Finding comparatively important concepts between texts

Renaud Lécœuche*†
renaudl@dai.ed.ac.uk

Abstract

Finding important concepts is a common task in requirements engineering. For example, it is needed when building models of a domain or organising requirements documents. Since a lot of information is available in textual form, methods to identify important concepts from texts are potentially useful. Traditional methods for finding important concepts from texts rely on the assumption that the most frequent concepts are the most important. In this paper, we present an approach that does not depend on this assumption. It makes use of two texts to find important concepts comparatively. We show that this approach is viable. It discovers concepts similar to those found by traditional approaches as well as concepts that are not frequent. Finally, we discuss the possibility of extending this work to requirements classification.

Keywords: requirements extraction, concept discovery, text analysis

1 Introduction

The aim of this paper is to present a method for finding important concepts in a text by comparing it to others. The task of finding important concepts in a text is an old one for requirements engineering. It still remains an important issue. For example, finding important concepts is needed when building models of a domain or when organising requirements documents. Numerous methods have been proposed to perform this task. These methods can roughly be divided into two groups: automated ones and semi-automated ones. The aim of the automated methods is to extract important concepts, often in the form of entity-relationship diagrams, without human intervention. These methods are usually based on the recognition of certain patterns in the text, which are then mapped to a part of the diagrams [2, 1, 10]. However, since natural language is ambiguous, it is not clear whether the diagrams built are correct. Moreover, given the fact that not all the requirements documents are in natural language, it is doubtful that these methods will extract all the important concepts needed to build complete diagrams [14]. The semi-automated approaches recognise these limitations. Their aim is to suggest a number of important concepts to the requirements engineer. The engineer then reviews the suggestions, integrates them with important concepts found from other sources and builds the relevant diagrams [6]. Semi-automated methods are useful because they can go through a lot of texts and point out important concepts, thus reducing the risk of information overload on the requirements engineer. These approaches are often based on a simple assumption: the more often a concept appears in a text, the more important it is. The problem is then to map from words to concepts and count those. Different mapping strategies exist, from direct words to concepts mapping to more complex sentence analyses. However, it is not clear whether the basic assumption really holds. Common concepts may not be the most relevant ones. In our approach, we do not rely on this assumption. Rather than searching for important concepts per se, we search for them compared to another text, called the baseline text. In other words, we consider important concepts to be those that best differentiate between the two texts. The fact that the important concepts are found with respect to the baseline text point of view seems natural. Indeed, it is unlikely that we would analyse a text always in the same way irrespective of what we want to do with the analysis, as is the case with methods that simply count concepts appearing in the text. This approach has two advantages: (1) it can find important concepts that are not frequent in a text; (2) it can replicate some of the results of traditional semi-automated methods while usually requiring less information. We show these two advantages on examples later in the paper.

Important problems remain. Notably, the mapping from words to concepts is not resolved. We chose to keep a rather straightforward mapping from words to concepts by using word root n-grams. Although this mapping is certainly not satisfactory in general, it seems to perform well on the examples we have tried. We discuss potential ways of improving on this simple syntactic approach in sections 3.2 and 5.
Many other methods exist for requirements extraction and analysis. Some of them are based on very different approaches such as modelling goals and scenarios [7, 13] and modelling goals, actors and constraints [3]. Our proposal is much simpler. Its scope is to serve as a tool that may point out concepts of interest to the requirements engineer. Our method should not be taken for a silver bullet. It is intended as a simple addition to the requirements engineer’s toolbox for text analysis. The requirements engineer retains a very important role, from selecting the baseline text, to setting the parameters of the tool and interpreting the results. Only s/he can decide whether the concepts highlighted by the approach are really relevant.

The paper is organised as follows. In section 2, we describe our method. In section 3, we show how our method works on two examples. Our approach offers the prospect of further improvements in text analysis such as requirements classification. We briefly discuss this possibility for future research in section 4. Finally, we conclude in section 5. We point out at several places in the paper limitations with our handling of natural texts. These limitations are quite often found in other tools as well. By clearly identifying them and offering some initial solutions, we hope to suggest some potential improvements.

2 Finding comparatively important concepts

In this section we first present the general approach used in our method. We then describe in more detail the different phases of the method.

