Mining Relations from Git Commit Messages — an Experience Report

Rikard Andersson*, Morgan Ericsson†*, Anna Wingkvist†

*Chalmers | University of Gothenburg
Gothenburg, Sweden
mail@rikardandersson.se, morgan@cse.gu.se

†Linnaeus University
Växjö, Sweden
anna.wingkvist@lnu.se

1. Introduction

Software repositories contain rich data that can be used to uncover interesting and actionable information about software systems, projects, and development. Hassan and Xie (2010) coin the term *software intelligence* as a parallel to business intelligence and suggests that up-to-date information derived from software repositories is useful to developers, decision makers, and researchers. Much of this data, such as specifications, bug reports, etc. is expressed in free form natural language. We focus on commit messages, short texts written by developers to describe changes to the software repository. Our hypothesis is that such messages contain one or more relations that capture the intent(s) of the commit. For example, the commit message "Fixes #1097" with a "Fixes"-relation between the author and the issue #1097, suggests that the intent of the commit was to fix that specific issue.

We applied our relation extractor, Relex (Andersson, 2014), that performs well on web-style English texts to commit messages on Github and found a significant drop in performance compared to the text types we trained our relation extractor on. We describe the experiment setup and results, and perform a qualitative study to find the reasons for the drop in performance. We find that the grammar and structure used in commit messages combined with heavy use of domain specific terms made it difficult to find sentence boundaries, tokens, and part-of-speech (POS) tags, which Relex needs.

2. Experiment

We randomly selected and annotated 600 commit messages[1] from the *MSR 2014 Mining Challenge Dataset* (Gousios, 2013). Each message was annotated with relations and entities by a single person. However, the annotation guidelines (c.f., Appendix B in Andersson (2014)) and some examples where first discussed among the three authors. Since many commit messages refer to actions of an implicit Author entity (e.g., "Update CHANGELOG to mention the json_escape change") we consider relations between two explicit entities (Entity relations) and between an implicit author and an explicit entity (Author relations). To manage the latter, we add a mock Author entity at the beginning of every sentence. In the 600 messages there are 66 true Entity relations and 338 true Author relations.

Relex consists of a three-stage pipeline that (i) preprocesses the input, (ii) classifies binary candidate relations as true or false, and (iii) label the relation. In (i) we use Apache OpenNLP to detect sentences, tokenize these, and annotate each token with its POS. In (ii) we generate all possible binary relations between Noun Phrases (NP), Named Entities (NE), or a combination. These relation candidates are then classified as true or false using supervised learning (SVM). We rely on bag-of-words and POS sequences as features (Merhav et al., 2012; Banko et al., 2007). In (iii) we label true relations using a simple heuristic, inspired by Xu et al. (2013) and Etzioni et al. (2011), based on POS sequences between two candidates: (a) last verb, (b) last noun, or (c) last token.

We trained the three models (NE-NE, NE-NP, and NP-NP) in (ii) on data from CoNLL 2011 shared task (Pradhan et al., 2011). The relations for the NE-NE set contain all pairs of named entities, and a relation is considered true if it is found in the PropBank predicate-argument annotations. In the NE-NP and NP-NP datasets false relations are found by discriminating based on a dependency tree. We use the distance between the first and second entity in the dependency tree as a discriminator. If the distance is greater than three we label the candidate as false, less than three as positive, and if it is precisely three we leave it out entirely. Additionally, the arguments (noun phrases or named entities) must follow the predicate pattern with a subject and an object to be considered a true relation.

To evaluate the binary classifier in (ii) we use each of the three models to identify either Entity or Author relations from the pre-processed commit messages, and measure the Precision, Recall, and F-measure. Table 1 shows that the performance is significantly worse, no matter which model we use. Since (iii) depends on (ii) we only evaluate it on known correct data, i.e., the annotated true binary relations. Table 2 shows the ratio of suggested labels that are identical to the manual annotation.

3. Why did we lose accuracy?

We found that automatically generated commit messages occurred often and were problematic for Relex. These messages follow a wide range of patters, e.g., "svn

[1] Available at https://github.com/Rikard-Andersson/GHTorrent-Brat.

Relation type	Model	Git commit messages Precision	Recall	F-measure	CoNLL 2011 shared task Precion	Recall	F-measure
Entity	NE–NE	0.848	0.141	0.242	0.821	0.874	0.847
	NE–NP	0.273	0.167	0.207	0.840	0.745	0.790
	NP–NP	0.409	0.205	0.273	0.809	0.793	0.801
Author	NE–NE	0.556	0.242	0.337			
	NE–NP	0.473	0.432	0.452			
	NP–NP	0.701	0.507	0.589			

Table 1: Performance of the binary classifier on Git and CoNLL data.

Relation type	Accuracy Git	CoNLL (avg.)
Entity relations	0.515	0.947
Author relations	0.675	
Both kinds of relations	0.649	

Table 2: Comparing accuracy of the relation labelling.

path=/trunk/mono/; revision=111467" and "Merge pull request #370 from Memphiz/airplay". Messages such as the former should be disregarded, but given the vast number of tools and patterns the process to identify and filter these messages should be automated.

