Automated Measurement and Analysis of Open Source Software

Project Number 318736

D4.2 – Question/Answer Extraction System from Online Threads

Version 1.1
13 June 2013
Final

Public Distribution

University of Manchester

Project Partners: **Centrum Wiskunde & Informatica, SOFTEAM, Tecnalia Research and Innovation, The Open Group, University of L'Aquila, UNINOVA, University of Manchester, University of York, Unparallel Innovation**

PROJECT PARTNER CONTACT INFORMATION

Centrum Wiskunde & Informatica Paul Klint Science Park 123 1098 XG Amsterdam, Netherlands Tel: +31 20 592 4126 E-mail: paul.klint@cwi.nl	**Softeam** Alessandra Bagnato Avenue Victor Hugo 21 75016 Paris, France Tel: +33 1 30 12 16 60 E-mail: alessandra.bagnato@softeam.fr
Tecnalia Research and Innovation Jason Mansell Parque Tecnologico de Bizkaia 202 48170 Zamudio, Spain Tel: +34 946 440 400 E-mail: jason.mansell@tecnalia.com	**The Open Group** Scott Hansen Avenue du Parc de Woluwe 56 1160 Brussels, Belgium Tel: +32 2 675 1136 E-mail: s.hansen@opengroup.org
University of L'Aquila Davide Di Ruscio Piazza Vincenzo Rivera 1 67100 L'Aquila, Italy Tel: +39 0862 433735 E-mail: davide.diruscio@univaq.it	**UNINOVA** Pedro Maló Campus da FCT/UNL, Monte de Caparica 2829-516 Caparica, Portugal Tel: +351 212 947883 E-mail: pmm@uninova.pt
University of Manchester Sophia Ananiadou Oxford Road Manchester M13 9PL, United Kingdom Tel: +44 161 3063098 E-mail: sophia.ananiadou@manchester.ac.uk	**University of York** Dimitris Kolovos Deramore Lane York YO10 5GH, United Kingdom Tel: +44 1904 325167 E-mail: dimitris.kolovos@york.ac.uk
Unparallel Innovation Nuno Santana Rua das Lendas Algarvias, Lote 123 8500-794 Portimão, Portugal Tel: +351 282 485052 E-mail: nuno.santana@unparallel.pt	

DOCUMENT CONTROL

Version	Status	Date
0.8	First draft	25 April 2013
0.9	Draft version with additional tables	7 May 2013
1.0	QA review	31 May 2013
1.1	Final version	13 June 2013

TABLE OF CONTENTS

EXECUTIVE SUMMARY

The target of OSSMETER Task 4.2 is to develop unsupervised classifiers for classifying online communication messages into two coarse grained classes, Questions or Answers. The meaning of Questions and Answers in this context is different than the everyday use of these words. For the purposes of this analysis, questions are requests, i.e. communication messages that report some bug or problem of a piece of Open Source Software (OSS) or some difficulty in using the OSS and ask for a helpful reply. Answers are reply messages that respond to request messages or any other communication within an OSS community, e.g. announcements. Consequently, the task is challenging. Not all requests are expressed as interrogative sentences and also replies can be expressed interrogatively.

To investigate this task, we created a new corpus of approximately 1,000 online communication messages, downloaded from popular communications channels: NNTP newsgroups and the bug tracking systems Bugzilla and Github. We chose to download messages relevant to OSS for which OSSMETER industrial partners have expressed their interest, in the requirements documentation. Corpus messages were manually annotated as requests or replies.

While observing the corpus, we identified various textual properties that seemed to be characteristic in requests or replies. We used these observations to build simple unsupervised classifiers and we evaluated these classifiers on the corpus, by comparing classification predictions to the manually assigned classification values. We conclude that these unsupervised classifiers can actually achieve a good level of accuracy score for this task. However, since this classification is really crucial for the text processing workflow that we are building in OSSMETER work-package 4, we plan to investigate whether the accuracy level can be further improved. For this purpose, we aim to build a supervised classifier that uses our practical observations as features, in the context of task 4.3.

1. INTRODUCTION

1.1 OVERVIEW

In this document, we present our progress in relevance to the OSSMETER project task 4.2: "Extraction of Questions and Answers from Online Threads". The goal of this task is to develop a classifier able to distinguish between messages that pose a request to the online community interested in some specific piece of Open Source Software (OSS), and messages that intend to reply to request messages. The task was briefly discussed in section 3.8 of deliverable 4.1 and its position in the text processing workflow was illustrated in the block diagram, Figure 1, deliverable 4.1.

In section 2 of the current document we present the corpus of online communication messages that was created to serve as the experimental base for investigating task 4.2. The corpus consists of approximately 1,000 communication messages downloaded from the bug tracking systems Bugzilla and Github as well as from NNTP newsgroups relevant to OSS of interest to our industrial collaborators. Communication messages were annotated manually as requests or replies. Apart from this task, the corpus will be also used in task 4.3 "Thread analysis and classification", after being enriched with further annotations. Since task 4.3 involves the development of supervised classifiers, the corpus will be also used for training.

In section 3, we present our experimentation for building unsupervised classifiers able to predict if a given message is a request or a reply one. The task is quite challenging, since a request is not always formed as an interrogative sentence. In addition, interrogative sentences do not always indicate requests. We evaluated various shallow text processing methods and we report results in detail. We conclude that the best performance achieved is probably not adequate for the purposes of OSSMETER, thus we intend to use our observations as features to build a supervised classifier in task 4.3.

1.2 RELATED LITERATURE

Although there is extensive research in Natural Language Processing on question answering (QA) [1, 2], it mainly focuses on constructing answers from certain types of questions from a large document collection. This is significantly different from the current task, to classify online communication messages as requests and replies. The nature of OSS-related online forums, which contain noisy data, make the task quite challenging. Although there is no directly related literature dealing with the specific task in hand, there is a limited amount of literature performing classification tasks in the domain of online forum threads.

The closest piece of published research work to the current task is presented in [3, 4, 5]. The authors have constructed a corpus of online threads downloaded from the CNET forums. Entire online threads were downloaded, only. Then, the messages of each thread were classified as problems, solutions or miscellaneous and then further classified into subclasses that were defined inside each coarse grained class. The best classification F-

measure score reported is in the area of 69%, for the solutions class. In [4, 5], it is also attempted to predict whether a thread discussion is complete, solved or spam using supervised machine learners.

In [6], a general framework based on Conditional Random Fields is proposed to detect the contexts and answers of questions from forum threads. In [7], it is attempted to classify forum posts according to dialogue acts and, then, structure thread messages as a discourse. In [8], a method is proposed to detect attitudinal sentences in thread discussions, as well as the type of attitude expressed.

1.3 INTENTIONS

We intend to use the corpus of online communication messages, that was developed and annotated for task 4.2, in task 4.3, as well. In task 4.3 we plan to build a supervised multiple-class classifier for deciding the type of messages. The corpus after some enrichment in terms of annotations will be used for development and training of the supervised classifier.

We intend to use the evaluation results of the investigation presented in the current document so as to select the most informative textual properties to be used as features of the supervised classifier that will be developed in task 4.3.

