Semantics in user-added text for categorizing press releases

Simon Paarlberg (s062580)
28 June 2012

Kongens Lyngby
IMM-B.Eng-2012-22

Department of Informatics and Mathematical Modelling.
Technical University of Denmark (DTU)
Supervised by Michael Kai Petersen

Summary (English)

The aim of this thesis is to analyze and test whether Latent Semantic Analyses (LSA) can be used to improve the delivery of targeted press releases. This is done by using existing content of press releases as a base for finding relevant media outlets. The focus in the thesis is on how LSA works by examples, using the free software package gensim. Various approaches to using LSA are covered along with background information on the media industry.

The result of this thesis has been conducted on data from 138,363 articles from 28 Danish online news outlets and the Danish version of Wikipedia. The result is inconclusive, most likely because the dataset was not big enough.

Resumé (Dansk)

Formålet med denne afhandling har været at analysere og teste om Latent Semantisk Analyser (LSA), kan bruges til at forbedre leveringen af målrettede pressemeddelelser. Dette er forsøgt gjort ved at benytte eksisterende indhold af pressemeddelelser, som en base for at finde relevante medier. Afhandlingen har fokuseret på at vise hvordan LSA kan benyttes til at klassificere kendt data. Dette er sket ved brig af software-pakken gensim.

Resultatet af denne afhandling er blevet gennemført på data fra 138,363 artikler fra 28 danske online nyhedsmedier - og den danske version af Wikipedia. Konklusionen på arbejdet er at det ikke har været muligt at opnå en forbedring på klassificeringen ved brug af LSA. Dette kan skyldes at de brugte datasæt ikke har været store nok.

Preface

This thesis was written at the Department of Informatics and Mathematical Modelling (IMM) at the Technical University of Denmark (DTU) in the period between April 2nd 2012 and June 28th 2012 (13 weeks) as a conclusion to my Bachelor degree.

The project was made in collaboration with Press2go ApS. (press2go.com), and the documentation of the work can be found in this report.

Copenhagen, June 28th 2012.

Simon Paarlberg

Table of Contents

Report structure

This report will first introduce a problem from a particular Copenhagen company along with the environment of factors surrounding the procedures where the problem occurs. Details about the history of the media industry and where it is headed will be included to give the reader a sense of understanding of the problem in an industry context.

The theory chapter introduces some of the key concepts in working with LSA. Here, the gensim package is also introduced by demonstrating how to obtain topics from a minimalistic dataset.

The analysis chapter will focus on the wanted structure of the data set, the sources to get the data and lastly on how to process the data using LSA and what is possible to do with the setup that has been chosen.

The thesis will end with a conclusion on the analysis and a brief summary of what knowledge has been collected from the project.

Introduction

With the introduction of digital media broadcasting, the creation and consumption/use of information and entertainment have become even more available to the general public. This has resulted in a vast fragmentation of the media landscape beginning in the mid-90s and rapidly evolving every day. This has led to growing competition for the consumers' time and an increase in the amount of news stories released (Lund et al., 2009)

The changes are happening fast, and professionals working in the field of public relations (PR) have a hard time following the constantly growing and changing media market. As a way to assist these changes, the company Press2go was founded in 2005 with the goal of facilitating the connection between PR employees and the media outlets. The backbone of the company is its software for handling contact to the press. The software has two focus areas: one is a delivery system for broadcasting targeted emails and the other a media database containing the majority of all media outlets worldwide.

Press2go's PRM tool works in simple terms by writing a story into a predefined template which then transforms the text into the final press release. The software is then able to send the press release to a number of media outlets like The Daily Mail, NY Times, etc. The choice of media outlets is handpicked from an in-house database that holds information about a great number of media outlets from around the world. The typical approach to getting started using the software is to collect some initial media outlets that are relevant to a company. These outlets are then placed on a list for later use. This is called a media list and is meant to be used as a shortcut to sending out press releases.

This works great if it wasn't for the constant changes in the media market. The changes mean that the PR worker has to continuously refine the media lists so that they correspond to the media landscape. Unfortunately, this task is put off because of difficulty resulting in poor reception from the receiving media outlets.

This is where this project's goal comes into fruition. Ultimately, we want to use Natural Language Processing (NLP) in the form of Latent Semantic Analysis (LSA) to find relevant media outlets based on the content of the press release. The solution should work in the sense that relevant media outlets should be suggested in the user interface once the PR worker has completed writing the message. The PR worker should then get to choose which outlets are included in the upcoming broadcast.

Problem definition

This thesis will cover the analysis of using Natural Language Processing (NLP) in finding connections between written content of a press release and relevant media outlets. The approach will be to try to see if LSA (Latent Semantic Analysis) can be used to predict which media outlets have interest in a given press release.

The thesis will cover the following question parts:

- What is the theory behind LSA and how does it work?
- What kind of abilities will LSA approve on?
- How does it work in practice?
- What is the problem with the existing solution for finding relevant media outlets?
- How is the present solution structured?
- What kind of extra data would be needed to work with LSA?
- Where would the data come from?
- What are the results coming out of the analysis?
- Is it usable for what we want to do?

The media industry

Here follows an introduction to the existing media industry with emphasis on Press2go.

When working with PR employees of all sorts, Press2go have noticed a general tendency among its users not to make it a priority to target the media outlets precisely. Targeting should be understood as the choice of media outlets to which a press release will be delivered. The reason for this inaccuracy is that there has been a habit in the industry to inform a small static set of media outlets on all events. Because of the rapid change in the media market, the news landscape has been forced to change as well.
In 2008, a Danish research project called "Projekt Nyhedsugen" (Project News Week) documented the perception that the Danish population today is subject to more news than 10 years ago. In an in-depth study, they documented that in any given week in 1999, a collection of news outlets (See Appendix A for a detailed list) brought 32,000 "news" stories, where the corresponding figure in 2008 was 75,000 stories (an increase of ~134%). According to the Danish journalist union, the number of journalists has not increased accordingly in the same period. Neither has the money spent on original journalism. The thing that has increased is the number of PR employees in the public and private sectors. In the private sector, there has been an increase of ~139%[1], while the public sector has seen an increase of ~108% in the communication staff (Lund et al., 2009:165). This has the effect that there is an increase in news material being sent to the media outlets, thus making it harder for journalists to sort through the incoming mail for relevant information.
Another effect of the technical evolution is a fragmentation of the media industry, thus making the target of a story much narrower than ever before. The PR employees need to be more versatile, and, more importantly, they need to target the promotion of the message to the right outlets. Otherwise the story will not get out. This means that the PR employees need to show more craft and cunningness to get the same results as they achieved before.

While Press2go's software supports the tasks generally used in the media industry, they also encourage their customers to create tight segmented media lists that can be used in combination to deliver the message to a static set of media outlets. They do this to support the workflow of many PR employees, simply because the creation of media lists for each press release would be too time-consuming. At the end of the day, this means that a press release is being sent out to a cluster of media outlets based on a less specific media list, rather than by the content of the actual press release. Of course this only happens if the customer picks an existing media list without adjusting for the content of the message — which, according to the support staff at Press2go, there are strong indications that they do.

[1] The private sector had an increase from 398 in 1999 to 953 in 2008 while the public sector has an increase from 172 in 1999 to 357 in 2008.

Theory

This chapter will introduce some of the key theories and models used for understanding and working with LSA and gensim.

Bag of words

The bag of words (BOW) model is one of the simplest representations of document content in NLP. The model consists of content split up into smaller fractions. In this thesis, we will be restricted to plain text. In this case, the model stores all used words in an unordered list. This means that the original syntax of the document will be lost, but instead we will gain flexibility in working with each word in the list. As an example, the operation of removing certain words or simply changing them would become much easier, since ordinary list operations can be used. The BOW representation can be optimized even further if all words are translated to integers, by using a dictionary to store the actual words. This will be covered later in this chapter.

