A Virtual Assistant for Web Dashboards – Suport Bot

Ricardo Lopes
ricardo.mendes.lopes@tecnico.ulisboa.pt

Instituto Superior Técnico, Universidade de Lisboa, Portugal

October 2018

Abstract

The aim of this work is to implement a system capable of answering questions posed by the employees of BNP Paribas, regarding the web dashboards they use, in Natural Language – a Frequently Asked Questions problem. Since we lack data, we explore how useful paraphrases can be to automatically generate data. To do this we develop a Paraphrase Generation module based on paraphrasing rules and based on Neural Machine Translation, and apply it to our manually-created questions dataset. We find that using paraphrases to populate a Knowledge Base improves the accuracy of our system. Moreover, we propose to enhance our solution by asking users for their feedback, which enables the system to match new questions with the ones already in the Knowledge Base. The final solution is a system that generates paraphrases of the user question, searches for similar questions in the Knowledge Base, retrieves the final answer from a Question/Answer Database and learns from the feedback given by the users.

Keywords: Natural Language Processing, Question Answering, Paraphrase Generation

1. Introduction

BNP Paribas (BNP) is an international banking group with a presence in 74 countries and more than 190,000 employees[1]. As a result of their operations large amounts of data are recorded for analysis. Tools such as web dashboards are used by BNP employees to access the information they need to perform business activities. An example of a dashboard used by BNP is the "Rollover[2] Opportunities" dashboard, which displays information regarding trades about to expire and the history of trades of a given client. For instance, users can see how many times a particular client has performed a rollover and which clients might be more interesting to talk to, in order to perform a rollover (a rollover is the process of extending the settlement date of an open position). Currently, if dashboard's users have any questions, other than asking their coworkers, they can only send an email to the support team asking for help. However, replies are not immediate, and users are forced to wait, causing productivity to drop. Examples of questions users of the "Rollover Opportunities" dashboard might ask are: "How is the rollover ratio calculated?" or "What are the filtering options of the dashboard?".

[1] http://www.bnpparibas.pt/en/bnp-paribas/bnp-paribas-group/ (accessed 20-12-2017)

[2] https://www.investopedia.com/ask/answers/06/rollover.asp (accessed August 2018)

We propose to implement a Question Answering System (QAS), a type of system designed to automatically answer questions made by users in Natural Language to solve the dashboard users' problem of having to wait for a reply from the support team. Instead, users can ask the same question to the QAS and receive an answer much faster. This is a Frequently Asked Questions (FAQ) problem. As such, we will build a questions dataset (FAQ collection) based on the documentation of the dashboard. To deal with the problem of lack of data, we will create a Paraphrase Generation (PG) module, to automatically generate paraphrases of the questions in our dataset, in order to expand it.

1.1. Objectives

The objectives of this work are as follows:

- Implement a system capable of answering the dashboards users' inquiries;
- Implement a Paraphrase Generation module capable of expanding our collection of questions.

1.2. Organization of the Document

This work is organized as follows: Section 2 presents and evaluates our Paraphrase Generation module, used in several of our proposed solutions. In Section 3, we propose and evaluate several solutions to build a Question Answering System, based on the amount of work required to to

build a QAS for a different dashboard. Finally, Section 4 presents the conclusions of our work.

2. Paraphrase Generation

This chapter discusses the methodology followed in order to build a PG module for our Question Answering System. The main objective of this module is to generate paraphrases of the questions inputted by the users, which can be seen as a method of Automatic Query Expansion. We performed an experiment using a Paraphrase Generation framework and publicly available corpora. Since results were poor, we used 2 other methods for paraphrase generation: based on a database of paraphrases (PPDB-based) and based on Neural Machine Translation (NMT). In order to evaluate these approaches we first had to create a question dataset. We conclude this chapter with the evaluation of those 2 methods and the discussion of the results.

2.1. Preliminary experiment

A first experiment to generate paraphrases was made using a Question Generation framework developed by Rodrigues et al. [1]. The framework takes as input sentence/question pairs to create semantic patterns. Questions are then generated by applying the patterns created to new sentences. When a match is found, a question is generated. In our experiment we wanted to see if it was possible to utilize the framework developed by Rodrigues et al. to paraphrase questions, instead of only generating questions from sentences. That is, to input question/question pairs, create semantic patterns and generate questions by matching the patterns with new questions. To do this we used the crowdsourced paraphrase corpora made available by Wang et al. [2] as input seeds to the framework. There are 8 different corpora: basketball; blocks; calendar; geo880; housing; publications; recipes; regex; restaurants and socialnetwork. The corpora were created using Amazon Mechanical Turk (AMT)[3] by having crowd workers (turkers) paraphrase canonical utterances, such as “meeting whose date is at least date of weekly standup” or “player whose number of steals (over a season) is smaller than 3”.

