Requirements Analysis

Stephane Clinchant
ENSEEIHT
Lund University
stephane.clinchant@netcourrier.com

Abstract

This paper presents the attempts to adapt the minimal edit cost algorithm for sentences in order to compare two requirements. These requirements are two short texts strongly related we want to compare. What is similar ? What is new ? are the questions we want to answer.

1 Introduction

One of the first and fundamental activities in a software process is requirement analysis. The system's services constraints and goals are established by consultation with system users and defined in a manner which is understanbled by boh users and developpement staff. Domain understanding , Requirement collection , Classification , Conflict Resolution , Prioritization and requirement validation are main activities in requirement analysis. This project is related to classification: this activity takes the unstructured collection of requirement and organized them into coherent clusters. In market driven software development there is a strong need for support to handle congestion in the requirement enginneereing process. To meet the market demands it is important to have an effective and efficient requirement engineereing process to deal with a numerous flow of incoming requirements.

2 Aim of the project

This project was suggested by Obigo a mobile phone software company. An analyst work on 2 sets of requirements: the old requirements and the new requirements, which could be for example :

> **Old**: SMS messages may contain up to 140 characters
> **New**: SMS messages must contain 240 characters including ISO characters.

A part of the analysts' work is spent on matching pairs of old and new requirements and identifying changes. His first task is to link similar requirements, to detect requirements referring to the same need , functionnality. It is a classification activity to find the structure and the relations between the requirements. The second task is to find the change , the new constraints , the difference between the related requirements. In the example, the important changes here are $may \rightarrow must$, $140 \rightarrow 240$ and a new constraint: *including ISO characters.*

This project aims at improving requirement analysis and saving time of Obigo's analysts. We focused on sequential sets of requirements and compare the new and the old ones. To streamline analysis , the following process was proposed :

- Use the tool of Johan Natt och Dag, ReqSimile to find the similar requirements
- Find the difference, the change between the pair of requirements :this is the aim of this project.

The language processing project was to implement an algorithm to detect and quantify the changes between two similar requirements. Thus, the analysts would be able to explore quicly the sets of requirements, find relations and detect the changes.

3 Minimun Edit Cost Algorithm and Alignments

The idea to explore in order to compare two short texts was to try to adapt the minimun edit cost algorithm for sentences and provide an alignment to compare them. Alignments and the minimun edit cost algorithm (MCA) are strongly related since the minimun edit cost produce an alignment. A definition of an alignement could be a sequence of operation which

transforms a sequence of symbols (the source) into an other (the target). Operations acts on symbols , can be copy (of symbols), insertion, deletion, substitution and have a cost. Letters are the symbol type for the classical minimun edit cost algorithm. A sequence of symbols is a word and the algorithm measure the distance between two words. Symbols can also be ADN bases in order to compare genomes. In this project symbols are words in the same language. But what is important to keep in mind is that an alignment is a way of matching elements of the source with the target. Here is an example of a possible alignment for the previous requirements

[copy(sms),copy(messages),subs(may,must),
del(up),del(to),subs(140,240),
copy(characters),ins(including),ins(iso),
ins(characters)]

The MCA is part of dynamic programming and gives alignments with the minimal cost. It is significant to underline the fact that there exists most of the time several possible alignments for one source and one target.The MCA equations stand for the possibilities a sequence source can be transformed into the sequence target. Basically there are 3 ways to do this. Suppose we have two sequences $[a_1, \ldots, a_n]$ and $[b_1, \ldots, b_n]$. First we can suppress a_n and transform $[a_1, \ldots, a_{n-1}]$ into $[b_1, \ldots, b_n]$. A second way to proceed is : if $a_n = b_n$ or if we substitute a_n by b_n we only need to transform $[a_1, \ldots, a_{n-1}]$ into $[b_1, \ldots, b_{n-1}]$. Lastly , if we know how to transform $[a_1, \ldots, a_n]$ into $[b_1, \ldots, b_{n-1}]$, we just need to insert b_n.

More formally, if E is the edit cost of two sequence and $c(.)$ the cost of an operation The minimun edit cost algorithm is defined by these equations:

$$\mathrm{E}([a_1, \ , a_n], [b_1, \ , b_n]) =$$

$$\min \begin{cases} \mathrm{E}([a_1, \ , a_{n-1}], [b_1, \ , b_n]) + c(\mathrm{del}(a_n)) \\ \mathrm{E}([a_1, \ , a_{n-1}], [b_1, \ , b_{n-1}]) + c(\mathrm{subs}(a_n, b_n)) \\ \mathrm{E}([a_1, \ , a_{n-1}], [b_1, \ , b_{n-1}]) + c(\mathrm{ins}(b_n)) \end{cases}$$

$$\mathrm{E}([a_1, \ , a_k], [\,]) = k \quad \mathrm{E}([\,], [b_1, \ , b_k]) = k$$

The costs of operation in the classical algorithm are

$$cost(ins) = 1 \quad cost(del) = 1$$
$$cost(copy) = 0 \quad cost(subs) = 2$$

To get the alignment , a matrix is filled with the cost of intermediary transformed sequences. Then , an alignment is obtained by a path in this matrix.