Document-term matrix

The document-term matrix is a basic representation of our corpus (group of documents) that consists of rows of words and columns of documents. Each matrix value consists of the weight for a specific term in a specific document. The calculation of the words can vary, but in this thesis, it will represent the number of times the word is present in the document. The advantage of having the corpus represented as a matrix is that we can perform computational calculations on it using linear algebra.

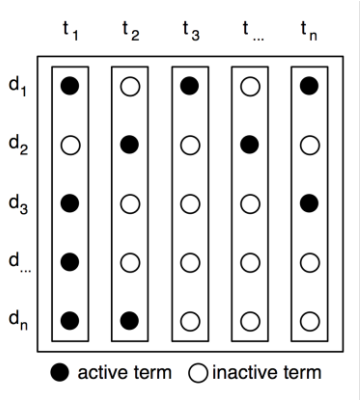
Latent Semantic Analysis

Latent Semantic Analysis (LSA) is a vector-based method that assumes that words that share the same meaning also occur in the same texts (Landauer and Dumais, 1997:215). This is done by first simplifying the document-term matrix using singular value decomposition (SVD), before finding closely related terms and documents, using the principals of vector-based cosine similarities. By plotting a number of documents and a query as vectors, it is possible to find the Euclidian distance between each document vector and a query vector. This can also be thought of as grouping together terms that relate to one another. This operation should, however, not be confused with clustering of the terms, since each term can be part of several groups.

In the following sub-chapter, we will introduce the most important building blocks of LSA before piecing it all together in the end and show how to perform queues and extend a trained index.

The corpora as vectors

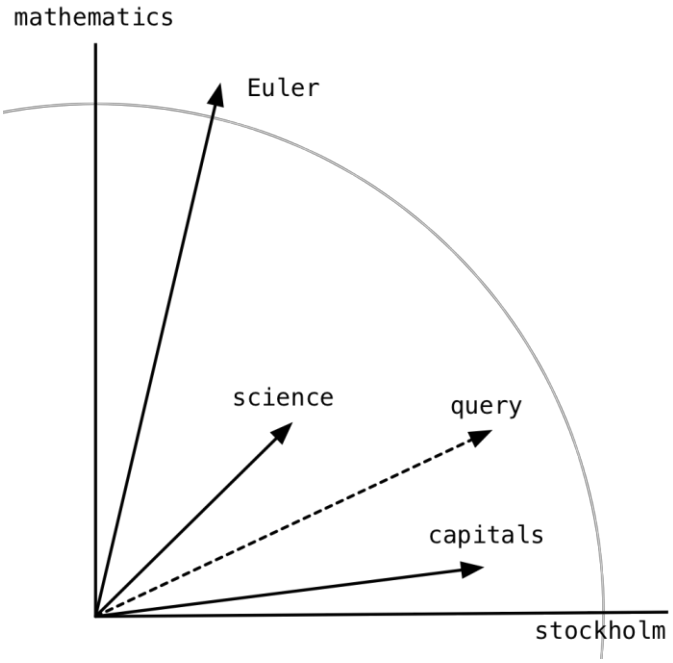
When our corpus is in the form of a document-term matrix, it is simple to treat it as a vector space model, where we perceive each matrix column as a vector of word weights.



Since only a minor part of the words are used to represent each document, this means that the vectors mostly consist of zeroes, also called sparse vectors.

The advantage of representing each document as a vector is that we then have the ability to use geometric methods to calculate the properties of each document. When plotting the vectors in an n dimensional space, the documents that share the same terms tend to lie very close to each other.
To demonstrate this, we will display a couple of vectors from a very minimalistic document-term matrix. This is because we are only able to visualize up to three dimensions in a three-dimensional world. Since paper only has two dimensions, we will simplify even further by only using two dimensions.

In the following the document vectors will be plotted. This means that the terms are equal to the dimensions used. Since we only have two dimensions, our term document matrix can only have two terms. We have randomly picked the terms "mathematics" and "Stockholm". This will be our two dimensions. If we then have three documents that we want to plot, we will take the weight of the two terms in each of the three documents and plot them as vectors (from the center). If the first document is about Leonhard Euler, the next is about Scandinavian capitals, and the third is about science in Sweden, then plotting the three documents by the two terms would perhaps look like this:



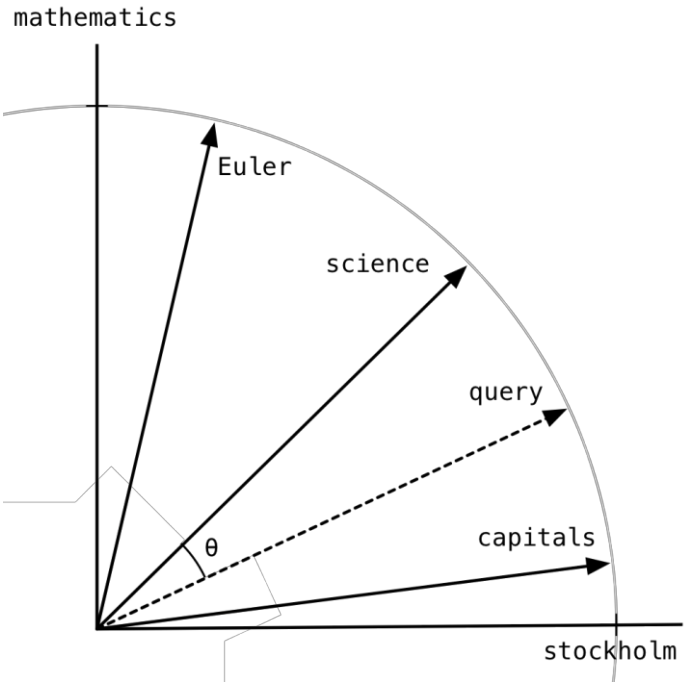
The length of the vectors tells us something about how many times the terms are present in each document. Let us say that the grey line marks a scale of 10, then the document about Euler would contain around 11-12 words about mathematics and around three words about Stockholm. The same principle goes into creating the rest of the vectors. In this way, we can plot any query onto the plane and see how close it is to any of our existing documents. This means that the closer two document vectors are, the closer their content is to each other. This again means that we want the angle to be as small as possible, and therefore we want cosine to be as big as possible. Since we are working with vectors, we know that we can calculate the angle between two vectors by

$$\cos(\theta) = \frac{A \bullet B}{||A||||B||}$$

where A and B are the two vectors. This is referred to as the cosine similarity of the two vectors. To make this even simpler, we can normalize the vectors so that their length is always equal to 1. This means that we can reduce the above formula to

$$\cos(\theta) = \frac{A \bullet B}{||A||||B||} \Rightarrow \frac{A \bullet B}{1*1} = A \bullet B$$

which is very easy to calculate on a large set of vectors. The method above is very important for finding similarities in large corpora. The result of the normalized vectors will render the model as follows:



The range of the cosine similarity is −1 to 1, where 1 indicates that the two vectors are exactly the same, 0 that the two vectors are independent and −1 that the two vectors are exactly opposite.

tf-idf

When using the values from the document-term matrix, commonly used words tend to have an advantage, because they are present in almost every document. To compensate for this, tf-idf (Term Frequency - Inverse Document Frequency) is very often used.
The point of using tf-idf is to demote any often used words, like "some", "and" and "i", while promoting less used words, like "automobile", "baseball" and "london". This approach easily removes any commonly used words while promoting domain-specific words that are significant for the meaning of the document. By doing this, we can avoid using lists of stop-words to remove commonly used words from our corpus.

The model is comprised of two different elements. First, the Term Frequency (TF) part that is the number of times a term is represented in a document, while Inverse Document Frequency (IDF) is the number of documents in the corpus divided by the document frequency of a word, but inverted.

Because tf-idf uses the logarithmic scale for its calculations of IDF, the result cannot be negative. This means that the cosine similarity cannot be negative either. The range of a tf-idf cosine similarity is therefore 0 to 1, where 0 is not similar and 1 is exactly the same.

