Paper prepared for the 2nd Annual Meeting of the
Comparative Policy Agendas Conference
The Hague, 18-19 June 2009

Strategies for Improving Semi-automated Topic Classification of Media and Parliamentary documents

Gerard Breeman, Wageningen University, Montesquieu institute.
Hans Then, Pythea company
Jan Kleinnijenhuis, VU University Amsterdam
Wouter van Atteveldt, VU University Amsterdam
Arco Timmermans, Leiden University, Montesquieu institute

This paper has been presented at the 2009 MPSA Conference, Chicago.

Work in progress

Introduction

Since 1995 the techniques and capacities to store new electronic data and to make it available to many persons have become a common good. As of then, different organizations, such as research institutes, universities, libraries, and private companies (Google) started to scan older documents and make them electronically available as well. This has generated a lot of new research opportunities for all kinds of academic disciplines.

The use of software to analyze large datasets has become an important part of doing research in the social sciences. Most academics rely on human coded datasets, both in qualitative and quantitative research. However, with the increasing amount of datasets and the complexity of the questions scholars pose to the datasets, the quest for more efficient and effective methods is now on the agenda.

One of the most common techniques of content analysis is the Boolean key-word search method. To find certain topics in a dataset, the researcher creates first a list of keywords, added with certain parameters (AND, OR etc.). All keys are usually grouped in families and the entire list of keys and groups is called the ontology. Then the keywords are searched in the dataset, retrieving all documents containing the specified keywords. The online newspaper dataset, LexisNexis, provides the user with such a Boolean search method.

However, the Boolean key-word search is not always satisfying in terms of reliability and validity. For that reason social scientists rely on hand-coding. Two projects that do so are the congressional bills project (www.congressionalbills.org) and the policy agenda-setting project (see www.policyagendas.org). They developed a topic code book and coded various different sources, such as, the state of the union speeches, bills, newspaper articles etcetera. The continuous improving automated coding techniques, and the increasing number of agenda setting projects (in especially European countries), however, has made the use of automated coding software a feasible option and also a necessity.

Supervised machine learning

Hillard et al. (2008) and Purpura et al. (2006) have been exploring the feasibility of the use of automated coding for the policy agendas project and the congressional bills project. They reached high score performance and their main conclusion is that the coding improves when mixed methods are being used. They integrated different machine learning techniques and raised the accuracy of the automated coding up to 90% – 95% (with a large training

file, though). The mixed coding tools can also predict how accurate the results are. Their main objective is to increase coding efficiency by setting aside the entries that have sufficient accuracy (95% or higher) and let human annotator only code those entries with low accuracy by hand.

In this paper we take their argument about mixed methods further and explore additional ways of mingling the various coding methods and algorithms. Still, however, we also need a human annotator to train the software and to code the entries with low accuracy. In the case of *supervised* machine learning human supervisors apply or develop a category system, defined as a set of mutually exclusive and jointly exhaustive categories. Human coders should code a large number of texts, to inform algorithms for supervised machine learning about typical features of texts that belong to the various categories of the category system, so as to enable classification of new texts. Elementary supervised "automated coding" considers a text as a *bag of words*. They will find some words, or some combination of words, that occur often in other texts that belong to the same category as well, but not quite as often in texts that belong to other categories. For three reasons, this procedure is not quite robust, however.

First, it should be recognized that the future performance of a learning algorithm will be low when it was trained on a limited number of examples. In this paper we will examine for the case of policy documents how performance is increased by (1) increasing the number of examples, by (2) balancing the examples over the categories of the category system, and by (3) improving the file of examples by adding new examples to it, in case no doubt existed at all with respect to the classification.

