Understanding Arguments

Tim van Rossum
0439541

Bachelor thesis
Credits: 18 EC

Bachelor Opleiding Kunstmatige Intelligentie

University of Amsterdam
Faculty of Science
Science Park 904
1098 XH Amsterdam

Supervisor
Dr. M. J. Marx

ILPS
Informatics Institute
University of Amsterdam
Science Park 904
1098 XH Amsterdam

Dr. D. Grossi

ILPS
Informatics Institute
University of Amsterdam
Science Park 904
1098 XH Amsterdam

August 10th, 2012

Contents

1 Introduction

There have been a number of studies in recent years dealing with the subject of opinion mining, the automatic classification of an opinion represented in a piece of written or spoken text. This research generally takes the shape of sentiment analysis applied to independent documents. A comprehensive overview of the subjects of opinion mining and sentiment has been published by Pang and Lee [1]. The research that is presented here is related to the topic of opinion mining, but takes a somewhat different approach. The focus will not be on the classification of the sentiment of a independent pieces of text, but rather on the sentiment of a discussion. Our purpose is to us transcripts of debates of Bill Committees of the UK Parliament to train a classifier to predict the outcome of the a debate, where an outcome is defined as the withdrawal or acceptance of a proposal by consensus, or acceptance or rejection of the proposal based on a vote taken by a divided committee. The difference, then, between this goal and that of previous studies is that, rather than an individual statement or piece of writing, the structure of a debate as a whole will need to be represented in such a way that useful features can be extracted that can be used as input for learning algorithms. The approach that will be presented in this paper can be divided into two stages. In the first stage structural qualities of the discussion will be used to predict the outcome of the debate. This means that the features that are used are not related to the semantic content of what was said. The data includes information on the number of speeches, the number of speakers, references to other speakers and the number of participants per party.These features will be used to set a baseline for the accuracy of the classification of outcomes. Because the focus of this research is on the representation of the debate, several pre-existing implementations of learning algorithms will be used to create a predictor from these features.

In the second stage it will be attempted to improve the predictor by using information on the opinion of individual speakers. Thomas et al. [2] have shown that it is possible to predict individual speakers in debates similar to those used here (in US Congress), with a significant measure of accuracy.

The final result will be a chain of binary classifiers which starts with the individual speaker preference classification. Next a classifier will be trained to predict whether there will be a vote at the end of the debate or whether a consensus will be reached. Then, for both sets of debates, a classifier will be added to decide whether the proposal is accepted or not. Finally, the results of this approach will be presented.

2 Data

2.1 Available Data

The data that will be used here consists of xml files containing discussions of bill committees (previously knows as standing comittees) of the British parliament, regarding proposed amendments or changes to bills. The format in which the data was available divided into 1971 files, each covering a single sitting of a bill committee. Each sitting may contain discussions regarding several changes and amendments to multiple clauses in a bill. The files were annotated with the name of the bill that is being discussed an a list of the members of the committee. Each particular discussion is headed by the clause under discussion, but information on which particular part of the clause is discussed is not indicated. The outcome of each debate was only included as a paragraph within a speech. If the committee is divided and votes on the proposal, the division of the vote is included behind the speech that includes the outcome and is clearly marked with its own tag. All speeches regarding a clause were located behind the header indicating the clause, resulting in a rather flat representation of the data. The name and unique id of the speaker are included as attributes of a speech. Further information regarding the speaker, such as party affiliation is included in a separate file and readily accessible.

2.2 Desired format of the data

In order to efficiently be able to use the data as input for a learner, its representation needs to be adjusted as to make all relevant data easily accessible. The first of the two main issues that are to be addressed is that the outcome of the debate is not explicitly indicated in the data. The first change that must be made is that the outcome must be extracted and indicated as a separate piece of information, as it is, of course, vital to both the training and the validation of a predictor.
Secondly, the different debates must be separated. Considering that a debate, as it is of interest here, is a discussion (sequence of speeches) ending in a single outcome, the position of the outcomes is to be used to delimit the debates. The next step is to group the speeches regarding a particular proposal together. In the original data the speeches regarding different topics are separated by the headings. However, it is preferable to have them nested under a header, rather then on the same level, making it much easier to iterate over the debates and extract the relevant features for each debate. Finally, the date, which is only included in the filename, needs to be included in the file, in order match speaker id to their position and party affiliation at the time the debate was held. The date can then be used to acquire accurate information regarding the speaker.