1.4 OUTCOME

The outcomes of the research presented in this document are the following:

- A corpus consisting of online communication messages, downloaded from online newsgroups and bug tracking systems about OSS of interest to our industrial partners. The corpus includes a manual annotation for each message, identifying it as a request or reply message.
- An evaluation of a variety of simple unsupervised classification methods on the corpus of online communication messages. An investigation of other useful features for this classification task.

2. ONLINE COMMUNICATION CORPUS

2.1 OVERVIEW

In this section, we present the development of a corpus of approximately 1,000 online communication messages. In section 2.2, we discuss the source and nature of the messages. In section 2.3, we discuss details about the corpus annotations. In section 2.4, we present a number of messages that comprise difficult instances for a request/reply classifier.

2.2 CORPUS CREATION

Task 4.1 involves developing a classifier able to decide if an online communication channel message is a request or a reply message, i.e. if it reports a bug or asks for some help from the community related to a piece of Open Source Software (OSS) or replies to a request submitted previously. As discussed in the OSSMETER proposal and deliverable 4.1, the classifier should be unsupervised, i.e. no machine learner is involved. Alternatively, the classifier should be based on various characteristics of the messages, such as the presence of interrogative sentences. To investigate the extent to which these characteristics can aid in deciding whether a online communication message expresses a request or replies to a request expressed previously we created a corpus consisting of a selection of communication instances.

In deliverable 4.1, it was mentioned that we had already developed an NNTP newsgroup reader using a freely available application programming interface (API). Now, we have developed similar readers for the bug-tracking systems Bugzilla and Github, which have been discussed in deliverable 4.1. In addition, the NNTP newsgroup and the Bugzilla readers have already been adapted to the OSSMETER platform, currently under development in work-package 5.

We employed the NNTP newsgroup, Bugzilla and Github readers to create the online communication corpus. To ensure that the projects that are of interest to our industrial collaborators, we tried to locate NNTP newsgroups, Bugzilla bugs and Github issues relevant to their chosen projects in the industrial requirements document.

Communication Channel Type	***Channels (#)***	***Messages (#)***	***Requests (#)***	***Replies (#)***
Bugzilla	8	410	131	279
Github	22	412	98	314
NNTP newsgroups	4	208	77	131
Total:	34	1030	306	724

***Table 1**: Online communication corpus statistics*

Table 1 presents some high-level statistics of the corpus. Approximately 400 messages were downloaded from Bugzilla and Github, while approximately 200 messages were downloaded from NNTP newsgroups. We chose to download fewer messages from newsgroups, due to the fact that we found fewer newsgroups about projects of interest

to the industrial partners than Bugzilla bugs and Github issues. The last two columns refer to the annotations added to the downloaded messages and will be discussed in section 2.3.

Product	*Comments (#)*	*Selection*	*Requests (#)*	*Replies (#)*
Bugzilla	52	Random comments	13	39
Fedora	52	Random comments	13	39
Issue-Tracker	52	Random comments	22	30
Pulp	52	Random comments	24	28
Red Hat Database	46	All comments	18	28
Red Hat Enterprise Linux 7	52	Random comments	10	42
Red Hat Linux	52	Random comments	10	42
Topic Tool	52	Random comments	21	31
Total:	410		131	279
Bugzilla server:	bugzilla.redhat.com			

***Table 2**: Statistics of the Bugzilla sub-corpus*

Table 2, 3 and 4 show detailed statistics of the Bugzilla, Github and NNTP newsgroup parts of the online communication corpus, respectively. There were 8 Bugzilla channels chosen, 7 of which were longer and only 1, *Red Hat Database*, shorter than 50 comments. All comments of the latter were downloaded, while just 52 random selected comments were downloaded from all the other 7 channels.

Project	*Issues (#)*	*Comments (#)*	*Selection*	*Requests (#)*	*Replies (#)*
acts_as_geocodable	3	5	All comments	1	4
amazon-ec2	5	6	All comments	1	5
attachment_fu	14	43	All comments	10	33
audited	11	15	All comments	3	12
braid	3	6	All comments	1	5
cache_fu	2	3	All comments	0	3
capsize	1	4	All comments	3	1
chronic	15	56	Random comments	13	43
enum_field	1	1	All comments	0	1
eycap	7	11	All comments	2	9
forgery	7	16	All comments	5	11
git-wiki	1	4	All comments	1	3
god	10	23	All comments	5	18

grit	17	24	All comments	4	20
low-pro-for-jquery	1	1	All comments	0	1
resource_controller	5	8	All comments	2	6
restful-authentication	14	23	All comments	8	15
rubinius	16	56	Random comments	10	46
ruby-git	10	23	All comments	5	18
ruby-on-rails-tmbundle	11	25	All comments	8	17
signal-wiki	2	3	All comments	1	2
thin	20	56	Random comments	15	41
Total:	176	412		98	314
Github URL:	https://api.github.com/repositories				

***Table 3**: Statistics of the Github sub-corpus*

Table 3 shows similar details from comments downloaded from Github. Due to the limited number of Github repositories on the main server *(https://api.github.com/repositories)* we employed all projects on the server and just performed a random selection of comments for popular projects. The second column mentions the number of issues which contain the comments of the third column. Consequently, in the case of random selected comments, the actual number of issues available might be greater than the number mentioned in the second column.

NNTP Server	***Newsgroup***	***Articles (#)***	***Selection***	***Requests (#)***	***Replies (#)***
news.eclipse.org	eclipse.technology.subversive	52	Random articles	17	35
news.eclipse.org	gmane.comp.java.sonar.general	52	Random articles	15	37
news.eclipse.org	eclipse.hudson	52	Random articles	20	32
news.gmane.org	eclipse.platform	52	Random articles	25	27
Total:		208		77	131

***Table 4**: Statistics of the NNTP newsgroup sub-corpus*

Table 4 shows statistics of the messages downloaded from NNTP newsgroups. A random selection of 52 articles was downloaded from each of the 4 newsgroups about OSS of interest to our industrial collaborators.

2.3 CORPUS ANNOTATION

The online communication channel messages that were downloaded form Bugzilla, Github and NNTP newsgroups, as described in the previous section, were stored in XML formatted files. Each XML file contains downloaded messages from a single Bugzilla channel, Github channel or NNTP newsgroup.

To investigate the extent to which simple (shallow) linguistic characteristics can serve as unsupervised features to build a classifier able to distinguish between request and reply messages, each message was annotated manually as request of reply by a computational linguist. An extra XML field was added to the XML schema to host this annotation.

The fundamental criterion for annotating a communication channel message is whether it hosts a new request of a user to the community interested in some piece of OSS. There is a variety of different types of requests:

- A message reporting a newly discovered bug.
- A message asking whether a previously reported bug has been fixed.
- A message reporting a difficulty in installing or using the OSS.
- A message stating that the user is facing a previously reported problem, bug or difficulty.