Next, it has been widely acknowledged in the research literature that texts are not bags of words at all. Texts exhibit structure, composition, style and humor. Features that could be used by supervised machine learning programs need not be isolated words in a bag of words, but could also be defined as higher-order syntactic and sementic features of the texts at hand. Various suggestions have been made:

1. Do not use the word forms, but the *word stem* (Porter, 1980), or use a lemmatizer to find the *word lemma*, so as to enable the recognition of words, in spite of a variety of word forms.
2. Use a thesaurus to define as additional features of texts the *word groups to which separate words belongs*. This will enable supervised learning algorithms to extract common topics from texts that belong to the same category, in spite of a large variety of different, but nearly synonymous words.
3. Use a natural language grammar parser to get rid of the variation in the meaning of words that is due to word order. A attacks B does not

resemble B attacks A, but B is attacked by A, although B attacks A is synonymous with A attacks B from a bag of words perspective.

4. Use the grammar parser output to define higher order semantic features of texts, for example about who is quoted or paraphrased, or about the political sentiment of political actors viz a viz each other (Van Atteveldt et.al. 2008ab).

In this paper we will ask whether performance can be improved by stemming (ad 1). The other improvements have to wait until future research.

Third, words in policy documents at a given point in time do not only reflect more or less stable characteristics of a policy field, like employment, unemployment, inflation and the interest rate in the case of economic policy, but also highly transitory, fleeting and perishable words, like the names of an obscure stakeholder who will not survive in the political struggle, or the label of a law that will be discarded by the next government, or the names of the policy domain experts from the current government coalition. Algorithms for supervised automated coding will often "learn" many of these transitory words. At first sight this helps them reproduce human codings appropriately, but learning such words is a waste of time when the aim is to classify also texts from a different period of time or from a different nation. This problem can be avoided in principle by leaving out all transitory words from the documents that serve as the training set for the supervised learning program, but distinguishing between short-lived and long-lived words will presumably cost more time than classifying each and every text without any computer at all. Another solution would be to use an entirely new training set for every new year, and for each new context, but this increases the amount of human efforts as well. In this paper we have adopted still another approach. For each of the categories of the category system we defined search strings that would presumably extract many relevant texts, although with a fairly low precision and a low recall. The hits generated by these search strings can be considered as features that can be fed into a supervised learning algorithm also. On the basis of a preliminary analysis, we will discuss whether this approach is worthwhile.

Description of the agenda project

Our interest in efficient topic coding tools originates in the agenda-setting projects. The purpose of these projects is to study the politics of attention. By counting the amount of topic attention in parliament, media, public opinion, and government documents we want to highlight the moments of high and low political attentions Based on the work of Baumgartner and Jones (1993, 2004) the common assumption in these projects is that attention for policy topics changes incrementally, but that this is occasionally punctuated with

high levels of attention. In these moments of high attention policies can change dramatically. The goal of the agenda projects is to study the mechanisms that explain why certain topics get attention and other, just as important, do not. Is media for instance only responding to political attention or vice versa? Who is framing the issue: politicians, experts, or journalists? And so on.

The topic coding is based on a topic code book, which was originally developed by Baumgartner and Jones in the 1990ties and updated by Wilkerson and Adler in 2006 (www.policyagendas.org). It contains 19 main topics and 225 subtopics (8 extra topics have been added for media coding, see appendix 1). Since 2000 various European scholars have started to code their national data too, using the same code book, although they made minor adjustments to meet the country specifics. By now, there are teams coding in Belgium, Canada, Denmark, England, France, Italy, The Netherlands, Spain, Switzerland, and the United Kingdom. In addition to the national projects, some are also starting to code EU activities, such as the COM documents (EU directives) and EU parliamentary questions.

