Automatic Tag Cloud Summarisation of US Presidential Speeches

Savelie Corneguta, Elena Koritskaya, Maithreyi Venkatesh and Gloriose Hitimana

13th April 2015

1 Introduction

The aim of our project was to summarise a large document with a tag cloud. We chose to analyse and generate tags for important speeches made by US Presidents. Crucially, we set out to generate relevant tags as opposed to generate tags based on frequency alone.

Speeches made by US Presidents are often about important political decisions or in the case of inaugural speeches, they outline the issues the newly elected President intends to tackle. Using a tag cloud to summarise these speeches, by extracting the most relevant and important tags, would allow users to search for speeches which tackled certain issues. It would also help in summarising the issues a President hoped to tackle during his term in the White House.

2 Data

Python[1] was used to acquire and process the data described below The pre-processing and analysis of the data was also carried out using Python in addition to the Natual Language Toolkit (NLTK)[2].

2.1 Data Acquisition and Dataset Description

The Miller Center's "Presidential Speech Archive" [3] is a hand-curated dataset comprising of the transcripts of the most important speeches given by Presidents of the United States of America.

The transcripts of speeches from this archive were scraped to make up our dataset. It should be noted that this dataset is unlabelled so each speech was not tagged with a set of keywords that describe each speech.

The dataset comprised of 893 of the most important speeches made by US Presidents from 1789-2010; from George Washington to Barack Obama.

2.2 Dataset Pre-processing and Analysis

The transcript of each speech was broken down into tokens using the NLTK tokeniser with a custom regular expression. The NLTK tokeniser, with a custom regular expression, was used instead of simply splitting the text based on spaces to ensure that abbreviations, numbers, words with hyphens and other special characters were preserved. The position of the word, or token, in the speech was preserved so that they could be used in the tag generation models.

Following tokenization, we obtained a list of the top 100 most common words in the English language and used this for stop word removal. Not all the words on this list were removed as they were deemed as being useful in the context of our dataset, e.g. "oil". Some extra words specific to this dataset were also

[1]https://www.python.org/
[2]http://www.nltk.org/
[3]http://millercenter.org/president/speeches

removed, e.g. ”bless” due to their frequent occurrence.

It should be noted that this customised stop word removal was primarily useful when working with bigrams. Our models successfully eliminated these common words, or ”noise” when working to generate unigrams.

We preserve the case of the words except for the first word of the sentence which is transformed to upper case. This is for better visual presentation of the tags.

3 Methodology

We implemented all the methods and evaluation using Python, making use of the Natural Language Toolkit (NLTK) for tokenization.

3.1 Evaluation

The dataset was unlabelled so each speech did not come with a set of relevant tags that described or summarised the speech. To obtain labels, we generated and extracted the top 20 most relevant tags for each speech using a keyword extraction tool, AlchemyAPI[4].

Statistical IR measures, precision and recall measures were employed to evaluate each of the methods described in this section.

Precision is the fraction of retrieved tags that are relevant:

$$precision = \frac{|relevant\ tags \cap retrieved\ tags|}{|retrieved\ tags|}$$

Recall is the fraction of relevant tags that have been retrieved:

$$recall = \frac{|relevant\ tags \cap retrieved\ tags|}{|relevant\ tags|}$$

The relevant tags are those keywords extracted by AlchemyAPI. In order to compute these measures, the top 20 relevant tags and the tags extracted by each of the algorithms were converted to lowercase then compared.

However using these statistical measures alone was not sufficient for evaluating our keyword extraction methods. Using only these measures would result in us trying to essentially replicate the extraction algorithm being used by AlchemyAPI. This is because these statical measures only compare how different our tags are from AlchemyAPI’s tags for a speech. This could be argued as being subjective unless a great deal of crowd sourcing is used to obtain tags that multiple, reliable users agree on. Ideally, for each method we would use crowd sourcing to manually analyse the tags extracted by each method for each speech and assess their suitability. However given the time frame this is not really possible.

These statistical measures were not used for all keyword extraction methods described in this report. In some cases these measures were simply not suitable. If time permitted we would have devised our own evaluation measures for these methods.

Hence in addition to using statistical measures, we carried out an in-depth analysis of our methods. This was carried out by selecting a particular speech and then manually comparing and contrasting the generated tags with the ones extracted by AlchemyAPI.