2.3 Processing of the data

2.3.1 Outcome tagging

As was mentioned in the previous subsection, the first step is to recognize the outcome of the debate. Casual inspection of the original data showed three sentences that characterize the possible outcomes of the debates, these are:
Question put and agreed to.
Amendment, by leave, withdrawn.
The house divided: Ayes: 6, Noes: 9.
This can be recognized using a very simple regular expression, consisting a simple disjunction of these sentences, where the numbers (of which a random example was chosen above) are indicated as a sequence of one or more numerical characters. However, more extensive inspection of the data showed that the formulation of the outcomes were not quite so consistent. There were exceptions that are small deviations from the examples above, such as the same sentence, but without the period at the end. Also, it should be taken account that it are not only amendments that are discussed, but also, for example, schedules or clauses as a whole. This is reflected in the description of the a negative outcome (the second example above). Furthermore, there are several instances where a negative outcome of the discussion is described as: *Question put and negatived.*
In the case of a positive outcome, the vast majority is described with the sentence given above. There are however, some exceptions that should be included, which are of the form:
Amendment agreed to.
The word amendment may be something else here, in the same way as with the most common phrase with negative outcome.
The finding and tagging of the outcome was implemented in Java. Because Java does not (unless instructed otherwise) match the regex for a random character (the period) to a linebreak, these differences pose little problem, as the start of the sentence can be inidcated simply as a sequence of arbitrary characters of any length. The fact that linebreaks are not matched asw being such a character avoid computational issues with this approach.
There are, however, not only extra phrases that must be included as outcomes, but also a few that must be excluded from the search. There are two situations in which phrases were used that would match a regex searching the phrases described above. The first is a description of the outcome of a vote, after the vote and the division of the vote were already mentioned. These are characterized by the sentences:
Question accordingly agreed to
Clause 3 disagreed to
The second situation is where a bill is accepted, without a general discussion.

That is to say, all debates pertain to 'smaller' subjects, and the outcome regarding this subjects has already been indicated with one of the sentences discussed here. These are indicated by the phrase:
Main Question put and agreed to.
Excluding this is a matter of excluding phrases starting with the word 'Main'. The requirements discussed above resulted in the following Java regex:

```
(The Committee divided: Ayes \\p{Digit}+, Noes \\p{Digit}+.?|
?(?!(Main)).* (?!(accordingly ))(?!(dis))agreed to.?|.* by leave, withdrawn.?|
Question put and negatived.?)"
```

The data files were read and represented as a string, to which this regex was applied. The paragraph tags ($< p >$) were replaced by outcome tags ($< outcome >$) and the result was written to a new file, making some of the most important information in the data recognizable to an xml parser and preparing the data for the next step in thr preparation.

2.3.2 Dividing the debates

Once the outcomes have been clearly indicated in the data, they can be used to delimit the debates, considering that a debate will here be considered to consist of a range of speeches followed by an outcome. An XQuery was used in order to seperate the debates by adding headers between the debates. It should be noted that this operation was applied to files which contain all debates regarding a particular bill. Combining several files, divided into sittings, into one file per bill is a trivial process and does not warrant further description here. The only thing that is worth mentioning is that for sake of simplicitly only the date of the first sitting was included in these files. It is assumed that no important information regarding the speakers has changed over the course of the committees debates.
The operation of separating the topics of the debates consists, besides the trivial copying of dates, bill title and committee members, of the addition of headers after speeches containing an outcome. In order to achieve this an XQuery was written that loops through the headers each file, which are labelled as 'minor-heading'. For each header, a list is created of the speeches that follow it, excluding those that follow the next header. Next, these speeches are looped to and directly returned if they don't contain an outcome, with the exception of the first speech, which is returned in a tuple where it follows the header that is directly above it in the original file. If the speech does contain an outcome, it is checked wheteher this is the last outcome belonging under this header. The first speech containing an outcome following the current one is compared to the first speech containing an outcome following the next header. If they are the same, the current speech is the last one containing an outcome belonging to this header. In this

