Discovering Speech Acts in Online Discussions: A Tool-supported method

Itzel Morales-Ramirez[1,2] and Anna Perini[1]

[1] Software Engineering Research Unit. Fondazione Bruno Kessler - IRST
Via Sommarive, 18, 38100 Trento, Italy
imramirez,perini@fbk.eu
[2] University of Trento, Italy

Abstract. The increasing participation of users of software applications in online discussions is attracting the attention of researchers in requirements elicitation to look at this channel of communication as potential source of requirements knowledge. Taking the perspective of software engineers who analyse online discussions, the task of identifying bugs and new features by reading huge threads of e-mails can become effort demanding and error prone. Recognising discussants' speech acts in an automated manner is important to reveal intentions, such as suggesting, complaining, which can provide indicators for bug isolation and requirements. This paper presents a tool-supported method for identifying speech acts, which may provide hints to software engineers to speed up the analysis of online discussions. It builds on speech act theory and on an adaptation of the GATE framework, which implements computational linguistic techniques.

Keywords: Speech Acts, GATE framework, Intentions

1 Introduction

Researchers in requirements elicitation recognise that the increasing participation of users of software applications in online discussions turns these type of discussions into an attractive source of information. Such an information can be exploited for different purposes and one purpose can be to derive requirements knowledge. In this context we define requirements knowledge as the knowledge that contributes to the definition of software systems requirements, as well as to the modifications of requirements already specified.

In this line, if we take the perspective of software engineers who need to analyse probably many text files comprising the content of online discussions, the task of identifying bugs and new features by reading huge threads of messages or e-mails could make this task effort demanding and error prone. For instance, in an open-source software development that rests on distributed communities, open forums and mailing lists are communication means commonly used to enable the collaboration tasks to perform solution design, code writing, software deployment, maintenance and evolution.

Recognising discussants' speech acts in an automated manner can be seen as an important task to reveal intentions, such as suggesting, complaining, which are becoming of crucial importance to understand discussants' comments. We take the inspiration from the Speech Act Theory (SAT), originally formulated by Austin and Searle [1], and from the Grice's claim, in [2], that says "speaker's utterances automatically create expectations, which guide the hearer towards the speaker's meaning". Indeed, the speaker may aim at persuading, inspiring or getting the hearer to do something.

In this paper we present a tool-supported method that aids the discovery of speech acts, a task that we address using information extraction techniques. We believe that this identification of speech acts can provide software engineers hints to speed up the analysis of online discussions. Our approach builds on SAT and its application in computational linguistic [3]. Specifically, we exploited the GATE framework [4].

The remainder of the paper is structured as follows. In Section 2 we give some background on SAT and on the NLP framework used in our approach. In Section 3 we describe our tool that identifies speech acts as annotate intentions in text. The related work is presented in Section 4 and the conclusion in Section 5.

2 Speech Act Theory and GATE Framework

In this section we recall basic definitions from the philosophy of language, namely Speech Act Theory (SAT), we build on, and about the General Architecture for Text Engineering (GATE) tool for information extraction that has been adapted to implement our tool. The basic claim of the SAT developed by Austin and Searle in the field of philosophy of language [5, 2] is the following. When a person says something she/he attempts to communicate certain things to an addressee by getting him or her to be affected by the speaker's intention, in other words each utterance in a conversation corresponds to an action performed by the speaker. A speech acts involves then three types of acts, namely, locutionary, illocutionary and perlocutionary acts. A locutionary act is the act of "saying something", an illocutionary act makes reference to the way in which the locutions are used and in which sense, and a perlocutionary act is the effect on the audience that may be achieved. So, for instance, considering the utterance "I'll bring you a chocolate", the locutionary act corresponds to the utterance of this sentence, the illocutionary act corresponds to the speaker's intention to make the audience aware that she is committing to bring a chocolate, and the effect, i.e. the perlocutionary act, is that the audience got convinced about the speaker's intention.

Our tool makes use of a classification of speech acts that is proposed by Bach and Harnish in [1]. There are four main kinds of acts, namely constantives, directives, commissives, and acknowledgements. *Constantives* express the speaker's belief and her intention or desire that the hearer has or forms a like belief, e.g. "I must confess I'm a good chef". *Directives* express the speaker's attitude toward some prospective action that should be performed by the hearer and her intention that her utterance must be taken as a reason for the hearer's action. For example, if I say to you: "I want you to walk the dog in the evening", I intend to motivate you to perform the action. *Commissives* express the speaker's intention to commit to do something. As for instance when I say to you: "I am going to bring you a chocolate", I intend to make you believe that I'm committing to buy and bring a chocolate for you. Finally, *acknowledgements* express feelings regarding the hearer or the speaker's intention that her utterance satisfies a social expectation. For instance, "Please forgive me".