Singular Value Decomposition

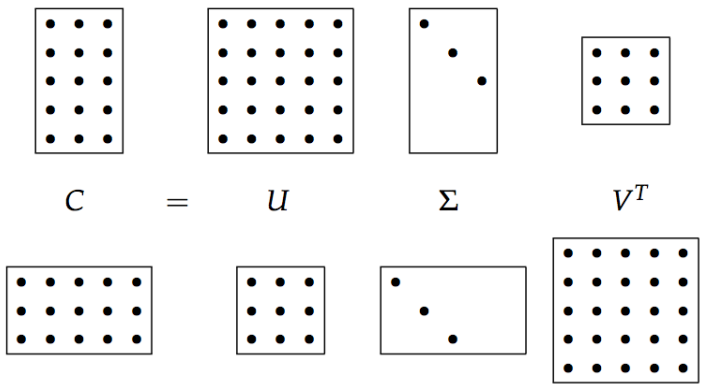
Finding document similarities by using vector spaces on a document-term matrix can often become a costly process. Since even a modest corpus is likely to consist of tens of thousands of rows and columns, a query on the model can prove to take up a lot of time and processing power. This is because each document vector has a length of all the terms in the matrix. Finding the closest vectors in all dimensions is not a small task.
What singular value decomposition (SVD) gives us is the ability to perform two important tasks on our document-term matrix, namely reducing the dimensions of our matrix and in this process finding new relationships between the terms across our corpora. This will create a new matrix that approximates the original matrix, but with a lower rank (column rank).

The basic operation behind SVD is to unpack our matrix, promote non-linear terms and use this to reduce the size of our matrix without losing much quality. Then afterwards, the data is packed up again into a smaller, more efficient version. All of this while keeping the essence of our original matrix. A fortunate side-effect of this process is a tendency to bring the co-occurring terms closer together.

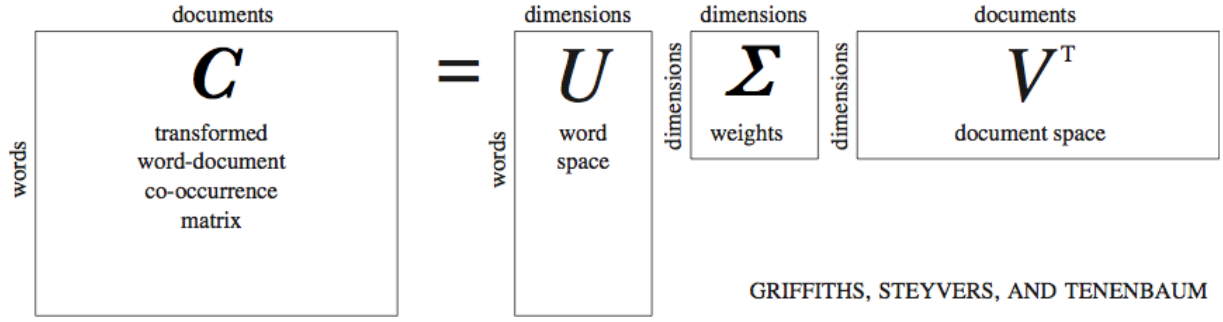
In the following sub-chapter, we will try to go a little deeper into the workings of SVD.
If we have the M × N document-term matrix C, then we can decompose it into three components by the following

$$C = U\Sigma V^T$$

where U is an M × M matrix where the columns are the orthogonal eigenvectors of CC^T, V is an N × N matrix where the columns are the orthogonal eigenvectors of C^TC and Σ is an M × N zero matrix with the eigenvalues of CC^T (or C^TC) placed on the diagonal in descending order and then squared (Manning et al. 2008:408). These are known as the singular values of C



[Fig. 1] *Depiction of two cases of SVD factorization, first where M > N and second where M < N.*



[Fig. 2] *Different viewpoint that reflects the words, document and dimensions of the products. "The transformed word–document co-occurrence matrix, C, is factorized into three smaller matrices, U, Σ, and V. U provides an orthonormal basis for a spatial representation of words, Σ weights those dimensions, and V provides an orthonormal basis for a spatial representation of documents." (Griffiths et al., 2007:216)*

To reduce the size of our matrix without losing much quality, we can perform a low-rank approximation on matrix C. This is done by keeping the top *k* values of Σ and setting the rest to zero, where *k* is the new rank. Since Σ contains eigenvalues in descending order, and the effect of small eigenvalues on matrix products is small (Manning et al. 2008:411), the zeroing of the lowest values will leave the reduced matrix C' approximate to C. How to retrieve the most optimal *k* is not an easy task, since we want k top large enough to include as much variety as possible from our original matrix C, but small enough to exclude sampling errors and redundancy. To do this in a formal way, the Frobenius norm can be applied to measure the discrepancy between C and C_k (ibid.:410). A less extensive way is just to try out a couple of different *k*-values and see what generates the best results.

When reducing the rank of a document-term matrix, the resulting matrix C' becomes far more dense compared to the original matrix C, which means that although the dimensions of our original matrix C become smaller, the content of C' becomes more compact, thus requiring more computational power. Because of this, we do not reduce the dimensions of C.

Piecing it together

Once we have applied tf-idf and SVD to our document-term matrix, we can again apply the cosine similarity procedures. With the rank reduction of the original matrix, what we have is an approximation of the document-term matrix, with a new representation of each document in our corpus.
The idea behind LSA is that the original corpus consists of a multitude of terms that in essence have the same meaning. The original matrix can in this sense be viewed as an obscured version of the underlying latent structure we discover when the redundant dimensions are forced together.

Another important advantage of LSA is its focus on trying to solve the synonymy and polysemy problem. Synonymy describes the instance where two words share the same

meaning. This could be "car" and "automobile", "buy" and "purchase", etc. This can cause problems when searching for documents with certain content, only to receive results not bearing the synonyms of the query text. As for polysemy, this describes words that have different meanings depending on the context. This could be "man" (as in human species or human males or adult human males), "present" (as in right now, a gift, to show/display something or to physically be somewhere (Wikipedia, 2012)), etc. In cases of polysemy, documents that have nothing to do with the intent of the query can easily become falsely included in the result.

When looking at the vector space model, the problem of synonymy is caused by the query vector not being close enough to any of the document vectors that share the relevant content. Because of this, the user that makes the query needs to be very accurate in searching, or the search engine needs to have some kind of thesaurus implemented. The latter can be a very costly and inaccurate affair.
When putting our original document-term matrix through LSA, synonyms are usually not needed, since similar terms should be grouped together by the lowering of rank. Since applying LSA also lowers the amount of "noise" in the corpus, the amount of rare and less used terms, should also be filtered out. This of course only works if the average meaning of the term is close to the *real* meaning. Otherwise, since the weight of the term vector is only an average of the various meanings of the term, this simplification could introduce more noise into the model.

Performing searches

To perform queries on a model, the query text must be represented as a vector in the same fashion as any other document in the model. After the query text has been converted to a vector representation, it can be mapped into the LSA space by

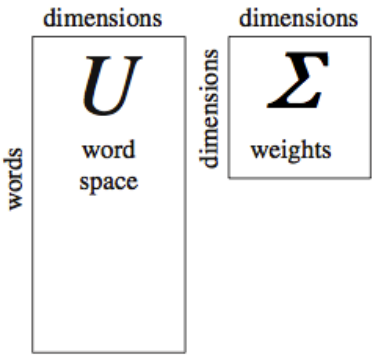
$$\vec{q}_k = \sum_k^{-1} U_k^T \vec{q}.$$

where $\vec{q}$ is the query vector and *k* is the number of dimensions. To produce the similarities between a query and a document or two documents or between two words, we can again use cosine similarities.

Since a query can be represented as just another document vector, the equation above also works for adding new documents to the model. This way we do not have to rebuild the entire model every time a new document needs to be indexed. This procedure is referred to as "folding-in" (Manning et al. 2008:414). Of course with folding-in, we fail to include any co-occurrences of the added document. New terms that are not already present in the model will be ignored. To truly include all documents in the model, we have to periodically rebuild the model from scratch. This is, however, usually not a problem since a delta-index can easily be rendered in the background and be switched in when ready.

