Negation Identification and Scope Resolution in NL Software Requirements

Ali Samad
Department of Computer Science & IT
The Islamia University of Bahawalpur
Pakistan-63100
ali6345@hotmail.com

Rafaqut Hussain Kazmi
Faculty of Computing
Universiti Teknologi Malaysia, Malaysia
rafaqutkazmi@gmail.com

ABSTRACT: *In this paper, we presented an approach that finds the negation in software requirements, and scope of negation. Our approach works at different levels of parsing each level has its own importance. The main phase of our approach is the deep semantic parsing this phase is further divided into three phases that are classification of negation, argument detection and scope detection. First phase classifies the negated sentences into different types of negation as implicit negation, explicit negation, double negation and single negation. Argument detection finds arguments that are negated. The last phase finds the scope of negation the arguments that are actually negated by the negation signal. Experimental works shows that our approach scores better than earlier studies.*

Keywords: Negation Detection, Negation Scope Resolution

Received: 2 June 2016, Revised 8 July 2016, Accepted 21 July 2016

1. Introduction

Requirement analysis is one of the main phases for software development. We can achieve good quality software if the requirements analysed correctly. Requirements are written in the form of natural language. There are many software available used for automatic conversion of Natural language requirements to software models. These software models analyse the natural language text semantically and build model on that basis. As natural language is ambiguous there can be different types of ambiguities such as be lexical, syntactic, semantic and pragmatic ambiguity [7] as there are different types of ambiguities one should know the level of ambiguity and handle it correctly.

Negation in a NL sentence can result in an ambiguous semantic s to the sentence. Negation can be single or double. Single negation can easily be identified but in case of double negation it is difficult to identify its meanings. For example in sentence like

"I haven't got no money" the semantic in the sentence is he has no money, but due to two negations it is difficult to understand. Negation can give multiple meanings for example sentence like customer does not applied for two accounts semantics are not clear whether customer haven't applied for the account or he has applied for the account but not for the two accounts. Other types of negations can be explicit and implicit. Explicit negation can easily identified and handle but implicit negation difficult to find as it hidden negation the words that give hidden negation are abnormal, bad, illegible. To find the negation in the sentence is simple but to find the scope which word is affected due to the negation is difficult and important [6].

There are different approaches available to find the negation in sentences. Generally people have worked on negation in biomedical field they applied classification techniques to find the negation and scope of negation.

We are using deep semantic parsing and Markov classification approach to find the negation and scope of negation in the sentences.

2. Related Work

Morante et. al. [1] proposes a system for finding the scope of negation. They used the bioscope corpus a freely available resource for their system. There system consists of two parts in first part k-nearest neighbour classification algorithm is applied to find that sentence is negated or not in second phase scope finding algorithm is applied to the sentence that are negated. Scope finding classifier uses different features to extract the token that are affected by negation. Algorithm is applied to the extracted blocks these blocks are random. Algorithm first get all the blocks that affected by the negation signal and in next step find and save block that is inside the scope and exclude all block that are outside scope. For evaluation purpose they use Fscore, precision and recall.

The resource text they have used is annotated text which shows the boundaries of negation. But in real scenario sentences does not show their negation boundaries. However the algorithm they have applied for scope finding dependent upon the length of non-scope blocks between two blocks that have negation. Algorithm needs some improvement.

Dadvar et. al. [2] used the approach of finding negation for opinion detection. They used the data set of movie reviews. Theses reviews contain different positive and negative opinion. They obtain the wordlist of positive and negative opinion and classify the review on the basis of these words. After this classification they detect the negation the words that were negated are saved in the wordlist of negation. They repeat the test for all the reviews and find the final conclusion of a movie review. They evaluated there system on the basis of accuracy and compare their system with nave baseline.

Cruz Díaz et al. [3] develop a system that finds the scope of negation using machine learning techniques. The system works in two phases in first phase Naïve Bayes and c4.5 classifier algorithm applied to find the sentences that are negated in second phase SVM classifier finds the signal that is affected by the negation token. The system works on sentence level so POS tagger is used to parse the sentences and features are extracted from sentences. Classifier algorithm is applied to these features and finds the sentence is negated or not if negation signal is present in the sentence then scope finding classification is applied and after that an post processing algorithm is applied to the signal that stores the negation signal in scope data test to use it for testing data test. First the system is trained with given resource and after that a test phase is applied to find system working correctly or not. The system is evaluated on token level and signal level. Measures for the evaluation are precision, recall and FScore.