[4]`http://www.alchemyapi.com/`

3.2 Method 1: Vanilla TF-IDF

The first method implemented is the term frequency-inverse document frequency (TF-IDF) to extract unigrams. TF-IDF is used to reflect the importance of a word in a document in a corpus. In this case the corpus is the set of speeches while the vocabulary is made up of the all the unique tokens in the corpus.

There are many variations of TF-IDF with multiple ways to compute both the term frequency and inverse document frequency components. In this project, we make use of the most basic variation in which the term frequency (TF) component is the raw term frequency (or count) and the inverse document frequency (IDF) component give by:

$$log(\frac{N}{df_t})$$

where:

- N is the number of documents in the collection.
- df_t is the document frequency i.e. the number of documents in the collection that contain the term t.

3.3 Method 1.1: TF-IDF with Modified Count

This method is similar to method described in the previous section however, the term frequency has been modified to incorporate the position of the word in the transcript. We hypothesised that the introduction and conclusion of the speech contains terms that outline what the speech is about. We analysed the beginning of the speech sets out the topics the President intends to discuss while the end summarises the topics covered in the speech.

To incorporate this analysis into our method we modify the weights given to the words depending on their position in the document. So instead of using the raw term frequency of a word, we use the sum of weights related to the position each time a word is seen in the document.

The weight of a term based on its position is given by $w(k) = 1 + |\frac{2k}{DL-1}|$. This is a concave function that produces a weight whose value is around 1 when a word is seen at the beginning or at the end of the speech. Words that appear in the middle of the speech are assigned a weight of around 0.5.

$$f^m(q_i, D) = \sum w(k) \times I(q_i = D(k))$$

where:

- $f^m(q_i, D)$ is the modified q_i's term frequency in the document D.
- $D(k)$ is the kth word in the document.
- k is the position in the document.
- DL is the document length i.e. the number of tokens in the document.

As we can see only the term frequency (TF) component of the TF-IDF method (described in the previous section) has simply been altered to use this modified term count. The IDF component of the method remains the same as the one previously described.

3.4 Method 2: Modified Okapi BM25

The Okapi BM25 [2] method has been modified and used for the purposes of keyword extraction by Karkali et al. [1]. Similarly, we also modified the BM25 algorithm for this task. Unlike the method proposed by Karkali et al. we do not use the temporal document frequency. Our modified BM25 method is:

$$BM25M(t,d) = \frac{tIDF(t) \times tf(t,d) \times (k_1 + 1)}{tf(t,d) + k_1 \times (1 - b + b\frac{|d|}{avgdl})}$$

where:

- $tf(t,d)$ is the number of times the term occurs in the document (or speech).
- $|d|$ is the number of tokens in document d.
- $avgdl$ is the average document length in the corpus.
- k_1 and b are free parameters; in our case $k_1 = 1.6$ and $b = 0.75$. These parameters control the document length normalisation and the term frequency saturation, respectively.

$tIDF$ is the defined as:

$$tIDF(t) = log(\frac{N - df(t) + 0.5}{df(t) + 0.5})$$

where:

- N is the number of documents in the corpus.
- $df(t)$ is the document frequency i.e. the number of documents that contain the term t.

Note that each document corresponds to a single speech.

3.5 Method 2.1: Modified Okapi BM25 with Modified Count

This method implemented the modified BM25 model described in section 3.6 with the modified term count described in section 3.3.

3.6 Method 3: Clustering

In this method use clustering to group the tags generated by each of the methods described above in an attempt to construct a better description of each speech. We used the word to vector model which produces a vector of length 1 for each term (this approach is described in section 3.8. Based on the n tags achieved from each of the models above we use the k-means clustering algorithm to generate $fracn4$ clusters.

It should be noted that this is one of the methods for which statistical measures were not employed. Instead manual analysis of a number of examples were carried out. In the results section we present on of the examples studied.

3.7 Method 4: Merging Multiple Methods

This method extracts keywords by combining the results of previously described algorithms; the BM25 algorithm executed to generate both unigrams and bigrams. In order to merge these two set of keywords without duplication we compare unigram keywords with all words in bigram tags. To remove duplicate tags, we exclude all unigram tags which are also seen as a part of bigram keywords.