The GATE tool [4] is a framework, developed by the University of Sheffield in UK, for building and deploying software components to process human language. GATE can support a wide range of NLP tasks for Information Extraction (IE). IE refers to the extraction of relevant information from unstructured text, such as entities and relationships between them, thus providing facts to feed a knowledge base [6]. GATE is widely used both in research and application work in different fields (e.g. cancer research, web mining, law). This tool is composed of three main components for performing language processing tasks: *Language Resources* represent entities such as lexicons, corpora or ontologies; *Processing Resources* represent entities that are primarily algorithmic, such as parsers, generators or ngram modellers; and *Visual Resources* represent visualisation and editing components that are used in GUIs.

GATE offers the flexibility to replace or extend the *Processing Resources* component. For our purposes we have adapted the framework to build our tool by considering the following modules: (i) *Sentence splitter:* this module split the text into sentences, using RegEx splitter that is based on regular expressions; (ii) *Tokeniser:* is the module that identifies basic "tokens", such as words, punctuation symbols, and numbers; (iii) *Part-of-speech (POS) tagger:* this module associates tokens with parts of speech such as noun, verb, and adjective, based on the Hepple tagger [3]; (iv) *Morphological analyser:* this module is used to lemmatise the tokens and to provide words in their root form, e.g. running – run; (v) *Gazetteer:* this module can be adapted and it is a list of lists, where each list is a group of words that are associated with the domain; and (vi) *Java Annotation Patterns Engine (JAPE):* this is the main module that has been adapted in our tool and it enables the creation of rules in the form of regular expressions. The left-hand-side of the rule refers to what should match a fragment of text, and annotations in the right-hand-side says what should be done with the matched text.

3 Tool for Discovering Speech Acts

Our tool is based on a knowledge-heavy approach [7], this means the use of a POS tagger, JAPE rules, a tokeniser, a lemmatiser and gazetteers. In Figure 1 we present the different modules that have been used to build the tool. The bottom part of the figure shows Java as the platform on which GATE is built on and we have also reused to access to our dataset, XML is one of the input format that GATE allows and we have chosen to parse the processed files. On top of this layer are the modules that can be exchanged, and the JAPE and Gazetteers modules, which are flexible modules to be adapted according to the objective of the IE tasks. Gazetteers and JAPE rules have been tailored to annotate the intentions applying some tags that we have defined previously. These tags are used to annotate fragments of text, the tags are the subcategories of speech acts defined in Table 1. The first column with heading "Category" shows the main categories of speech acts and the column "Subcategory" is a specialisation used as the names for the tags to annotate. For instance, the first category *Constantives* is specialised into several subcategories, including the subcategory *Suggestives*, which corresponds to a linguistic act expressing the intention of a sender to make the receiver to consider as an option what he or she is suggesting. To adapt the JAPE module we

[3] Part-of-speech tags taken from http://gate.ac.uk/sale/tao/splitap7.html#x37-761000G

have formulated lexico-syntactic rules, by using a bag of words inspired from the given examples in [8] and by an empirical exploration of users' comments given in online platforms like the bugzilla issue tracking system.

The Gazetteers used in our approach are the lists of verbs taken mainly from [1] for each subcategory of speech act. Some JAPE rules use the Gazetteers to annotate intentions. The linguistic analysis is executed on discussion threads in the format of text files, which are the input. The tool is used to annotate intentions on the text messages of each thread. After this, the files annotated with intentions are parsed to extract the intentions found in each message. Finally, an analysis of intentions is performed, following an analysis model that we are elaborating.

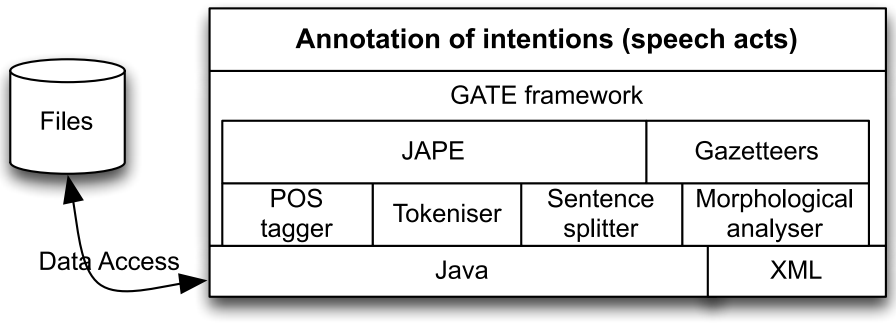

Fig. 1. Architecture of the tool.