All other messages are considered as replies. Clearly, the class of reply messages is broader than the class of requests. This fact is also verified by the numbers of the two rightmost columns of tables 1, 2, 3 and 4, which report the numbers of requests and replies for the entire corpus and the sub-corpora downloaded from Bugzilla, Github and NNTP newsgroups, respectively. It can be observed that the reply/request ratios for Bugzilla (2x), Github (3x) and NNTP (2x) are different, in particular approximately 2, 3 and 2, respectively. This difference might be due to the different nature of discussions or might also be depending on the process of selecting messages randomly.

The main reason for having more reply than request messages is that the range of reply messages is much wider. It contains:

- All communication between developers and users after a bug is reported and before the bug is fixed.
- User replies after the bug is fixed or it is decided that it cannot be fixed at a given time.
- Communication between developers that concerns their progress on fixing bugs and improving the OSS.
- Communication between developers for assigning and reassigning bugs and programming jobs to each other.
- Notices and announcements for users made by the developers about news, new releases or other changes concerning the lifecycle or the availability of an piece of OSS.

The rationale behind the division of communication messages into the request and reply classes is based on the usability of the text processing workflow within the OSSMETER project. Since the overall target is to score online communications in terms of quality of provided user support, we choose to classify together messages that are considered to contribute similarly to the evaluation score, either positively or negatively. In particular,

request messages are considered negative towards the quality score: a large number of requests paired with a small number of replies probably indicates that the developers are cannot adequately support users. A large number of reply messages indicates increased activity levels and probably successful handling of user requests.

However, very few or zero request messages might be an indication of absence of interest from the side of users, and thus, should be considered as negative towards the quality score. The effect of request and reply messages will become clearer when messages are organised in threads. Then, we will be able to measure the number of requests that remain unaddressed and also the actual time between a request and the corresponding reply.

2.4 AUTOMATIC ANNOTATION CHALLENGES AND NOTEWORTHY CASES

Automatically annotating request and reply bugs, as defined in the previous section, presents a number of challenges. Question marks and/or WH question words, i.e. words that introduce a question, such as *what*, *who* and *where*, could potentially be considered as indications of requests. Question marks are only present in direct interrogative sentences. In written communications, it is very usual that questions are expressed indirectly. Sometimes questions are expressed directly but the question mark is omitted, although this is not syntactically correct. In indirect questions, the WH question words are still present.

The hypothesis that request messages are characterised by the presence of question marks or WH question words might be valid to a certain extent, however there are noteworthy exceptions. Tables 5, 6 and 7 present request messages from two NNTP newsgroups and a bugzilla project. None of the messages contain direct or indirect interrogative sentences. In table 5, it is mentioned that the user encountered some error. Similarly, in tables 6 and 7, reported deficiencies are expressed by the words “failed” and “incorrect”.

NNTP Server	*news.eclipse.org*
Newsgroup	*eclipse.hudson*
Annotated as	Request
Text	I just started the Tomcat server, which is hosting Hudson, using jdk1.6.0_27 as JAVA_HOME. I still get the error, when Hudson attempts to send mails.

Table 5*: Example of a request, where there is no question mark or WH question word used.*

NNTP Server	*news.eclipse.org*
Newsgroup	*eclipse.hudson*
Annotated as	Request
Text	I do have the groovy-support plugin installed. I reduced the groovy script to only a single println and even that failed.

Table 6*: Example of a request, where there is no question mark or WH question word used.*

Bugzilla Server	*bugzilla.redhat.com*
Project	*Red Hat Linux*
Annotated as	Request
Text	tin in its default configuration tries to read usr/lib/news/active, when a newsspool operation is requested. This is incorrect, active is in /var/lib/news under RH.

Table 7*: Example of a request, where there is no question mark or WH question word used.*

In contrast to tables 5, 6 and 7, table 8 presents a reply message that contains both a question mark and a WH question word. It is a direct interrogative sentence, but concerns communication between the developer that attempts to address the request and the user that reported the deficiency, initially.

NNTP Server	*news.eclipse.org*
Newsgroup	*eclipse.hudson*
Annotated as	Reply
Text	What version of jboss as are you deploying to (According to the stack trace it's version 5x)?

Table 8*: Example of a reply, where there is a question mark or WH question word. The message is classified as a reply, since it is a question asked by an OSS developer to an OSS user that has previously submitted a request message.*

Table 9 shows a reply message that contains the actual request message, indented with the greater than symbol (>). This feature is typical in email replies and also adopted by a number of OSS-related bug tracking systems and newsgroups. The presence of the request message text might fool a classifier irrespectively if it is supervised or not. To tackle this cases, we have applied a cleaning step that excludes lines of text indented with the greater than symbol, discussed in section 3.

Github URL	*https://api.github.com/repositories*
Project	*rubinius*
Annotated as	Reply
Text	> is that needed at all for this change? No, this is feature from ruby 2.0. Each element in `$LOAD_PATH ` is frozen and `$LOAD_PATH` is cached in exactly the same way. So I think we can remove that part from pull request. > Also, is there a reason for all the synchronization in it? I'm not sure - I was based on the implementation of `$LOAD_FEATURES`
 > I think we should address that separately so we don't mix different changes, which makes discussing and reviewing them harder. Agree, I will update this pull request and I will remove all changes related to `$LOAD_PATH`

Table 9*: Example of a reply, which contains the actual request message indented with the greater than symbol (>). A developer has replied by adding their answers in between indented questions.*

Tables 10 and 11 present two examples of cases of request messages difficult to recognise automatically. In table 9, a description of the problem typed by the user is accompanied with an error message output by the piece of software that is failing, i.e. Topic Tool. The message in table 10 is even harder than the one in table 9, since the entire text but the last sentence consists of the output of an installation process. The installation process fails to recognize that a gcc compiler is already installed. The last line of the message is typed in by the user and states that a gcc compiler is already installed in their system. In the next section, we discuss our experimentation with simple, unsupervised methods to classify the messages of the online communication corpus and present extensive evaluation results.

Bugzilla Server	*bugzilla.redhat.com*
Project	*Topic Tool*
Annotated as	Request
Text	Description of problem: Tool gracefully ignores the fact that a topic could not be read (sometimes due to a validity issue) and keeps on retrieving subsequent topics: [Fatal Error] :234:6: The element type "step" must be terminated by the matching end-tag "</step>". ERROR: Unable to load topic (http://topicrepo.englab.bne.redhat.com/TopicRepository/Tasks/IPA/Installing_the_IPA_Server.xml). ERROR: Unable to parse 'Installation_Guide_Export/en-US/Infrastructure.xml'. Ideally I think we want to allow this but keep track of which topics couldn't be read and display a list at the end of the run to highlight that it wasn't successful.

Table 10*: Example of a request, where there is no question mark or WH question word used. The message contains a copied and pasted description of the error. At the end of the message, the sender describes how the problem should be addressed, in their opinion.*

Bugzilla Server	*bugzilla.redhat.com*
Project	*Red Hat Linux*
Annotated as	Request
Text	checking host system type... i586-pc-linux checking target system type... i586-pc-linux checking build system type... i586-pc-linux checking for a BSD compatible install... /usr/bin/install -c checking whether build environment is sane... yes checking whether make sets ${MAKE}... yes checking for working aclocal... found checking for working autoconf... found checking for working automake... found checking for working autoheader... found checking for working makeinfo... missing checking whether make sets ${MAKE}... (cached) yes checking for gcc... gcc checking whether the C compiler (gcc) works... no

	configure: error: installation or configuration problem: C compiler cannot create executables. I have egcs and gcc installed.

