relation by building rapport and avoiding (embarrassing) silence [4].

"We tend to think of a computer's replies ought to be fast, accurate, concise and above all truthful"[8] however, human communication is not like that, containing errors, misunderstandings, disfluencies, rephrases, etc. This way, adding some **human-like failures** to a chatbot system may also be a good trick.

2.4. Edgar

Edgar is a virtual butler designed for answering natural language questions, posed either verbally or written, about Monserrates Palace, developed by L2f under the project FalaComigo - Enhance the Cultural Tourism through the Interaction with Virtual Characters.

Like many of the other systems reported through this section, the core of Edgar's architecture is responsible for matching the received input against its knowledge base in order to obtain a response to give back to the user. This matching is accomplished through the use of a technique, from now on *Edgar's matching technique*, combining natural language normalizations (e.g. stopwords removal), the Jaccard, Overlap and TfIdf algorithms.

[7] `http://thomwhalen.com/ThomLoebner1995.html`

[8] `http://www.alicebot.org/anatomy.html`

3. Chat corpus

In this Section we present the procedure of gathering, processing and analyzing chatbot's corpora which led to the identification of some problems that could rise from roughly adding it to a chatbot system.

Finding other chatbot's knowledge bases is not a trivial task as this type of data is not usually available since writing it is a slow process and its results are not shared easily. This way, the only corpus obtained was the one from the ALICE foundation[9]; this is constituted by AIML (Artificial Inteligence Markup Language) files containing question/answer (QA from now on) pairs, manually written. As for the remaining chatbots, it is quite ordinary to find logs of chats led by these. The better examples can be found in the chatbots community (like PERSONALITY FORGE[10]) or on the own chatbot page (like CLEVERBOT[11] or JABBERWACKY[12]) where their creators, wanting to show how well the system works, expose snippets where the talk was really well handled.

Depending how the gathered interactions were ordered, we processed it to one of two formats:

- *dialog format*: the different interactions are ordered as consecutive utterances, intercalated between users;

- *QA format*: the interactions are organized as QA pairs;

It should be noted that processing the collected data had its challenges, for instance, with the AIML files we had to deal with their numerous tags, each one with its own function, and use of recursion. With the processed interactions we built the *Chat corpus*, containing a total number of interactions pairs rounding the 7800.

Finally, and as mentioned in Section 1, through this process of collecting and processing information, we identified some problems which would occur if the data was not subjected to a second processing before being incorporated in a chatbot:

[9] `http://www.alicebot.org/downloads/sets.html`

[10] `http://www.personalityforge.com/forum.php?ForumID=1&StartAt=16707\#16707`

[11] `http://cleverbot.com/cleverness`

[12] `http://www.jabberwacky.com/j2conversations`

- Assuming that a chatbot knowledge base already exists, roughly incorporating new interactions in it may lead to undesired overlaps;

- Incoherencies between personal information cotained in the interactions (e.g. answers to personal questions) and the chatbot's profile;

- Presence of undesired words, expressions or topics, that one may want to filter, in the interactions.

4. The Just.Chat platform

In this section we present the architecture of Just.Chat, our platform for transforming a raw *Chat corpus* into one free of some of the problems already seen.

A sketch of Just.Chat architecture, with its main components and input and outputs, is presented in Figure 1.

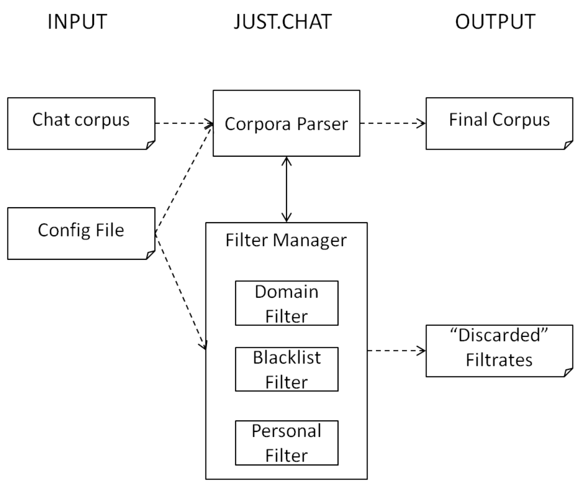

Figure 1: Just.Chat architecture

A run of Just.Ask comprises a series of steps that lead to transform the rough inputted *Chat corpus* into the *Final Corpus*.