We defined 2 experimental cases: canonical/paraphrase and paraphrase/paraphrase. In the first case we used the canonical utterance given to the turkers and one paraphrase as seeds. We used the first 26 pairs of canonical utterance and paraphrase as input. In contrast, in the second case we used two paraphrases generated by the turkers, from the canonical utterances. We used one paraphrase/paraphrase pair for each canonical utterance. Additionally, to generate new questions we used canonical utterances in the first case and human-generated paraphrases in the second case.

An example of a pair used in the first case is “player whose number of steals (over a season) is smaller than 3” and “name a player with fewer than three steals?”. For the second case, an example of a pair we used is “name a player with fewer than three steals?” and “how many players have less than 3 steals on them in a season?”

Unfortunately, results from preliminary experiments were extremely poor. Countless questions where generated but close to none had any syntactic or semantic correctness.

2.2. Main problems with this approach

Here we will briefly discuss which were the problems behind this approach and the reasons why it failed.

2.2.1 Dataset problems

As previously mentioned, we used the corpora made available by Wang et al. as input to this experiment. Furthermore, we had assumed that paraphrased sentences would have the same meaning, but found that it was not always the case.

After a brief examination, we found that some workers write questions with a different meaning. For the original canonical utterance “season of player kobe bryant whose number of blocks (over a season) is 3” one worker wrote the question “what player had more blocks than kobe bryant whose number of blocks is 3”, which has a completely different meaning from the original. Another worker wrote “in which season did kobe bryant average 3 blocks”. By adding “average” to the question, it also changes its meaning from “total blocks” to “average blocks”.

However, the fault does not always lie with the workers. Some canonical utterances are vague and open to interpretation. For instance, the canonical utterance “number of position”. While most questions mean “how many different positions are there”, one worker wrote “what is the number of the player in this position”, which has a different meaning.

2.2.2 Framework limitations

We also found some limitations with the framework. It requires that every predicate argument in the sentence (or first question, in our case) to exist in the question. Taking as an example the questions “player who had less than 3 steals in a season” and “who has had less than 3 steals?”, since “player” does not exist in the second question, no pattern is created. Furthermore, using the question pair “In which seasons did kobe bryant make

[3] https://www.mturk.com/ (accessed 30-05-2018)

3 blocks?" and "Season in which 3 blocks were made by kobe bryant?" as example, in the first sentence, "In which seasons" is labeled as *AM-TMP* denoting a temporal relation, but on the second sentence "Season in which" is labeled as *Loc* relating to a location. As such, no pattern is created between those questions either. Additionally, since the framework is designed to receive as input a sentence/question pair, it requires the first input to be bigger than the second input. While this is not a limitation per se, as it can be overcome by always placing as first input the question with bigger length, it was inconvenient.

2.3. Creating a Dataset for evaluation

Since we wanted to evaluate our paraphrasing approaches we first had to create a questions dataset, to which we would apply our paraphrasing methods, to generate paraphrases.

The development process began by manually writing questions based on the documentation of the dashboard "FXLM Rollover Opportunities". The documentation of this dashboard consisted of approximately 6 pages of information, which resulted in 606 different questions.

The next step was to manually group the 606 questions by expected answer, that is, to group the questions such that each group contains only questions that have the same answer – we did this in order to be able to test the quality of our paraphrases as we assume that paraphrases of a question have the same answer. The outcome of this grouping were 130 different groups. Examples of questions from the group with highest frequency are "How can I see what clients are more likely to rollover?", "How can we find potential clients to speak with to rollover with BNPP?" and "How can I know what clients will rollover?". The following table and figure present more details regarding the dataset created. In Table 1 we can see the number of questions, number of groups, maximum, minimum and mean questions per group, as well as the standard deviation.

Table 1: Original Dataset Details

Original Dataset Details	
Nr. of questions	606
Nr. of question groups	130
Max questions per group	72
Min questions per group	1
Average questions per group	4.662
Standard Deviation	7.967

2.4. Generation based on the ParaPhrase DataBase

Paraphrase Database (PPDB) [3] is, as the name suggests, a Database of paraphrases. It offers paraphrase packs in 22 different languages in sizes ranging from S (small) to XXXL (extra-extra-extra-large). The paraphrase packs are plain text files with a paraphrase rule per line. The rules for the English pack are formatted as follows:

$$LHS|||PHRASE|||PARAPHRASE|||(FEATURE=VALUE)^*|||ALIGNMENT|||ENTAILMENT$$

Where LHS is the constituent label for the paraphrase pair, PHRASE is an expression, PARAPHRASE is its paraphrase, (FEATURE=VALUE)* is a list of features and respective values, ALIGNMENT is the alignment between the PHRASE and PARAPHRASE, and ENTAILMENT denotes an entailment relation between PHRASE and PARAPHRASE (e.g. Equivalence for the pair restriction/limitation or ForwardEntailment for the pair dog/animal).