***Table 11**: Example of a request, consisting just an installation log in which an error has occurred. The last line of the message is manually typed and mentions pieces of software already installed. The user intends to point out a contradiction: the installation fails although prerequisite software is already installed.*

3. UNSUPERVISED CLASSIFICATION METHODS

3.1 OVERVIEW

In this section, we present all details about our experimentation to inspect the extent to which simple textual based features are useful to build an unsupervised classifier for request as opposed to reply communication channel messages. Unsupervised classifier are not based on machine learning techniques, are not trainable and, thus, do not require annotated training instances. In section 3.2, we present classification methods. In section 3.3, we provide details about the evaluation procedure and we discuss evaluation results. Finally, in section 3.4, we investigate another feature useful for the current classification task.

3.2 CLASSIFICATION METHODS

In this section we discuss a number of classification criteria, useful to classify online communication messages as requests or replies. The classification criteria are based on observations on the corpus of messages, presented in section 2. They do not require any pre-processing of the input text, such as part-of-speech tagging or parsing, thus the resulting simple classification methods would be quick to execute and suitable for an online classification algorithm. The basic evaluated methods are:

- **Question mark method**: The method classifies as request all messages that contain question marks and as replies all messages that do not contain question marks.
- **RE method**: This method takes into account the subject of communication messages. If a message starts with *"RE: "* or *"Re: "* it is classified as reply, otherwise as request. Unfortunately, a separate subject field is not available in messages downloaded from Bugzilla or Github comments; thus, the method is only applicable to NNTP newsgroup articles.
- **Question words method**: This method is a generalisation of the question mark method, so as to capture indirect questions in addition to direct ones. The method looks for the following WH question words: *what, when, where, which, who, whom, whose, why* and *how*. After observing the corpus, we also added the words *help* and *please*, that are typically present in requests. Communication messages that contain one or more of these words are classified as requests, otherwise as replies. Matching is performed in a case insensitive manner.
- **Cleaning:** Cleaning is used as a method component rather than a method itself. It refers to removing text lines that start with the greater than symbol (>). As discussed in section 2.4, this is indicative of previous communication message text included in the current message.

Apart from the basic methods above, we have evaluated combinations of them:

- **Question mark or words method**: It classifies as request all messages that contain question marks or WH question words. Otherwise, messages are classified as replies.

- **RE Question mark method**: The method applies to NNTP newsgroup articles, only. It classifies as requests all messages whose subject does not start with *"RE: "* or *"Re: "*. The remaining messages are classified as requests if they contain a question mark or as replies, otherwise.
- **RE Question mark or words method**: A combination of the two methods above. NNTP newsgroup articles that are not classified as requests by the *RE method* are classified as requests if they contain question marks or WH question words. Otherwise, articles are classified as replies.

All the above simple and combinatorial methods were designed for the English language and the experiments presented in this document also concern messages in English. However, the methods are applicable to other languages directly or after some minor modifications. In particular, **Cleaning** can be applied directly to any language, since the greater than symbol (>) indentations is language independent. The remaining methods can be applied to other languages after translation. The **Question mark method** should be modified to capture the symbols that encode direct questions. The **RE method** should be modified to capture the initials or prefixes that denote "reply" and are used in the subject of reply emails. The **Question words method** should be modified to capture the words that introduce questions in other languages.

We experimented with all methods discussed above, with or without *Cleaning* as a preprocessing step. In addition, since the *RE method* is applicable to NNTP newsgroup articles only, we paired this method with other methods applied to the Bugzilla and Github part of the corpus to compute evaluation results over all online communication messages. Experimental results are discussed in the next section.

3.3 EVALUATION RESULTS

In this section, we discuss the evaluation measures employed and we present the experimental results of this investigation. Due to the nature of these experiments, accuracy, i.e. the number of correctly predicted instances over the total number of instances is adequate to measure performance. However, the accuracy score cannot indicate what is the performance for specific classification classes, i.e. how many request messages were classified as requests and how many of the actual replies were classified as replies. For this reason, apart from accuracy we computed precision and recall scores. For all score coputations we used the java class PrecisionRecallEvaluation[1], part of the LingPipe API[2].

Table 12 shows the evaluation results of 17 methods evaluated on the corpus of online communication messages, described in section 2. The *RE method* as well as combined methods which use the *RE method* as a component are only evaluated on NNTP newsgroup articles, because the subject field is not available in Bugzilla and Github messages. To measure the effect of these methods on the entire corpus, we classified the messages of the Bugzilla and Github sub-corpora using other applicable methods. For example, the method *Question mark (B, G) + RE (N)* classified Bugzilla *(B)* and Github

[1] Documentation of the PrecisionRecallEvaluation class is available at: alias-i.com/lingpipe/docs/api/com/aliasi/classify/PrecisionRecallEvaluation.html

[2] The LingPipe API is available at: alias-i.com/lingpipe

(G) comments according to the *Question mark* method and NNTP newsgroup articles according to the *RE method*. In these cases, the accuracy scores referring to the three sub-corpora are copied in grey colour.

The best performing method on the Bugzilla sub-corpus is *Question words* with *Cleaning* as a pre-processing step. However, the score is only slightly better than the scores achieved by any other method applicable. In contrast, the same methods tested on the Github sub-corpus lead to more variable results. The best accuracy is achieved by the *Question mark method* after *Cleaning*. In the NNTP newsgroup part, methods based on question marks and WH question words perform much worse that in the Bugzilla and Github part. Probably, this is due to the differences in the nature and usage of these three different sources. However, the *RE method* achieves a high accuracy score. Looking at the accuracy scores achieved for the entire corpus, the method *Cleaning, Question mark (B, G) + RE (N)* leads to the highest accuracy, 73.592%. It consists of the *Question mark method* after *cleaning* applied on Bugzilla and Github messages, combined with the *RE method* applied on NNTP newsgroup articles.