Firstly, the *Config File* is read and provides the needed configurations to be applied to Just.Chat. Then, the *Corpora Parser* receives the files, corresponding to the *Chat corpus*, and parses the contained interactions, pair wise. Each of these pairs is passed to the *Filter Manager*; this invokes the different filters to have them test if the pair should be filtered due to the presence of one of the following problems:

- The information contained in the corpora is already present in the knowledge base of a chatbot (*Domain Filter*, Section 5.1);

- The question is personal (*Personal Filter*, Section 5.2);

- Either the input or output have undesired topics or expressions (*Blacklist Filter*, Section 5.3).

The result whether the pair was filtered or not is communicated back to the *Corpora Parser* as a boolean: *true* or *false*, respectively. The interactions not filtered are incorporated in the *Final Corpus*; as for the filtered ones, the filter responsible for its exclusion adds them to the *"Discarded" Filtrates.*

5. Filters

Through this section we will depict the different filters to resolve some problems of incoherency raised in Section 3.

5.1. *Domain filter*

The *Domain Filter* aims at avoiding collisions between existing interactions in the chatbot's knowledge base and the interactions from the new corpora to be added to the chatbot. To achieve such goal, and since restricting the filter to exact matches would be very limiting and result in many false negatives, we decided to incorporate the core of Edgar's system: its matching technique (the *Edgar's matching technique*).

This way, when the *Domain Filter* is created it takes the chatbot's knowledge base and saves the exact information contained in the files. Then, when the *Corpora Parser* sends a new pair of interactions, the filter takes the first one (as it potentially corresponds to the "question") and applies *Edgar's matching technique* to test if it discovers a match in the chatbot's knowledge base. If no match is found it means the pair can be added to the *Final Corpus.* Otherwise, the pair is kept in a different file (inserted in the *"Discarded" Filtrates*) which contains the resultant filtrates from the appliance of the *Domain Filter.*

Finally, we have that the "question" of the filtered pair, used for the matching, can be a paraphrase of the chatbot's knowledge base. This way we decided to add the option of having the *Domain Filter* replacing the answer of the filtered pair with one belonging to the knowledge base. So, we have that two files will be generated to the *"Discarded" Filtrates*: one with the original pair that was filtered and another one with the original question and the new corrected question. As a final remark, we decided to keep this file in the *"Discarded" Filtrates*, instead of immediately the corrected pair in the *Final Corpus*, as the matching against the chatbot's knowledge base may be wrong.

5.2. *Personal Filter*

The *Personal Filter* was created in order to identify and "discard" interactions pairs with personal questions so that, its answer can be mold to the chatbot's character. To do so, a sequence of tests over the first interaction of the pair (the potential question) is executed. These tests start by indentifying if the interaction is a question or not; if such verifies, another test is run to see if that question can be categorized as PERSONAL. Pairs that pass both tests are added to the *"Discarded" Filtrates*, so that the answer can be customized, while the others go to the *Final Corpus*.

Through the combination of part-of-speech (POS) tags returned by a syntactical parsing (achieved by using the Berkeley Parser [14]) and grammatical rules we came up with three rules for the identification of a sentence as being a question:

- '?' rule: the presence of a question mark at the end of the sentence;
- SQ/SBARQ rule: the run of the Syntactic Parser over the sentence and a SQ or SBARQ tag[13] at the parsed tree;
- REGEX rule: an inversion between the subject (either a personal pronoun, manually written, or a NP[13]) and the auxiliary/modal verb is detected by a regular expression.

For the classification of questions as being personal or not personal, we used a machine-learning classifier, Just.Ask, trained with a corpus built by us and the feature BU+POS (binary unigram + part-of-speech tags).

It should be noticed that Just.Ask came with a set of features which has been incremented throughout the years to accomplish diverse types of classifications [20], [15]. So, we systematicaly performed tests with these features (Section 6.2) until concluding that the best results were achieved when combining binary unigrams (BU) with POS tags.