Comparing two terms

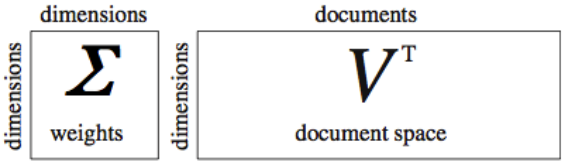
We also know previously that the matrix U (the word space) consists of tokens and the number of dimensions of the SVD, while matrix Σ (the weights) is an m×m matrix where m is also the number of dimensions of the SVD.



To compare two terms, we know from Deerwester that the dot product between the i^{th} column and the j^{th} row of matrix UΣ will give us the similarity between the two words chosen. This is because the UΣ matrix consists of the terms and the weights of those terms in the SVD.

Comparing Two Documents

The approach of comparing two documents is similar to comparing two terms. The difference is that instead of using matrix UΣ, we instead take the dot product between the i^{th} and j^{th} rows of the matrix VΣ and that gives us the similarity between the two documents.



Comparing a Term and a Document

The approach of comparing a term and a document is a little more different than in the method used above. It will simply come down to the weight of a specifc term on a specific document. For this we will need the values from both UΣ and VΣ, but since we are going to combine the result, the values of the 2 matrices will have to be divided by 2.

To complete the operation, the dot-product between the i^{th} and the j^{th} row of matrix $U\Sigma^{1/2}$ and the j^{th} row of matrix $V\Sigma^{1/2}$ (Deerwester, 1990:399)

Gensim

Gensim is a free memory-efficient Python framework for extracting semantic topics from documents. It holds a solid implementation of LSA. Its main advantage is that it can process very large corpora of data by switching the data to disk, thus using a limited footprint in memory.

In the following chapter, we are going to use the corpora.dictionary to keep track of our tokens while models.tfidfmodel and models.lsimodel are used for transforming our corpora. By applying the LSA transformations to our corpora, we should be able to expose new relationships between documents and terms. Finally, we conclude the example by applying the similarity module to the example. This is done to demonstrate how to search the corpora for related documents.

The following is a short review of the gensim tutorial with emphasis on theory.

From documents to vectors

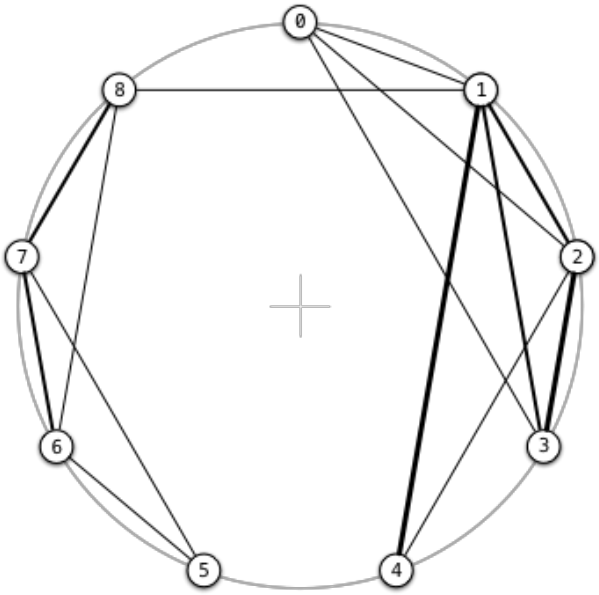
The objective of applying LSA or LDA to a corpus is to reveal any underlying semantic structure of the documents. To demonstrate the approach, the documents from the Deerwester paper will be used (Deerwester et al., 1990:396)

The following are the 10 documents from Deerwester, where the first five are about the topic of human-computer interaction, while the last four are about graphs.

```
>>> documents = ["Human machine interface for lab abc computer applications",
>>>               "A survey of user opinion of computer system response time",
>>>               "The EPS user interface management system",
>>>               "System and human system engineering testing of EPS",
>>>               "Relation of user perceived response time to error measurement",
>>>               "The generation of random binary unordered trees",
>>>               "The intersection graph of paths in trees",
>>>               "Graph minors IV Widths of trees and well quasi ordering",
>>>               "Graph minors A survey"]
```

Since we do not care about the actual syntax of each document, but only about the frequency of words in each document, the first thing to do is to tokenize the text in each document. This way we are left with each of the documents represented as a BOW. In the tutorial on the website, some stop words are removed from each document along with all words only used once. This is done to simplify matters even more, since LSA works by connecting documents with similar words. The downside of removing these words is that we cannot find similar documents by searching for the words we removed. The kept words are underlined in the text above.

As pointed out before, the first five documents are related and so are the last four. The following model depicts the documents and their word relations.



[Fig. 3] *Each document has been plotted on a circle, and lines are drawn between those documents that share words. The thickness of the lines indicates how strong the relation is.*

It is clear that there is a separation between the two document groups. They only share one word between document #1 and #8 — namely "survey".

Since working with large sets of text in memory can be demanding, what gensim does is constructing a dictionary that maps all words to unique integers. This way the word "human" might become "0", "interface" might become "1" and so forth. Apart from the mapping, the dictionary also keeps track of the word frequency, since this can be used for determining the weight of the word against the corpus.
By feeding the BOW representation of the documents into the dictionary, it will map each unique word with an integer value.

```
>>> dictionary = corpora.Dictionary(texts)
```

To illustrate the mapping, we can call the token2id method on the dictionary object to get the terms shown along with their IDs.

```
>>> print dictionary.token2id
{'minors': 11, 'graph': 10, 'system': 5, 'trees': 9, 'eps': 8, 'computer': 0,
'survey': 4, 'user': 7, 'human': 1, 'time': 6, 'interface': 2, 'response': 3}
```

We can now use the dictionary method doc2bow to determine the weight of any words that appear in our dictionary.

```
>>> new_doc = "Human computer interaction with another human"
>>> print dictionary.doc2bow(new_doc.lower().split())
[(0, 1), (1, 2)]
```

Since only the words "computer" (with ID 0) and "human" (with ID 1) are in our dictionary, only they are included. The rest gets discarded. The second dimension of the result is the frequency of the word in our corpora.
It is worth pointing out that the result omits the elements of zero, since these do not bring any information to the table anyway. The above dictionary has twelve distinct words (also called features or terms), so any vector representation coming out of the dictionary object will be a vector of 12 dimensions (12D), but since the rest of the dimensions are (0.0) they are not included.

Finding topics

A transformation is a process where the corpus or similar data (like a query) is converted from one vector space to another. The reason for transforming the models is, as described earlier, to expose a hidden structure along with decreasing the number of dimensions. For LSA, this means lowering the rank to make it possible to find the best approximation for the original data set.

It is crucial to use the same vector space for both training and transformation. This is because each word is matched with an integer in the dictionary, which means that the words will be mismatched if another dictionary is used.

We will start by training with a tf-idf model with the corpus from before:

```
>>> tfidf = models.TfidfModel(corpus)
```

The tf-idf model will calculate the IDF weights for all terms in the corpus on the fly (tfidfmodel , 2012). We can then use the tf-idf object to convert any plain document-term vector into a tf-idf represented model. To demonstrate this, we can take the vector representation we obtained from the doc2bow method from before: `[(0, 1), (1, 2)]` and apply that to the tf-idf model

```
>>> doc_bow = [(0, 1), (1, 2)]   # 0:computer, 1:human
```

```
>>> print tfidf[doc_bow]
[(0, 0.44721360), (1, 0.89442720)]
```

We see that the token with ID 1 is greatest which means that the word "human" says more about the query than the word "computer" compared to the number of times the words were used in the rest of the corpus. Since the weights are normalized, the square root of the sum squared weights must be equal to 1. SQRT(0.4472^2+0.8944^2) ≈ 1 , therefore the weights will change if the number of duplicate words in the query changes.