Method	***Bugzilla***	***Github***	***News-groups***	***Entire Corpus***
Question mark	67.561%	75.000%	51.923%	67.379%
RE	-	-	**81.731%**	-
Question mark (B, G) + RE (N)	67.561%	75.000%	81.731%	73.398%
RE Question mark	-	-	55.769%	-
Question mark (B, G) + RE Question mark (N)	67.561%	75.000%	55.769%	68.155%
Question words	69.756%	63.835%	44.712%	62.330%
Question mark or words	67.805%	63.592%	47.115%	61.942%
RE Question mark or words	-	-	47.115%	-
Question mark or words (B, G) + RE Question mark or words (N)	67.805%	63.592%	47.115%	61.942%
Cleaning, Question mark	67.561%	**75.485%**	64.904%	70.194%
Cleaning, Question mark (B, G) + RE (N)	67.561%	75.485%	81.731%	**73.592%**
Cleaning, RE Question mark	-	-	68.269%	-
Cleaning, Question mark (B, G) + Cleaning, RE Question mark (N)	67.561%	75.485%	68.269%	70.874%
Cleaning, Question words	**70.488%**	64.078%	52.404%	64.272%
Cleaning, Question mark or words	68.537%	64.078%	54.808%	63.981%
Cleaning, RE Question mark or words	-	-	54.327%	-
Cleaning, Question mark or words (B, G) + Cleaning, RE Question mark or words (N)	68.537%	64.078%	54.327%	63.883%

***Table 12**: Evaluation results – Accuracy of unsupervised methods tested on the online communication corpus*

The differences in accuracy of a method on difference sources may also be affected by the ratio of reply/request in the dataset. However, since the messages were selected randomly, we accept for this experimentation that the ratios in the dataset reflect the actual ratios in the entire resource, Bugzilla, Github or NNTP newsgroup.

Appendix A at the end of the current document presents detailed results of the experiments presented in Table 12. Table A.1, A.2, A.3 and A.4 shows the results for experiments evaluated on the Bugzilla sub-corpus, the Github sub-corpus, the NNTP newsgroup sub-corpus and the entire corpus, respectively.

The conclusion drawn from this experimentation is that simple textual characteristics are very important for classifying a communication message as request or reply. However, the accuracy achieved by the best performing method is probably inadequate for the purpose of OSSMETER. For this reason, we investigate another potentially useful source of features in the next section. We plan to merge all these features together with features encoding the words that occur in text into a supervised classifier, which might potentially lead to higher accuracy.

3.4 INVESTIGATION OF ANOTHER, PROBABLY USEFUL FEATURE

In this section we investigate an additional feature that might be useful as indicator for classifying messages as requests or replies. Frequent users are more likely to be developers and thus more likely to contribute reply than request messages. This is a reasonable claim, but needs to be inspected quantitatively.

For each part of the online communication message corpus, i.e. Bugzilla, Github and NNTP newsgroups, we counted the number of requests and replies of each user separately. Then we sorted the users in order of decreasing number of messages. For each user, we computed the percentage of requests and replies over all their messages collectively. If our initial claim is true, we expect to see large reply percentages at the top of the list. Traversing the list from top to bottom, reply percentages are expected to decrease while request percentages are expected to increase.

Tables 13, 14 and 15 show these lists for the NNTP newsgroup sub-corpus, the Github sub-corpus and the Bugzilla sub-corpus, respectively. For presentational reasons, we have grouped together users that have sent the same number of messages. For example, the last two rows of table 13 report that 19 users have sent just one message to some NNTP newsgroup and this message was a reply, while 41 users have sent only one message to some NNTP newsgroup and this was a request. The last two rows present the average percentage of requests and replies per user class. For example, in table 13, for the class of users that have sent 12 messages, a random selection from their messages is a request with probability 31.25% and a reply with probability 83.33%.

Users (#)	*Messages per user (#)*	*Requests per user (#)*	*Replies per user (#)*	*Requests per user (%)*	*Replies per user (%)*	*Average requests (%)*	*Average replies (%)*
1	42	20	22	47.62%	52.38%	47.62%	52.38%
1	38	15	23	39.47%	60.53%	39.47%	60.53%

1	26	4	22	15.39%	84.62%	15.39%	84.62%
1	22	14	8	63.64%	36.36%	63.64%	36.36%
1	16	2	14	12.50%	87.50%	31.25%	68.75%
1		8	8	50.00%	50.00%		
1	13	0	13	0.00%	100.00%	0.00%	100.00%
1	12	1	11	8.33%	91.67%	16.67%	83.33%
1		3	9	25.00%	75.00%		
1	11	3	8	27.27%	72.73%	27.27%	72.73%
1	10	6	4	60.00%	40.00%	60.00%	40.00%
1	9	7	2	77.78%	22.22%	77.78%	22.22%
1	7	0	7	0.00%	100.00%	7.14%	92.86%
1		1	6	14.29%	85.71%		
1	6	0	6	0.00%	100.00%	8.33%	91.67%
1	5	1	5	16.67%	83.33%		
1		2	3	40.00%	60.00%	40.00%	60.00%
1	4	0	4	0.00%	100.00%	46.43%	53.57%
1		1	3	25.00%	75.00%		
3		2	2	50.00%	50.00%		
2		3	1	75.00%	25.00%		
3	3	0	3	0.00%	100.00%	38.89%	61.11%
4		1	2	33.33%	66.67%		
5		2	1	66.67%	33.33%		
9	2	0	2	0.00%	100.00%	21.43%	78.57%
4		1	1	50.00%	50.00%		
1		2	0	100.00%	0.00%		
19	1	0	1	0.00%	100.00%	68.33%	31.67%
41		1	0	100.00%	0.00%		

***Table 13**: List of decreasingly active Bugzilla comment senders*

Looking at the tables reveals that there is a tendency of active users to send more replies than requests. However, it is neither very stable nor as clear as we would imagine theoretically. For example, there is a NNTP newsgroup user that has sent 22 messages, of which 63.64% are requests. This indicates that the behaviour of developers may differ per community, for instance some developers file issues to keep track of work to be done. In addition, to build an unsupervised classifier we would need to define some threshold of activity above which the corresponding user is considered as developer and his messages are considered more likely to be replies. Defining this threshold is problematic for a variety of reasons. Sometimes, a user starts with asking a lot of questions about some software and becomes more and more involved until they are able to reply to requests of other users. Moreover, there is danger of circular reasoning in this task: in order to classify messages as replies or requests, we need to distinguish between

developers and users and this influence in its turn the classification of messages that we started with.

The conclusion that can be made here is that this observed tendency would be a valuable feature for a trainable classifier, however an unsupervised method based on it, would most probably achieve results not higher that the accuracy levels achieved by the methods of table 12.

We have investigated the observation that the more active an online communication forum user is the more of their messages tend to be replies. Apart from this observation there may exist other correlations useful for the current classification task. For example, it is common that long time after a problem or bug is addressed, some user submits another relevant request. Thus, it might be useful to investigate whether there is a correlation between the time gap from the current message to the previous one and the class that the current message was assigned to, requests or replies. It seems theoretically reasonable that the longer the time gap from the previous message the more probably the current message is a request. Unfortunately, this claim cannot be tested on the corpus of online communication messages, discussed in section 2, because the messages in the corpus have been selected randomly and/or irrespectively of the actual bugs to which they were submitted. As a result the corpus does not always contain the previous of any given message. We plan to investigate this feature while developing a supervised classification system in task 4.3.