In this system they have used only 14 features in training phase for finding negation some of them are ignored but if want to use the system in any other domain other features that are ignored can be important. Test phase is dependent on training phase if a new negation signal present in test data set system will not take it as negation.

Fujikawa et. al. [4] propose the approach to find the scope of negation. They used supervised classification method for finding the negation in sentences and scope in negation. After finding the scope of negation they applied the grammar parsing approach for adjusting scope. They applied different features for finding the scope of negation. Features that are used for detecting negation signal includes main word (no, not), position of words with respect to negation and number of words. Features for the classification of token includes negated word, token present in left and right of the negated word. In last they have found chunk of words that are negated and they applied algorithm for adjusting the scope of negation with respect to grammatical rules.

Li et al [5] find the negation signal and its scope by using shallow semantic parsing approach. They have taken a parsed tree as input and applied shallow semantic parsing on it.

They have taken negation signal as predicate signal and all its related argument in the scope of negation further they applied argument pruning and argument identification for finding the particular word that is affected by negation. They applied different features for finding the scope. Arguments that are affected by the negation are identified these arguments can be continuous or discontinuous a formula is applied to find the particular argument that is affected. By using the probability formula they found that argument that is present in the left is affected or right. (if the probability is same?).

3. Types of Negation

There are different types of negations that can exist in one sentence. Every type of negation has different effect on the sentence and handling/finding the scope of that negation is different. Types of negation shown below:

Single Negation: this is simple type of negation we have one signal of negation and this signal can affect one more words in that sentence. if there is single negation in the sentence then finding the scope of negation is easy in that sentence is easy for example: “unauthorized person cannot login to the site.”

Double negation: this type of negation gives complex result as there exists two negation signals and both affecting the words differently. Finding the scope of negation in this case is quite difficult.

For example: “There wasn’t anyone who did not applied for the discount”

Implicit Negation: there are some words that give negative meaning to the sentences other than no, not can be said as hidden negation. It is difficult to find. Words that make the sentence implicitly negated are as bad, Illegible, incomplete, unpleasant.

For example: “He is illegible”

Explicit Negation: It is a type of negation that can easily be found. It is simple and open the words are no, not.

4. Used Approach

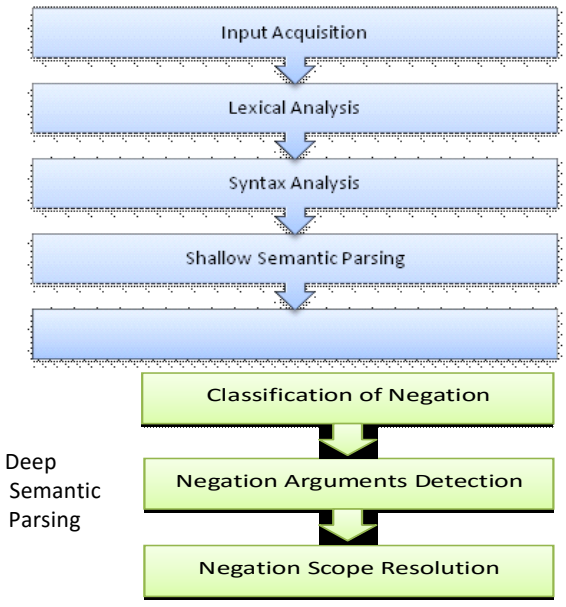

Figure 1. Architecture of the system for Negation resolution

These steps are explained as following:

1) Lexical Analysis
It is first step we input the textual format requirements these requirement are pre-processed and sentences are split. Each sentence can be identified as unique sentence. After splitting the sentences tokenization is applied different tokens in the each sentence is identified. In next phase POS tags are applied to each token. Nouns, verbs are separately identified in each sentence and tagged accordingly for further application. In last step lemmatization applied to the token. Lemmatization identifies the base word and removes its suffix. For example word in "suggested" ed is removed and word become suggest.

2) Syntactic analysis
When lexical analysis applied on sentence it will out a parse tree then in next phase syntactic analysis is applied it has next three steps that are named a syntax tree generation , classification of active and passive voice and conjunction and disjunction finding. In syntax tree generation phase dependencies of the words are identified and means nouns, verbs are separated according to their influence on sentence. In next step as mentioned above voice classification is applied to the sentences each sentence has grammatical feature that represents the voice of sentence and sentences are easily classified according to that. When voice of the sentences identified then subject, object and verb are classified. After this classification conjunction and disjunction is identified. Conjunction is combine two sentence by a specific word for example "customer can have two accounts but he can login from one account at one time". The words identified as conjunction in sentence are as, and, but, so.

