Textual Analysis or Natural Language Parsing? A Software Engineering Perspective

Sebastiano Panichella
University of Zurich

***Abstract*—The problem of designing effective methodology to summarize, and analyze the amount of textual information produced by developers remains particularly challenging especially when the goal is to help developers in making better development/maintenance decisions. Moreover, contrasting results might be obtained depending on the communication channel being mined and the technique adopted for its analysis. In our work we investigate the usage of Natural Language Parsing (NLP) and Textual Analysis (TA) techniques to automatically classify development content. Results of our study highlight the superiority of NLP techniques over the traditional TA techniques when used to analyze the textual data produced in software development. We also show the benefits of NLP when used to enhance software engineering recommenders.**

***Index Terms*—Unstructured Data Mining, Natural Language Parsing, Empirical Study.**

I. Context

In many open sources and industrial projects, developers make an intense usage of written communication channels, such as mailing lists, issue trackers and chats. Developers' communication contains valuable information for guiding developers to accomplish different maintenance and evolution tasks. For example, such information have been extensively analyzed by software engineering researchers to build recommenders aimed at performing bug triaging, profiling developers, re-documenting existing source code [2], or simply classifying textual development content [1]. However, because of the rapid increasing of the amount of textual data to analyze, the ability to summarize and understand such information remain challenging.

II. Motivational Example

We argue that approaches based on NLP analysis can be highly beneficial to determine the information needed by developers for making better development/maintenance decisions. The usage of NLP is motivated by the need of better capturing the intent of a sentences in a discussion, a task for which techniques based on lexicon analysis, such as Vector Space Models (VSM), Latent Semantic Indexing (LSI), or Latent Dirichlet Allocation (LDA) would not be sufficient. For example, let us consider the following two sentences:

1) *We could use a leaky bucket algorithm to limit the bandwidth.*
2) *The leaky bucket algorithm fails in limiting the bandwidth.*

A topic analysis technique (e.g., LDA) will reveal that these two sentences are likely to discuss the same topics: *"leaky bucket algorithm"* and *"bandwidth"*. However, these two sentences have completely different *intentions*: in sentence (1) the writer proposes a solution for a specific problem, while in sentence (2) the writer points out a problem. This example highlights that understanding the *intentions* in developers' communication could add valuable information for guiding developers in detecting text content useful to accomplish different and specific maintenance and evolution tasks.

III. Empirical Study and Major Findings

In our work we propose an approach, named DECA (Development Email Content Analyzer), that uses NLP to capture linguistic patterns and classify emails' content [3]. Thus, we investigate the use of NLP and TA techniques to automatically classify development content in according to a taxonomy of high-level categories of sentences, obtained by manually classifying development emails using grounded theory: *feature request*, *opinion asking*, *problem discovery*, *solution proposal*, *information seeking* and *information giving*.

Results of our study highlight the limits of traditional TA techniques and the superiority of our NLP technique when used to classify development content. Specifically, DECA outperforms traditional ML techniques in terms of recall, precision and F-Measure when classifying e-mail content. Moreover, we also successfully used DECA for re-documenting source code of two Java systems, improving the recall, while keeping high precision, of a previous approach based on adhoc heuristics.

IV. Perspective

The proposed approach can be used for a wider application domain, such as the preprocessing phase of various summarization tasks (as partially showed in our study). For example, it could be used as a preprocessing support to discard irrelevant sentences within emails or bug report summarization approaches. Furthermore, DECA can be used in combination with topic models for retrieving contents with the same intentions and treating the same topics from developers discussions in order to plan a set of change activities.

References

[1] A. Bacchelli, T. Dal Sasso, M. D'Ambros, and M. Lanza. *Content classification of development emails*. In Proceedings of the 34th International Conference on Software Engineering (ICSE), 2012, pp. 375-385.

[2] S. Panichella, J. Aponte, M. Di Penta, A. Marcus, and G. Canfora, *Mining source code descriptions from developer communications*. In Proceedings of the 20th IEEE International Conference on Program Comprehension (ICPC), 2012, pp. 63-72.

[3] A. Di Sorbo, S. Panichella, C. Visaggio, M. Di Penta, G. Canfora, H. Gall, *Development Emails Content Analyzer: Intention Mining in Developer Discussions*. Proceedings of the 30th international conference on Automated Software Engineering (ASE 2015), 2015, To appear.