Selecting FAQs as Responses to IT Service Tickets

Costanza Calzolari
Masters of Data Science
ETH Zürich
calzolac@ethz.ch

Margherita Rosnati
Masters of Data Science
ETH Zürich
mrosnati@ethz.ch

Brian Regan
Masters of Data Science
ETH Zürich
bregan@ethz.ch

Abstract

This project aims to automate the handling of incoming tickets to the help desk leveraging an internal FAQ list. Our solution, given an incoming ticket, either selects the most likely FAQ answers, or flags the lack of appropriate registered answer. The model consists of an unsupervised method and a supervised classifier which generate and learn a "question-to-FAQ" mapping. We obtain a 89.4% top three mean accuracy and 53.4% top three F1 score, thus unlocking the potential to improve the IT staff's efficiency in addressing tickets and identifying opportunities for new FAQ entries.

1 Introduction

This project is a collaboration with the ETH IT Services and seeks to tackle a help desk's common problem: the same questions are asked and answered over and over again, however there is no system to automate the answering to natural language questions. This problem results in employees performing repetitive and menial tasks, when they could be working on more impactful projects.

Given an incoming ticket, our solution either selects the most likely FAQ answers from an inhouse FAQ inventory, or flags the lack of appropriate registered answer; hence allowing the ticket staff to reply more efficiently and identify opportunities to update their internal FAQ list.

The data provided consist of a selection of uncleaned email based ticket conversations and a set of internal FAQs. For more details on the data structure see Section 3. Since the data doesn't include explicit labelling of a ticket-to-FAQ mapping, an unsupervised approach is required. To tackle the problem, we use a two stage approach. First, we create a mapping using existing tickets and FAQ by matching embeddings based on ticket answers and FAQ. Then we build a supervised classifier on the learned labels dataset. The details of the model are specified in Section 4. We report experiments with several methods of embedding and several methods used to alleviate some of the problems found with the classification task.

2 Related Work

Common methods to answer questions using Frequently Asked Questions leverage lexical similarity and ontology to determine the closest answer to a given question [1]. On the other hand, extensive research has developed around leveraging community based questions and answers to extract knowledge and automate question answering [8]. Methods developed by T. Hope et. al. [2] use mixtures of semantic and structural similarities to find a latent representation within the dataset question and answer items. Another interesting finding uncovered by J. Jeon [3] include community questions and answers structure: when looking at similarity between question and answer sets, similarity calculated over answers scores better than similarity calculated over questions.

However, virtually no work has been carried out on utilizing both conversational questions and answers and Frequently Asked Questions datasets.

32nd Conference on Neural Information Processing Systems (NeurIPS 2018), Montréal, Canada.

3 Data

The data provided comes from two different sources: a database of ticket conversations and a database of FAQ questions and answers. Significant cleaning was required for both datasets and specific details are outlined below.

3.1 Tickets Dataset

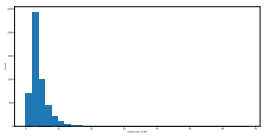


Figure 1: Histogram of article count per ticket

The ticket dataset come in the form of a several tables database, most of which are not relevant to the project context. The structure of tickets is different to that of FAQs in that they are *conversational*. A ticket is composed of several *articles* which are messages for the service desk agent, the user or system messages issued when events such as ticket-transfers occur.

The initial dataset is composed of 36,221 articles, which make up 5,147 tickets. We first drop all explicitly tagged system articles, of which there were 4,894, from tickets and retain only those issued from the user or agent. We finally remove articles which are system messages but are not explicitly tagged as so. These are identified by subject lines such as *"Close!"*, *"Note!"*, *"Responsible Update!"*, *"Priority Update!"* etc. These commonly occurring subject lines are found manually by looking at the most common subject lines and 12,890 articles are removed as a result, meaning 18,437 articles remain which make up 5,035 tickets. The median amount of articles per ticket is 3 and a histogram of counts can be seen in Figure 1.