Users (#)	***Messages per user (#)***	***Requests per user (#)***	***Replies per user (#)***	***Requests per user (%)***	***Replies per user (%)***	***Average requests (%)***	***Average replies (%)***
1	19	2	17	10.53%	89.47%	10.53%	89.47%
1	18	1	17	5.56%	94.44%	9.26%	90.74%
2		2	16	11.11%	88.89%		
2	6	1	5	16.67%	83.33%	16.67%	83.33%
1	5	0	5	0.00%	100.00%	10.00%	90.00%
1		1	4	20.00%	80.00%		
1	4	0	4	0.00%	100.00%	30.00%	70.00%
3		1	3	25.00%	75.00%		
1		3	1	75.00%	25.00%		
8	3	0	3	0.00%	100.00%	25.00%	75.00%
4		1	2	33.33%	66.67%		
4		2	1	66.67%	33.33%		
22	2	0	2	0.00%	100.00%	26.67%	73.33%
22		1	1	50.00%	50.00%		
1		2	0	100.00%	0.00%		
113	1	0	1	0.00%	100.00%	28.93%	71.07%
46		1	0	100.00%	0.00%		

***Table 14**: List of decreasingly active Github comment senders*

Users (#)	*Messages per user (#)*	*Requests per user (#)*	*Replies per user (#)*	*Requests per user (%)*	*Replies per user (%)*	*Average requests (%)*	*Average replies (%)*
1	13	1	12	7.69%	92.31%	7.69%	92.31%
1	8	0	8	0.00%	100.00%	0.00%	100.00%
1	7	0	7	0.00%	100.00%	14.29%	85.71%
1		2	5	28.57%	71.43%		
1	5	1	4	20.00%	80.00%	20.00%	80.00%
1	4	0	4	0.00%	100.00%	0.00%	100.00%
2	3	0	3	0.00%	100.00%	26.67%	73.33%
2		1	2	33.33%	66.67%		
1		2	1	66.67%	33.33%		
9	2	0	2	0.00%	100.00%	25.00%	75.00%
3		1	1	50.00%	50.00%		
2		2	0	100.00%	0.00%		
59	1	0	1	0.00%	100.00%	51.24%	48.76%
62		1	0	100.00%	0.00%		

***Table 15**: List of decreasingly active NNTP newsgroup message senders*

4. RISKS

In the context of the current task, there were no risks identified. As discussed, in section 2.3, we concluded that the accuracy scores achieved by the evaluated unsupervised methods is not adequate for the purposes of this task. Potentially, there is a risk that the supervised improvement that we plan to develop does not improve the accuracy result further. However, in most natural language processing tasks, supervised methods are proven to perform better than unsupervised ones.

5. CONCLUSION

In this document, we have presented our progress concerning OSSMETER Task 4.2. To investigate this task, we have constructed a corpus consisting of approximately 1,000 online communication messages, relevant to OSS projects of interest to our industrial partners. In section 2, a description of the corpus was accompanied with details about the annotations that were added and a discussion of interesting message examples.

The corpus of online communication messages was used as an evaluation base for a number of unsupervised methods to classify messages as requests or replies. The types of message contents that should be classified as request or relies were specified in section 2.3, while the unsupervised classification methods are presented in section 3.2. Section 3.3 presents the results of this evaluation and section 3.4 presents a further investigation of other features potentially useful to the current classification task.

In conclusion, shallow unsupervised techniques were shown to achieve a good level of accuracy. However, since this task is fundamental for the text processing analysis in turn crucial to the entire OSSMETER project, we believe that the accuracy achieved is not adequate for this purpose. In the context of task 4.3, we plan to implement a supervised classifier that takes into account all features discussed in this document to address this shortage.

REFERENCES

[1] *Sanda Harabagiu, Finley Lacatusu, and Andrew Hickl.* 2006. **Answering complex questions with random walk models.** In Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval, SIGIR '06, pages 220–227, New York, NY, USA. ACM.

[2] *Hoa T. Dang, Diane Kelly, and Jimmy Lin.* 2007. **Overview of the TREC 2007 question answering track.** In NIST Special Publication: SP 500-274 The Sixteenth Text REtrieval Conference (TREC 2007) Proceedings.

[3] *Li Wang, Su N. Kim, and Timothy Baldwin.* 2010. **Thread-level analysis over technical user forum data.** In Proceedings of the Australasian Language Technology Association Workshop 2010, pages 27–31, Melbourne, Australia.

[4] Timothy Baldwin, David Martinez, Richard B. Penman. 2007. **Automatic thread classification for Linux user forum information access.** In Proceedings of ADCS 2007.

[5] *Timothy Baldwin, David Martinez, Richard B. Penman, Su N. Kim, Marco Lui, Li Wang, and Andrew MacKinlay*. 2010. **Intelligent Linux information access by data mining: the ILIAD project.** In Proceedings of the NAACL HLT 2010 Workshop on Computational Linguistics in a World of Social Media, WSA '10, pages 15–16, Stroudsburg, PA, USA. Association for Computational Linguistics.

[6] *Shilin Ding, Gao Cong, Chin-Yew Lin, and Xiaoyan Zhu.* 2008. **Using conditional random fields to extract contexts and answers of questions from online forums.** In Proceedings of ACL-08: HLT. Association for Computational Linguistics.

[7] *Su N. Kim, Li Wang, and Timothy Baldwin.* 2010. **Tagging and linking web forum posts.** In Proceedings of the Fourteenth Conference on Computational Natural Language Learning, CoNLL '10, pages 192–202, Stroudsburg, PA, USA. Association for Computational Linguistics.

[8] *Ahmed Hassan, Vahed Qazvinian, and Dragomir Radev.* 2010. **What's with the attitude?: identifying sentences with attitude in online discussions.** In Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing, EMNLP '10, pages 1245–1255, Stroudsburg, PA, USA. Association for Computational Linguistics.

APPENDIX A

***Table A.1**: Detailed evaluation results of unsupervised methods tested on the **Bugzilla** part of the online communication corpus – Contingency tables, number of correctly and incorrectly predicted instances, accuracy, precision, recall and F-measure scores "Ann" stands for annotations, while "Pred" stands for method predictions.*

Experiment information		Contingency table				Evaluation scores			
Experiment id:	**e01Bugzilla**		Pred:Request	Pred:Reply	Sum			Accuracy	**67.561%**
Method:	**Question mark**	Ann:Request	24	106	130	Correct	277	Precision	47.059%
Collection:	**Bugzilla (410)**	Ann:Reply	27	253	280	Incorrect	133	Recall	18.462%
		Sum	51	359	410			F-measure	26.519%
Experiment id:	**e02Bugzilla**		Pred:Request	Pred:Reply	Sum			Accuracy	**69.756%**
Method:	**Question words**	Ann:Request	74	56	130	Correct	286	Precision	52.113%
Collection:	**Bugzilla (410)**	Ann:Reply	68	212	280	Incorrect	124	Recall	56.923%
		Sum	142	268	410			F-measure	54.412%
Experiment id:	**e03Bugzilla**		Pred:Request	Pred:Reply	Sum			Accuracy	**67.805%**
Method:	**Question mark or words**	Ann:Request	78	52	130	Correct	278	Precision	49.367%
Collection:	**Bugzilla (410)**	Ann:Reply	80	200	280	Incorrect	132	Recall	60.000%
		Sum	158	252	410			F-measure	54.167%
Experiment id:	**e01cleanBugzilla**		Pred:Request	Pred:Reply	Sum			Accuracy	**67.561%**
Method:	**Cleaning + Question mark**	Ann:Request	23	107	130	Correct	277	Precision	46.939%
Collection:	**Bugzilla (410)**	Ann:Reply	26	254	280	Incorrect	133	Recall	17.692%
		Sum	49	361	410			F-measure	25.698%
Experiment id:	**e02cleanBugzilla**		Pred:Request	Pred:Reply	Sum			Accuracy	**70.488%**
Method:	**Cleaning + Question words**	Ann:Request	74	56	130	Correct	289	Precision	53.237%
Collection:	**Bugzilla (410)**	Ann:Reply	65	215	280	Incorrect	121	Recall	56.923%
		Sum	139	271	410			F-measure	55.019%
Experiment id:	**e03cleanBugzilla**		Pred:Request	Pred:Reply	Sum			Accuracy	**68.537%**
Method:	**Cleaning + Question mark or Words**	Ann:Request	78	52	130	Correct	281	Precision	50.323%
Collection:	**Bugzilla (410)**	Ann:Reply	77	203	280	Incorrect	129	Recall	60.000%
		Sum	155	255	410			F-measure	54.737%