Following the work of Dong et al. [4] we focused on **LEXICAL** (single word) and **PHRASAL** (multiword to single/multiword) paraphrases and used the S sized pack for precision.

Instead of using all the rules in the pack, we selected only those with an entailment relation we thought appropriate for paraphrases, namely: Equivalence, ForwardEntailment (hyponym) and ReverseEntailment (hypernmym). A hyponym is "A word of more specific meaning than a general or superordinate term applicable to it" [4] as exemplified above, whereas is the opposite.

Additionally, we also filtered the number of rules based on the features "PPDB1.0Score" and "PPDB2.0Score", and a threshold for those features. We selected the thresholds for filtering after a small empirical experiment.

We chose 4 different thresholds, based on quartiles:

- PPDB1.0Score above second quartile and PPDB2.0Score above second quartile (50-50);
- PPDB1.0Score above third quartile and PPDB2.0Score above second quartile (75-50);
- PPDB1.0Score above second quartile and PPDB2.0Score above third quartile (50-75);
- PPDB1.0Score above third quartile and PPDB2.0Score above third quartile (75-75).

We then filter the rules in two ways:

- By selecting those with PPDB1.0Score and PPDB2.0Score above the threshold (Case A);
- By grouping the rules according to their Entailment Relation and then selecting those with

[4] https://en.oxforddictionaries.com/definition/hyponym (accessed 22/09/2018)

PPDB1.0Score and PPDB2.0Score above the threshold, for each group (Case B).

We then applied each set of rules to our manually created 606 questions and removed duplicates.

2.5. Generation based on Neural Machine Translation

NMT is the an approach to Machine Translation (MT) that uses Neural Networks (NNs) to translate text or speech from one language to another. We follow the work of Dong et al. [4] to generate paraphrases by translating sentences from English to German and then translating back to English. At first we attempted to use the same library as Dong et al., however we faced many problems due to a conflict of versions between several libraries. We then searched for other NMT libraries and found tensorflow's Tensor2Tensor (T2T) library[5] [5], which we eventually used. More specifically, T2T's implementation of the Transformer model from the paper by Vaswani et al. [6].

We trained 2 models with Hparams "transformer_base" on data released as part of the WMT15 task (4.2 million English and German aligned sentences) for 21k steps each, on 8 K80 GPU's, using Google Cloud[6].

To decode (i.e. translate), we set the parameters "Beam_size" to 10 and "Alpha" to 0.6, after a brief empirical experiment.

An example of a paraphrase generated using this method is "What is a billing date?". The original question was "What is a settlement date?" and the German-translated question was "Was ist ein Abrechnungsdatum?".

Using this method, each original question may, or not, generate a paraphrase since the final translated question may be the same as the original question. Additionally, it is possible that different questions generate the same paraphrase. From the 606 generated paraphrases only 562 were left after removing duplicates. Additionally, from those 562, 232 already existed in the original questions.

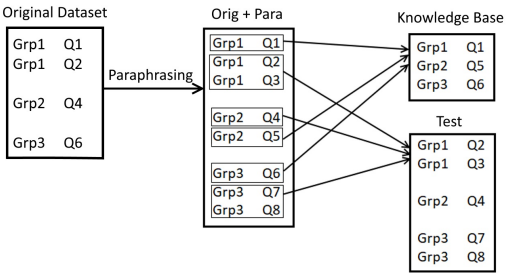
2.6. Evaluating our approaches

In order to evaluate our PPDB-based and NMT-based approaches we designed 3 experiments: Big Knowledge Base, Small Knowledge Base and Real Case Simulation. In all experiments the goal was to classify the "test" questions using the questions in our "Knowledge Base (KB)". For example, if we had in our KB a question manually classified as being in group 1 "What is the main goal of the dashboard?" and a question manually classified as being in group 2 "What does the dashboard show?", given a new question "What can I see in the dashboard?" the goal would be to classify it as belonging to group 1 or 2.

In the first experiment, Big KB, we simulate the setting of having a big knowledge base. We start by generating paraphrases of our original questions. Then, we choose 1 question from each group for testing and use the remaining as our KB. After that, we compute the distance between the test questions and the KB questions and classify the group of each test question as the group of the KB question closest to it.