The common code book that all countries use makes an easy comparison of policy attention between the different countries possible. The country teams meet regularly to exchange information and coordinate comparative projects. One of the informal agreements is to code the data sources at least back to 1978 (further, if possible). The teams also coordinate what and how to do the coding. The bills are, for instance, coded per bill, but the government agreements per section and the yearly opening speech of parliament per quasi-sentence. The data sources that have been coded differ somewhat per country, but in general the following data sources are being coded or will be coded:

- The opening speech of the parliamentary year. In the US this is the State of the Union, in the UK the King's or Queenspeech and in Denmark the Prime- minister's speech.
- The bills and laws.
- Media data. In the US the New York Times has been coded, in Denmark the summary transcripts of the radio news service at noun.
- Parliamentary activities. Most teams are coding the questions of the members of parliament, but some also code the minutes of the parliamentary debates or committee meetings.
- Government agreements.
- Public opinion.
- Attention in political party. This is usually done by coding party manifestoes.

In this paper we use the Dutch dataset to test and train the automated topic classification software. Thus far, the software has been tested on English datasets, in particular on the bills dataset of the congressional bills project (Hillard et al. 2008). The Dutch dataset contains the following data:

- Queen speeches (QS), coded per quasi-sentence (1945 – 2008, n=8122)
- Laws (LAW), titles and summaries, coded per law (1990-2006, n=2791)
- Government Agreement (GA), coded per section 1963-2006 (n=4455)
- NRC newspaper articles, full text and titles of front page articles, coded per article. We used a pre-coded sample of 1104 entries and appr. 40.000 uncoded articles 1990-2007.
- EU documents, coded per document. We coded directives and changes to directives only (not white books for instance), 1975-2008 (n=2300).

The coding tool

'The challenge is that if we are coding cased that humans have not coded, we can not judge whether the prediction is accurate or not (there is nothing to compare it to)' (Wilkerson in Wolfgang 2008). This means that the tool we are using to code the unseen data should be excellent, well tested, and should also give us feedback about the accuracy of the results. In previous research a mixed method, based on various classifiers, delivered very good results (Hillard et al 2008).

The central principle of all automated coding programs is the same: They all are based on hand coded training files. The various programs, however, use different algorithms to learn from the uploaded trainingfile. Based on the trainingfile the programs create models, and these models, in turn, classify new fresh data.

The basic strategy thus fare is to combine these different algorithms, because the accuracy is very high when three algorithms give the same code (idem). However, before using different algorithms in an ensemble, we first wanted to improve the trainingfile. Improving training files is usually done by adding more examples to the training file especially in those areas where the accuracy is low. This strategy is however time consuming and in this paper we suggest, and test an alternative.

The algorithms we use in this paper are gathered in an easy to use toolkit, Tooltexts (Wolfgang 2008). The different algorithms are all existent tools. We use the support vector machine (SVM), the Maximum Entropy classifier (Mallet or MAL), a Naïve Bayes classifier and a character N-Gram classifier (LingPipe of LING). (For more information about these algorithms see Hillard et al. 2008). We did not use all algorithms in all runs, because of time

constraints. Besides, after a series of tests, the N-Gram (LING) classifier turned to be most effective and therefore we focused more on this classifier.

Improving training files

We first wanted to know how well the different algorithms did when we simply uploaded the whole database and then let it code the same data again. The results of these tests are shown in table 1.

1. Coding based on the entire trainingset (with stemming and removal stopwords, see below)

	Eu com docs (n=2300)	Gov. agree (n=4455)	Laws (n=2791)	Nrc (n=1104) Txt &Title		Queenspeech (n=8122)
SVM	96,5	89,8	92,8	73,3	86.1	82,8
Maxent	98.4	98.7	97.9	100	97.2	87.9
Ling	98.8	98.4	98.4	99.9	100	86.9

The real performance test of the models is however if they are tested on entries that have not been part of the trainingfile. Therefore we used all even entries of the hand coded dataset to train the algorithms and used all odd numbers to test the model. The results are in table 2

2. Coding based on the 50% train (even numbers) and 50% test (odd)

	Eu com docs (n=2300)	Gov. agree (n=4455)	Laws (n=2791)	Nrc (n=1104) Txt & title		Queenspeech (n=8122)
SVM	81,0	50,2	57,8	32,2	32,6	28.8
Maxent	80.6	55.5	61.5	49.4	35.9	38.9
Ling	81.2	56.3	65.6	48,2	37.4	43.0