Examples of design of JAPE rules are illustrated below, the full set of rules are available online[4]. The bottom part of the rule for tagging the speech act *Suggestives* represents the name of the tag to be annotated by the tool that processes the NL text, in our example the intention *Suggest*. In the second layer are the POS tags and tokens that are used by the tool to annotate such an intention. The POS tags $< PRP >$ and $< MD >$ refers to the initial set of words to annotate, the $< Keyword >$ refers to a list of verbs that we have defined in the Gazetteer modules and that are used by the JAPE rules[5].

$$\frac{\dfrac{\text{You}}{< PRP >} \quad \dfrac{\text{can}}{< MD >} \quad \dfrac{\text{“try”|“check”}}{< Keyword >}}{Rule\ to\ tag:\ Suggestives}$$

In the case of the rule to tag the speech act *Questions*, the second layer presents the starting and ending type of tokens that are used by the tool to annotate a question. In the left side of this second layer are the POS tags that can be found at the beginning

[4] JAPE files are available at http://selab.fbk.eu/imramirez/JAPErulesFeb2014/files.zip

[5] Gazetteer files are available at http://selab.fbk.eu/imramirez/GazetteerFeb2014/files.zip

Table 1. Categories of *speech acts* (excerpt of categories).

Category	**Subcategory**
Constantives	Assertives Concessives
	Suggestives
	Suppositives
	Responsives
Directives	Requestives
	Questions
	Requirements
Expressives	Thank
	Accept
	Reject
	Negative opinion
	Positive opinion
Attach (non-linguistic)	Link
	Code
	Log

of a question, i.e. ($< WRB >$ and $< VBZ >$), and in the right side, the *content* of a question plus the *question mark* indicating the end of a question. The top layer shows a concrete example.

$$\underbrace{\underbrace{\underbrace{\text{Where}}_{< WRB >}\quad \underbrace{\text{is}}_{< VBZ >}}_{Question\ begins}\quad \underbrace{\underbrace{\text{the option to delete...}}_{< content >}\quad \underbrace{\text{“?”}}_{Question\ mark}}_{Question\ ends}}_{Rule\ to\ tag:\ Questions}$$

We manually designed the rules considering some characteristics for extracting the intentions, such as preceding and succeeding words, length of the words, root of the words, special types of verbs, using the bag of words, syntax and the codification of the POS tagger used by GATE. The tag is used to label a text fragment when one of the corresponding rule matches it. Each rule is formulated as a regular expression. The regular expressions $< content >$, $(< MD >)*$ and [Hh], for example, make reference to a set of words in the middle of two keywords or POS tags, to the presence or absence of the POS tag and to the uppercase or lowercase of the first letter of a word, respectively.

Each rule was then translated into JAPE rules, the tool uses the rules and tags to process the discussion threads in text format (i.e. txt files). This tool implements the modules of GATE as follows, see Figure 2 that depicts this process:

1. Cleaning and pre-processing: this process makes a document reset to clean the text of any previous tags, and noisy data.
2. Splitting into sentences: we use RegEx to split the text into sentences.
3. Splitting into tokens: we make use of a tokeniser provided in GATE to split the sentences into words.

4. Tagging tokens: then we run a POS tagger to tag each token into categories of nouns, verbs, punctuation, etc.
5. Lemmatising: the morphological analyser provided by GATE is used to lemmatise each word to its root, the example in the figure is the verb "wondering" into "wonder".
6. Gazetteers: in this process the Gazetteer' lists are used to identify specific verbs we have selected and that refer to intentions.
7. Applying rules: we use the JAPE rules to annotate the speech acts in the text. This is the last step that needs each word to be tagged with a POS tag, lemmatised and recognised by the Gazetteers.