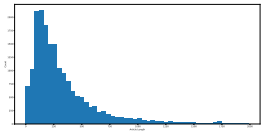


Figure 2: Histogram of article lengths after cleaning

Next we clean individual articles. The median length of an article is 1,129 characters. The article body is unstructured email content and thus contains many artifacts of such a format that need to be removed. The first commonly occurring feature of these messages is the appending of all previous articles within the ticket to the bottom of an article. We decide, given observations of a large amount of tickets, to clip all tickets after the tokens: "–", "__", "**", " From:", "From:", and "Von:"[1]. After this procedure we also remove all articles of length above 2000 characters which resulted in a loss of only 149 articles and retained 18,288 articles (99.19%). A histogram of the article lengths post-cleaning can be seen in Figure 2.

The articles, originally both in German and English, are then translated using the Google Translate API[2]. See Section 3.3 below for details on this process.

Finally, in order for our ticket data to be comparable to FAQ data it is necessary to convert from the conversational nature of the ticket data to a question-answer format. We evaluate (on the entire pipeline) several methods including keeping only the first article from the user as a question and the first article from the IT service agent as an answer but settled on concatenating all user articles as a question and all agent articles as an answer, as the preliminary training results looked more promising.

We also manually remove some tickets which on manual inspection have an empty question or answer section or are otherwise corrupted. 200 tickets are also removed for testing and validation (see Section 5.1) In conclusion, we pass 4,007 tickets to the final model for training.

3.2 FAQ Dataset

The FAQ dataset is contained in a relational database under a mixture of raw text and HTML. In the initial dataset there are 382 questions and 406 answers related by a correspondence unique id. The questions are single line sentences, usually not longer than a few words; whereas the answers carry the majority of the information. All answers have a title, where sometimes the title is equal to the

[1] We don't claim these tokens account for all article endings but they were sufficient for a large amount of observations

[2] `https://console.cloud.google.com/apis/api/translate.googleapis.com/overview`

corresponding question. Some answers are also directly taken from tickets more as a template for future conversations than as a ready-made answer to use directly.

We take the inner join of the two tables. We extract the raw text from the answer articles and concatenate the answer titles to the remaining corpus, in order to keep the formatting consistent with the tickets. FAQs, like tickets, are then translated using the Google Translate API. Further details can be found in Section 3.3 below.

Finally, we manually inspect all FAQs and remove duplicates (some FAQs were present in both languages and therefore duplicated on translation). We remove all FAQs with near-empty or *"junk"* content which may have appeared as a result of accidental entry of text as an FAQ. In conclusion, we pass 199 FAQs to the final model.

3.3 Translation

Both the ticket and FAQ dataset contained a mix of both German and English language. Since throwing away data in either of these languages would've lead to a significant loss of information, we opt to translate all tickets and FAQs to English using the Google Translate API. FAQs contain an explicit language tag which mean only German tickets are translated. Tickets, however, have no such tag and so if the *detected language* returned by the API is German the English translation is kept. If the API detects English, the original is left untouched. Tickets with other detected languages are dropped (99.71% of the ticket data is retained).

4 Model

The task is to create a mapping from a ticket question to an FAQ answer in the absence of labelled matches. Intuitively, one approach would be to look for similarities between the question posed by the ticket and the question posed by the FAQ. However, there are two issues with this approach. First, as in [3], we observe that tickets answers and FAQs are often written by the same group of people and thus retain similar syntax and structure while questions are often much more heterogeneous. Second, the FAQ questions are much less informative and are lacking both in length and sufficient semantic content. Thus, our model first matches all tickets in the training set to an existing FAQ based on the content of their answers; then with the new learned labels dataset we build a classifier from ticket to FAQ index based on the content of the ticket question. We discuss the specifics of the model in the sections below. Figure 3 shows a schematic of our pipeline.