The relative high score of the EU documents could be explained by the dominance of two policy areas: agriculture and environment. The low accuracy scores of the NRC newspaper titles and Queens' speeches are probably caused by the short and ambiguous statements of these sources. The accuracy scores of the government agreements and laws of about 50% to 60% is comparable to the tests that are done with other comparable sources (Spanish laws, and US Bills, see texttools.blogspot.com). Important observation here, however, is the difference in accuracy rates for different types of texts. This is an issue which should be taken further in future experiments.

Based on this starting point, we tried to improve the training files per algorithm so the accuracy per algorithm would increase too.

1. Stemming and stopword removal

The first improvement is to remove all kinds of unnecessary stop words from the texts and to simplify the text with shortening the verbs to its stem (so-called stemming, see for more detailed information about this Porter 1980). Wolfgangs' Tooltexts contains several stemming algorithms for various languages. The results in table 2 have already been improved with the stemming and stopwords removal. Table 3, however, compares the results with and without stemming and stopword removal.

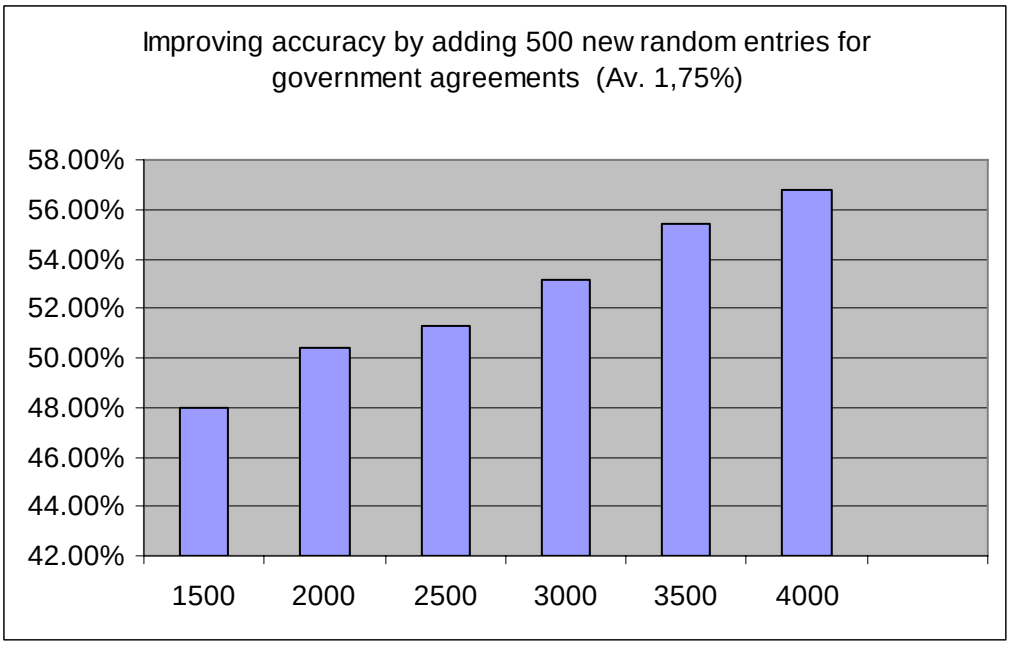
3: improvements before and after stemming and stopword removal

	Eu com docs (n=2300) Before / after		Gov. agree (n=4455)		Laws (n=2791) Before / after	
SVM	80.3	81.0	49.2	50.2	56.3	57.8
Maxent	76.8	80.6	52.1	55.5	57.0	61.5

Table 3 shows that in all cases the results improved when using a stemming program and stop word removal: it various from 0.7% up to 4.5%. However, compared to the results with the N-Gram classifier (LING), that we used without any language preprocessor (because we do not have that yet), the best results are still with this algorithm.