Now that we have our corpus represented as tf-idf, we can use it for decomposing and reindexing our corpora using LSA. To project the corpus into latent topic space, we simply parse our corpus wrapped inside the tf-idf transformation.

```
>>> lsi = models.LsiModel(tfidf[corpus], id2word=dictionary, num_topics=2)
```

This produces a model of two topics/dimensions — we have picked two topics here, because we know our initial corpus is split into two topics; one about human-computer interaction and one about graphs. By taking advantage of LSA's ability to find topics, it should split the words into roughly two topics.

The following is a table containing the two topics. The words marked in green are words from the human-computer interaction part. While the words marked in red are words from the graphs part of the text. Any words shared by the two parts are marked in yellow.

Topic 0	Topic 1
-0.703 : "trees"	-0.460 : "system"
-0.538 : "graph"	-0.373 : "user"
-0.402 : "minors"	-0.332 : "eps"
-0.187 : "survey"	-0.328 : "interface"
-0.061 : "system"	-0.320 : "time"
-0.060 : "time"	-0.320 : "response"
-0.060 : "response"	-0.293 : "computer"
-0.058 : "user"	-0.280 : "human"
-0.049 : "computer"	-0.171 : "survey"
-0.035 : "interface"	0.161 : "trees"
-0.035 : "eps"	0.076 : "graph"
-0.030 : "human"	0.029 : "minors"

Green: Words from the human-computer interaction part.
Red: Words from the graphs part.
Yellow: Words shared equally by the two topics.

Each column in the table lists the terms ordered by normalized non-zero-weighted tokens. Because of the limited amount of documents and features of our dataset, we are able to see all 12 terms from the two topics. When working with LSA, this is always the case. The ordered list of each topic is as long as the number of tokens. It is up to the operator to decide how many are relevant.
When looking at the two columns, it is easy to see that the first topic is about graphs, while the second is about human-computer interaction. We also notice this on the weights, where there is a sudden drop between the words. This means that we should be able to isolate the group of words that tells us something about the topic. When setting up the number of dimensions in LSA, one should be careful to calibrate it to the size of the corpus. Failing to do so, will likely lead to a bland set of topics.

Now that we have seen that there are two topics in the corpus, we can use the LSA model to determine of which topic a given query text is more likely to be a member. To do this, I use doc2bow to make the query text into a BOW representation and then transpose it using the tfidf model onto the LSA:

```
>>> text = 'A survey of user opinion of computer system response time'
>>> q_bow = dictionary.doc2bow(text.lower().split())
>>> lsi[tfidf[q_bow]]
[(0, 0.197), (1, 0.761)]
```

This indicates that this document is most similar to topic 1 — which we know is true, since it originates from the same corpus. We could however use another constellation of the words in our dictionary to construct a new document. By using this approach, we can figure out what topic(s) any document will be most similar to.

Finding similar documents

In the gensim introduction up until now, we have looked at how to transform the vector space model between different vector spaces. This leads up to using gensim for finding similarities between a query document and a whole set of other documents. The gensim similarity module works by building an index of documents to search by using the principals of comparing two documents.

To initialize the similarity module and build an index from our corpus, we can either choose to use the plain corpus or the tf-idf transformed corpus, as it will only affect the cosine weight of the results (as explained in the tf-idf sub-chapter). For now, I will skip the tf-idf transformation for simplification, since I do not have any excess words.

```
>>> lsi = models.LsiModel(corpus, id2word=dictionary, num_topics=2)
```

```
```

Now that I have an LSA representation of the corpus, I can transform it to LSA space and index it, like so:

```
>>> index = similarities.MatrixSimilarity(lsi[corpus])
```

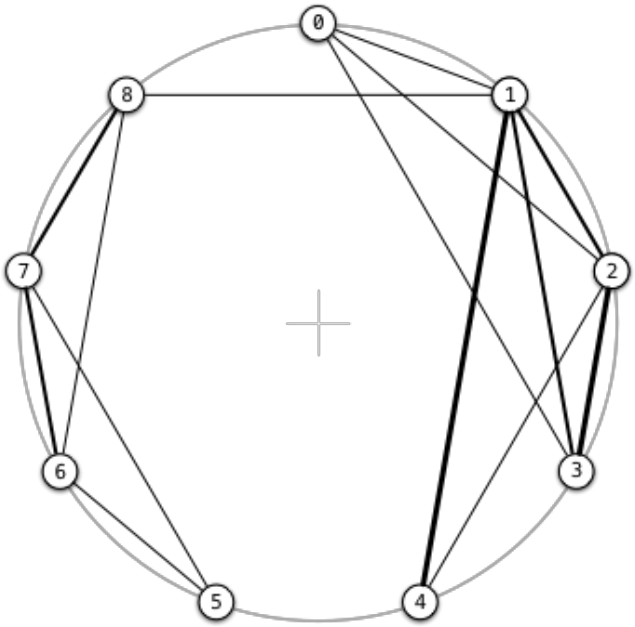
The index can now be used to perform queries on the corpus. It is worth noticing that the index returns the results as cosine similarities and not as the normalized weights that have been used this far. This means the result outputted will be in the range -1 to 1.

Let us use the index to come up with a list of documents that are similar to the following query text: "Human computer interaction". To perform queries against the index, we simply transform the bow->lsi of our query text onto our index which in turn prints each document as a tuple containing the document ID and the cosine similarity.

```
>>> doc = "Human computer interaction"
>>> vec_bow = dictionary.doc2bow(doc.lower().split())
>>> lsi[vec_bow]
[(0, 0.4618), (1, 0.0700)]

>>> index_result = index[ lsi[vec_bow] ]
>>> print sorted(enumerate(index_result), key=lambda item: -item[1])
[(2, 0.99844527), (0, 0.99809301), (3, 0.9865886), (1, 0.93748635), (4, 0.90755945), (8,
0.050041765), (7, -0.098794639), (6, -0.10639259), (5, -0.12416792)]
```

The cosine similarity expresses the relation between the query text and each document in the index. Since the closer the query text is to each of the documents, the higher the cosine value is, the cosine value needs to be as high as possible for it to make a match. This means that according to LSA, documents 2, 0, 3, 1 and 4 look very much like the query text. Likewise, we can see that documents 8, 7, 6 and 5 do not look like our query text. This complies with our relation model from before.



One very important thing to note, which is also mentioned in the tutorial, is that document 2 "The EPS user interface management system" and document 4 "Relation of user perceived response time to error measurement" would never have been returned by a standard Boolean full text search, since none of the words in the query text "Human computer interaction" are present in either of the two texts. They are nonetheless relatable for us humans. When performing this operation, it is important to look at the cosine score next to each document ID. This tells us something about how probable it is that this is in fact a match. The reason for this, as mentioned before, is that even if it is a poor match, all the documents of the corpus will still be present. It is merely a matter of ranking. Document 2 is actually the most similar document to the query text. This is a beautiful example, since it is obvious for us humans that the text "Human computer interaction" is more or less the same as ".. user interface management system".

Comparing Two Terms

One thing the gensim tutorial does not include is the process of how two terms are compared. This is vital to working with LSA since this is one of the core features. According to the author of the gensim project (Radim Řehůřek) it is something he is working on for the next version.
We can however easily set this up on our own, since it is just a matter of applying the correct operations to the LSA model.

The first thing we need to do is to pull out the U and Σ matrices from the LSA model and then multiply them like so:

```
>>> US = lsi.projection.u * lsi.projection.s
```

This gets us an m x m matrix containing the weights of each word in relation with each other. According to Deerwester we should be able to get a matching result by dotting two terms together:

```
>>> dictionary.id2token
{0: 'computer', 1: 'human', 2: 'interface', 3: 'response', 4: 'survey', 5: 'system', 6:
'time', 7: 'user', 8: 'eps', 9: 'trees', 10: 'graph', 11: 'minors'}
```

To compare "computer" with "graph" we simply do this:

```
>>> numpy.dot(US[0, :], US[10, :])
0.019056
```

Since the word "computer" has very little relation to "graph", the score is very low, which is good. If we instead take two words that occur together like "user" and "system", we get:

```
>>> numpy.dot(US[7, :], US[5, :])
0.382907
```

The number is higher, which is also good. The results from this approach are displayed in weights and not in cosine similarity. If we make a comparison between the same word, we will therefore not get the number 1:

```
```

```
>>> numpy.dot(US[0, :], US[0, :])
0.193149
```

To make this happen, we will need to normalize the vectors first. This can be done with the gensim.matutils.unitvec method, like so:

```
>>> numpy.dot(unitvec(US[0, :]), unitvec(US[0, :]))
0.99999999999999978
```

Which is very close to 1.