Finally, we saw that despite the existence of expressions like "do you" in the training corpora, repeating over and over again, others like "did you" (similar to the previous one but with a change in the time of the verb) were not covered. This led, consequently, to some wrong classifications that could well be avoided just by having a small set of written rules. This way we wrote some regular expressions to immediately catch things like an inversion between the personal pronoun "you" and a modal/auxiliary verb or, this same pronoun followed by a verb with positive or negative polarity (e.g. "love" and "hate" respectively).

[13]`http://bulba.sdsu.edu/jeanette/thesis/PennTags.html`

5.3. *Blacklist Filter*

We created the *Blacklist Filter* with the objective of putting aside (that is, adding to the *"Discarded" Filtrates*) the interactions containing either undesired words, directly indicated by the user, or derived from a certain topic.

For expanding topics we used WordNet[14]. WordNet is a lexical database where nouns, verbs, adjectives and adverbs are grouped in synsets, a set of cognitive synonyms denoting the same, distinct concept. These synsets are then interlinked by relations like the super-subordinate one [9]. Taking advantage of these connections between synsets and, assuming that the topic can be viewed as the synset at the higher level, we recursively iterate the synsets linked to it until it is not possible to go further down, while, at the same time, the words that form the synsets are kept. For accessing WordNets database we used the JAWS framework[15] which not only allowed to obtain the synsets containing certain words but also to extract from these other synsets, according to a desired relation type between them (one of the listed above).

Lastly, we found that it would be fairly restrictive to confine the matches to direct ones. So, for augmenting the probability of correctly identifying and matching "bad words/expressions" in the interactions we included the following features:

- Removal of all punctuation and diacritical marks both from the "bad words/expressions" and the input strings;
- The stemming of the interaction reducing its words to their root form, achieved through the use of Java Implementation of the Porter Stemmer 2[16];
- Use of Levenshtein Distance to allow a certain number of differences (Levenshtein's editions: insertions, deletions and substitutions) between the "bad words/expassions" list's and the interaction's content.

6. Evaluation

In this section we evaluate individually each one of Just.Chat's filters and their performance in accomplishing the goal for which they were created. We also evaluate how incorporating the *Chat corpus* in Edgar, after processed by Just.Chat, improved its parlance skills.

[14]`http://wordnet.princeton.edu/`
[15]`http://lyle.smu.edu/~tspell/jaws/`
[16]`http://snowball.tartarus.org/algorithms/english/stemmer.html`

6.1. Evaluating the *Domain Filter*

In a first experiment, we randomly extracted 300 interactions from the gathered *Chat corpus* and passed them through the *Domain Filter* to see if any collided with the information encapsulated in Edgar's corpus. The results are shown in Table 1, whereas it is possible to observe the specificity of Edgar's corpus with only one of the interactions actually appearing in it (a true positive).

Filtered		Non Filtered	
Correct	Incorrect	Correct	Incorrect
1	3	296	0
	Total: 4		Total: 296

Table 1: Domain filtering results (with samples from the *Chat corpus*)

Having a filtering, or its absence, corresponding to a positive or negative prediction by the *Blacklist Filter* and whether it was correctly or incorrectly done corresponding to the actual true or false classification, we are able to calculate the precision, recall and accuracy from the tabled results, which assume the values of 25% and 100% and 98%, respectively. The attained results also point to the fact that it is likely to have some interactions wrongly matched against one of the corpora, which can be explained by the use of *Edgar's matching technique* which allows matches when only some of the terms are shared.

Due to Edgar's specificity, we were apprehensive that the estimated values when using the *Chat corpus* were not illustrative of the performance of the *Domain Filter*. This way, to simulate a "conflicting" corpus with the one of Edgar, we used a set of 427, not repeated, real user interactions posed to Edgar between the months of July and September. Also, as we had the statistics of these interactions, indicating whether they were or not covered by Edgar's corpus, we could withdraw the values summarized in Table 2.

Filtered		Non Filtered	
Correct	Incorrect	Correct	Incorrect
206	59	106	56
	Total: 265		Total: 162

Table 2: Domain filtering results (with real user interactions)

Now these tabled values conduct to an accuracy of 73%, with a precision of 78% and recall of 79%, a more moderate values than the ones attained in the first experiment. To explain the decrease of these values, we have that, besides the false positives (the 59 wrongly filtered interactions), this experiment also presented some false negatives (the 56 interactions that should have been filtered). This last situation is comprehensible as in natural language the same sentence can be expressed in a number of other ways (paraphrases). This way, though a human can easily identify these situations, for a computer this is a problem whose resolution is still a field of investigation.