2. Increasing number of entries table

A second method to improve the training file is by adding randomly new entries to it. To test this we started to train and test the government agreements with 1500 entries (750 (even) in the train file and 750 (odds) in test file) and increased the number with 500 new entries. The graph showed a rather steady increase of the overall accuracy. On average, every 500 new entries shows an improvement with 1,75%. However, it is likely that this is not a linear function.



3. *Balancing*

The next step we have taken to improve the trainingfile is by balancing it. We simply selected 20 entries of each code from the precoded dataset (if available) and used the rest as the testfile. Table 4 shows the results of these tests.

Table 4: Results Before and after balancing

	Eu com docs (n=2300) Before / after		Gov. agree (n=4455) B		Laws (n=2791) Before / after		Nrc (n=1104) Text, before /after		Queenspeech (n=8122) Before / after	
Maxent	80.6	71.1	55.5	47.2	61.5	54.4	49.4	35.2	38.9	45.7
Lingpipe	81.2	77.4	56.3	59.0	65.6	63.2	48,2	55.2	43.0	46.5

This strategy shows mixed results. The maximum entropy improved only with the Queenspeeches. LingPipe did better with the Government agreements, the NRC, and the Queenspeeches. The reasons for these different results are not clear, and will be part of further research.

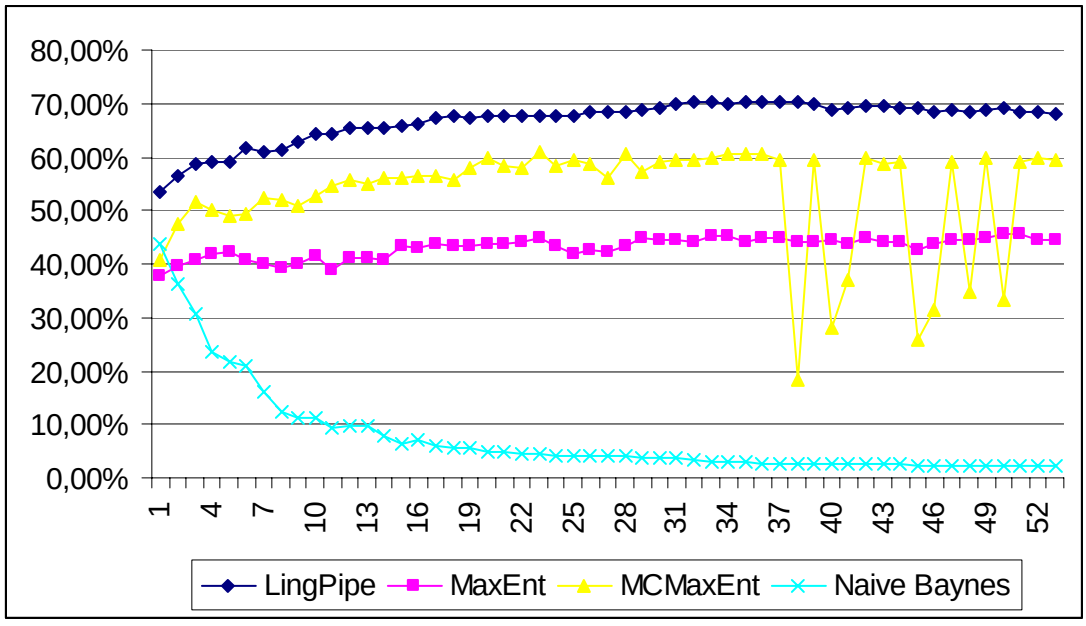
4. Improving trainingfile automatically

As said earlier, the most promising strategy for improving the trainingfile is to raise the number of entries. However, this is time consuming when done by hand, and we wanted to have a more efficient method to add new entries. Therefore we developed a small program that could add new entries with a high accuracy to the training file.