In the Small KB experiment we begin as in the previous experiment, by generating paraphrases of the original questions, and then do the opposite: choose 1 question from each group for the KB and use the remaining questions for testing. This is better shown in Figure 1. After selecting questions for the KB and testing, we compute the distance between the test questions and the KB questions, and select as the group for each of the test questions the group of the KB closest to it, as in the Big KB experiment. This setting can also be seen as a "lazy" setting and more likely to occur in the "real world" as it simulates a FAQ setting, where we have 1 answer for each FAQ.

Figure 1: Small KB Setting



The experiment "Real Case Simulation" is similar to the Small KB experiment since we also choose 1 question from each group for the KB and use the remaining for testing. However, in contrast to the Small KB experiment, we first choose the questions for the KB and testing and only then generate paraphrases of the questions in the KB. The process of selecting the questions can be seen in Figure 2

For each experiment we tested using:

- only the manually generated questions;
- only PPDB-based generated paraphrases (each paraphrase set individually);
- only NMT-based paraphrases;
- original questions + best lexical and phrasal PPDB-based paraphrases + NMT-based paraphrases

[5] https://github.com/tensorflow/tensor2tensor (accessed June 2018)

[6] https://cloud.google.com/

Figure 2: Real Case Simulation Setting

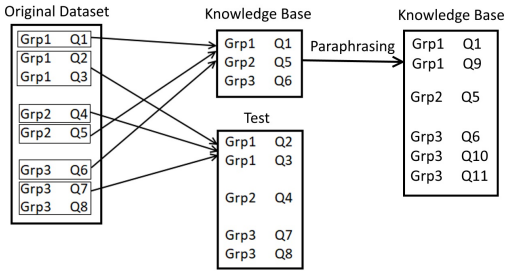


Table 2: Big KB Original Questions Results

	Big KB Results
Metrics	**Original Questions**
Lev-char	0.666
Lev-word	0.513
Jac-char	0.655
Jac-uni	0.603
Jac-bi	**0.678**
Jac-tri	0.659
W2V-sum	0.650

We classified the questions using 7 different distance metrics based on the Levenshtein Distance [7], Jaccard Distance [8] and Word2Vec [9]:

- Character-wise Levenshtein Distance
- Word-wise Levenshtein Distance
- Character-wise Jaccard Distance
- Unigram Jaccard Distance
- Bigram Jaccard Distance
- Trigram Jaccard Distance
- Distance between the sums of Word2Vec Embeddings

2.7. Results

Since we randomly choose a sample of questions for the knowledge base and testing, for each of the 2 experiments we classify the questions and compute the accuracy of the classification 20 times and average the results. Accuracy is defined as

$$Acc = \frac{TP + TN}{TP + TN + FP + FN}$$

Where TP where TP are correctly identified instances, or True Positives, TN are correctly rejected, or True Negatives, FP are incorrectly identified, or False Positives and FN are the incorrectly rejected instances, or False Negatives.

2.7.1 Big Knowledge Base

The results for this experiment can be seen in Tables 2 and 3.

For the original questions, whose results can be seen in Table 2, the best distance metric was the Jaccard Distance using bigrams (0.678). However, all except for the Word-based Levenshtein Distance (0.513) and Unigram-based Jaccard Distance (0.603) have comparable performances.

The best result for the PPDB-based lexical is obtained by the question set "Case A-50-50" and using the Character-based Levenshtein Distance (0.778). Moreover, except for the question set "Case B-75-75", where the best metric is the Jaccard Distance using bigrams, the best metric was the Levenshtein Distance for characters. Comparing these results with the ones obtained using only the original questions, we can see that except for the "Case B 75-75" paraphrase set, the best result was always better than the best result of the original questions. Furthermore, the overall best result for this experiment – "Case A 50-50", improved the original results by 10% (0.778 vs. 0.678).

Similarly to the lexical results, the best metric for the PPDB-based phrasal paraphrases is almost always the Levenshtein Distance. When it is not the best metric it still gives a comparable performance. The best overall result is obtained by the "Case B-50-50" question set and the Levenshtein Distance based on characters (0.723). Once again, by comparing the results of using phrasal paraphrases with the results of using only the original questions, we see an increase in the best accuracy when using paraphrases (0.723 vs. 0.678).

Regarding the NMT paraphrases, the best metric are Word Vectors. However, results are slightly poorer in comparison with the original questions (0.650 vs 0.643). This is perhaps due to having less questions in the knowledge base, since the NMT approach generated less paraphrases than the number of original questions.

The results of combining the Original Questions with the best PPDB questions (best lexical and best phrasal) and NMT questions are shown in Table 3.