case, the speech is returned by itself. If it is not the last outcome belonging to the current header, the speech is returned in a tuple, followed by a copy of the header, resulting in a structure where each header is preceeded by a sequence of speeches, of which exactly one contains an outcome. The new headers are copies of the original ones, except that their id is extende with the number the speech which 'decides' that the header must be added. This way, each header, and therefore each topic, has a unique id. The XQuery code for this operation can be found in appendix A.

2.3.3 Final Restructuring

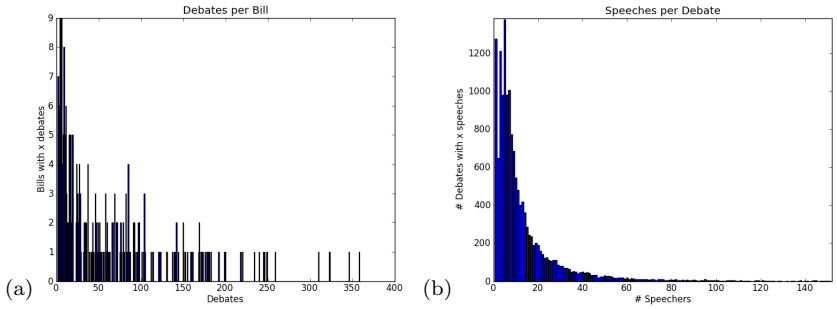
The main purpose of the final step is to collect all information to a particular debate under a node, making each debate - which will be a data point for our learner, accessible as a unit. For each header, there will be a 'debate' node. This node will contain a 'topic' node, which is a copy of the header, except that the 'minor-heading' tag is 'replaced' by 'topic' for clarity, a sequence of speeches and an outcome.
In order to achieve this structure, an XQuery which loops through all headers and through the speeches between that header and the next header, excepting those that follow the speech containing the outcome. For each of the speeches, the paragraphs are returned one by one and placed between speech tags, creating a new speech node containig the exact same paragraphs as the original speech. The only exception is that in the speech that contains the outcome of the debate, all paragraphs following the outcome are discarded. Finally, the outcome, which was not copied with the speech that is what in, as it is no longer labelled as paragraph, is added as the last property of the debate. A simple regex matching is used to establish whether the outcome is positive, negative of divided and an attribute with an integer value is added the outcome node, indicating the type of outcome. In the case where the committee is divided, a conditional statement is used to find the next divisioncount (i.e. the division of votes belonging with this outcome) and added as a subnode of the outcome. This results in a structure in which the data can be used for our purpose without further difficulties. The XQuery that performs this operation can be found in Appendix A.

2.3.4 Data Validation and Results

Due to the large amount of data available, it has been restructured automatically, rather than manually. Consequently, the quality of the new data is dependent on the correctness of the transformations that have been applied to it. It is therefore desirable to validate the quality of the output. Unfortunately, the quality of matters of content, such as the outcome recognition, cannot be automatically validated, because the recognition of what correct is would depend on the same criteria that have been used to find the outcomes

	Agreed to	Withdrawn/Negatived
Consensus	6452	6613
Vote	632	1180

Table 1: Distribution of outcomes over the available debates



(a) (b)

Figure 1: Distribution of debates over the bills (a) and distribution of speeches over the debates (b)

in the first place. However, the structure of the resulting representation of the data can be validated using Relax NG schema's. Relax NG is a schema that defines the structure that an xml file should have. In this case, it is a tree consisting of of a root node, under which there are a 'date' node a 'committee node' and one or more 'debate' nodes. A schema that shows this structure has been created and applied to all the processed data files. This schema can be seen in appendix A. Although most files were in order, some issues were revealed during this process, consisting mostly of empty nodes or disallowed types and unusual characters that appear at random places within the data. A number of these issues have been manually fixed and some files with major issues have been discarded. When this process was complete, all files were succesfully validate. Taking into account this result as well as some random manual checks, the new representation of the data is now considered fit for use.