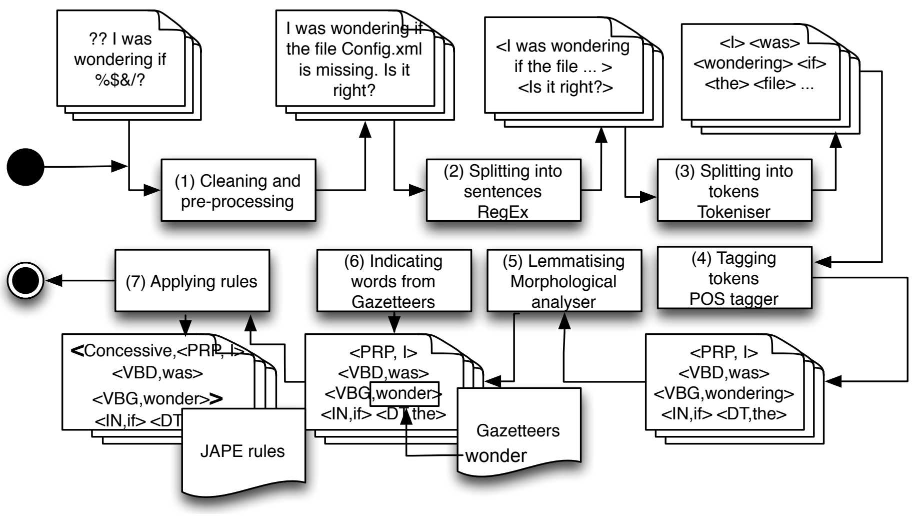
?? I was wondering if %$&/?
I was wondering if the file Config.xml is missing. Is it right?
<I was wondering if the file ... > <Is it right?>
<I> <was> <wondering> <if> <the> <file> ...
(1) Cleaning and pre-processing
(2) Splitting into sentences RegEx
(3) Splitting into tokens Tokeniser
(7) Applying rules
(6) Indicating words from Gazetteers
(5) Lemmatising Morphological analyser
(4) Tagging tokens POS tagger
<Concessive,<PRP, I> <VBD,was> <VBG,wonder>> <IN,if> <DT
JAPE rules
<PRP, I> <VBD,was> <VBG,wonder> <IN,if> <DT,the>
Gazetteers wonder
<PRP, I> <VBD,was> <VBG,wondering> <IN,if> <DT,the>

Fig. 2. Process to identify speech acts and annotate intentions.

After the annotation is executed, the tool parses the files to extract only the speech-acts tags and the corresponding intentions found in the text and a CSV file is generated by each discussion thread. Each file contains the discussants' name and the intention(s) identified in their messages.

We propose an analysis model of the intentions in a discussion thread that can be performed at different levels of granularity, namely, sentence and message level. At the sentence level we can identify single and nested speech acts. For instance, in the sentence "I suggest you to make a copy of your data", the single intention *Suggest* is the speech act "I suggest you", which refers to the subcategory *Suggestives*. An example of nested intentions is expressed in the sentence "Why don't you try to use the wizard?". In this case there are two speech acts, one is "Why don't you try to. . . ?" , and the other one is "don't you try", representative of the intentions *Quest* and *Suggest* respectively. At the message level the occurrences of pairs of intentions is analysed, called compound intentions, and we claim can be indicators of *Bug*, *Feature*, or *Clarification* requests. For example a combination of speech acts from the subcategory *Negative opinion* ("There is a problem") and *Question* ("Can anyone help me?") can be an indicator of a bug. Therefore, a set of nested or compound linguistic and non- linguistic acts can be considered as indicators of bug, features, and clarification. Currently, we are working on this model to incorporate it into the tool.

4 Related work

The analysis of NL textual messages in online discussion forums, bug-tracking systems or mailing lists has been addressed by research works that we briefly recall in this section. An automated identification of intentions is presented in [9]. This work proposes a tool that is based on SAT, dialogue acts and fuzzy logic to analyse transcripts of telephone conversations. One purpose of identifying intentions is that of detecting deception among participants in conversations by deriving participant profiles based on a map of the intentions expressed in such conversations. The classification of e-mails using speech acts is investigated in [10]. They are interested in classifying e-mails regarding office-related interactions as negotiation and delegation of tasks. Besides, they consider non-linguistic acts, such as *deliver*. In [11] the investigation of speech acts on thread of discussions, in student forum, aims at identifying unanswered questions to be assigned to an instructor for their resolution. They present some patterns of interaction found in the threads, the patterns correspond to the responsive and question speech acts.