Table 3: Big KB Combination of All Methods Results

	Big KB Results
Metrics	**Original + NMT + PPDB**
Lev-char	**0.819**
Lev-word	0.657
Jac-char	0.716
Jac-uni	0.704
Jac-bi	0.718
Jac-tri	0.675
W2V-sum	0.725

From the results we can see the best met-

ric is the Character-based Levenshtein Distance (0.819). Furthermore, we can conclude that using our paraphrases to populate the Knowledge Base has a positive impact – we see an increase in the accuracy when using the paraphrases and our best overall result is achieved when we combine the original questions with both of our paraphrasing methods. This suggests that having more examples increases the accuracy, which is expected as we have more data in our KB on which to base our classification. However, the quality of the paraphrases is also important seeing that the accuracy for some of the phrasal paraphrases question sets (e.g. Case B-50-75) have accuracy comparable to the original questions and NMT paraphrases (0.659 vs. 0.678 vs. 0.643), while having more questions in the KB (1,097 vs. 606 vs. 562 questions).

2.7.2 Small Knowledge Base

The results for this experiment can be seen in Tables 4 and 5.

Table 4 shows the results for the original questions. The best metric was Word2Vector (0.682).

Table 4: Small KB Original Questions Results

	Small KB Results
Metrics	**Original Questions**
Lev-char	0.648
Lev-word	0.624
Jac-char	0.478
Jac-uni	0.681
Jac-bi	0.667
Jac-tri	0.611
W2V-sum	**0.682**

The best result for lexical paraphrases is obtained by the question set “Case A-50-50” and using the Unigram-based Jaccard Distance (0.753). We can see that by using lexical paraphrases our accuracy increased, in comparison with the accuracy of using only the original questions (0.753 vs. 0.682). Additionally, the best metric for all the question sets was either the Unigram-based Jaccard Distance or Word2Vector and both have always comparable performance to each other, except for the “Case A-75-50” question set.

The best result for phrasal paraphrases is obtained by the “Case B-75-50” question set and Word2Vector (0.788). Once again, we see that using paraphrases improves the classification (0.788 vs. 0.682). The best metrics are the Jaccard Distance computed on unigrams and Word2Vector, except for the “Case A-50-50” question set, which is the Bigram-based Jaccard Distance.

Regarding the NMT approach, the best results, as with the original questions, are obtained with Word Vectors. However, a worst performance than with the original questions is achieved (0.587 vs. 0.682).

The results of combining the Original Questions with the NMT questions and the best PPDB questions are shown in Table 5.

Table 5: Small KB Combination of All Methods Results

	Small KB Results
Metrics	**Original + NMT + PPDB**
Lev-char	0.600
Lev-word	0.605
Jac-char	0.403
Jac-uni	0.691
Jac-bi	0.638
Jac-tri	0.521
W2V-sum	**0.703**

The best metrics, in line with the individual results, are Word2Vector and Unigram Jaccard Distance, with the overall best being Word2Vector (0.703). Using all the paraphrases we managed to obtain a better performance than by using only the original questions (0.703 vs. 0.682). However, while this result is better than the one obtained using the original questions and NMT paraphrases, it is worse than using only PPDB lexical or phrasal paraphrases. Perhaps if we had not included the NMT paraphrases, which had a worse accuracy than the original questions, we would have achieved a better result. However, we did not test this hypothesis. It is interesting to see than the best metric for the Small KB experiment, a setting where we classify many questions based on a single question per group, are Word Vectors. This means that the classification is not “bound” to the structure of the sentences.

2.7.3 Simulating a real case

Instead of having a big knowledge base, in a real case it is far more likely to have one question of each group in the knowledge base. As previously mentioned, in this experiment we select one question per group for our knowledge base, expand those questions using our Paraphrase Generation Module (NMT + PPDB generation) and classify the remaining questions. The results can be seen in Table 6.

If we compare these results with the ones using only the original questions (seen in Table 4), we see that the overall accuracy of the classification decreased but the best results for each case do not differ that much (0.681 vs. 0.657). This might mean that the quality of our generated paraphrases is not the best.

Table 6: Real Case Simulation Results

	Simulating a Real Case
Metrics	**Original + NMT + PPDB**
Lev-char	**0.657**
Lev-word	0.606
Jac-char	0.467
Jac-uni	0.618
Jac-bi	0.634
Jac-tri	0.619
W2V-sum	0.653

2.8. Effects of pre-processing

After evaluating our 3 experiments we decided to evaluate the effects of pre-processing on the classification as we believed it would yield better results. For instance, without using any pre-processing the occurrences "(Client Contribution)", "Client Contribution", "client contribution" and "(client contribution)" are all different due to differences in casing and characters due to the brackets. While this does not greatly affect the Character-based Levenshtein Distance, others are more affected. We experimented only with the "Small KB" method and using the original questions as well as the NMT and PPDB paraphrases. We set all letters to lower-case, separated symbols with spaces (e.g "that is," $\rightarrow$ "that is ," and "+/-" $\rightarrow$ "+ / -") and removed stop-words using the default stop-word list for the English language from Natural Language Toolkit (NLTK) [10].