A more optimal way of doing this would be to use the MatrixSimilarity module on the transposed word space like it was done on the document space previous in the chapter.

To do this, we will first need to create the index:

```
term_corpus = Dense2Corpus(lsi.projection.u.T)
index = MatrixSimilarity(term_corpus)
```

The Dense2Corpus method simply treats a dense numpy array as a sparse gensim corpus, so we can use it instead of gensim.models.LsiModel.

```
result = list(term_corpus)[0]
cos_sim = index[result]
print sorted(enumerate(cos_sim), key=lambda item: -item[1])
```

The first line pulls out the result for token 0, while the second line gets the cosine similarity of the query to each one of the 12 terms. The third line prints out the result sorted by relevancy.

```
[(0, 1.0), (7, 0.99992669), (3, 0.9998644), (6, 0.9998644), (5, 0.99940175), (2, 0.99820149),
(1, 0.99810249), (8, 0.9980489), (4, 0.78768295), (11, 0.09390638), (10, 0.026982352), (9, -
0.058117405)]
```

From the result, we see that words "user", "response", "time", etc. lies closer to the word "computer", than the words "trees", "graph" and "minors" . Which matches the result has had

up until now.

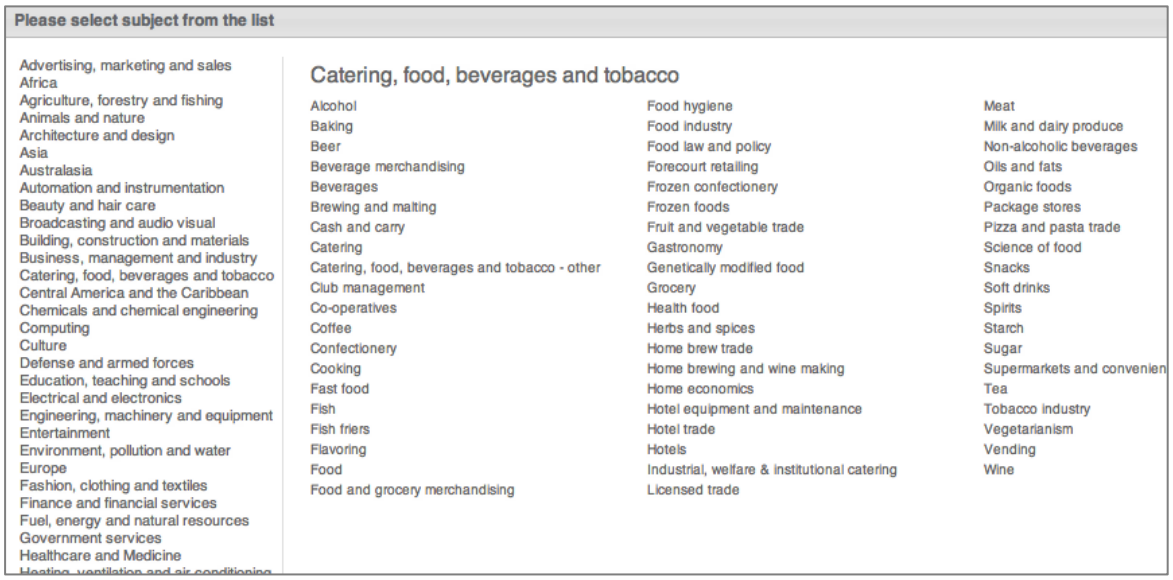
Analysis

This chapter will start by analyzing the problem of the environment we are trying to improve by the use of technology. It is important to understand why there's a problem and why LSA could be the answer. This chapter will then go in to the analysis of the kind of data that is needed for the experiments to work. Lastly the gensim package will be used for analysis on relevant data to see if there can be basis for using LSA in the future.

Existing subjects

Press2go's media database contains about 180,000 media outlets and about 500,000 media employees (reporters, editors, producers, etc.) spread out over most of the world's countries. To better search and sort them, they have all been sorted into 55 main subjects (like Chemicals and chemical engineering, Government services, Safety and security, etc.) and then again, into about 1,400 sub-subjects (like Regional construction, Brewing and malting, Military logistics, etc.). Users of the software are able to search the media database with these parameters to find media outlets that fit their interests.
The subjects are used as a segmentation parameter to single out media outlets that bring stories of similar matter. It is the most used parameter in Press2go's arsenal, and when the researchers contact the media outlets for updated information, the media outlets often know exactly what subjects they bring.

The downside of the existing classification of subjects is its biased nature and its large amount of finely divided subjects. Since all subjects are found and supplied by a staff of human researchers (together with the media outlet), and because it is up to each researcher to make sure that the distribution of subjects is correctly applied, the data often suffers from a variety due to each researcher's perception of the subject definition. On top of this, there is the task of finding the correct subjects in the pool of sub-subjects, which means that an average subject group has ~25 sub-subjects. Even the task of displaying an overview of all these subjects becomes a hassle.



[Fig. 4] *Excerpt of a screenshot from the existing solution.*

On top of this, the researcher also needs to collect other information from the media outlet, often while being in the middle of an active telephone call with the media outlet. Keeping the call to a minimum can become a challenging task that sometimes results in the collected information being of a varying standard. Because the subjects are considered to be very important in the process of finding relevant media outlets, a good quality of these subjects is of great importance.

One way of improving the classification process could be to automatically base the relevant subjects on already published content, instead of supplying subjects that very well could prove to be misunderstood or wishful interpretations of the outlets readers' interests.
To be able to base the subjects on another structure than the one presently in use, we need to analyze the content that is published by the media outlets. This could mean that to keep up the pace of the media industry, we would need to continuously monitor the content of each media outlet and assign appropriate subjects to each of them.
The discovery of subjects (or topics) is where LSA could prove to be a successful match. With its bottom-up parsing approach, a number of relevant subjects can come to light. It would, however, probably mean that the existing structure would have to be abandoned, since it is doubtful that the existing ontology approach will actually hold up in the real world. A shift in origin could also prove to dramatically cut back on the large amount of subjects that are present in the existing system.
To perform this sort of classification, we will need a text corpus that is relevant to the press releases that are being sent out. This means that we not only need a large corpus, but also a relevant one.

Datasets

The following sub-chapters will list and analyze the various possibilities to get relevant

Danish content for use in our experiment. To get the LSA to work, the content should not be too broad, since this will most likely create topics containing mixed set of words, making it difficult to interpret. It is also important that the content is no more than 10 years old, since the outcome of the LSA should point is in the correct direction of today's media landscape. The structure of the content has no importance, since this is what LSA will discover for us. It would, however, be nice to have as a measurement of correctness.

Wikipedia

Wikipedia is an online user-collaborated encyclopedia that, because of its free use, has become an Internet standard for reference of information. Since the driving force behind Wikipedia is its open and free content, it is also possible to download it in its entirety.
At the time of writing this thesis, the Danish version of Wikipedia contains about 160,000 [2] articles about a large set of topics. When looking in the gensim documentation, this is also their choice of data source for showing the use of LSA on a large corpus.
While the amount of articles should be enough to get a large amount of usable topics back, the broad set of topics in an encyclopedia may prove to become a problem. This is because an encyclopedia is meant to be a place of strictly summarized and narrow segmented elaborations on specific topics. An article can therefore contain a great deal of information about the subject domain, without going into the necessary detail to satisfy the NLP models. The collection of articles may still prove to be a good data source. It is certainly a comprehensive collection of data, while it is also constantly being expanded and updated.