The remaining data consists of 14877 debates concerning 245 bills with an average of 14.6 speeches per debate. Figure 1(a) shows the distribution of the debates over the bills and figure 1(b) shows the distribution of speeches over the debates. Table 1 shows the distribution of the different types of outcome.

3 Classification Using Structural Features

3.1 Features

The features that will be initially used as parameters for the learners can be extracted from the data quite straightforwardly and mostly consists of simply counting entities. The following features are used:

1. Number of speeches in the debate
2. Number of speakers in the debate
3. Number of speakers partaking in the debate per party
4. The number of committee members per party
5. Average speech length (number of words) in the debate
6. Number of references to other speakers
7. Number of speakers referencing other speakers

The reference to other speakers will be counted as both the names of committee members appearing in speeches, as well as the use of phrases commonly used in the committee debates, such as 'the hon. member' or 'my hon. friend'.

3.2 Implementation

The features were extracted using XQueries, applied by a Java program and stored in a file readable by Weka. The different features were selected as follows:

Number of speeches in the debate A straightforward count of the number of speeches

Number of speakers in the debate A count of the distinct speaker id's of all speeches.

Number of speakers partaking in the debate per party A list of parties was created from the member data file. For each distinct speaker in the debate, their party was selected and added to the count. When inspected, it was shown that speakers from parties other than the three largest (Conservatives, Labour, Liberal-Democrats), so only the count of speakers from these parties was included as features.

The number of committee members per party Count kept based on the committee member list. Limited to the three largest parties for the same reason as the speaker count.

Average speech length (number of words) in the debate The content of the speech was separated into words using XQueries built-in regex representation of a word. The number of words returned by the regex is used as word count.

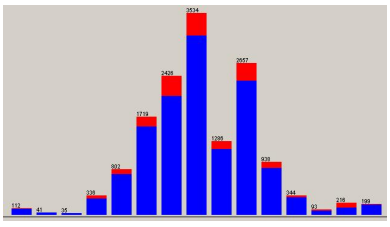
Number of references to other speakers A regex expression was created, that matches the following: 'hon. member', 'hon. friend', 'hon. Gentleman' and 'hon. lady'. The same strings with a capital 'H' instead also match. The string representing the speech was split around these expressions and the length of the resulting list minus one is used as the number of references.

Number of speakers referencing other speakers The same regex as above was used to find references. The speaker ids of each speech were collected and the number of distinct speakers in this list indicates the number of speakers referencing other speakers.

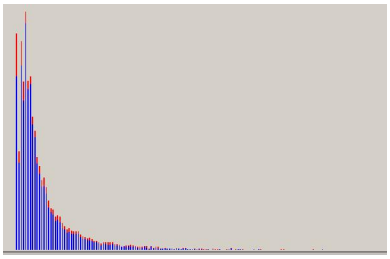
The predictor was trained and tested using five-fold cross-validation using the j48 decision tree and naive Bayes as they were implemented in Weka, with default parameters. SVM proved to require too much memory using this dataset on the machine used in this research. The entire data set was used in the first predictor, which decides whether the debate ended with a vote or general agreement. The datapoints where there was a vote and where there wasn't were separated and the prediction of whether or not the proposal was accepted were trained and tested separably.

3.3 Results

Figure 2 shows some examples of the distribution of the vote/no vote distinction over the values of different features. Table 2 shows the accuracy of the predictor of this distinction.



(a) Number of committee member from the Labour party



(b) Number of speeches in the debate

Figure 2: Overview of the distribution of values of features. Red indicates a vote, blue shows agreement at then end of the debate.