3.8 Method 5: Generalisation of Bigrams

This method makes use of Word2vec[5] which is a tool that provides a framework for computing vector representations of words. Word2vec was published by Google and is a neural network implementation of vector representation of words. This network takes a text corpus as input and generates the word vectors as output. These vector representations have special characteristics which allows us to compare vectors to see words

[5] `https://code.google.com/p/word2vec/`

that have similar meanings. This tool allows us to train models for different vector dimensionality.

We have trained the Word2vec tool on our corpus of speeches to produce vectors which have a dimensionality of 50. Word2vec constructed vectors for our vocabulary representing the context similarity of words. Following this, we use the modified BM25 model with modified count to extract the top 20 most relevant bigrams instead of unigrams.Then we used the Word2vec representations to find the most similar unigram to each bigram.

Hence, the Word2vec model gives us the ability to get the most similar unigrams for a given bigram. This gives us a generalisation of bigram tags and provides us with related unigram tags based on the meaning rather than the term frequency.

4 Results

In this section we present statistical measures for some of the methods described in addition to a more detailed evaluation of the tags generated by each method for Ronald Reagan's First Inauguration speech he gave on the 20th of January 1981 ("reagan_speeches_speech-3407" in the data/ folder).

The top 20 most relevant tags as generated by AlchemyAPI for Reagan's speech are as follows:
America; American; people; orderly; transfer; economic; decisions; momentous; occasion; personal; indignity; Speaker o Neill; human; misery; Reverend Moomaw; idle; industries; Senator Hatfield; commonplace; occurrence

4.1 Method 1: Vanilla TF-IDF

Precision	0.101317
Recall	0.107286

Table 1: Precision and recall measures for the vanilla TF-IDF method used to extract tags. Compared against the tags generated by AlchemyAPI.

The top 20 most relevant tags as extracted by this method are as follows:
heroes; Treptow; penalizes; markers; crosses; dreams; weapon; row; productivity; Martin; Americans; Earth; heal; Salerno; mortgaging; Pork; Rainbow; crushes; Chop; Chosin

As shown in the table above, the precision and recall aren't extremely high. The reason for this is highlighted when we compare the tags generated by this method for Reagan's speech against those generated by AlchemyAPI. Firstly, AlchemyAPI generated tags that were not unigrams so this clearly affected the measures. These tags were mostly names. In comparison, the tags generated by our method only produced unigrams.

4.2 Method 1.1: TF-IDF with Modified Count

Average Precision	0.098102
Average Recall	0.103900

Table 2: Precision and recall measures for the TF-IDF with modified count method used to extract tags. Compared against the tags generated by AlchemyAPI.

The top 20 most relevant tags as extracted by this method are as follows:
Treptow; penalizes; markers; crosses; heroes; Martin; row; Moomaw; weapon; every-4-year; Rainbow; barbershop; Reservoir; Chosin; Chop; Pork; Salerno; crushes; Belleau; mortgaging

The tags produced by this method for Reagan's speech do not really vary wildly compared to those produced by the previous method. In fact, a decrease in the precision is observed.

4.3 Method 2: Modified Okapi BM25

Average Precision	0.055516
Average Recall	0.057183

Table 3: Precision and recall measures for the modified Okapi BM25 method used to extract tags. Compared against the tags generated by AlchemyAPI.

The top 20 most relevant tags as extracted by this method are as follows:
Treptow; penalizes; markers; crosses; Pork; every-4-year; crushes; mortgaging; Rainbow; barbershop; Chosin; Chop; Moomaw; Reservoir; exemplar; truckdrivers; smother; Salerno; Belleau; cabbies

The average precision and recall measures greatly decreased when compared to the TF-IDF models indicating that this model didn't perform as well. Analysing the tags generated by the this method for Reagan's speech as compared to those compared by AlchemyAPI we can see why this is the case.

4.4 Method 2.1: Modified Okapi BM25 with Modified Term Count

Average Precision	0.074313
Average Recall	0.076321

Table 4: Precision and recall measures for the modified Okapi BM25 with modified term count method used to extract tags. Compared against the tags generated by AlchemyAPI.

The top 20 most relevant tags as extracted by this method are as follows:
Treptow; penalizes; markers; crosses; Reservoir; every-4-year; Reflecting; crushes; cabbies; Belleau; mortgaging; Salerno; exemplar; Pork; Chop; Chosin; Moomaw; Rainbow; barbershop; smother
The average precision and recall measures increase slightly as compared to the previous method. However it still performs worse than the first two methods. Again, the tags generated by this method for Reagan's speech are almost exactly the same as the previous method.