Website: http://da.wikipedia.org

Infomedia

Infomedia is a private Danish company that specializes in media monitoring and cutouts of media articles. Even though other players[3] are trying to compete, when it comes to archives of Danish media articles, they have monopoly status.

Infomedia is a joint venture between the media companies JP/Politikens Hus and Berlingske Media. The two companies both have strong ties to the Danish media industry and are considered to be two of the largest media companies in Denmark.
Infomedia's flagship product is a well-known search portal that gives access to an archive of articles from almost all Danish media outlets. The search interface is very good for manual searches and is highly valued in many research-heavy tasks.

---

[2] According to http://da.wikipedia.org/wiki/Speciel:Statistik (2012-04-28)
[3] A newly started Danish company called NewsWatch (newswatch.dk), is trying to gain market share in the same field.

For the purpose of gathering information from thousands of articles for this project, the search interface is not optimal. To get a better footstep, I decided to contact Infomedia to arrange a meeting where I could explain my intentions to them. They were friendly and very interested in my hypothesis. What I wanted was access to articles from 12 Danish media outlets. For the meeting I created a map of relations between the media outlets that I wanted. It was based on the existing subjects, so that it would reflect media outlets that had as little to do with each other as possible.
I had chosen a narrow segment of trade journals, as I assumed they were more divided into specific topics than daily newspapers. The map is included in Appendix B and shows the subject relations between the most active Danish trade journals.
The map is made so that the media outlets that share the same subjects are clustered together. I had picked eight media outlets from the map, ensuring that they would have the least to do with each other. On top of this, I had chosen another four that were located in the middle of the map. It was my intention for them to be more relatable to all other media outlets, located on the edge of the map. The result was supposed to show how the LSA model would be able to sort out the topics of the 12 media outlets in a way that they would not overlap.

Unfortunately, the partnership fell apart when the confidentiality agreement was about to be signed. It turned out that Infomedia wanted exclusive rights to the results. Something I could not see the point of giving them, since I wanted to be able to share this thesis with others, if possible.

Website: http://www.infomedia.dk

KorpusDK

KorpusDK is a comprehensive collection of texts from various genres. The origin website explains that the texts are processed in such a way that it is possible to make linguistic studies of the material. The collection consists of 56 million words and are picked and processed from various sources to give a broad impression of modern Danish language used in the years of 1990 and 2000.

When doing searches on the web page, the results are primarily excerpts from various Danish books. While it could prove to be a great data source, the texts are cut up into roughly 1,000 character snippets. It could be that the Society for Danish Language and Literature would release the full texts, but when I called them, they did not seem too enthusiastic about the idea. I think it has a lot to do with copyrights.

The collection would most likely have fitted the classification of books, screenplays and subtitles better, since they all share the same origin of more or less narrative realism. Another problem with this collection is that it is 12 and 22 years old which is a lot considering that the media landscape moves very rapidly. The advantage of using this, though, is that the authors have gone to great lengths to pick the texts in such a way that it is as broad as

possible.

Website: http://ordnet.dk/korpusdk/

Web scraping

Web scraping is the process of automatically collecting useable information embedded in websites. This is done by traversing the site's pages with a crawler, looking for specific patterns and then one by one extracting relevant information for further analysis. It can be a cumbersome task since almost all websites have their own specific DOM[4] structure. This means that each websites' structure will have to be analyzed separately, so that the correct data will be extracted. This uniqueness also means that the structure of the website can change at any moment, meaning that the crawler will be unable to pick up new information or, even worse, will pick up faulty information.
Most news websites use a streamlined process in publishing the articles on their websites. This often means there is an underlying structure to the data on the website. Analyzing and understanding this structure is mostly easy to do, and once found, it can be reversed to our advantage. This way we would be able to seek out a handful of sites containing relevant articles and extract their data.

The limitation of this approach is first and foremost the workload in keeping the crawler up to date, but for our experiment, it would be feasible, since this would not have to work for very long. Then there is the copyright infringement problem in harvesting data from third-party sites. In Denmark, there have been cases where companies have sued for compensation for their scraped data. The most recent of these cases is Boligsiden vs. Boliga.dk, where Boligsiden (which is the real estate companies' joint portal) admitted that they were obfuscating the data on their site to prevent Boliga.dk from scraping it. The case has not been heard yet, but the Danish Competition Authority has stated in a press release of January 25th 2012 that it intends to report the matter to the public prosecutor for serious economic crime[5]. This will probably have an outcome of scraping not being directly illegal, but perhaps making it a felony to publish the scraped data in a way that infringes on the content creator's intellectual property.
This project has no intention of publishing anything of that nature, so it is my opinion that this would be within the realm of ethical use for this project as long as the scraped data is not published.

Taking into account that this is a fairly expensive and not a particular smart way of getting data, for a small dataset this is, however, doable since many of the Danish media outlets are

[4] Document Object Model
[5] According to the website: http://www.kfst.dk/index.php?id=30783

publishing their articles online. However, not all of them are doing this in open formats like HTML and PDF. Other variants like Flash are also very common, making it difficult to automatically separate the articles from each other.

Getting the right data

Now that we have listed the various data sources, the time has come to pick one. The choice of Wikipedia is an easy one. It is a well-known large corpus of very relevant data. It is easy to obtain and use, which makes it a good choice for our topics discovery tests. The downside of this is that it most likely is too diverse to be used in an actual production environment. On top of this, the articles are updated all the time, which means we cannot fold them in, but will have to compile the entire set of articles on a regular basis. We can, of course, fold new articles in, but they will most likely not bring anything newsworthy to the corpus, since they are just starting up. The Wikipedia corpus is not ideal, but it does hold some interesting aspects that are worth analyzing further.
The next data source is Infomedia. While probably being the best data source for a production environment, since it is constantly updated with new media outlets and articles, the fact is that they do not want to cooperate on this project. Perhaps if they would have an economic benefit, they would once again become interested, but for now, we have to find our data elsewhere.
The data from KorpusDK also looks very interesting, but like Infomedia they are perhaps too big players for this project. Their data would perhaps not fit our needs anyway, since the data they offer is very old and of other literary origin than news.

This leaves us with having to use Wikipedia and in part having to scrape the data we would otherwise have been asking Infomedia for. Since far from all media outlets have their content shared on the net, this will dilute our dataset. We will in turn have to scope out the interesting sites and see if we cannot get a fragmented picture of the topics to demonstrate the use of LSA.

The process of analyzing the Wikipedia corpus and the process of scraping data for use in a similar analysis will be covered in the next two sub-chapters.

Topics from Wikipedia

By using the Danish version of Wikipedia as a data source, I will now run through the steps I used to extract the initial LSA topics using gensim.

The first thing I did was to download the files used in the gensim tutorial. Specifically the corpora.wikicorpus which is the script used on the website tutorial to work the English version of Wikipedia. Since I was to do a similar job, I decided to use this script as a starting point. The script is pretty straightforward an extension on the corpora.textcorpus.TextCorpus class with some logic for traversing through the huge Wikipedia XML file. It works by running through the XML line by line and looking for the <text> and </text> elements that

encapsulate each of the pages. Once it finishes a <text> element, it cleans the article for Wikipedia markup using a set of regular expressions. Once cleaned, it is tested to see if the remaining length of the article is at least 500 characters. Otherwise, it will get discarded. Before the final articles are passed on the dictionary, the utils.tokenize splits up each word and places them on a list. That list again gets stripped of any word that has a length below three and above 15 and does not start with an underscore. This is done to slim down the list by removing any unusual small or large words. The underscore words must be something specific to Wikipedia.
Once all the articles have been fed into the dictionary, where they are being mapped from words to IDs, the textual extremes are weeded out. This means (in my case) that tokens that appear in less than 50 documents or in more than 50% (60,675 docs) of the corpus are filtered out. On top of this, to minimize the number of unique tokens, only the first 100,000 most frequent tokens are kept.
The result we get from running in the articles is a corpus containing 121,350 documents (320,415 documents before weeding out the ones that did not make the cut) and 33,534 unique tokens, leaving me with a 121,350 x 33,534 Document-Term matrix. The density is 0.351%. Now, we have a dictionary containing all 33,534 tokens with a unique identifier and Term-Document matrix containing the frequency of each token paired with each document. The script can be found in Appendix C.