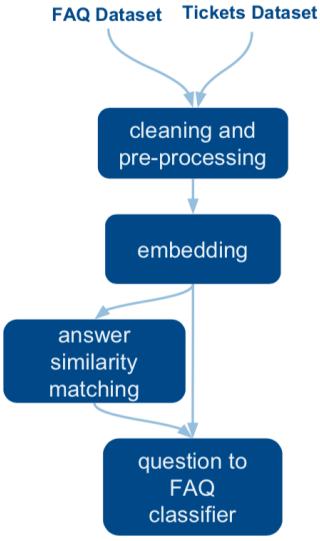


Figure 3: Pipeline

4.1 Pre-processing

After cleaning the data and translating all the tickets and FAQ to English we proceed with the pre-processing. The pre-processing steps applied depend on the embedding we use later on. When using Tf-idf embedding (see Section 4.2), the pre-processing already present in the Scikit Learn function `Tf-idfVectorizer` is used. In particular all accents are removed, all words are set to lower case and afterwords tokenized. No stopwords are removed as Tf-idf should recognize such words as very frequent in all documents and therefore assign them a low score. A personalized pre-processing is instead used when implementing Word2Vec and Doc2Vec embeddings (Section 4.2). In particular, text is stripped of numeric elements, white spaces and punctuation. For what concerns IP addresses and email addresses, they are substituted with the tags `<IP>` and `<EMAIL>` respectively. Initially, urls were also substituted with tags, but we decide to keep the raw version since we found it to carry information in some cases. Stopwords are removed, in particular words typically found in emails e.g. *"Dear", "Sincerely",...* The text is afterward tokenized and stemmed.

4.2 Embedding

In order to calculate similarity between two bodies of text (or documents) it is first necessary to create a vector which summarizes the semantic content of the document. We explore several options, and mixtures for such an embedding: Term Frequency–Inverse Document Frequency (Tf-idf), Word2Vec [7] and Doc2Vec [4].

4.2.1 Tf–idf

Tf-idf is a frequency based model to calculate the importance of words within a document. We calculate each term frequency within the document and normalize such count by the number of documents containing such term:

$$\text{Tf-idf}(t,d) \;=\; \text{tf}(t,d) \times \left(\log \frac{n_d + 1}{\text{df}(t,d) + 1} + 1 \right)$$

Where $\text{tf}(t,d)$ is the term frequency, n_d is the number of documents and $\text{df}(t,d)$ is the count of documents containing term t.

The model is hence dependent on the document corpus given. As such, we train a combination of different embeddings and evaluate their performance. To perform the unsupervised mapping between ticket answers and FAQ answers, the obvious corpus to consider is the ticket answers and FAQ answers alone, so to eliminate any noise from the user written questions. On the other hand, for the classification task embedding all documents together - ticket question, answer, FAQ question, answer - is the most intuitive solution in order to capture the importance of both words often used in the answers and ones used to answer queries. After training embeddings on different combinations of the four datasets we reach the conclusion that the marginal difference is negligible, hence we settle for the aforementioned procedure, that is embed on tickets and FAQs separately for the unsupervised matching and together for the classification task.

4.2.2 Word2Vec Based Models

Word2Vec, first described in [7], can be trained using either Continuous Bag of Words (CBOW) or skip-gram methods. The key difference is that the first is agnostic to word-order, whereas the latter takes order into consideration. According to the authors notes [6] CBOW is faster while skip-gram is, quoting, "slower, better for infrequent words". For all implementations of Word2Vec in this work we use CBOW with 5 epochs of training and a word-vector size of 128. As with Tf-idf, we try different combinations of embeddings and settle to train Word2Vec embeddings on the full corpus. We implement two main methods which use the generated word embeddings to create a document embedding.

Mean Word2Vec (mWord2Vec) In order to create a document representation we average out all the word embeddings over each word contained in a document.

Tf-idf Weighted Word2Vec Document embedding is created in the same way as in ***Mean Word2Vec***, but this average is combined with Tf-idf values so *"important"* words for a document are emphasized. Different combinations were studied:

- Detect the 5 most important words in a document i.e. 5 highest Tf-idf scores and average their word embedding representation (named mWord2Vec_top5)
- Detect the 5 most important words in a document i.e. 5 highest Tf-idf scores and average their word embedding representation using Tf-idf scores as weights (named wWord2Vec_top5)
- Average out all the word embeddings for each word contained in a document, weighting each word by its Tf-idf score (named wWord2Vec)

4.2.3 Doc2Vec

Finally we experiment with paragraph vectors [4]. We use the Distributed Memory (DM) training method (which can be seen as a "document version" of CBOW word embeddings) rather than Distributed Bag of Words (DBOW), again with an embedding size of 128. DM also takes into