To sum up, on one hand we have 13% (56/427) of interactions wrongly not being matched against Edgar's corpus, while on the other hand we have 14% (59/427) extra interactions being filtered in excess (due to a wrong match). Though the first situtation is something that we would like to avoid, the second one does not distress us as we believe that it is better to have more interactions filtered, even if incorrectly, than letting the right ones escape.

6.2. Evaluating the *Personal Filter*

As already explained in Section 5.2, the Personal Filter is composed by two parts: the identification of questions, and its classification as being PERSONAL or IMPERSONAL. Thereby, its performance is directly related to how well these two tasks are accomplished.

6.2.1 Question Identification

For testing how well the *Personal Filter* identifies questions we used the 300 interactions extracted for testing the *Domain Filter* (Section 6.1) and ran the module responsible for the question identification, to obtain its predictions and then analyzed how correct these were. To note that during our evaluations we assume that the identification of a question or non question corresponds to a positive or negative, with correctness or incorrectness of such identification corresponding to a true or false value (e.g. a sentence that is incorrectly identified as a question is a false positive). Results from the conducted experiment are shown in Table 3.

Chatbot's Corpora Set (300 clauses)			
Identified as Questions		Identified as Non Questions	
Correct	Incorrect	Correct	Incorrect
173	13	114	0
	Total: 186		Total: 114

Table 3: Final results from evaluation with Chatbot's Corpora

From the tabled results it is possible to see that all the questions were identified as such, that is, there were no false negatives. Moreover, as for the non questions we have that 13 of these were wrongly classified as questions. This corresponds to what we wanted: we prefer to have more clauses being identified as questions, even if wrongly, than missing questions. In fact, we have that the recall was of 100%, while the precision was of 93%; as for the accuracy of this final evaluation, with our final rules, we have that it rose to the 96%.

6.2.2 Classification of questions as PERSONAL and IMPERSONAL

As described through Section 5.2, in order to classify questions as being PERSONAL or IMPERSONAL we used a classifier developed by L2f under the project Just.Ask. Since the classifier used a SVM we had to gather corpora to train it and then, choose the best features to accomplish the best classification results. As Just.Ask came equipped with a set of features we systematically tested its performance with each one of them individually and then combining the ones that performed best.

We ran diverse tests always varying the test corpus. From these we could conclude that the feature that performed best was the combination of binary unigrams with POS tags with an accuracy of 84,4% (for a testing set containing the 173 correctly identified questions from the previous test). It should be noticed that the attained result was accomplished with the usage beforehand of the developed regular expressions for the immediate identification of questions (before even having these passed to Just.Ask), as without these the result would drop down to 73,9%. Also, when dwelling into the wrong and correct predictions of Just.Ask (when using the mentioned testing set and feature), we have that only one personal question was not identified as such (wich corresponds to a recall of 99%).

In conclusion, we stipulated that the combination of binary unigrams with POS tags as the feature to be used in the *Personal Filter* since it conducted to the best results. We have also saw that the inclusion of regular expressions covering patterns not present in the training corpora help in accomplishing better results. We observed, as well, that the classification of a question as PERSONAL or IMPERSONAL is not a trivial one since there are questions that do not fit directly into none of these categories. The cases that followed this scenario could be "tiebreaked" if some information about the context was provided. Such led to our final observation that it is difficult to have a universal classifier with a good performance as such would require to train it with an example of all the possible inputs and that, in natural language, is impossible; nevertheless we reached values of accuracy rounding the 84%.

6.3. Evaluating the *Blacklist Filter*

When talking about the *Blacklist Filter* we have two main features to underline: the expansion of topics using WordNet and the matching of the resulting words/expressions (or the ones introduced by the user). The second feature was something we tested informally just to verify if the normalizations, stemming and use of Levenshtein Distance were working as desired (which was the case).

This way, in this section we will discuss the results of using WordNet and its relations between synsets, to form a web of words/expressions associated to a certain subject. In order to conduct this test we asked a group of ten persons to send a topic and around ten words that they could derive from it. We passed these topics to the *Blacklist Filter* and obtained a set of words which we compared both with the topic and its related words.