3) Semantic Analysis
Semantic analysis phase understands the exact meanings of the sentence. This analysis phase divides the sentence in different chunks and finds relationship between the chunks. Semantic analysis has two further steps shallow semantic parsing and deep parsing.

a- Shallow Semantic Parsing: To find the exact meaning of the sentences we have to identify the different action that is performed by some actor. Each sentence has its own meaning with respect to the action and agent performing action on it. To identify the exact meaning of sentence labels are applied on the different argument of the sentence and classify each argument according to it. These different labels are explained as follows:

a. Actor object: Actor is the object that is performing some type of action in the sentence for example "customer is paying the bill". Customer is the actor in the sentence.

b. Co-actor object: Co-actor it is the object that is working with the actor in the sentence it is combined by some word for example "customer is paying bill by visa card". Visa card is co-actor in the sentence.

c. Recipient object: This is the object for which an action is performed. For example "Customer is paying bill for dresses". Dresses is that object.

d. Thematic object: the object on which the action is performed for example "customer is paying bill". Paying is thematic object.

e. Conveyance object: This is object on which someone travels. For example: "employee travel by buses"

f. Route object: This is the object that shows the route of actor from source to destination. For example "the order will be parcelled from Karachi to Lahore tomorrow".

g. Site object: it is the object that shows the location of one action. For example "customer can pay bill online".

Time object: this object shows the time when an action is occurred. For example "customer A applied for account an 16-june".

h. Duration object: This object specifies how long an action takes. For example "Procedure of account verification takes 1 day".

b- Deep Semantic Parsing: Deep semantic parsing is divided into three sub-phases :

i. Classification of Negation:
As explained earlier there can be different types of negation can exist in a sentence. Each negation has different effect in our first phase of negation classification we find the type of negation in all sentences. Types can be single negation, double negation, explicit negation, implicit negation. For example we have sentences like: "No person is eligible". "A few persons are not eligible". "A few persons are illegible." "A few persons are not illegible".

All sentences above give the classified as negated sentences but all are treated differently for finding the scope. So for this

purpose in the first step of our deep semantic parsing classification of negation we identify the type of negation and classify allsentence in particular category as single negation, double negation, implicit negation and explicit negation.

The above four sentences first one “No person is eligible” is classified to single negation, “A few persons are not illegible” is classified as double negation, “A few persons are illegible.” Is classified as explicit negation, A few persons are not eligible” classified as implicit negation.

When all negated statements are classified into their categories the next step is applied that find which argument is affected by the negation.

ii. Negation Argument Detection:
After classification of negation we find which argument that is affected by the negation symbol. These arguments can be more than one all the arguments that seems to be affected are identified.

Arguments of Implicit Negation

iii. Negation Scope Detection:
when the arguments are found we find the scope of algorithm for finding the scope we use the algorithm.

5. Experiment and Results

In this section, an example is processed to test the performance of the designed system. The designed system performs set of processing steps and finds negation in the input software requirements. Following example is an extract that is taken from a document ‘Software Requirement Specification’ (Adams, 2004).

1) Problem statement of Case Study
The example used to test performance of the tool is taken from a document ‘Software Requirement Specification’ that was prepared in 2004 (Adams, 2004). Figure 4.1 shows the scenario of the requirements that are shown below:

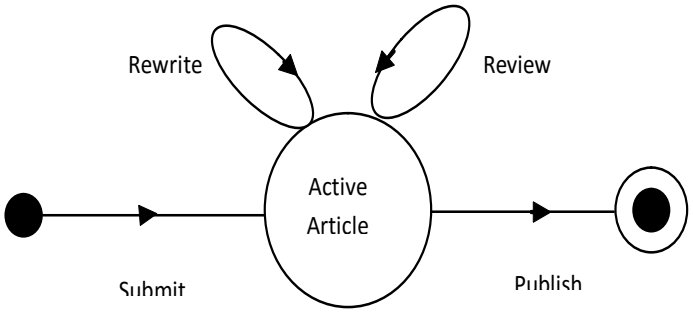

Figure 4.1 Article Submission Process

Below is the problem statement of the case study processed for negation identification in the later part of this section:

“An Author submits an article for consideration. The Editor enters it into the system and assigns it to and sends it to at least three reviewers. The Reviewers return their comments, which are used by the Editor to make a decision on the article. Either the article is accepted as written, declined, or the Author is asked to make some changes based on the reviews. If it is accepted, possibly after a revision, the Editor sends a copyright form to the Author. When that form is returned, the article is published to the Online Journal. Not shown in the above is the removal of a declined article from the system.”