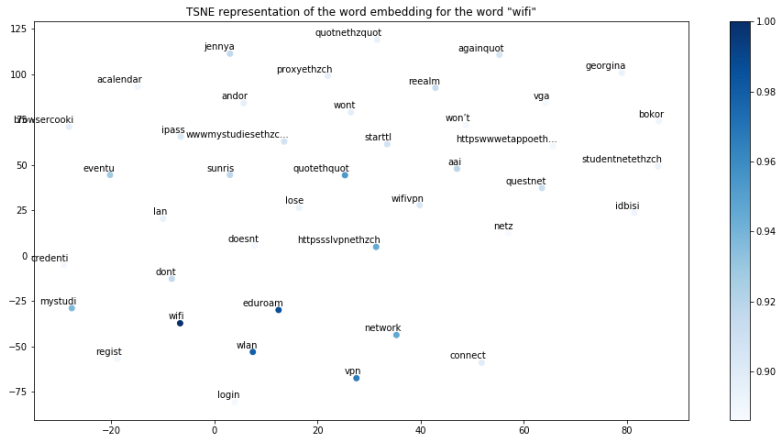


Figure 4: Word2Vec representation of the word "wifi" using TSNE [5]. The heat-map represents the similarity score between "wifi" and the 40 most similar words. As a validation of the embedding we expect darker dots to be closer to "wifi"

account word order (or rather which words are "close" to others). However, Doc2Vec methods are often trained on datasets with substantially more samples. For example, [4] trains on 75,000 IMdB movie reviews. Given our dataset contains < 5,000 documents Doc2Vec doesn't preform as well as it potentially could. We experimented with lowering the embedding vector size but there was no significant increase in performance. In the case of Doc2Vec we train the similarity embedding on ticket and FAQ answers, whereas we train the classifier embedding over ticket questions.

4.3 Similarity Matching

Given a set of of ticket answer embeddings and FAQ answer embeddings, we compute the similarity matching in the following way: we allocate a ticket to the FAQ with the maximal cosine similarity between their embeddings. As discussed in section 5.1, approximately 75% of the tickets are not relevant to a specific FAQ, and hence should not be assigned to one. To deal with these tickets we introduce a *"non-FAQ"* class. For each ticket to FAQ mapping, if the similarity between the two is lower than a set threshold, we assign the ticket to the "non-FAQ" class. The threshold mirrors the labelled empirical findings and is set at the 75th percentile of the training tickets.

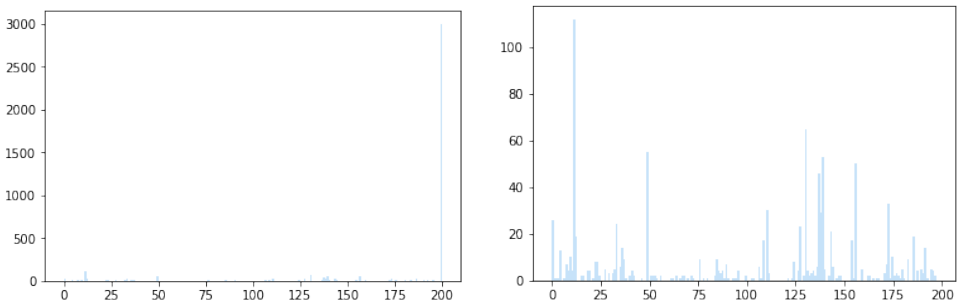
When similarity matching is complete we pass a FAQ label (including a label for *"non-FAQ"*) to the classifier.

4.4 Classification

The classification task takes the learned labels dataset generated by the Similarity Matching (see Section 4.3) and the ticket questions embeddings and aims to predict the labels. It is important to note therefore, that there are two errors associated with this classifier:

- The error of the classifier relative to the learned labels dataset. That is, given the ticket-FAQ labels how well does the classifier preform.
- The inherited error of previous stages of the pipeline. A classifier can have 0 errors with respect to the labelled dataset. However, if this dataset was ill formed, then it will have high error with respect to any labelled ticket-faq data.

The methods to measure the latter error are explained in Section 5.1, whereas we use cross validation to evaluate a classifier assuming the learned labels dataset is correct.



(a) Incl. "non-FAQ" class (cl. 200) (b) Excl. "non-FAQ" class

Figure 5: Tf-idf FAQ mapping class frequency

4.4.1 Random Forest Classifier

To perform the classification task a Random Forest classifier with 100 trees is implemented. Two main challenges arise during this step. First of all, as we can see in Figure 5 the classes are unbalanced. In particular, as discussed in the section 5.1, the "non-FAQ" class contains about 75% of the tickets (see Fig. 5a: the bar representing the number of tickets mapped to "non-FAQ" significantly outweights all other bars). Three methods have been explored to address this problem: one is introducing a Random Forest pre-classifier for "non-FAQ" tickets, the second is an initial threshold using embedding similarity and the third is balancing classes (Section 4.4.2). The second challenge consists of often having only one or no ticket associated to the FAQs (Section 4.4.3).