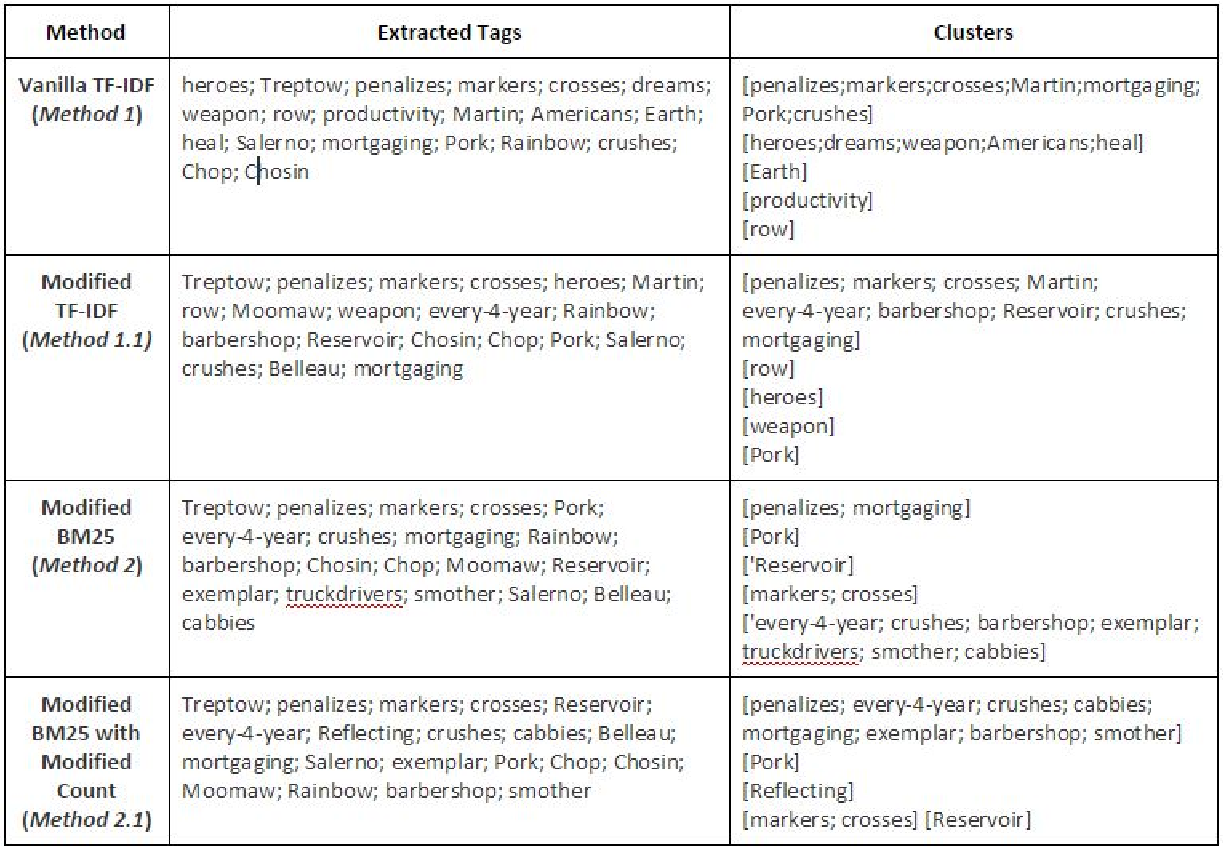
4.5 Method 3: Clustering

As shown by the results, the clusters produced by the method were not very good because the tags generated by the models were not extremely accurate or helpful. While in theory this method proved to be good, in practice using k-means clustering to group the tags was not successful. Hence, this method was not employed in the implementation of the next few methods.

4.6 Method 4: Merging Multiple Methods

Upon inspection of the output, the precision and recall measures were not computed for this method due to time constraints and processing power constraints. We did not have sufficient computational power required to compute these measures on the entire corpus in the time frame. Hence the evaluation was carried out using manual analysis of the tags generated by this method for certain speeches. Of course we acknowledge that this is not a scientific approach.