Following is the input statement that was processed by the Stanford parser and output is also shown below:

Input: Either the article is accepted as written, declined, or the Author is asked to make some changes based on the reviews.

Tagging: Below is the output of POS Tagging generated by the Stanford POS tagger after tokenization and POS tagging:

Either/CC the/DT article/NN is/VBZ accepted/VBN as/IN written/VBN ,/, declined/VBD ,/, or/CC the/DT Author/NN is/VBZ asked/VBN to/TO make/VB some/DT changes/NNS based/VBN on/IN the/DT reviews/NNS ./.

Parse Tree: Below is the parse tree generated by the Stanford parser

```
(ROOT
 (S (CC Either)
  (S
   (NP (DT the) (NN article))
   (VP (VBZ is)
    (VP (VBN accepted)
     (SBAR (IN as)
      (S
       (S
       (VP (VBN written)))
       (, ,)
      (VP (VBD declined)))))))
  (, ,)
  (CC or)
  (S
   (NP (DT the) (NN Author))
   (VP (VBZ is)
    (VP (VBN asked)
     (S
      (VP (TO to)
       (VP (VB make)
        (NP (DT some) (NNS changes))
        (PP (VBN based)
         (PP (IN on)
         (NP (DT the) (NNS reviews)))))))))
  (. .)))
```

Syntactic dependencies: Below are the typed dependencies (collapsed) generated by the Stanford parse:

preconj(accepted-5, Either-1)

det(article-3, the-2)

nsubjpass(accepted-5, article-3)

auxpass(accepted-5, is-4)

root(ROOT-0, accepted-5)

mark(declined-9, as-6)

dep(declined-9, written-7)

advcl(accepted-5, declined-9)

det(Author-13, the-12)

nsubjpass(asked-15, Author-13)

xsubj(make-17, Author-13)

auxpass(asked-15, is-14)

conj_or(accepted-5, asked-15)

aux(make-17, to-16)

xcomp(asked-15, make-17)

det(changes-19, some-18)

dobj(make-17, changes-19)

prepc_based_on(make-17, on-21)

det(reviews-23, the-22)

pobj(make-17, reviews-23)

Negation Identification: The used algorithm identifies the sense of the words and finds that decline is a based on negative sense and by using the above shown dependencies, it is found that the decline is related to article.

2) Results and Discussion

In this section, we calculate the results using above solved case studies 4.1, in this thesis and three more case studies were also done, in which we extract the possible negations (both implicit and explicit) and also find out the scope of the identified negation. On the basis of the achieved results, we can calculate the value of Precision, Recall and F-measure using these following formulas.

$$Recall = \frac{Matched\ negations}{Total\ negations}$$

$$Precision = \frac{Relevent\ negations}{Extracted\ negations}$$

Where P is the precision value and R is the Recall value. Table 4.2 shows the results of P, R and F-measure and Figure 4.1 represents the results.

Case Studies	Total Incidences of Negation	Correct Identifications	Incorrect Identifications	Miss Identifications
1	3	3	0	0
2	5	4	0	1
3	8	6	1	1
4	5	4	1	0

Table 4.1 calculate the values of P, R and F-measure

Case Studies	Precision (%)	Recall (%)	F-Measure (%)
1	100	100	100
2	80	80	80
3	85.7	75	79.9
4	80	80	80

Table 4.2 calculate the values of P, R and F-measure

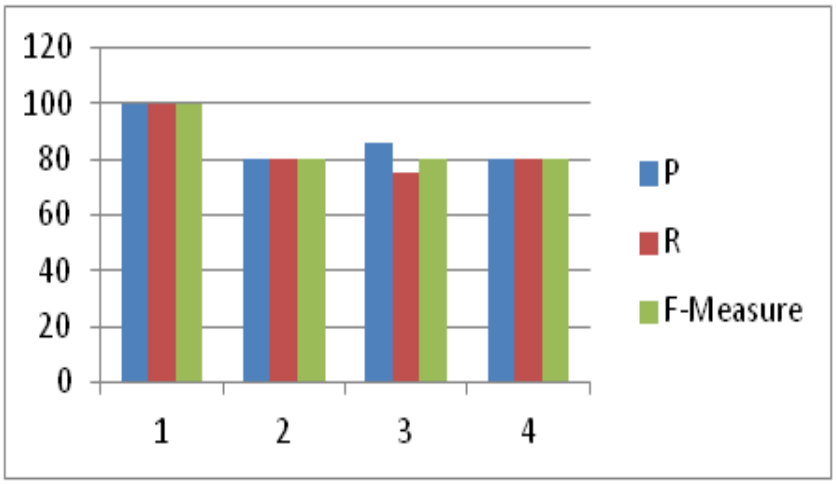

Figure 4.1 Graph showing results of the system

In this section, a case study was discussed in detail. We present a set of experiments to test the performance of the approach discussed in this chapter. The results are very encouraging.