***Table A.2**: Detailed evaluation results of unsupervised methods tested on the **Github** part of the online communication corpus – Contingency tables, number of correctly and incorrectly predicted instances, accuracy, precision, recall and F-measure scores "Ann" stands for annotations, while "Pred" stands for method predictions.*

Experiment information		Contingency table				Evaluation scores			
Experiment id:	**e01Github**		Pred:Request	Pred:Reply	Sum			Accuracy	**75.000%**
Method:	**Question mark**	Ann:Request	51	47	98	Correct	309	Precision	47.664%
Collection:	**Github (412)**	Ann:Reply	56	258	314	Incorrect	103	Recall	52.041%
		Sum	107	305	412			F-measure	49.756%
Experiment id:	**e02Github**		Pred:Request	Pred:Reply	Sum			Accuracy	**63.835%**
Method:	**Question words**	Ann:Request	34	64	98	Correct	263	Precision	28.571%
Collection:	**Github (412)**	Ann:Reply	85	229	314	Incorrect	149	Recall	34.694%
		Sum	119	293	412			F-measure	31.336%
Experiment id:	**e03Github**		Pred:Request	Pred:Reply	Sum			Accuracy	**63.592%**
Method:	**Question mark or words**	Ann:Request	65	33	98	Correct	262	Precision	35.714%
Collection:	**Github (412)**	Ann:Reply	117	197	314	Incorrect	150	Recall	66.327%
		Sum	182	230	412			F-measure	46.429%
Experiment id:	**e01cleanGithub**		Pred:Request	Pred:Reply	Sum			Accuracy	**75.485%**
Method:	**Cleaning + Question mark**	Ann:Request	51	47	98	Correct	311	Precision	48.571%
Collection:	**Github (412)**	Ann:Reply	54	260	314	Incorrect	101	Recall	52.041%
		Sum	105	307	412			F-measure	50.246%
Experiment id:	**e02cleanGithub**		Pred:Request	Pred:Reply	Sum			Accuracy	**64.078%**
Method:	**Cleaning + Question words**	Ann:Request	33	65	98	Correct	264	Precision	28.448%
Collection:	**Github (412)**	Ann:Reply	83	231	314	Incorrect	148	Recall	33.673%
		Sum	116	296	412			F-measure	30.841%
Experiment id:	**e03cleanGithub**		Pred:Request	Pred:Reply	Sum			Accuracy	**64.078%**
Method:	**Cleaning + Question mark or words**	Ann:Request	65	33	98	Correct	264	Precision	36.111%
Collection:	**Github (412)**	Ann:Reply	115	199	314	Incorrect	148	Recall	66.327%
		Sum	180	232	412			F-measure	46.763%

***Table A.3**: Detailed evaluation results of unsupervised methods tested on the **NNTP newsgroup** part of the online communication corpus*
Contingency tables, number of correctly and incorrectly predicted instances, accuracy, precision, recall and F-measure scores
"Ann" stands for annotations, while "Pred" stands for method predictions.

Experiment information		Contingency table				Evaluation scores			
Experiment id:	**e01Newsgroup**		Pred:Request	Pred:Reply	Sum			Accuracy	**51.923%**
Method:	**Question mark**	Ann:Request	55	22	77	Correct	108	Precision	41.353%
Collection:	**Newsgroup (208)**	Ann:Reply	78	53	131	Incorrect	100	Recall	71.429%
		Sum	133	75	208			F-measure	52.381%
Experiment id:	**e02Newsgroup**		Pred:Request	Pred:Reply	Sum			Accuracy	**81.731%**
Method:	**RE**	Ann:Request	45	32	77	Correct	170	Precision	88.235%
Collection:	**Newsgroup (208)**	Ann:Reply	6	125	131	Incorrect	38	Recall	58.442%
		Sum	51	157	208			F-measure	70.313%
Experiment id:	**e03Newsgroup**		Pred:Request	Pred:Reply	Sum			Accuracy	**55.769%**
Method:	**RE Question mark**	Ann:Request	67	10	77	Correct	116	Precision	44.966%
Collection:	**Newsgroup (208)**	Ann:Reply	82	49	131	Incorrect	92	Recall	87.013%
		Sum	149	59	208			F-measure	59.292%
Experiment id:	**e04Newsgroup**		Pred:Request	Pred:Reply	Sum			Accuracy	**44.712%**
Method:	**Question words**	Ann:Request	58	19	77	Correct	93	Precision	37.662%
Collection:	**Newsgroup (208)**	Ann:Reply	96	35	131	Incorrect	115	Recall	75.325%
		Sum	154	54	208			F-measure	50.216%
Experiment id:	**e05Newsgroup**		Pred:Request	Pred:Reply	Sum			Accuracy	**47.115%**
Method:	**Question mark or words**	Ann:Request	70	7	77	Correct	98	Precision	40.462%
Collection:	**Newsgroup (208)**	Ann:Reply	103	28	131	Incorrect	110	Recall	90.909%
		Sum	173	35	208			F-measure	56.000%
Experiment id:	**e06Newsgroup**		Pred:Request	Pred:Reply	Sum			Accuracy	**47.115%**
Method:	**RE Question mark or words**	Ann:Request	73	4	77	Correct	98	Precision	40.782%
Collection:	**Newsgroup (208)**	Ann:Reply	106	25	131	Incorrect	110	Recall	94.805%
		Sum	179	29	208			F-measure	57.031%
Experiment id:	**e01cleanNewsgroup**		Pred:Request	Pred:Reply	Sum			Accuracy	**64.904%**
Method:	**Cleaning + Question mark**	Ann:Request	51	26	77	Correct	135	Precision	52.041%
Collection:	**Newsgroup (208)**	Ann:Reply	47	84	131	Incorrect	73	Recall	66.234%
		Sum	98	110	208			F-measure	58.286%