It is difficult to do a formal analysis of the obtained results however, by computing the accuracy dividing the in-domain terms returned by WordNet (the true positives) with the total terms returned (the in-domain and out-of-domain terms, the true and false positives, respectively) to which we also added, whenever they existed, the sent terms not matched (the false negatives), we were able to estimate a mean accuracy of 69% for the *Blacklist Filter*. So, we were confronted with two main problems: on one hand we have that, in some cases, we could not find all the desired in-domain terms; on the other hand, we were presented with some out-of-domain terms.

The first problem can be explained by the lack of relations for reaching from one synset to another. But that is an inevitable reality when using ontologies and, taking into account that WordNet does not restrict solely to a certain topic, and instead tries to cover almost every known word, we can say that the attained accuracy is quite satisfactory; we do not think that we could do better in relating terms without the use of WordNet. We believe that the right choice of the topic, even including more than one word, can solve this potential problem; in fact, we have that two of the topics had all the words given matched against the ones returned by the *Blacklist Filter*. Also, despite the large database that is behind WordNet there is always the possibility, as seen, that some words are not included in it. It is for resolving these cases that we think that some future work could be done.

As for the terms unrelated to the given topic, we have that these are due to expanding the wrong synsets. As sometimes this happens due to a deep expansion of some synsets, limiting this expansion to a certain depth could be an approach to resolve some of these problems; however that would reduce dramatically the returned terms, even the ones related to the given topic.

6.4. Parlance Improvements when using the *Chat corpus*

For a final test we thought that it would be interesting to see a concretization of all the work done, from the collection and processing of other chatbot's knowledge bases to its filtering using Just.Chat and its addition to Edgar (and test possible improvements in this system).

In order to obtain a corpus that could be added to Edgar (from now on called the *Test corpus*) we started by passing the *Chat corpus* through Just.Chat. We used all of Just.Chat's filters so we could test its full capacities:

- For the interactions that should be caught by the *Domain Filter* we used, once more, Edgar's corpus;
- We considered as interactions that should be caught by the *Blacklist Filter* the ones containing terms derived from the topic "dirty words". Also, to complement the terms derived from "dirty words", we added four softer terms that we considered as undesired.

With all the filters configured we ran Just.Chat over the *Chat corpus*. Table 4 shows the total number of interactions processed and, from these, which ones were filtered (and went to the *"Dicarded" Filtrates*) and which ones were not (and went to the *Final Corpus*).

filtered interactions (*"Discarded" Filtrates*)	¬filtered interactions (*Final Corpus*)
26848	51092
# total interactions: 77940	

Table 4: Statistics from the interactions processed by Just.Chat

We took all the resultant interactions, simulating that the ones "discarded" were properly corrected:

- The 3660 interactions caught by the *Domain Filter*, thanks to its additional feature, had their answers automatically replaced by one from Edgar;
- The 21241 interactions caught by the *Personal Filter* had their answers corrected according to Edgar's profile;
- The 1947 interactions caught by the *Blacklist Filter* had their answers rewritten accordingly to whether these were replies to questions containing the undesired terms, or if the answers per se were the ones containing one of these unwanted terms.

This way, with the *Final Corpus* and the, supposedly corrected, *"Discarded" Filtrates* we formed our *Test corpus* which we added to Edgar's knowledge base.

With Edgar's knowledge base enriched with the *Test corpus*, we passed the 427 interactions used in the *Domain Filter* test (Section 6.1). By doing such we registered some fluctuations in previously registered matches, which can be explained by changes in the weights attributed to a word by the TfIdf algorithm (due to the addition of the *Test corpus* to Edgar) and the occurence of 62 matches against the *Test corpus* (27 interactions matched came from the *"Discarded" Filtrates*, while the remaining 35 came from the *Final Corpus*). Assuming that all the questions matched against the *Test corpus* are true positives and that all the inputs that could be matched against this corpus were actually matched (the only false negatives remain the ones evaluated against Edgar's corpus), we obtain the new results depicted in Table 5, where "KB" and "¬KB" (KB stands for knowledge base) indicates whether the received input exists, or not, in the new enriched Edgar's knowledge base.

Answered Inputs		Not Answered Inputs	
Correct (KB)	Incorrect (¬KB)	Correct (¬KB)	Incorrect (KB)
266	63	60	38
Total: 329		Total: 98	

Table 5: Edgar's results when using the *Test corpus*

As can be seen in the Table 5, Edgar could now answer to 329 inputs, which, comparatively to the 265 (Table 2) corresponds to an increase of 24% of the responses given.