The results, which can be seen in Table 7, are overall worse than without preprocessing but the best accuracy for both cases is comparable (0.703 without pre-processing and 0.700 with pre-processing). Nonetheless, the results with pre-processing and using the paraphrases are still better than without pre-processing and only the original questions (0.700 vs. 0.682), which, once again, shows that using the paraphrases is beneficial for the classification.

Table 7: Small KB Pre-Processing Vs. No Pre-Processing Results

	Original + NMT + PPDB	
Metrics	**Pre-Process**	**No Pre-process**
Lev-char	0.576	0.600
Lev-word	0.591	0.605
Jac-char	0.393	0.403
Jac-uni	**0.700**	0.691
Jac-bi	0.515	0.638
Jac-tri	0.366	0.521
W2V-sum	0.323	***0.703***

It is possible that removing the stop-words decreases the accuracy. Since we used the default stop-word list, the question-words (e.g. "what", "how", etc.) are removed. Thus, some questions are reduced to only one or two words. For instance the question "what is the maturity date ?" is transformed to "maturity date ?". Perhaps a better accuracy could be achieved without removing the stop-words or by choosing our own stop-words instead of using NLTK's. We did not, however, test this.

3. General Solution

This section presents our solution to build a Questions Answering system capable of solving BNP's problem. We begin by explaining the general architecture of our solution. After that, we propose and evaluate different setups for our solution. Next, we discuss how to enhance our solution by taking advantage of user feedback. Finally, we discuss how to apply our solution to new dashboards.

3.1. General Architecture

Our general architecture, is fairly simple.

We begin by taking as input a user question. Then, we generate paraphrases of that question using our Paraphrase Generation module, which we presented and evaluated in the previous section. The original question and the paraphrases are passed on to the Information Retrieval (IR) module, which returns the final answer to the user question.

Depending on the setup, which we will discuss in more detail in the following section, our IR module may use Elasticsearch[7], a pre-existing IR engine or perform retrieval using question/question similarity based on the metrics mentioned in the previous section. Furthermore, when using ElasticSearch we use as our knowledge source the documentation of the dashboard.

Since we only index and return paragraphs as the answer to the users' questions, we manually create a new documentation, by reorganizing the original documentation, so that the information we consider to be the answer to a question is all in the same paragraph. For instance, consider the question "Into how many steps is the workflow logic split?". While in the original the answer to this question would be split into different paragraphs, one for each step, in the new documentation we put all this information into the same paragraph. Notice that if the answer was contained in different paragraphs it would be impossible to give a correct answer. We use either the original documentation or the new documentation as the knowledge source, when using ElasticSearch, depending on the setup.

When we perform retrieval based on question/question similarity we use the dataset we manually created, extended with paraphrases, as our knowledge source. We classify the user question

[7] https://www.elastic.co/

given our knowledge source, as we did to evaluate our paraphrase methods in the previous section, and then retrieve the answer from a Question Group/Answer database that we manually created.

3.2. Proposed Setups

As mentioned in Section 2, we manually created a dataset consisting of 606 questions based on the documentation of a web dashboard, and manually grouped them according to their expected answer. Manually creating and grouping questions from text is a tedious work and involves allocating human resources that could be being used in a different task, perhaps one more relevant to BNP's business activities. As such, we present in this section different solutions, each requiring more human work than the one before. The solutions we present are:

- H_0: ElasticSearch + Original Documentation
- H_1: Paraphrase Generation + ElasticSearch + Original Documentation
- H_2: ElasticSearch + New Documentation
- H_3: Paraphrase Generation + ElasticSearch + New Documentation
- H_4: Paraphrase Generation + Retrieval Based on Question/Question Distance + Question/Answer Database

3.2.1 H_0: ElasticSearch + Original Documentation

This solution is our baseline. We index the original documentation by paragraph, using ElasticSearch and, when a question is asked, we perform a query on ElasticSearch and return the paragraph with the highest score.

3.2.2 H_1: Paraphrase Generation + ElasticSearch + Original Documentation

In this solution we index the original documentation as in the previous solution but instead of querying using only the original question we first perform a step of paraphrase generation. We query ElasticSearch with the original questions and paraphrases, with each query yielding a set of paragraphs and the corresponding scores. We sum the scores of the paragraphs individually and return the one with the highest score. An example would be a user asking a question, Q_1 and our system generating a paraphrase $Para_1$. We then query ElasticSearch with Q_1 and $Para_1$, which returns paragraphs P_1 and P_2 with scores 0.5 and 1.0, for Q_1, and paragraphs P_1 and P_3 with scores 1.0 and 0.8, for $Para_1$. Summing the scores for each paragraph returns 1.5 for P_1, 1.0 for P_2 and 0.8 for P_3. As such, we return P_1 as the answer to Q_1.