	Decision Tree	Naive Bayes
Vote/No Vote	87.96	84.07
Accepted/Rejected without vote	67.93	70.79
Accepted/Rejected with vote	64.99	67.31

Table 2: Accuracy using simple structural features

classified as →	No Vote	Vote
No Vote	11715	1249
Vote	952	822

Table 3: Confusion matrix for naive Bayes vote/no vote predictor

The acceptance/rejection prediction shows poor accuracy, which is somewhat to be expected, as it seems intuitive that debate structure tells us more about the level of disagreement then the sentiment of the content of what is said.
Although the numbers in the vote/no vote prediction look like a promising result, it should be taken into consideration that the different values of the predicted quality are unevenly distributed. Consequently, the best result is achieved by the decision tree, which, on inspection of the confusion matrix, is shown to consistently predicts that there will be no vote. So the percentage that is correctly predicted is simply the percentage of data points where no vote was taken.
The naive Bayes method does predict both outcomes, but at about the same ratio that they appear. This means that its result can also largely be explained by the uneven division of values for the predicted property. This is reflected in the fact that the classification as having a vote is more often incorrect than it is correct, as can be seen in table 3.

4 Classification Using Individual Speaker Preference

4.1 Features and Implementation

The features that have been extracted from the speaker preference are a simple count of the speakers that are for and those that are against the proposal, as well as a count of the total number of speeches that have been held by speakers of either side of the debate.
It was intended to implement the speaker preference predictor from described in Thomas et. al [2] in order to complete the chain of classifiers. However due to time constraints there as no time for this. It is still possible, though, to test if the features that can be derived from the individual speaker

	Decision Tree	Naive Bayes	SVM
Accepted/Rejected	72.79	74.65	77.10

Table 4: Accuracy using speaker preference information

classified as →	Accepted	Rejected
Accepted	11715	1249
Rejected	952	822

Table 5: Confusion matrix for SVM Acceptance/Rejection predictor

opinions would aid in the prediction. In order to achieve this, the data has been limited to those in which there has been a vote. As the vote cast by each speaker is contained in the data, this vote can be used to characterize the speaker's preference. This way, a perfect opinion predictor (perfect as trained on the votes beind indicative of the opinion represented in a speech) is imitated. The same learners as before are used, as well as SVM, which was useable here due to the smaller data set.

4.2 Results

Due to the fact that only those data points where there was a vote were available for this experiment, only the prediction of whether the proposal was accepted or rejected given that there was a vote could be tried. Table 4 shows the results.

There are 1231 data points in this collection, of which 896 result in a rejection of the proposal. The decision tree once again achieves its result by consistently predicting the most frequently occurring value. The best result, however, is shown by the SVM, which predicts both outcomes right more often than wrong, as can be seen in the confusion matrix shown in table 5.

5 Conclusion

The results that are presented here show that the features that have been used are not sufficient to accurately predict the outcome of the debates. Using the structural features, both in the prediction of whether or not there was voted as in the prediction of whether the proposal was accepted, the best results can be achieved by consistently predicting the same outcome. This is a clear indicator that the predictor is not working properly. Furthermore, inspection of the distribution of feature values shows no strong separating value in any of the features as themselves.

The prediction of positive or negative outcome of the vote using individual speaker information showed more promise. SVM classification scored several percentage points higher in accuracy than the ration of the most frequent of

two values. There is still a lot of room for improvement here, but there is also a lot of features that can still be extracted from the information Ultimately, it cannot be definitively concluded that the structural features cannot aid in prediction when used together, as only a limited range of learners has been tried. To conclude that the outcomes of the debates cannot be predicted from the speeches would certainly be a mistake, as only a limited set of rather simple features has been applied. Future research using more complex features might improve on these results.

6 References

Matt Thomas, Bo Pang, and Lillian Lee. Get out the vote: Determining support or opposition from Congressional floor-debate transcripts. *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 327335, 2006.
Bo Pang, and Lillian Lee. Opinion Mining and Sentiment Analysis. *Foundations and Trends in Information Retrieval*, v.2 n.1-2, p.1-135, January 2008.