Finally, we wanted to test the quality of the original unchanged responses from the *Chat corpus* versus the new ones from the *Test corpus*. To do so, we took Edgar's 62 matches against the *Test corpus*, from these we created two sets:

- A first one with the original answers (the ones from the *Chat corpus*) to the 62 matches;
- A second one with manually written answers to the 62 matches (except for 7 matches, corresponding to interactions caught by the *Domain Filter* which the answer was automatically replaced).

Then, we asked a group of 10 persons to rate, in a scale of one to five, how correct each one of the answers of the two sets were, before a given input. We also informed each involved person about Edgar's profile, like its advanced age, and the role it plays, emphasizing that the system existed and was working and so, the ratings should reflect how appropriate the diverse responses were.

From the scores given by our 10 evaluators the following inferences could be taken:

- As expected, the manually written answers had the best scores;
- The discrepancy between unchanged *Chat corpus'* answers and corrected ones is bigger when these correspond to ones which have been filtered by Just.Chat. This means, the difference of score between original and handwritten answers is bigger if these correspond to interactions that Just.Chat "discarded". Such indicates that interactions filtered by Just.Chat should be reviewed by a human and re-written. As for the remaining unfiltered interactions, even if indiscriminately used, conduct to satisfactory results (even if not so good like the ones attained with manually written answers).

To end this test, we also asked our testers what score they would give if, instead of the presented answers, these were replaced with a "Sorry I didn't understand". The mean result attained was of **1,6** which shows that, despite some answers not being so good, it is still preferable to have these being outputted than admitting ignorance before the received input.

7. Conclusions

Motivated by the amount of resources provided by the chatbot's community, which could be used to build chatbots, we decided to use these same resources, in our case examples of possible interactions, and have them incorporated in Edgar, a butler chatbot-like system, to enrich its knowledge base. So, taking the interactions from ALICE's AIML files, CLEVERBOT's and *Personality Forge*'s chatbots, we build the *Chat corpus*, containing around 78000 pairs of interactions.

However, after analyzing and processing the *Chat corpus*, we were confronted with some problems that could derive from crudely adding it into a chatbot system, being it Edgar or another one. Such led to the creation of our platform: Just.Chat.

Just.Chat aims at identifying and filtering (that is, putting aside) interactions containing the following problems: a) conflicts with previously created interactions; b) personal questions, which answer should be customized accordingly to the chatbot system profile; c) undesired topics or words. We developed a filter for addressing each one of these cases: the *Domain Filter*, the *Personal Filter* and the *Blacklist Filter*. We also tested individually each one of these filters, estimating its accuracies which were, respectively, 73%, 84% and 69%.

At last, we decided to have some of the collected interactions passed through Just.Chat. We took all the resultant interactions, simulating that the ones "discarded" were properly corrected, and added them to Edgar. This enrichment of its knowledge base resulted in an improvement of 24% in the number of answers given. Also, when asked to score the correctness of the answers, from the new interactions added, the users inevitably scored higher to the handwritten answers. However, if the original answers were not corrected, the ones not filtered by Edgar, that is, the supposedly right ones, reached higher values than the ones filtered, "discarded"; also, before the possibility of having any of these answers returned or a simple "I didn't understand", all the users preferred the first option. These results seem to corroborate our initial idea that external sources can be in fact used to either build a chatbot from scratch, or improve an existing knowledge base; but, even more, point to the fact that while it is "safe" to roughly include some of these new possible interactions, it is necessary to review and even rewrite some others (either because they overlapped others scripted, or contained personal questions or even undesired terms). Just.Chat comes into play here, appearing as a helping tool when dealing with all these interactions, as it provides a way for filtering these "others" interactions from the remaining ones, so that the user only has to look at the set of interactions filtered, instead of them all.