3.2.3 H_2: ElasticSearch + New Documentation

This solution is the same as H_0, but using the new documentation as the knowledge source.

3.2.4 H_3: Paraphrase Generation + ElasticSearch + New Documentation

This solution is the same as H_1, but using the new documentation as the knowledge source.

3.2.5 H_4: Paraphrase Generation + Retrieval Based on Question/Question Distance + Question Group/Answer Database

This solution corresponds to the Big Knowledge Base experimental case described in the evaluation section of Section 2 (Section 2.6). It uses as the Knowledge Base the question dataset we created manually (question + question group) and the paraphrases we generated using our Paraphrase Generation Module (Big KB setting). Additionally, we manually created a Question Group/Answer Database which we use to find the answer to the user question.

We use what we learned from our evaluation in the previous section and only use the paraphrase sets that gave us the best results, for the Big KB case (original + best lexical – "Case A 50-50" + best phrasal – "Case B 75-50" + NMT). Furthermore, we noticed that the best accuracy for this experiment was achieved using the Character-based Levenshtein Distance. As such, we discarded all other metrics and base our classification only on the Levenshtein Distance.

To retrieve the answer to a user question we first generate paraphrases of the input question. Then, we compute the distances between the original question and its paraphrases, and the questions in our Knowledge Base. In the following step, we sum the all the distances for the original question and respective paraphrases, and select the most similar question (i.e. smallest distance). We assume that the group of the user question is the same of the most similar question and use the Question Group/Answer Database to return the answer to the question.

3.3. Results and Discussion

In this section we present and discuss the results of the solutions shown in the previous section.

We manually evaluate solutions H_0 through H_3 using a sample of 20 questions from the dataset we created. For solutions where ElasticSearch is

used (H_0 – H_3) we check if the top ranked paragraph is the correct and if the correct answer is in the top 5 ranked paragraphs. For H_4 we consider the accuracy of this solution as the best accuracy of the Big Knowledge Base setting in Section 2. Since we assume that the Question Group/Answer Database we created is correct, evaluating this solution corresponds to evaluating the classification of the questions and paraphrases, which is what was evaluated in Section 2.

The results can be seen in Table 8.

Table 8: Solutions Results

	Accuracy	Top-5 Accuracy
H_0	0.65	0.80
H_1	0.65	0.80
H_2	0.70	0.90
H_3	0.70	0.90
H_4	0.82	No data

In order to evaluate the impact of paraphrasing in the retrieval process we must compare the results of H_0 and H_1, and the results of H_2 and H_3. From those results, we see that the accuracy is the same with or without paraphrasing. However, upon inspecting the number of generated paraphrases for each question we find that our system was unable to generate a single paraphrase for 8 of the 20 questions. Furthermore, for the questions that our H_0 and H_1 setups wrongly classified (7 questions out of 20), our system generated 1 paraphrase for 1 of the questions and 0 for the rest. Regarding the wrongly classified questions for the H_2 and H_3 setups (6 out of 20), our system generated 2 paraphrases for 1 question and 0 for the remaining questions. As such, it is hard to evaluate the impact of paraphrasing in the retrieval process.

It seems our system is capable of generating multiple paraphrases for a single question, but, in turn, it fails to generate paraphrases for many questions. This is perhaps due to the fact that we use domain-general rules, while our vocabulary is domain-specific.

Comparing the H_0 results with the H_2 results, we see that using our new documentation improved not only the accuracy but also the top-5 accuracy, which means that while the answer is not the paragraph with the highest score, it is contained in the top-5 paragraphs with the highest score.

Our best overall accuracy is given by the H_4 setup, which greatly outperforms the other setups. Notice that the accuracy achieved in section 2, which we also report as the accuracy of our H_4 solution in Table 8, is obtained considering that the test questions are not present in the Knowledge Base. In practice, in a FAQ setting, that is not always the case. Users tend to ask multiple times the same question.

We can conclude that paraphrases improve the accuracy when we use them to populate a Knowledge Base of questions. However, we can not take any conclusions from the impact of paraphrases on the accuracy of a retrieval engine since our system was unable to generate paraphrases for 8 questions from our sample of 20. Perhaps if we had used another sample of questions, or a bigger sample, we would have achieved different results.

3.4. Enhancing H_4 with User Feedback

In this section we propose a method of enhancing our H_4 setup by taking advantage of user feedback.

As previously mentioned, in this solution we first take a user question as input, generate paraphrases and find the closest question in our Knowledge Base to the original question and its paraphrases. Using the group of that question, the one in our KB, we retrieve the answer from our manually-created Question Group/Answer Database.