References

[1] Bo Pang and Lillian Lee. Opinion mining and sentiment analysis. *Found. Trends Inf. Retr.*, 2(1-2):1–135, January 2008.

[2] Matt Thomas, Bo Pang, and Lillian Lee. Get out the vote: determining support or opposition from congressional floor-debate transcripts. In *Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing*, EMNLP '06, pages 327–335, Stroudsburg, PA, USA, 2006. Association for Computational Linguistics.

A XQueries for data processing

Division of debates

```
let $file := doc('file:///FILENAME')
    for $content in $file//publicwhip
        return
        <publicwhip>
        {
           for $bill in $content//bill
               return $bill
        }
        {
           let $date := $content/date
               return $date[1]
        }
        {  for $topic in $content//minor-heading
           let $speeches := $topic/following-sibling::speech except
            $topic/following-sibling::minor-heading/following-sibling::speech
           let $nextTopic := $topic/following-sibling::minor-heading
           let $nextTopic := $nextTopic[1]
           let $otherBlockEndSpeeches := $nextTopic/following-sibling::speech/outcome/parent::speech
             for $speech at $i in $speeches
             let $header := <minor-heading id="{concat($topic/@id, "*", $i)}" time="{$topic/@time}"
              nospeaker="{$topic/@nospeaker}" url="{$topic/@url}">{data($topic)}</minor-heading>
             let $laterEndSpeeches := $speech/following-sibling::speech/outcome/parent::speech
             let $print := if(exists( $speech/outcome ) and matches(data($speech/outcome[1]), "divided")) then
              ($speech, $speech/following-sibling::divisioncount[1]) else $speech
return if ( $speech is $speeches[1]  )
             then ($header, $print) else if( exists( $speech/outcome ) and not($laterEndSpeeches[1] is
              $otherBlockEndSpeeches[1]) ) then ($print, $header) else $print
             }

        {   let $committee := $content/committee
            return $committee[1]}
        </publicwhip>
```

Final restructuring of the data.

```
let $file := doc('file:///FILENAME')
    for $content in $file//publicwhip
        (:let $speeches := $content//speech:)
        return
        <publicwhip>
        {
           for $bill in $content//bill
               return $bill
        }
        {
           let $date := $content/date
               return $date[1]
        }
           {for $topic in $content//minor-heading
           let $prevHeading := $topic/preceding-sibling::minor-heading
           let $speeches := $topic/following-sibling::speech
           let $lastSpeech := $speeches/outcome/parent::speech
           let $prevLastSpeech := $prevHeading[1]/following-sibling::speech/outcome/parent::speech
           where not($lastSpeech[1] is $prevLastSpeech[1])
               return
               <debate>
               <topic id="{data($topic/@id)}" time="{data($topic/@time)}">
               {data($topic)}
               </topic>
               {
                   for $speech in $speeches
                   except $lastSpeech/following-sibling::speech
                       let $outcome := $speech/outcome
                       return
                       <speech id="{data($speech/@id)}" speakerid="{data($speech/@speakerid)}"
                        speakername="{data($speech/@speakername)}" time="{data($topic/@time)}"
                        nospeaker="{data($topic/@nospeaker)}">
                       {   for $p in $speech//p except $outcome/following-sibling::p
                              (: where not( exists( $speech//outcome ) ):)
                              return $p }
                        </speech>
                }
                <outcome  type="{if( matches(data($lastSpeech[1]/outcome[1]), "agreed")) then '0'
                 else( if( matches(data($lastSpeech[1]/outcome[1]), "withdrawn|negatived"))then '1' else '2')}">
                {data($lastSpeech[1]/outcome[1])}
                   {
                   if( matches(data($lastSpeech[1]/outcome[1]), "divided"))
                   then (let $div := $lastSpeech/following-sibling::divisioncount return $div[1]) else ""}</outcome>
```

```
        </debate>
     }
    {let $committee := $content/committee
    return $committee[1]}
</publicwhip>
```