4.4.2 "non-FAQ" Pre-Classification

Random Forest To deal with the problem of the oversized "non-FAQ" class, we experiment with different pre-classifiers with the purpose of early detecting the tickets not associated with any FAQ, hence a binary classifier. Our intuition is that doing so, a second classifier solely working on "FAQ" tickets would be able to identify more fine-grained characteristics and therefore improve the performance. First we consider an additional Random Forest with the objective to discriminate between "FAQ" and "non-FAQ" tickets, the ones found to be related to an FAQ are then passed to the Random Forest described in Section 4.4.1.

Initial threshold We also try to separate "FAQ" and "non-FAQ" tickets using an initial threshold on the similarity measure between the ticket questions and FAQ answers. All the tickets whose distance to the closest FAQ is greater than the threshold are automatically assigned to the "non-FAQ" class. The remaining tickets are then passed to the classifier described in Section 4.4.1. Notice that this differs from the similarity measure described in 4.3: first, the matching done previously serves the only purpose of facilitating the unsupervised tagging and as such is a "one-off" process, whereas the initial threshold similarity would be run for every new ticket. Secondly, the initial threshold is designed to work on new tickets and as such is calculated using ticket questions, whereas the the matching similarity uses ticket answers.

Balanced Classifier Another strategy to tackle the problem is to balance out classes in the main classifier. To do so, we exploit the built-in Scikit learn functionality of the `RandomForestClassifier` that allows to set classes weights to be inversely proportional to the class frequency. The weights are computed based on bootstrap samples. By doing so we hoped that the classifier would learn equally well to distinguish between "non-FAQ" and "FAQ" tickets and to assign each "FAQ" ticket to the correct one. Additionally, balancing classes would help with other FAQ classes.

4.4.3 One-shot Classification

To tackle the one-shot classification problem, we build a probability distribution over the 5 most probable FAQ classes according to the similarity measure described in 4.3. We then train the main random forest classifier on an expanded version of the tickets, where each ticket is assigned to a given

FAQ with a weight given by the probability aforementioned. Doing so, the likelihood of an FAQ class appearing only once drops drastically.

However, during preliminary results we found that this latter method performs poorly and decreases the speed of the full process, hence we do not pursue it further.

5 Experimental Framework

5.1 Labelled Data

In order to evaluate the final model we tag manually 300 tickets with the relevant FAQ, approximately 7.5% of the total tickets dataset. It is important to notice here that we empirically estimate 75% of all tickets to be of "non-FAQ" class. We divide the 300 tickets in three pools: training, validation and testing.

5.2 Metrics

We develop different metrics for the different parts of the model as well as overall metrics to evaluate the performance. We focus on metrics on the classifier and on the overall model, implying the performance of the learned labelling.

Overall Model To evaluate the performance of the overall model, we look at three metrics: the mean class accuracy, mean class recall and the corresponding F1 score. The mean class accuracy, and similarly for the recall, is calculated by determining the accuracy of each represented class and taking their weighted average with regards to the class occurrence. As reported in section 4.4, we use probability prediction over classes. A true positive is determined by looking at the top three highest probable outputs of the classifier.

Similarity We evaluate the similarity learned labels based on a qualitative review of the similarity strength, together with sampling manually ticket mappings of different strength. We do so by plotting a histogram of the similarity strength for all tickets and looking at the distribution skew.

In particular, given the large "non-FAQ" class, we look at the distribution of the tagged training set to determine if the thresholding mechanism explained in 4.3 separates well the "non-FAQ" class.

Classification The metrics used to evaluate the classification and the totality of the model are the same. The difference lies in which dataset is used as ground truth. When only looking at classification in isolation, we use the learned labelling as reference and proceed with cross validation, as briefly mentioned in Section 4.4.

6 Results

6.1 Similarity

When looking at the similarity distributions with the highest achieving FAQ per ticket (Figure 6) as described in section 4.3 we notice that the similarity is skewed towards higher values for semantic embeddings and lower values for the frequency embedding.