We started with the balanced trainingfile of 20 entries per code of the NRC dataset of 1104 manually coded newspaper articles. The remaining hand coded entries were used as the test file. When tested, the accuracy per algorithm was the following:

Ling: 55.2 %
MaxEnt: 32. 90%
MCMaxEnt: 37.12%
NaiveBayes: 49.49%

Then we coded the entire dataset of approximately 40.000 newspaper articles with all 4 algorithms and add after every run 15 new entries automatically to the trainingfile from those 2 codes which had the highest accuracy in the initial test round. For instance, if we found an accuracy of 0.94 with Maxent for code 19 and an accuracy of 0.89 for code 20 with Lingpipe, and all other algorithms scored lower, we added 15 form code 19 and 15 from code 20. After adding these 30 new entries to the training file we tested it against the test file, which in turn delivered us with a new accuracy table. We then run this new trainingfile for a second time against the 40.000 articles and added again 2*15 new entries to the trainingfile. We repeated this learning cycle 54 times (automatically). And the accuracy improved for the algorithms of LingPipe, and McMaxEnt to a certain level. The Naïve Baynes declined in accuracy, which is still a puzzle to us. The graph shows the results. The most important problem with this strategy is that the codes with the highest accuracy will be oversampled in the trainingfile. Fixing that, will be part of new test.



Using outcomes of key word searches for categories as textual features

To overcome the problem that supervised learners use transitory and fleeting words that are unique for a specific category in a given period of time only, human coders could delete such short-lived words from texts. They could also define meaning search strings that do not make use of such words, and feed the outcomes of each of these search strings as features of texts to a supervised learner. Here we will follow the last approach. We will use multinomial logistic regression analysis, which resembles closely MAXENTropy, as our learning algorithm. For the sake of the discussion it suffices to apply this approach to the NRC (n=1104) sample of human coded news articles.

Defining search strings in LUCENE, which enables quite advanced searches with distances between words in addition to Boolean searches, for the categories of the CPA-project is not an easy task, however. Category 301 deals with Health Care Reform, for example. Therefore it is insufficient to find texts about the *policy domain* of "Health Care". Only those texts should be highlighted that deal also with *policy measures* labeled as "reform". Google and Wikipedia can be used to find various synonyms and subtypes of health care, as well as near synonyms of reform. The lucene-string below consists of Dutch near synonyms of the policy measure "reform" separated by / followed by near-synonyms of the policy domain "health care", also separated by /.

301HealthCareReform
"stelselhervorming/hervorming/reorganisatie/stelselwijziging/marktwerking/privatisering/fusie/fusies/schaalvergroting/grootschalig/grootschalige/automatisering/efficiency/doelmatigheid/efficiëntie gezondheidszorg/zorg/zorgsector/zorgstelsel/medische/ziekenhuis/ziekenhuizen/zorgaanbieders/mantelzorg/awbz/ziekenfonds/ziekenfonds/ziektekostenverzekering/ziektekostenverzekeraar/ziektekostenverzekeraars"~20

In many other cases the definition of the category according to the manual coding instruction asks not only for a conjunction, for an AND-combination, of a specific policy domain and a specific policy measure, but also for a conjunction with specific *stakeholders*, or with specific *laws and procedures*. The problem with classification is that applying a fourfold conjunction will only result in a limited number of article, whereas the coding book does not prescribe precisely how to classify the 15 other types of combinations. Human coders apply such rules quite relaxed. They know that not every possible fourfold combination is defined in the coding instruction. They will apply their own coding rules when they encounter the other 15 combinations, which may result in a low reliability. Often the coding book does not mention explicitly whether a unifold, a twofold, a threefold or a fourfold conjuction applies to policy domain, policy measures, stakeholders, judicial procedures and policy outcomes. Therefore the definition of search strings for a given category, as well as the number of hits for that category, depend heavily on

arbitrary choices. To reduce this dependency, the number of hits for a given category is normalized in this paper by a division with the total number of hits in an article.