Merging procedure in particular example to not exclude many words. The only unigram generated by the modified BM25 with modified term count method and a part of bigrams is 'Treptow'. So in case of merging we would have to keep all unigrams except 'Treptow' and all bigrams.



Method	Extracted Tags	Clusters
Vanilla TF-IDF (*Method 1*)	heroes; Treptow; penalizes; markers; crosses; dreams; weapon; row; productivity; Martin; Americans; Earth; heal; Salerno; mortgaging; Pork; Rainbow; crushes; Chop; Chosin	[penalizes;markers;crosses;Martin;mortgaging; Pork;crushes] [heroes;dreams;weapon;Americans;heal] [Earth] [productivity] [row]
Modified TF-IDF (*Method 1.1*)	Treptow; penalizes; markers; crosses; heroes; Martin; row; Moomaw; weapon; every-4-year; Rainbow; barbershop; Reservoir; Chosin; Chop; Pork; Salerno; crushes; Belleau; mortgaging	[penalizes; markers; crosses; Martin; every-4-year; barbershop; Reservoir; crushes; mortgaging] [row] [heroes] [weapon] [Pork]
Modified BM25 (*Method 2*)	Treptow; penalizes; markers; crosses; Pork; every-4-year; crushes; mortgaging; Rainbow; barbershop; Chosin; Chop; Moomaw; Reservoir; exemplar; truckdrivers; smother; Salerno; Belleau; cabbies	[penalizes; mortgaging] [Pork] ['Reservoir] [markers; crosses] ['every-4-year; crushes; barbershop; exemplar; truckdrivers; smother; cabbies]
Modified BM25 with Modified Count (*Method 2.1*)	Treptow; penalizes; markers; crosses; Reservoir; every-4-year; Reflecting; crushes; cabbies; Belleau; mortgaging; Salerno; exemplar; Pork; Chop; Chosin; Moomaw; Rainbow; barbershop; smother	[penalizes; every-4-year; crushes; cabbies; mortgaging; exemplar; barbershop; smother] [Pork] [Reflecting] [markers; crosses] [Reservoir]

Figure 1: Table shows the clusters produced for the tags generated by each method.

4.7 Method 5: Generalisation of Bigrams

The precision and recall measures were again not computed for this method due to time and processing power restrictions as for the previous model.

The top 20 bigram tags generated by the modified BM25 model with modified count were: *believe Americans; dreams hopes; hopes goals; believe fate; Martin Treptow; fate fall; act worthy; fall us; happiness liberty; re sick; freedom those; provide opportunity; no misunderstanding; go away; growth government; us renew; few us; look answer; each Inaugural; taken aimed*

As we can see bigram tags are more meaningful in comparison to term-frequency based unigrams. Unigram tags achieved from bigrams tags as a closed to bigrams by meaning seem to be more descriptive as well:

think; inspire; tasks; indignation; Airborne; peril; propriety; must; freedom; high-quality; just; ample; inference; without; system; me; themselves; determine; unseated; law-defying

While these tags are still very different from those extracted by AlchemyAPI, out of the methods we tried this was the most successful method.

5 Discussion

As shown by the results, the best method was the that described in section 3.8. While it cannot be compared to the tags generated by AlchemyAPI the tags generated by the generalisation of bigrams method made the most sense w.r.t. to Reagan's speech.

If we were to repeat this task again, we could make use of the Word2vec tool to compare keywords extracted by the methods we implemented and computed the distance to tags extracted by AlchemyAPI.

6 Data Visualisation

In conjunction with the aim of our project, we observed that data visualisation is a valuable way to communicate important information at glance. Hence we decided to visualise the tags extracted by the best method which was the generalisation of bigrams and those using alchemy using a word clouds.

We used the D3.js [6] Javascript library which can be used to manipulate document based on data to produce word clouds for the speeches. Figures **??** and 3 present word clouds made up of the tags generated by our generalisation of bigrams method and generated by the AlchemyAPI, respectively.