2.1 Overview

Our approach is based on work done for information extraction. In particular, we reuse some ideas developed for the Message Understanding Conference (MUC) experiments and text classification. The aim of MUC is to compare approaches that automatically extract information, such as the date and persons involved in terrorist attacks or the name of companies involved in mergers, from texts, such as newspaper articles. The methods used for this task usually try to match templates to the text. If a template matches the text, it defines part of the information to extract. For example, the template “murder of <np>” can be used to extract the names of victims in a terrorist attack. One problem is then to select an efficient set of templates, extracting as many relevant pieces of information as possible. The first thing to do during this selection is to get rid of spurious templates, i.e., templates extracting information that is not relevant. This is done by computing the “precision” of templates. The precision is defined as the proportion of times relevant information was extracted compared to the number of times information (relevant or irrelevant) was extracted. The proportion tends to 1 if the template is relevant, and to 0 otherwise. A second problem is to get templates that can be applied very often, since having a relevant template that applies only in very few cases is less useful than a relevant template that can be applied often. This is done by computing the “recall” rate, which is the number of times a template gives relevant information compared to the number of times relevant information should have been extracted. By playing with two thresholds, one to ignore irrelevant templates and one to ignore infrequent templates, the information extraction designer can select a set of templates relatively easily.

However, in many cases the precision and recall rates cannot be computed because we do not have a corpus specifying which pieces of information should have been extracted. Work has been done to evaluate the quality of templates without a corpus by approximating these measures [11]. In particular, the relevancy measure discussed later in the paper can be used, in addition to counting the number of times a template can be applied. The same measures can also be used for finding text classification templates [12].

We propose to use a similar selection mechanism in our approach. That is to say, we propose to consider not only concepts that appear frequently but also those that have a high estimated precision. Our approach is different from the pieces of work developed for MUC in that we apply it to find important concepts rather than extract information or classify texts. We also have a phase of machine learning where the importance of the concepts is judged. Finally, we expect the requirements engineer to interact with the method. In particular, the requirements engineer may want to try different settings of parameters depending on the amount of important concepts he/she wants to retrieve.

Our approach consists of four stages, namely:

1. cleaning up of the texts;
2. identification of concepts;
3. extraction of potentially important concepts;
4. evaluation of the concept importance.

In these steps, two texts are used. One text is to be analysed for important concepts while the second text is used as a baseline text. We now describe the steps in more detail.

2.2 Cleaning up of the texts

The first step in the approach consists of getting rid of pieces of data that are not useful. There are two kinds of such pieces of data:

- Pieces of data that cannot differentiate texts. Although these pieces of data would be eliminated in step 2
whose role is to extract potentially important concepts, we can speed up the process by eliminating pieces of data that are very unlikely to differentiate texts. For example, determiners such as “a” or “the” are unlikely to differentiate texts. In order to clean up the texts, we use a part-of-speech tagger\(^1\), which is a tool associating a word with its syntactic category, to detect determiners. We then remove them from the texts. (We do not recommend to always ignore determiners when analysing texts. They may play an important role in the interpretation process. However in the examples we have tried with our method, they can be removed without changing the results.)

- Pieces of data that could differentiate texts but are not interesting. For example, if we want to know the concepts that differentiate an interview from a requirements document, e.g., to know which concepts may have been spoken about in the interview but not reported in the document, it is likely that we will retrieve the names of the interviewer and interviewee. Indeed, it is likely that these names are mentioned in the interview but very unlikely that they are in the requirements document. However, these names are not important concepts. We remove them by again using a part-of-speech tagger and removing proper names. The kind of data to remove depends on the type of texts to compare.