Whereas SVM, Maxent and Ling all succeeded in predicting 73.3%, 100% and 99.9% of the NRC n=11004-sample texts correctly, the method described here resulted in a performance percentage of 62% correct predictions only ($R_{Nagelkerke} = 0.87$). Nevertheless, this method has some potential. It's easy to put this method in top off bag of words approaches. Next, the search strings can be improved by looking at the number of mismatches and by finding new terms by means of the usage of newspapers and television, but also of television and the internet. Last but not least, each of the search terms can be split up systematically in four different search strings: policy area, policy aims, political support and political success.

Conclusions

This paper raises more question than it answers. However, we believe that we still can improve the trainingfiles first in various ways before we proceed to use the algorithms in an ensemble. This strategy adds to the efficiency of automated coding. To summarize our results: Stemming and stopword removal improves the results with some percentages. Balancing we don't know yet, although the N-gram classifier (LingPipe) shows some promising results. Adding new entries by hand seems always to be a good strategy, but it hurts efficiency. Adding new entries automatically, however, is especially promising for the N-Gram classifier (LingPipe) algorithm. Moreover, this last classifier gives in the Dutch case the best results in nearly all cases. The basic score starts with approximately 48% accuracy, after balancing, at least the NRC, the results go up to 55% and after automatic training the results improve 70%. Prediction can be based only on explicit search strings as well. The relatively poor performance percentage of 62% suggests that this method is possibly useful on top of other approaches. The meaning of the tests discussed here will be part of further research.

Appendix 1

1 Macroeconomics
2 Civil Rights, Liberties and integration
3 Health
4 Agriculture
5 Labor & Employment
6 Education
7 Environment
8 Energy
10 Transportation
12 Law, Crime, & Family issues
13 Social Welfare
14 Housing & Community Development
15 Banking & Commerce
16 Defences
17 Science & Technology
18 Foreign Trade
19 International Affairs & Aid
20 Government and Government Operations
21 Public Lands and watermanagement

Added for media analyis:
26. Weather and Natural Disasters
27. Fires
28. Arts and Entertainment
29. Sports and Recreation
30. Death Notices
31. Churches and Religion
99. Other, Miscellaneous, and Human Interest

References

Baumgartner, F., and B. Jones (1993) *Agendas and Instability in American Politics*. Chicago: Chicago University Press

Hillard, D., S. Purpura and J. Wilkerson (2008) "Computer-Assisted Topic Classification for Mixed-Methods" Social Science Research." *Journal of Information Technology and Politics 4(4).*

Jones, B and F. Baumgartner (2005). *The Politics of Attention: How Governement Prioritizes Problems.* Chicago: Chicago University Press.

Porter, M. (1980). "An Algorithm for suffix Stripping". *Program 16(3), 130-137.*

Purpura, S and D. Hillard (2006) "Automated Classification of Congressional Legislation" In: *Proceedings of the International Conference on Digitial Government Research* (pp. 219-225) New York: Association for Computer Machines.

van Atteveldt W, Kleinnijenhuis J, Ruigrok N. Parsing (2008a), Semantic Networks, and Political Authority: Using Syntactic Analysis to Extract Semantic Relations from Dutch Newspaper Articles. *Political Analysis*. 16(3): 428-446.

van Atteveldt W, Kleinnijenhuis J, Ruigrok PC, Schlobach S. (2008b), Good news or bad news? Conducting sentiment analysis on Dutch text to distinghuish between positive and negative relations. *Journal of Information Technology & Politics*. 5(1): 1-22.

Wolfgang (2008) in Wolfgang: *Text Classification Tools Version 0.6.* Available: http://docs.google.com/Doc?id=ddb38zh2_66dw4m3bc4 (last accessed12/03/2009).