4 Hypothesis

To investigate the adaptation of MCA , I first limit my approach by not doing any linguistic process but just to use and apply the the basic algorithm to observe basic results. For example, we could say that the insertion of the word "and" costs zero because it does not bring significant information in the new requirement.So I did not try to tune this algorithm linguistically. One of the most important observation, is that there are several possible alignments.When I developped my algorithm, my goal was to construct all the possible alignments and to choose the best one with an heuristic.There is one simple reason for multiple alignment. In the example of requirements, the word "characters" occurs two times in the new requirement. So the "character" from the old requirement can be linked in two different places in the new requirement.

5 Factorization

Another reason is if we use the classical MCA we will obtain similar alignments which leads to the same matching of words. If x and y are two symbols, inserting x then deleting y is the same than deleting y then inserting x which is the same than substituting y by x.

$$ins(x), del(y) \sim del(y), ins(x) \sim subs(y, x)$$

We get the same matching of words but we are only intereted in a reduced form, the shortest alignment among its similarity class. So I decided to suppress the substitution operation from the algorithm so that it produces sequences of operations which are only insertion, deletion or copy. The next step was to factorize , to reduce the alignment in order to find the set of possible and interesting alignment. $copy(x)$, $subs(y, w)$ is the same than $copy(x)$, $ins(w)$, $del(y)$ but the first form is the best to keep. If we want to reduce an alignement, we have to consider more complicated similarity forms. As I suppressed the substitution operation, I can obtain alignement whose parts are such that:

$$ins(x_1), del(y_1), ins(x_2), del(y_2), \ \ldots \ , ins(x_k), del(y_k)$$

I need to factorize it and the problem was there was several way to do it. For example if we

have:

$$ins(x_1), del(y_1), ins(x_2), del(y_2)$$

This can be factorized in two ways:

$$subs(y_1, x_1), subs(y_2, x_2)$$

$$ins(x_1), subs(y_1, x_2), del(y_2$$

and we can not know which form is the best. If F is the Factorization function, the simple case are

$$F(Copy :: tail) = Copy :: F(tail)$$

$$F(Ins :: Ins :: tail = Ins :: F(Ins :: tail)$$

$$F(Del :: Del :: tail = Del :: F(Del :: tail)$$

$$F(Del :: Ins :: a :: tail = Subs :: F(a :: q)$$

where a = Copy or Ins

$$F(Ins :: Del :: a :: tail = Subs :: F(a :: q)$$

where a = Del or Copy
The complicated case is whenever there is a sequence of ins, del or a sequence of del,ins. Wether the number of couple ins,del be pair or not , there is always two possibilities to understand and to factorize this sequence:

- Take the first pair of ins,del (or del,ins) and group the other pairs after. If the number of sequence is unpair there is one operation left.
- Take the first operation and form the first pair with the second and the thrid operation. If the number of sequence is pair there is one operation left.

6 Results

So I developped a component for reducing alignment and I obtained 18 factorized alignment for the requirement example. I should have worked on heuristic at this time but I developped a simple GUI to display alignment and the matching of words with color codes. I had one idea about a heuristic at this time: alignments wih insertions at the end are more likely to be better than the others because people tend to add new elements at the end of a requirement when they rewrite them.

Then, I ran my algorithm on real requirements from Obigo which were much longer and I obtained something like 8 thousands alignments ! It was a exponential and combinatorial problem. To face this, I had to think to new ways to proceed:

- The first idea is to "to divide and conquer" and split the requirement in sentences and align sentences first and then align words.
- Drop the idea of finding the "best" alignment among the set of all possible alignment but just take one alignment with an heuristic.

I only had time to explore the second idea: my heuristic was to favorize insertion at the end. I had also a new idea : highlight the new words in the new requirement. New words in the requirement show the analyst which part of the requirement has been changed and need to be read. All the copy operation in the alignment show what is identical in the requirements.

7 Conclusion

Aligning two sentences is a very difficullt task if we do not take into acccount linguistic information. In translation where they also align sentences they exploit the structure of the sentence and linguistic information to find an alignment. But what the analyst is interested in , is to see what part of the requirement are identical and what is new. Taking one alignment is a simple and effective solution but there is work to do on how to display graphically the difference and the similarity of two text so that it is easy to see for the human eye.

8 Acknowledgements

I would like to thank Pierre Nugues, Johann Natt och Dag from Departement of Computer Science at Lund Universoty and Sven Olof Karlsson from Obigo for their help and advices during this project.

References

K.Yamada C.Goutte and E.Gaussier. Aligning words using matrix factorisation. *Xerox Research Centre Europe.*

C.Brockett C.Quirk and W.Dolan. Monolingual machine translation for paraphrase generation. *Natural Language Processing group Microsoft Research.*

S.Brinkkemper J. Natt och Dag, V.Gervasi and B. Regnell. Speeding up requirements management in a product software company.

Pierre Nugues. *An Introduction to Language Processing with Perl and Prolog.*

H.Garcia-Molina S.Chawathe, A.Rajaraman and J.Widom. Change detection in hierarchically structured information. *Departement of Computer SCience Stanford University.*