2.3 Identification of concepts

The next step consists of identifying concepts. This phase is particularly difficult. The same concept may appear in different forms because of, for example, synonyms. The same word can also refer to a different concept depending on its context in the text. As mentioned in the introduction, the approach we take here is very simple. It works reasonably well on the examples we tried. We identify concepts with n-grams in the text. A n-gram is a sequence of n words which appear consecutively in a text. However, rather than just collecting word n-grams, we compute word root n-grams. A problem with natural language is that the same concept may appear in different forms. For example, the notion of operation can appear as the common noun “operation” (singular or plural), the verb “operate” (and its conjugated forms), the adjective “operational” and so on. In order to judge the importance of a concept, it is necessary to normalise these different forms. In this example, every word should be changed to the base form “operate” in order to be evaluated. We do this by using a part-of-speech tagger as well as a stemmer. The stemmer uses the part of

\(^1\)We use a part-of-speech tagger available at http://www.ims.uni-stuttgart.de/Tools/DecisionTreeTagger.html

speech information in order to reduce words to their base form. For example, the “operational” adjective would be first transformed to the “operation” noun that would then be transformed to the “operate” verb. Other approaches have been proposed to deal with this problem. For example, it has been suggested to compare words at the character level [6]. In the previous example, the common part would be “operate”. However, such an approach introduces a lot of noise, such as words with common parts but which are not related, and still fails to identify some connections between words, such as, for example, “went” and “go”. We also consider n-grams where a part is replaced by wild cards. This can make their common structure more apparent. For example, the n-gram “call operator in” and “call him in” can both be replaced by “call * in”. Despite its shortcomings, we chose this mapping partly because it does not require any semantic knowledge. This knowledge is unlikely to be available for many domains and would require major work to build.

2.4 Extraction of potentially important concepts

We use two techniques to extract potentially important concepts as described in section 2. First, infrequent concepts are pruned. It is unlikely that a concept which appears, for example, only once in a text will be important. The requirements engineer can therefore set a threshold on the number of times a concept appears to be considered for further analysis. If a concept does not reach that threshold, it is eliminated. Of course, the requirements engineer may choose to set this threshold to 0. In that case no concept is ignored (see section 3). However, it is usually not sufficient to know which concepts are present in a text to determine which ones are important. Even the most frequent ones may not be the most important ones. This is where we use two texts to make a comparative study of the concepts in the texts. Concepts that can differentiate between the two texts are usually very important because they are salient compared to the baseline text. In this step, we make a first evaluation of how good concepts are to differentiate texts. This is done by calculating the frequency with which a concept appears in the text under analysis compared to the baseline text. That is to say, we compute \( \frac{F_2}{F_1} + \frac{F_1}{F_2} \) where \( F_a \) is the frequency with which the concept appears in the text under analysis and \( F_b \) is the frequency with which it appears in the baseline text. This “relevancy” ratio varies from 0 if the concept does not appear in text but appears in the baseline up to 1 if it appears in the text but not the baseline. If the ratio is high, then the concept may be important. The requirements engineer can then put a threshold on the ratio. Concepts that do not reach this threshold are not considered further. Note that a low ratio could also indicate an important concept. In our approach we ignore it. It
would be analysed if we reversed the role of the text under consideration and the baseline.

Obviously, there is a question of how to choose the baseline text. This depends on the use we want to make of concepts that are highlighted. For example, if we want to know whether some peculiar domain concepts are spoken about, we can compare the text with a domain textbook and see what concepts are considered important. Several texts may also potentially be combined in a single baseline. More concepts can also be found by varying the thresholds on the n-grams appearance times and the relevancy ratio, by removing the important concepts already found from the text under analysis.

2.5 Evaluation of the concepts importance

The previous step usually still provides too many concepts for the requirements engineer to analyse. The aim of the final step is to use machine learning to find out which concepts are the most efficient at distinguishing the texts. The assumption is that the higher the efficiency, the more important the concepts. To do this, we give the texts and potentially important concepts to a learning algorithm. This algorithm tries to learn how to distinguish between the texts using the potentially important concepts. It returns the concepts that can be used for this task and how well they perform. In our approach we use C4.5, which is a standard learning algorithm in UNIX systems [9, ch. 3; see man c4.5 and man c4.5rules]. C4.5 creates a decision tree deciding whether a text is part of the baseline or part of the document under review. Each node in the decision tree is a test on the presence of a particular concept in the text. The efficiency of a concept is given by the information gain provided by its related test, i.e., by how much it reduces the uncertainty about the text classification. Only tests that provide enough gain are added to the decision tree. The concepts that are mentioned in the tree are the ones we consider important. This algorithm cannot be applied up-front on the texts, before the steps described above are applied, because it is not robust enough to deal with the sheer number of words in the texts.