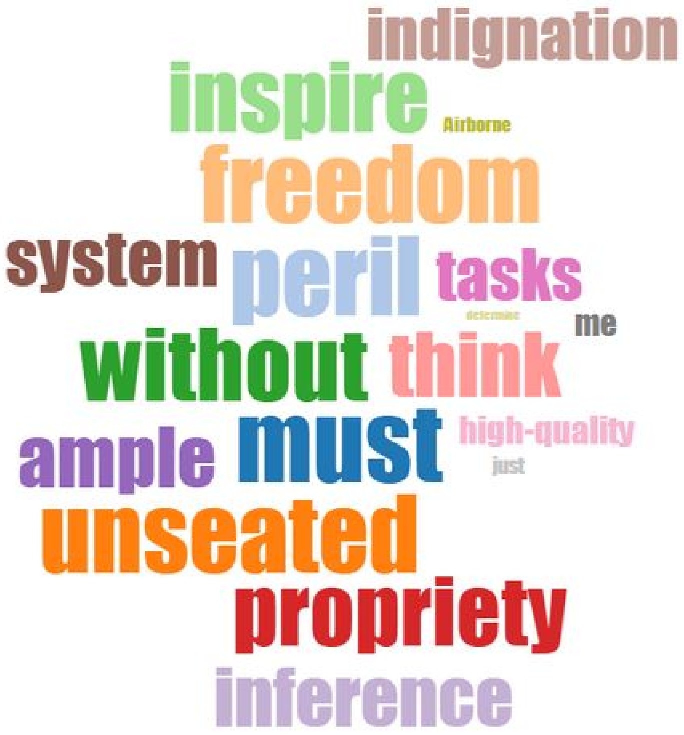


Figure 2: A word cloud made up of tags generated by the generalisation of bigrams method that summarises Ronald Reagan's speech.

The tag-cloud-visuals/clouds folder contains tag clouds generated for all of Barack Obama's speeches and Ronald Reagan's Inaugural speech which we have made use of throughout this report. It should be noted that these tag clouds were meant to be included as part of an interface that would have showcased the effectiveness of our method vs. AlchemyAPI for each speech. The files are marked by the same id's used to tag the speeches in the data/ folder. The info.tsv file can be used to find the appropriate speech in the data folder.

Please note that the colours or weight do not signify anything in particular. However if we were to implement this tool properly we would use the weights assigned to the tags to signify relevance.

[6]`http://d3js.org/`

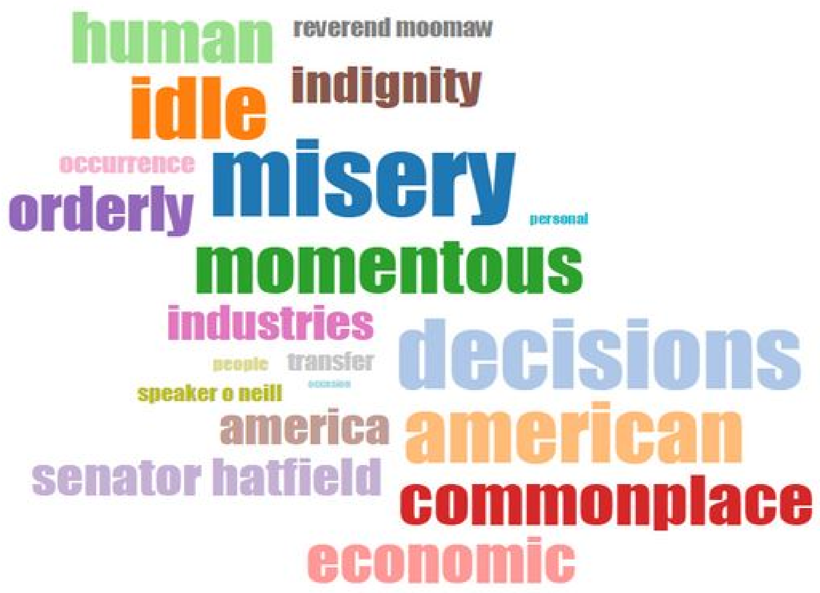


Figure 3: A word cloud made up of tags generated by AlchemyAPI for Ronald Reagan's speech.

References

[1] M. Karkali, V. Plachouras, C. Stefanatos, and M. Vazirgiannis. Keeping keywords fresh: a bm25 variation for personalized keyword extraction. In *Proceedings of the 2nd Temporal Web Analytics Workshop*, pages 17–24. ACM, 2012.

[2] S. Robertson and H. Zaragoza. *The probabilistic relevance framework: BM25 and beyond*. Now Publishers Inc, 2009.