With reference to Requirements Engineering tasks, Knauss et al. [12], analyse discussion threads for requirements elicitation purposes. They are focused on the content of communication between stakeholders to find patterns of communication used by stakeholders when they are seeking clarification on requirements. Their approach is based on a Naive Bayesian classifier, a classification scheme of clarification and some heuristics, with interesting results. Worth mentioning is also the work presented in [13] that aims at analysing messages, or *comments*, from users of software applications. Information extraction techniques and topic modelling are exploited to automatically extract topics, and to provide requirements engineers with a user feedback report, which will support them in identifying candidate new/changed requirements. All the above mentioned research works in the area of Requirements Engineering use NL text messages or documents to discover patterns, relevant topics or identify domain key terms, but none of the them consider SAT based techniques to understand stakeholders' intentions behind their messages. We consider that the application of SAT in Requirements Engineering can be a powerful strategy to understand stakeholder's intentions, thus supporting the analysis of the messages they exchange in current distributed collaboration and deriving requirements knowledge.

5 Conclusion

In this paper we presented a tool-supported method to identify speech acts in online discussions of OSS communities, which are developed by using unstructured natural language text. Indeed, we build on the idea that the recognition of discussants' speech acts is key to reveal their intentions, such as suggesting, or complaining. By automating speech acts recognition we may contribute to lower the software engineering effort in analysing huge discussions, also allowing for a better quality result.

The proposed approach exploits a taxonomy of speech acts based on the one proposed by Harnish and Bach. We illustrated the revisited taxonomy including linguistic and non-linguistic acts (such as code lines, URL links, and log files). Moreover, we described the adaptation of the framework GATE, specifically the modules Gazetteers and JAPE rules that can be configured according to the purpose of extraction, in our case the extraction of intentions. We explained the general process for annotating intentions and have explained the design of rules used to match fragment of text. Concerning the effectiveness of our approach, preliminary experimental evaluation on data sets extracted from online discussions in a OSS project are encouraging. We are currently planning further experimental evaluations to measure the accuracy of the tool by building a gold standard based on the annotations of three software developers. Our tool is currently based on a heavy-knowledge approach but in our future work we will target a machine learning approach.

References

1. Bach, K., Harnish, R.M.: Linguistic Communication and Speech Acts. MIT Press, Cambridge, MA (1979)
2. Wilson, D., Sperber, D.: Relevance theory. Handbook of pragmatics (2002)
3. Stolcke, A., Coccaro, N., Bates, R., Taylor, P., Van Ess-Dykema, C., Ries, K., Shriberg, E., Jurafsky, D., Martin, R., Meteer, M.: Dialogue act modeling for automatic tagging and recognition of conversational speech. Comput. Linguist. **26**(3) (September 2000) 339–373
4. Cunningham, H., Maynard, D., Bontcheva, K., Tablan, V., Aswani, N., Roberts, I., Gorrell, G., Funk, A., Roberts, A., Damljanovic, D., Heitz, T., Greenwood, M.A., Saggion, H., Petrak, J., Li, Y., Peters, W.: Text Processing with GATE (Version 6). (2011)
5. Searle, J.R.: Speech acts: An essay in the philosophy of language. Volume 626. Cambridge university press (1969)
6. Cowie, J., Lehnert, W.: Information extraction. Commun. ACM **39**(1) (January 1996) 80–91
7. Ahrenberg, L., Andersson, M., Merkel, M.: A knowledge-lite approach to word alignment. In: Parallel Text Processing. Springer (2000) 97–116
8. Jurafsky, D., Shriberg, L., Biasca, D.: Switchboard SWBD-DAMSL shallow-discourse-function annotation coders manual, draft 13. Technical report, University of Colorado at Boulder Technical Report 97-02 (1997)
9. Twitchell, D.P., Nunamaker, J.F., J.: Speech act profiling: a probabilistic method for analyzing persistent conversations and their participants. In: Proceedings System Sciences, 2004. (2004) 10 pp.
10. Carvalho, V.R., Cohen, W.W.: On the collective classification of email speech acts. In: Proceedings of the 28th annual international ACM SIGIR conference on Research and development in information retrieval, ACM (2005) 345–352
11. Ravi, S., Kim, J.: Profiling student interactions in threaded discussions with speech act classifiers. Frontiers in Artificial Intelligence and Applications **158** (2007) 357
12. Knauss, E., Damian, D., Poo-Caamano, G., Cleland-Huang, J.: Detecting and classifying patterns of requirements clarifications. In: RE 2012. (2012) 251–260
13. Carreño, L.V.G., Winbladh, K.: Analysis of user comments: an approach for software requirements evolution. In: ICSE, IEEE / ACM (2013) 582–591