After showing the user the answer to their query we ask for feedback: “Was this answer helpful?”, a “yes” or “no” question.

If the user answers affirmatively, the user question and its paraphrases are added to the Knowledge Base, with the group associated with that answer in the Question Group/Answer Database.

On the other hand, if the user answers negatively, we show them the top-5 answers, corresponding to the top-5 closest question groups and ask the user if any of those answers is satisfactory. If so, we add the question and paraphrases to the KB, with the group associated with the answer chosen by the user. Otherwise, we ask the user to reformulate their question and try again. While we expect the results to improve with the addition of user feedback, we were unable to evaluate this.

3.5. Applying the setups to new dashboards

An important part of this work is the “plug and play” capability of the setups. Since BNP has many dashboards it is important to be able to apply the setups as automatically as possible, while retaining a good accuracy.

All in all we:

- Created a question dataset and grouped the questions
- Reorganized the documentation
- Created a Question Group/Answer Database (requires dataset and grouped questions)

However, not all setups require these steps.

If an effort is made to create a question dataset, a question group/answer database and reorganize

the documentation, then the best solution is H_4 since it had the best accuracy in our evaluation, which we also expect to increase with user feedback. If, on the other hand, only an effort to reorganize the documentation is made, then the best solution is H_3 or H_4, since we could not conclude anything regarding the usage of paraphrases in the process of retrieval. Finally, if no effort is made, the best solution is H_0 or H_1.

We recommend the final solution, H_4 for its accuracy and the ability to learn from user feedback. However, it is up to whoever wants to create a Question Answering System for a new dashboard to decide which of the solutions to apply, given the results of each solution and the work associated with it.

4. Conclusions

With this work we presented a Question Answering System capable of answering the questions of the dashboard's users – a FAQ problem. Since we lacked data, we created a PG module to automatically generate paraphrases from our data, and use them to populate a KB. Our final solution leverages the power of the automatically generated paraphrases and user feedback to greatly improve our baseline, and learn over time.

From our evaluation we concluded that using our generated paraphrases improved the accuracy of the system when used to populate the KB, but could not conclude anything when we applied the PG module to single questions, for querying. This was due to the fact that we were unable to generate a single paraphrase for 8 of the 20 questions we used on our manual evaluation.

In order to develop and evaluate our system we created a question dataset, based on the documentation of the dashboard, where questions were grouped based on their expected answer (i.e. questions with the same answer belong to the same group). Furthermore, since we return paragraphs as the answer to users' questions, we reorganized the original documentation so that information we considered was the answer to a possible user question, and was contained in multiple paragraphs, would be contained in a single paragraph. Finally, and to support our final solution, we created a Question Group/Answer Database.

Since BNP has many dashboards, it was important to develop a system that was able to be easily applied to a different setting. With that in mind, we proposed different setups based on the amount of work required to implement the system. While it is up to whoever wants to create a QAS for a new dashboard to choose which solution to use, we suggest our final solution, as it is the only one to take advantage of paraphrases and user feedback.

References

[1] Hugo Rodrigues, Luísa Coheur, and Eric Nyberg. Improving question generation with the teacher's implicit feedback, 2018.

[2] Yushi Wang, Jonathan Berant, and Percy Liang. Building a Semantic Parser Overnight. *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1332–1342, 2015.

[3] Juri Ganitkevitch, Benjamin Van Durme, and Chris Callison-Burch. PPDB : The Paraphrase Database. *Proceedings of NAACL-HLT*, (June):758–764, 2013.

[4] Li Dong, Jonathan Mallinson, Siva Reddy, and Mirella Lapata. Learning to Paraphrase for Question Answering. *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 875–886, 2017.

[5] Ashish Vaswani, Samy Bengio, Eugene Brevdo, Francois Chollet, Aidan N Gomez, Stephan Gouws, Llion Jones, Lukasz Kaiser, Nal Kalchbrenner, Niki Parmar, Ryan Sepassi, Noam Shazeer, and Jakob Uszkoreit. Tensor2Tensor for Neural Machine Translation. *CoRR*, abs1803.07, 2018.

[6] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention Is All You Need. 6 2017.

[7] Vladimir Iosifovich Levenshtein. Binary codes capable of correcting deletions, insertions and reversals. *Soviet Physics Doklady*, 10(8):707–710, 1966.

[8] Paul Jaccard. Nouvelles recherches sur la distribution florale. *Bull. Soc. Vaud. Sci. Nat.*, 44:223–270, 1908.

[9] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. Distributed Representations of Words and Phrases and their Compositionality. *CoRR*, abs/1310.4, 2013.

[10] Ewan Klein and Edward Loper. *Natural Language Processing with Python.* Number January 2009. O'Reilly Media, 2009.