Experiment id:	**e02cleanNewsgroup**		Pred:Request	Pred:Reply	Sum			Accuracy	**68.269%**
Method:	**Cleaning + RE Question mark**	Ann:Request	63	14	77	Correct	142	Precision	54.783%
Collection:	**Newsgroup (208)**	Ann:Reply	52	79	131	Incorrect	66	Recall	81.818%
		Sum	115	93	208			F-measure	65.625%
Experiment id:	**e03cleanNewsgroup**		Pred:Request	Pred:Reply	Sum			Accuracy	**52.404%**
Method:	**Cleaning + Question words**	Ann:Request	56	21	77	Correct	109	Precision	41.791%
Collection:	**Newsgroup (208)**	Ann:Reply	78	53	131	Incorrect	99	Recall	72.727%
		Sum	134	74	208			F-measure	53.081%
Experiment id:	**e04cleanNewsgroup**		Pred:Request	Pred:Reply	Sum			Accuracy	**54.808%**
Method:	**Cleaning + Question mark or words**	Ann:Request	69	8	77	Correct	114	Precision	44.516%
Collection:	**Newsgroup (208)**	Ann:Reply	86	45	131	Incorrect	94	Recall	89.610%
		Sum	155	53	208			F-measure	59.483%
Experiment id:	**e05cleanNewsgroup**		Pred:Request	Pred:Reply	Sum			Accuracy	**54.327%**
Method:	**Cleaning + RE Question mark or words**	Ann:Request	72	5	77	Correct	113	Precision	44.444%
Collection:	**Newsgroup (208)**	Ann:Reply	90	41	131	Incorrect	95	Recall	93.506%
		Sum	162	46	208			F-measure	60.251%

***Table A.4**: Detailed evaluation results of unsupervised methods tested on entire online communication corpus - Contingency tables, number of correctly and incorrectly predicted instances, accuracy, precision, recall and F-measure scores "Ann" stands for annotations, while "Pred" stands for method predictions.*

Experiment information		Contingency table				Evaluation scores			
Experiment id:	**e01Corpus**		Pred:Request	Pred:Reply	Sum			Accuracy	**67.379%**
Method:	**Question mark**	Ann:Request	130	175	305	Correct	694	Precision	44.674%
Collection:	**Entire corpus (1030)**	Ann:Reply	161	564	725	Incorrect	336	Recall	42.623%
		Sum	291	739	1030			F-measure	43.624%
Experiment id:	**e02Corpus**		Pred:Request	Pred:Reply	Sum			Accuracy	**73.398%**
Method:	**e01Bugzilla, e01Github, e02Newsgroup**	Ann:Request	120	185	305	Correct	756	Precision	57.416%
Collection:	**Entire corpus (1030)**	Ann:Reply	89	636	725	Incorrect	274	Recall	39.344%
		Sum	209	821	1030			F-measure	46.693%
Experiment id:	**e03Corpus**		Pred:Request	Pred:Reply	Sum			Accuracy	**68.155%**
Method:	**e01Bugzilla, e01Github, e03Newsgroup**	Ann:Request	142	163	305	Correct	702	Precision	46.254%
Collection:	**Entire corpus (1030)**	Ann:Reply	165	560	725	Incorrect	328	Recall	46.557%
		Sum	307	723	1030			F-measure	46.405%
Experiment id:	**e04Corpus**		Pred:Request	Pred:Reply	Sum			Accuracy	**62.330%**
Method:	**Question words**	Ann:Request	166	139	305	Correct	642	Precision	40.000%
Collection:	**Entire corpus (1030)**	Ann:Reply	249	476	725	Incorrect	388	Recall	54.426%
		Sum	415	615	1030			F-measure	46.111%
Experiment id:	**e05Corpus**		Pred:Request	Pred:Reply	Sum			Accuracy	**61.942%**
Method:	**Question mark or words**	Ann:Request	213	92	305	Correct	638	Precision	41.520%
Collection:	**Resource (1030)**	Ann:Reply	300	425	725	Incorrect	392	Recall	69.836%
		Sum	513	517	1030			F-measure	52.078%
Experiment id:	**e06Corpus**		Pred:Request	Pred:Reply	Sum			Accuracy	**61.942%**
Method:	**e03Bugzilla, e03Github, e06Newsgroup**	Ann:Request	216	89	305	Correct	638	Precision	41.618%
Collection:	**Entire corpus (1030)**	Ann:Reply	303	422	725	Incorrect	392	Recall	70.820%
		Sum	519	511	1030			F-measure	52.427%
Experiment id:	**e01cleanCorpus**		Pred:Request	Pred:Reply	Sum			Accuracy	**70.194%**
Method:	**Cleaning + Question mark**	Ann:Request	125	180	305	Correct	723	Precision	49.603%
Collection:	**Entire corpus (1030)**	Ann:Reply	127	598	725	Incorrect	307	Recall	40.984%
		Sum	252	778	1030			F-measure	44.883%

Experiment id:	**e02cleanCorpus**		Pred:Request	Pred:Reply	Sum			Accuracy	**73.592%**
Method:	**e01cleanBugzilla, e01cleanGithub, e02Newsgroup**	Ann:Request	119	186	305	Correct	758	Precision	58.049%
Collection:	**Entire corpus (1030)**	Ann:Reply	86	639	725	Incorrect	272	Recall	39.016%
		Sum	205	825	1030			F-measure	46.667%
Experiment id:	**e03cleanCorpus**		Pred:Request	Pred:Reply	Sum			Accuracy	**70.874%**
Method:	**e01cleanBugzilla, e01cleanGithub, e02cleanNewsgroup**	Ann:Request	137	168	305	Correct	730	Precision	50.929%
Collection:	**Entire corpus (1030)**	Ann:Reply	132	593	725	Incorrect	300	Recall	44.918%
		Sum	269	761	1030			F-measure	47.735%
Experiment id:	**e04clean Corpus**		Pred:Request	Pred:Reply	Sum			Accuracy	**64.272%**
Method:	**Cleaning + Question words**	Ann:Request	163	142	305	Correct	662	Precision	41.902%
Collection:	**Resource (1030)**	Ann:Reply	226	499	725	Incorrect	368	Recall	53.443%
		Sum	389	641	1030			F-measure	46.974%
Experiment id:	**e05cleanCorpus**		Pred:Request	Pred:Reply	Sum			Accuracy	**63.981%**
Method:	**Cleaning + Question mark or words**	Ann:Request	212	93	305	Correct	659	Precision	43.265%
Collection:	**Entire corpus Entire corpus (1030)**	Ann:Reply	278	447	725	Incorrect	371	Recall	69.508%
		Sum	490	540	1030			F-measure	53.333%
Experiment id:	**e06cleanCorpus**		Pred:Request	Pred:Reply	Sum			Accuracy	**63.883%**
Method:	**e03cleanBugzilla, e03cleanGithub, e05cleanNewsgroup**	Ann:Request	215	90	305	Correct	658	Precision	43.260%
Collection:	**Entire corpus (1030)**	Ann:Reply	282	443	725	Incorrect	372	Recall	70.492%
		Sum	497	533	1030			F-measure	53.616%