For future work, we consider that the following complements and improvements to Just.Chat would be interesting:

- Extend the *Personal Filter* to not only catch interactions based on the fact of the question being a personal one, but also consider to be filtered pairs which answer contains personal information;
- Implementing and adding to Just.Chat one more filter for detecting interactions with temporal and spatial incoherencies, as some data varies with the change of time and location: considering the interactions pair "Who is the president" + "George W Bush", we have that the answer given is not only outdated but also not applicable for non-american chatbots;
- Develop a mechanism for avoiding incoherencies when merging the interactions from different sources, for instance the same question leading to different results (e.g. "What is your name" having as answer "Alice", likely response extracted from the AIML files, or

"Cleverbot", the likely response from Cleverbot files).

At last, we have that the previously presented ideas are only some possible directives for a work that still has much space for growth and improvement.

References

[1] A. D. Angeli and S. Brahnam. I hate you! disinhibition with virtual partners. *Interact. Comput.*, 20:302–310, May 2008.

[2] R. E. Banchs. Movie-dic: a movie dialogue corpus for research and development. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Short Papers - Volume 2*, ACL '12, pages 203–207, Stroudsburg, PA, USA, 2012. Association for Computational Linguistics.

[3] R. E. Banchs and H. Li. Iris: a chat-oriented dialogue system based on the vector space model. In *Proceedings of the ACL 2012 System Demonstrations*, ACL '12, pages 37–42, Stroudsburg, PA, USA, 2012. Association for Computational Linguistics.

[4] T. Bickmore and J. Cassell. *How about this weather?- Social Dialogue with Embodied Conversational Agents.* Number Clark 1996. 2000.

[5] R. Carpenter and J. Freeman. Computing machinery and the individual: the personal turing test, 2005, paper available at http://www.jabberwacky.com.

[6] B. L. Catizone, B. Batacharia, D. Levy, R. Catizone, A. Krotov, and Y. Wilks. Converse: a conversational companion. In *Proceedings of the 1st International Workshop on Human-Computer Conversation*, 1997.

[7] B. Endrass, M. Rehm, and E. André. Planning small talk behavior with cultural influences for multiagent systems. *Comput. Speech Lang.*, 25:158–174, April 2011.

[8] R. Epstein. The quest for the thinking computer. *AI Magazine*, pages 81–95, 1992.

[9] C. Fellbaum. *WordNet: An Electronic Lexical Database.* Bradford Books, 1998.

[10] J. L. Hutchens. How to pass the turing test by cheating. Technical report, 1997.

[11] B. Kuipers, J. McCarthy, and J. Weizenbaum. Computer power and human reason. *SIGART Bull.*, pages 4–13, June 1976.

[12] B. Malinowski. *The Problem of Meaning in Primitive Socities*, page 38. 1923.

[13] M. L. Mauldin. Chatterbots, tinymuds, and the turing test: entering the loebner prize competition. In *Proceedings of the twelfth national conference on Artificial intelligence (vol. 1)*, AAAI '94, pages 16–21, Menlo Park, CA, USA, 1994. American Association for Artificial Intelligence.

[14] S. Petrov and D. Klein. Improved inference for unlexicalized parsing. In *Proceedings of HLT-NAACL*, pages 404–411, Rochester, New York, April 2007. Association for Computational Linguistics.

[15] R. M. Pires. Query classification and expansion in just.ask question answering system. Master's thesis, Instituto Superior Tcnico, 2012.

[16] A. P. Saygin, I. Cicekli, and V. Akman. Turing test: 50 years later. *Minds and Machines*, 10:2000, 2000.

[17] K. Schneider. *Small talk: analyzing phatic discourse.* Sprachwissenschaftliche Reihe. Hitzeroth, 1988.

[18] R. P. Schumaker, M. Ginsburg, H. Chen, and Y. Liu. An evaluation of the chat and knowledge delivery components of a low-level dialog system: The az-alice experiment. *Decis. Support Syst.*, 42:2236–2246, January 2007.

[19] H. Shah. A . l . i . c . e .: an ace in digitaland. *Artificial Intelligence*, 4(2):284–292, 2006.

[20] J. a. Silva, L. Coheur, A. C. Mendes, and A. Wichert. From symbolic to sub-symbolic information in question classification. *Artif. Intell. Rev.*, 35(2):137–154, Feb. 2011.

[21] A. M. Turing. Computing Machinery and Intelligence. *Mind*, LIX:433–460, 1950.

[22] J. Weizenbaum. Eliza - a computer program for the study of natural language communication between man and machine. *Commun. ACM*, 9:36–45, January 1966.