3 Examples

In this section, we present how our approach works on two examples. These examples are based on the finger-missle document presented in [6]. The document is composed of requirements descriptions and of an interview about the system. We first show that we can find out what the important concepts in the document are by comparing it with an unrelated document which is our baseline text. We then show how we can compare different parts of the document (especially the interview and the more formal requirements descriptions) to find out concepts that are dealt with in one part but not another.

The program performing the analysis is coded in Perl. In both examples, the whole analysis when ignoring unique n-grams takes about 5 min. on a Sun Ultra 5 workstation. Most of the time is spent on computing the concept frequencies.

3.1 Finding important concepts

We took the whole finger-missle document (2443 words) and combined it with an unrelated specification document (4017 words). We then processed the texts as explained in section 2. The cleaning-up of the pieces of data and the normalisation reduced the amount of word types from 1484 to 966. We then generated n-grams of length one to four. We did not try longer n-grams as we believe that it is unlikely that they would be frequent enough to produce useful results. The threshold on the n-grams appearance times was set to one, thus removing unique n-grams.

We then extracted potential important concepts and obtained 257 of these. The threshold ratio was set to 0.9. The best-ranked concepts are listed in table 1 (we rank best the shortest concepts with a high relevancy and high frequency). These concepts are very similar to the ones proposed in [5, p.128+]. We have added in the second column the abstraction number, corresponding to the concepts we found, proposed in that work. Some words are specifically ignored in that work, so that no abstraction can correspond to our concepts. We marked them as "ignored". Most of the concepts/abstraction are common to both approaches. While we find similar results, we had to specify much less information. For example, we did not have to state that "be" or "of" (the most common words in the document) were not relevant. They were automatically rejected because they are not good at distinguishing the document under review from the baseline document. We could have eliminated them (such as the determiners) to speed up the processing through.

We then gave the list of concepts to the learning algorithm. From the 257 potential important concepts, five were retained: "mode", "miss", "reconnaissance", "switchx", and "position". These concepts are therefore the ones our approach highlights as being important. The missile has indeed two modes of functioning, one of them being reconnaissance. Different parameters of the missile can be set by the switchx. The position of the missile is an important piece of data available to the missile operator. "Miss" refers to the missions the missile has to perform. The stemmer mistakenly reduces "mission" to "miss". This could be avoided by having more semantic knowledge on the meaning of words and using it to prevent the transformation.
<table>
<thead>
<tr>
<th>Concepts</th>
<th>Abstraction number</th>
</tr>
</thead>
<tbody>
<tr>
<td>mode</td>
<td>34</td>
</tr>
<tr>
<td>controlx</td>
<td>19</td>
</tr>
<tr>
<td>operate</td>
<td>1</td>
</tr>
<tr>
<td>switchx</td>
<td>21</td>
</tr>
<tr>
<td>flinger-missile</td>
<td>ignored</td>
</tr>
<tr>
<td>reconnaissance</td>
<td>55</td>
</tr>
<tr>
<td>speed</td>
<td>17</td>
</tr>
<tr>
<td>remote</td>
<td>19</td>
</tr>
<tr>
<td>position</td>
<td>2</td>
</tr>
<tr>
<td>remote controlx</td>
<td>19</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

Table 1. Best-ranked potential important concepts

It is possible to get an idea of how good these concepts are in distinguishing between the texts by evaluating the error made in classifying the two texts using them. To investigate this, we put aside a set containing 20% of the text under analysis and 20% the baseline text. This set was not used during the previous analysis. It contained 50 statements, 16 of which were related to the flinger-missile documents and 34 were not. Using the five concepts to classify these statements, nine were rightly classified as belonging to the flinger-missile documents and seven were missed. The 34 remaining statements were rightly considered as not belonging to the flinger-missile documents. We therefore have seven errors on the 50 statements, that is to say about 15%. The fact that some statements are not considered part of the flinger-missile documents when they actually are certainly means that we are missing important concepts. A more thorough analysis of the result quality could be done in terms of recall and precision if we knew the concepts we should find. Unfortunately, we do not have a corpus of requirements texts with their corresponding important concepts.