4. Conclusion

In this paper, we have presented the approach of finding the scope of negation in Software requirements. Negation cause ambiguity in the sentence. No tool available that finds the scope of negation in software requirements. Handling the negated signals is one of the difficult tasks in NL. Work has been done for finding the scope negation in biomedical field but no work is found for handling the negation in software requirements. We presented the approach that finds the negated sentences and find the scope of negation in that sentences. There are different types of negation exists in the sentence we classify the negated sentences into different types of negation and find the scope of negation in these sentences with respect to the type of negation.

References

[1] Morante, R., Liekens, A., Daelemans, W. (2008). Learning the Scope of Negation in Biomedical Texts, *In:* Conference on Empirical Methods in Natural Language Processing (EMNLP 2008). ACL, 2008, p. 715–724.

[2] Dadvar, M., Hauff, C., de Jong, F. (2011). Scope of Negation Detection in Sentiment Analysis, *In:* 11th Dutch-Belgian Information Retrieval Workshop (DIR 2011), 2011, p. 16–19.

[3] Cruz Dýaz, N., Mana Lopez, M.,Vazquez, J., Alvarez, V. (2012). A machine-learning approach to negation and speculation detection in clinical texts. *Journal of the American Society for Information Science and Technology*, 63 (7) 1398–1410.

[4] Fujikawa,K.Seki,K.Uehara, K. (2012). A Hybrid Approach to Finding Negated and Uncertain Expressions in Biomedical Documents, *In:* MIXHS'12.

[5] Li, J., Zhou, G., Wang, H., Zhu, Q (2010). Learning the Scope of Negation via Shallow Semantic Parsing. *In:* Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010), p. 671–679

[6] Blanco E., Moldovan. D. Some Issues on Detecting Negation from Text, *In:* Proceedings of the Twenty-Fourth International Florida Artificial Intelligence Research Society Conference.

[7] Kiyavitskaya, N., Zeni, N., Mich, L., Berry, D.M. Requirements for tools for ambiguity identification and measurement in natural language requirements specifications. *Journal of Requirements Engineering* 13 (3) 207–239.

[8] Umber A., Bajwa, I.S., (2011). Minimizing Ambiguity in Natural Language Software Requirements Specification. 6th IEEE International Conference on Digital Information Management (ICDIM-2011), Sep 2011, p. 102-107, Melbourne, Australia.

[9] Umber A., Bajwa, I.S., (2011). A Step towards Ambiguity Less Natural Language Software Requirements Specifications *International Journal of Web Applications (IJWA)*, 4 (1) Mar 2012

[10] Ghosh, S., Elenius, D., Li, W., Lincoln, P., Shankar, N., Steiner, W. (2014). Automatically Extracting Requirements Specifications from Natural Language. CoRR, abs/1403.3142.

[11] Xiao, X., Paradkar, A., Thummalapenta, S., Xie, T. (2012, November). Automated extraction of security policies from natural-language software documents. *In:* Proceedings of the ACM SIGSOFT 20th International Symposium on the Foundations of Software Engineering (p. 12). ACM.

[12] Galvis Carreño, L. V., Winbladh, K. (2013, May). Analysis of user comments: an approach for software requirements evolution. *In:* Proceedings of the 2013 International Conference on Software Engineering (p. 582-591). IEEE Press.

[13] Umber A., Bashir T., Bajwa I.S., Naweed, M.S.(2011). Requirements Elicitation Methods. 2nd IEEE International Mechanical, Industrial, and Manufacturing Technologies Conference (MIMT-2011), Feb 2011, p. 541-545, Singapore

[14] Sagar, V. B. R. V., Abirami, S. (2014). Conceptual modeling of natural language functional requirements. *Journal of Systems and Software,* 88, 25-41.

[15] Piskorski, J., Yangarber, R. (2013). Information extraction: Past, present and future. *In:* Multi-source, multilingual information extraction and summarization (p. 23-49). Springer Berlin Heidelberg.