The syntax and grammar used in commit messages are often not of the same standard as the English sentences we trained on and the vocabulary is different (i.e., domain specific). This causes problems for the sentence detection and POS tagging. Punctuation and capitalization are used differently from well-formed English sentences. Dots occur in tokens such as file or method names, e.g., "Fix check for browser.mozilla so that Safari is not flagged as mozilla" is interpreted as: "Fix check for browser.mozilla" and "so that Safari is not flagged as mozilla". We also found that semicolon and linefeed were commonly used to delimit sentences. We also found several messages with poor sentence structure (or grammar), e.g., "extracted the built in profiling out added pp=profile-gc-time". Since Relex relies on sentences as a boundary for relations and to find a suitable mock-entity for Author relations, incorrectly detected sentence boundaries will reduce the accuracy.

The domain specific vocabulary causes problems for the POS tagger, e.g., *commit* and *pull (request)* are often used as nouns. The relation in "postgresql conflicts with postgres-xc" should be trivial to detect, but the binary classifier fails since *postgresql* and *postgres-xc* are considered as adjectives. Incorrect POS tags can also result in incorrect labels; in "cluster: Rename destroy() to kill(signal=SIGTERM)" *Rename* is considered an adjective, so the relation becomes (*cluster*, *destroy()*, *kill(signal=SIGTERM)*). POS tags and sequences of these represent half of the features for the binary classifier and all of the features for the relation phrase extractor. When this pre-processing step fails, further processing will also falter.

Many problems can be attributed to specific use of syntax and terminology. However, there are examples where sentence detection and POS tagging is correct yet the classification fails due to different language use, e.g., Relex finds a true relation, (*Use*, *mono_save_args*, *ARGSTORE*) in "(mono_save_args): Use ARGSTORE instead of TEMPSTORE to handle soft float correctly." together with four false. Similarly, Relex finds an Entity relation between *login shell* and *#59* "Start login shell (fixes #59 github issue) (per Austin Clements)" rather than an Author relation to *#59*. These examples illustrate how differently syntax and vocabulary are used in commit messages compared to the more standard language in the CoNLL data we trained on.

4. Conclusions

We identified several issues that cause Relex to perform poorly; mostly based on the poor fit between Relex and the data. The sentence detector, POS tagger, and binary classifier all fail since there is a mismatch between model and data. So, should we use different models or algorithms that do not rely on models to improve accuracy?

There are, to our knowledge, no models trained specifically for commit messages. Existing models are either specific to their domain or generic to well-formed English sentences, i.e., what we use. In our experience, it is not a good option to create new models for this purpose; the language use differs between projects, third party software, and even individuals. To make model creation feasible, one could investigate repositories used by a large, single body, e.g., a company. We then expect better guidelines and thus, more uniform syntax and language use.

Both the binary classifier and the relation phrase extractor could benefit from models trained on hand-annotated, domain specific data. For the relation phrase extractor, an algorithm such as conditional random fields could be used. However, features used by the algorithm would still need domain specific models. Also, it is difficult for an annotator to comprehend what the commit message author tries to convey when the syntax and terminology is not familiar to the annotator or when the language strays too far from conventional English.

Another option is to use different features and algorithms that are not as sensitive to noisy data. One possibility is less supervised machine learning algorithms, such as semi-supervised, self-supervised, and distantly supervised. These have all shown previous success for NLP. Given the structured data available in software repositories we intend to investigate distantly supervised learning next.

Acknowledgements

Relex was developed as part of a Master's thesis in collaboration with Findwise (`http://www.findwise.com`).

References

R. Andersson. 2014. Mining Relations from Git Repositories. Master's thesis, Chalmers, Sweden.

M. Banko, M.J. Cafarella, S. Soderland, M. Broadhead, and O. Etzioni. 2007. Open information extraction from the web. In *Proc. 20th Int. Conf. Artifical Intelligence (IJCAI'07)*, pages 2670–2676.

O. Etzioni, A. Fader, J. Christensen, S. Soderland, and M. Mausam. 2011. Open information extraction: The second generation. In *Proc. 22nd Int. Conf. Artifical Intelligence (IJCAI'11)*, pages 3–10.

G. Gousios. 2013. The ghtorrent dataset and tool suite. In *Proc. 10th Work. Conf. Mining Sw. Repositories (MSR'13)*, pages 233–236.

A.E. Hassan and T. Xie. 2010. Software intelligence: The future of mining software engineering data. In *Proc. FSE/SDP Works. Future of Sw. Eng. Research (FoSER'10)*, pages 161–166.

Y. Merhav, F. Mesquita, D. Barbosa, W.G. Yee, and O. Frieder. 2012. Extracting information networks from the blogosphere. *ACM Trans. Web*, 6(3):11.

S. Pradhan, L. Ramshaw, M. Marcus, M. Palmer, R. Weischedel, and N. Xue. 2011. CoNLL-2011 shared task: Modeling unrestricted coreference in OntoNotes. In *Proc. 15th Conf. Computational Natural Language Learning*, pages 1–27.

Y. Xu, M-Y Kim, K. Quinn, R. Goebel, and D. Barbosa. 2013. Open information extraction with tree kernels. In *Proc. Conf. NAACL HLT*, pages 868–877, June.