We did the same experiment without ignoring the unique n-grams, i.e., by setting the frequency threshold to 0 and relying only on the relevancy threshold. Although the number of potentially important concepts was much greater (almost 2800) the most important concepts were the same. It took approximately 25 min. to make the analysis.

3.2 Comparing requirements descriptions and interviews

In the second example, we compared two different parts of the flinger-missile document. In particular, we searched for the important concepts in the requirements descriptions compared to the interview. The method is the same as in the previous example, except that the texts have been changed. We obtained 73 potentially important concepts. From them, the learning algorithm selected two: "select" and "low".

"Low" is indeed an important concept in the requirements descriptions. The effect of low power and low signal are discussed. This topic is not mentioned in the interview. It could be important for the requirements engineer to realise this and maybe discuss this topic in another interview. It is unlikely that a traditional approach based on concept counts would have spotted this concept. "Low" only appears twice in the document and would have been buried under more frequent concepts.

"Select" refers to the flinger-missile mode selection in the requirements description which is an important concept. However, this concept is also discussed in the interview. It should not have been highlighted as an important concept compared to the interview. This mistake comes from the fact that this concept was discussed in the interview using other words such as "flip" or "change". This shows a limitation of our system: concepts that are spoken about with different words in the text under analysis and the baseline text may not be matched. However, this also highlights the use in the text of different words for the same concept. This could be a useful piece of information to ensure consistency when formalising the requirements. Using a source of knowledge about synonyms such as Wordnet may help to reduce this problem [8]. However, it is likely that context plays a role in determining the synonyms of a word. It is therefore not straightforward to apply this knowledge in practice.

The quality of the two concepts (assessed using a 20% test set as before) is very good. All of the 16 statements contained in the set are correctly classified as belonging to the interview (13) or the requirements document (3).

Again, taking into account unique concepts increases greatly the number of concepts (almost 1200) but does not change the results found.

4 From concept finding to requirements classification

In this section, we investigate whether we can use our method to not only find important concepts in texts but also to classify the texts based on them [4]. The main difference with the work presented in the previous sections is that we use the results of the learning algorithm not just for evaluating the concepts' importance, but to learn how to recognise difference classes of texts. For example, we could try to associate statements in interviews with the requirements they refer to. This seems promising since the important concepts we found were good at differentiating texts and would be useful to organise requirements.

We performed an experiment to see if this idea was viable. We took a set of interviews which could be divided
into statements. Each statement is related to a part of a requirements document. We then searched for the important concepts in the statements related to one part, compared to the others, using the method presented in this paper. We therefore obtained for each part of the requirement document a set of concepts found in the interviews related to that part. Finally, we tested how well these concepts could classify the interview statements with respect to the part of the system they discuss. The results are unfortunately quite poor. Error rates are around 50%. These rates are much greater than those given in the previous section. Distinctions between the different statements are often subtle and small errors in finding important concepts (such as the problem with "select" in the previous section) may have big effects on the classification process. However, the error rate may also be due to the small number of training statements. 125 statements were used to obtain the important concepts and learn to classify the corresponding statements. This is an order of magnitude less than the usual training corpora for that kind of approach. Better mappings from words to concepts may also improve the classification results. It therefore remains to be seen whether this approach is viable or not.

5 Conclusion

In this paper, we presented a method for finding important concepts by comparing two texts. This method does not assume that the most frequent concepts are the most important ones. We then presented how this approach performs on two examples. In particular, we showed that we could find concepts similar to those discovered by other approaches while specifying less information, and that we could find important concepts with low frequency. We still have to improve some aspects of our approach, notably the handling of synonyms. While it may be relatively easy to add some extra knowledge sources to perform a better job, it is likely that we will need to perform a much more complex analysis of the text in order to take into account the context when computing the semantic meaning of words. Interesting advances have been made by the natural language community in this direction [15]. However, it not clear how much improvements they would provide in our case. A program implementing our approach performs in reasonable time which makes it practical. We then discussed the possibility of using this technique for requirements classification but left this issue open for future research.

Acknowledgements

I would like to thank Dave Robertson for his useful comments and suggestions on the paper and for providing some of the specification documents on which the method presented in this paper has been tested.

References