However this shouldn't be interpreted as a better performance of the model. Given the procedure, we force every ticket to be assigned to an FAQ regardless of whether they would naturally relate to one. However, the highly heterogeneous nature of the ticket dataset means that often no FAQ is a full match for the problem discussed, hence the similarity scores should be distributed over all values between zero and one. This hypothesis is confirmed by the validation data plotted in Figure 6: embeddings outputting high similarity scores are worse separators of the "non-FAQ" class. Indeed we find that Tf-idf is the best separator of classes and provides a more intuitive distribution of similarities.

Tf-idf Similarity Example	
Strong similarity (57%)	Medium similarity (31%)
Ticket answer Dear xxx Could you tell us your number? It should be printed on a blue sign with the inscription “Betriebinformatik” or “ETH Zurich ID Services Delivery”. Your ID team Rudolf xxx. Dear xxx, A support ticket has been opened for your request. We will process your request as soon as possible. If you have any further questions, please answer directly to this e-mail.	**Ticket answer** Dear xxx I see you have already been to ETH Zurich. Therefore, the old password will still be valid on [1] www.passwort.ethz.ch. If you do not know this, we would have to send you a new one. Your ID Team Joel xxx Dear xxx Unfortunately, we can only regenerate the password. In this case we can either send it by mail or you can pick it up from us. Your ID Team Joel xxx
FAQ answer ZO computer number Could you tell us your number? It should be printed on a blue sign with the inscription “Betriebinformatik” or “ETH Zurich ID Services Delivery”.	**FAQ answer** Password forgotten Unfortunately we are not able to send your new password by email or by phone. You can pick up your new password at ...

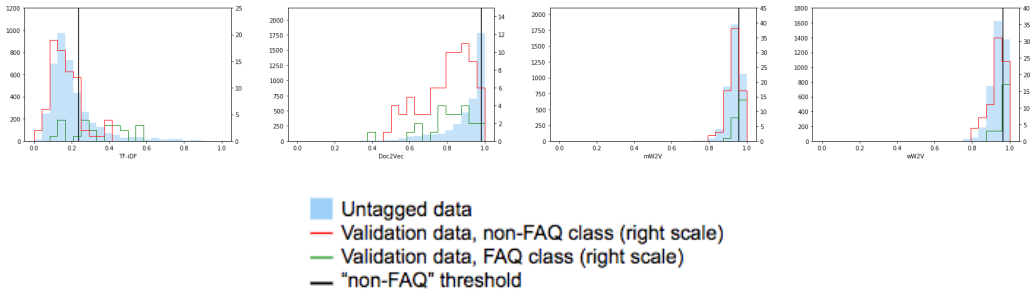


Figure 6: Similarity distributions

6.2 Classifier

When looking at the classifier in isolation, we do so through cross validation. The results in table 1 show that semantic based embeddings, especially mWord2Vec, perform better at classifying ticket questions given the learned mapping.

For the results regarding the addition of a random forest pre-classifier and the balanced classes classifier, please refer to the next section, whereas the thresholding pre-classifier is evaluated equally to the similarity mapping. As shown in Figure 7, initial results on Tf-idf as a pre-classification technique are not promising: the two validation classes are not separated by the threshold line. We hence dropped this approach before testing it with other embeddings.

Model	**Mean Accuracy (%)**	**Mean Recall (%)**	**F1 Score**
Doc2Vec	**98.6**	38.5	55.4
mWord2Vec	98.2	**39.6**	**56.4**
Tf-idf	95.6	33.1	49.2
mWord2Vec_top5	97.3	37.8	54.4
wWord2Vec_top5	97.1	37.1	53.7
wWord2Vec	97.9	35.8	52.4

Table 1: Classifier Cross Validation Results

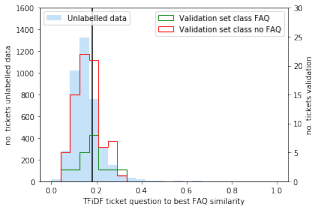


Figure 7: Pre-classifier similarity

Result Example	
Question	I have received an anomalous email. What should I do?
1st FAQ suggestion *0.5 prob.*	**No FAQ in Dataset** Please answer this manually
2nd FAQ suggestion *0.2 prob.*	**Phishing Mail warning** This is a phishing mail. You should delete it. If you clicked on the link and filled in your credentials, it is absolutely necessary to change the password immediately.
3rd FAQ suggestion *0.2 prob.*	**phishing mail** Thank you for the forward of the suspicious message. We will take care of it.

6.3 Overall model

However, once we look at the overall performance, the frequency based model performs better. In particular, when using a frequency based learned mapping and a semantic based embedding (Doc2Vec) for classification, we notice similar performance than when using a frequency based embedding (Tf-idf). We conclude that the key driver of the Tf-idf model performance is the similarity mapping (see figure 6 - Tf-idf). Interestingly, we also find that balanced classifiers perform better than most other combinations experimented with, but still worse than the full Tf-idf pipeline.

Model	Mean Accuracy (%)	Mean Recall (%)	F1 Score (%)
Baseline			
All as "non-FAQ"	72.0	36.0	48.0
RF Pre-classifier based models			
Tf-idf	66.6	2.1	4.1
Doc2Vec	63.1	2.5	4.8
mWord2Vec	74.2	2.8	5.4
wWord2Vec	71.3	2.7	5.2
No pre-classifier balanced models			
Tf-idf	88.5	37.8	53.0
Doc2Vec	72.6	36.0	48.1
mWord2Vec	82.9	36.9	50.6
wWord2Vec	88.9	36.9	52.2
mWord2Vec_top5	87.3	36.5	51.5
wWord2Vec_top5	83.3	36.3	50.6
No pre-classifier unbalanced models			
Doc2Vec	71.1	36.1	47.8
mWord2Vec	83.4	36.4	50.7
Tf-idf	**89.4**	38.1	**53.4**
wWord2Vec	84.0	36.9	51.3
mWord2Vec_top5	83.8	36.5	50.9
wWord2Vec_top5	79.3	36.3	49.8
Tf-idf mapping with Doc2Vec classifier	88.3	**38.2**	53.3

Table 2: Model Test Results

7 Conclusion and Further Work

7.1 Conclusion

In conclusion, we tackled the problem of automating a selection proposal of relevant FAQs given an incoming ticket. The method we propose uses frequency based word embeddings and relies on the finding that similarity between answers is stronger than similarity between questions. Given the training set, we first learn a mapping from the ticket answers to the FAQs in an unsupervised way, then use the learned mapping to train a classifier on ticket questions. Doing so, we can suggest three most likely FAQ answers to a new incoming question with mean accuracy of 89.4% and mean recall of 38.1%, improving our baseline by 17.4 and 2.1 percentage points respectively.

7.2 Further Work

A natural next step would be to dive deeper into the large non-FAQ class: in particular we could cluster similarly answered questions propose new FAQs automatically. Conversely, we could suggest the removal of FAQs no longer in use with the final goal to make a self updating FAQ list.

Regarding the work achieved, further work would consist in scaling the model on the full IT Desk dataset, estimated to be around 300,000 tickets. Given the large scale we could also attempt using more data-intensive methods within our pipeline such as recurrent neural networks.

Finally, towards the goal of full automation, a next step would be to automatically detect tickets requiring an action to be taken from an employee from the purely informative ones.

References

[1] R. D. Burke, K. J. Hammond, V. Kulyukin, S. L. Lytinen, N. Tomuro, and S. Schoenberg. Question answering from frequently asked question files: Experiences with the faq finder system. *AI magazine*, 18(2):57, 1997.

[2] T. Hope, J. Chan, A. Kittur, and D. Shahaf. Accelerating innovation through analogy mining. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 235–243. ACM, 2017.

[3] J. Jeon, W. B. Croft, and J. H. Lee. Finding similar questions in large question and answer archives. In *Proceedings of the 14th ACM international conference on Information and knowledge management*, pages 84–90. ACM, 2005.

[4] Q. Le and T. Mikolov. Distributed representations of sentences and documents. In *International Conference on Machine Learning*, pages 1188–1196, 2014.

[5] L. v. d. Maaten and G. Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9(Nov):2579–2605, 2008.

[6] T. Mikolov. Google Code Archive word2vec. `https://code.google.com/archive/p/word2vec/`. Accessed: 2018-12-17.

[7] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119, 2013.

[8] X. Xue, J. Jeon, and W. B. Croft. Retrieval models for question and answer archives. In *Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval*, pages 475–482. ACM, 2008.