Now that we have comprised a dataset, the next step towards finding topics is to apply the LSA to this data.

```
>>> dictionary = Dictionary.load_from_text('data/da_wiki_wordids.txt')
>>> tfidf_mm = MmCorpus('data/da_wiki_tfidf.mm')
...
accepted corpus with 121,350 documents, 33,534 features, 14,264,847 non-zero entries
lsi = LsiModel(corpus=tfidf_mm, id2word=dictionary, num_topics=40)
```

This generates 40 topics from the corpus, but because the result is rather large, only the first eight topics are listed here. A full overview can be found in Appendix D.

#0		#1		#2		#3	
0.3841	han	-0.7662	bebyggelse	0.4041	jeg	-0.3919	han
0.1288	hun	-0.3845	sogn	-0.3744	han	-0.3034	jeg
0.1231	hans	-0.3396	ejerlav	0.3397	cest	-0.2874	cest
0.1149	ikke	-0.1308	kommune	0.2489	cet	-0.1950	cet
0.1098	the	-0.1239	areal	0.1715	kan	0.1576	ligger
0.1071	ved	-0.1066	amt	0.1361	ikke	0.1514	byen
0.0997	jeg	-0.0966	herred	0.1220	man	-0.0980	bebyggelse
0.0995	kan	-0.0872	ligger	0.1058	eller	0.0967	eller
0.0961	sig	-0.0847	sognet	0.1055	skal	0.0948	kommune
0.0952	sin	-0.0845	stednavne	0.0989	artiklen	0.0864	jpg
0.0947	men	-0.0833	sogneportalen	0.0960	vil	0.0827	månen
0.0935	efter	-0.0821	autoriserede	0.0939	wikipedia	0.0807	indbyggere
...		...		...		...	

#4		#5		#6		#7	
-0.7474	hun	0.5237	the	-0.2147	landshold	-0.2615	the
-0.2960	the	-0.4629	hun	-0.2055	klubben	-0.1969	månen
0.2560	han	0.1315	and	-0.1901	kampe	-0.1748	kratere
-0.1617	hendes	0.1077	album	-0.1600	hold	-0.1618	cest
-0.0919	film	-0.0976	kommune	-0.1489	spillede	-0.1545	ligger
-0.0770	hende	0.0936	bebyggelse	-0.1391	cest	-0.1422	hovedkrateret
-0.0726	and	0.0918	bandet	-0.1379	fodboldspiller	0.1375	kan
-0.0670	skuespiller	0.0891	you	-0.1369	spillerinfo	-0.1335	kommune
-0.0669	album	0.0874	albummet	-0.1242	cup	-0.1226	månens
-0.0608	filmen	-0.0846	hendes	-0.1210	mål	0.1215	bebyggelse
0.0584	ligger	0.0825	guitar	-0.1203	kamp	0.1210	eller
-0.0569	født	-0.0801	kirke	-0.1182	spiller	-0.1168	jeg
...		...		...		...	

Although this report is written in English, the dataset is predominantly in Danish. This means that the result of the analysis is also in Danish. This poses a problem for people who do not speak Danish and for that I am sorry, but I feel that if I were to translate all the tokens, it would affect the focus of this report in a bad way. That is why there will be no translation of the Danish texts, unless it makes sense for the sake of the analysis.

In the result set above, the first 10 topics are shown. The first topic (#0) is mostly a set of adjectives that really does not bring any value as a topic. The next topic (#1) has something to do with land properties with special weight on parish and church relations. The following topics contain a lot of the vapid words which LSA unfortunately scores high in the ranking. To counteract this, I have chosen to strip the documents of these stop words. For that process, I have comprised a list of stop words from snowball.tartarus.org[6] along with words I have picked up myself. Since the Danish Wikipedia seems to hold a lot of documents in non-Danish languages like English, Swedish and Norwegian, I have also included stop words from these languages. A full list of stop words can be found in Appendix E. After applying the stop words, the first eight topics look a whole lot clearer.

[6] The files were pulled from the following website: http://snowball.tartarus.org/algorithms/ (2012-05-20)

#0		#1		#2		#3	
0.0838	henvisninger	-0.2664	sogneportalen	-0.1748	fodboldspiller	0.1292	landshold
0.0834	eksterne	-0.2641	stednavne	-0.1647	født	0.1260	kampe
0.0784	dansk	-0.2632	ejerlav	-0.1606	landshold	0.1248	fodboldspiller
0.0776	født	-0.2619	autoriserede	-0.1568	kampe	0.1114	ligger
0.0763	kilder	-0.2569	flg	-0.1335	spillede	0.1066	spillerinfo
0.0709	del	-0.2498	sognet	-0.1299	klubben	0.1018	klubben
0.0707	ligger	-0.2497	bebyggelse	-0.1260	spillerinfo	-0.0981	dansk
0.0699	tidligere	-0.2382	provsti	-0.1257	spiller	-0.0979	søn
0.0683	senere	-0.2365	herred	-0.1144	vandt	0.0960	nord
0.0665	danske	-0.2339	sogn	-0.1056	spillet	-0.0943	gift
0.0650	danmark	-0.2313	stift	-0.0922	debuterede	0.0928	syd
0.0639	sammen	-0.2216	amt	-0.0897	klub	0.0917	spiller
...		...		...		...	

#4		#5		#6		#7	
-0.1156	ligger	0.1477	månekratere	-0.1190	album	0.1411	bladene
-0.1116	nord	0.1472	iau	-0.1032	amerikansk	0.1338	blomsterne
-0.1093	delstat	0.1466	nedslagskrater	-0.1031	delstat	0.1228	frugterne
-0.1069	indbyggere	0.1461	kratere	-0.0953	albummet	0.1158	synlige
-0.1038	syd	0.1403	månens	-0.0938	landkreis	0.1133	arter
-0.1037	landkreis	0.1399	navigation	-0.0914	band	0.1070	hjemsted
-0.1008	byen	0.1392	astronomiske	-0.0898	diskografi	0.1067	blomster
-0.1002	kommunen	0.1337	måneatlas	0.0882	fodboldspiller	-0.1038	cest
-0.0958	vest	0.1295	månen	-0.0859	udgivet	0.1006	træk
-0.0956	øst	0.1270	karakteristika	0.0849	landshold	-0.1005	artiklen
-0.0945	kommune	0.1208	omgivelser	-0.0846	bandet	0.1004	udbredt
-0.0919	beliggende	0.1191	hovedkrateret	-0.0834	indbyggere	0.0938	almindelig
...		...		...		...	

All the topics can be found in Appendix F. We see that topic #1 has not changed; it is still about local geographic parishes?. Topics #2 and #3 are mostly about football (soccer). Topic #4 (like topic #1) also has something to do with geographic locations; the words "landkreis" (district in German), "delstat" (province), "bayern" and "tyske" (German) point to this topics being mainly German geography. Topic #5 is about space. Topic #6 looks like it is about bands, American provinces and football, so in essence it is an indecisive topic like #0. Topic #7 is about plants (flowers, trees, etc.).

Even though the removal of the stop words had an overall positive effect on the topics, it is still hard to make out what the topics are about in a definitive way. There seems to be a lot of noise in the topics. This could be because the chosen dataset is not big enough (there is only 33,534 unique words in this one). It can also be because it has not been calibrated properly. There are a lot of parameters that could prove to generate a better result. One of the key properties is the amount of topics that are generated. In the above example, I have used 40 topics, but it could also prove that 400 topics would be better. The general relation between the corpus and the amount of usable topics is the size of the corpus. This means that if we were to make 400 topics on this dataset, the resulting topics would definitely be harder to interpret. There would also be too many overlapping topics. For the initial testing of the dataset, I have tried with 20, 60, 80 and 120 topics, but it does not seem to improve the result. I have therefore decided that 40 topics will be sufficient.

Comparing a press release

Now that we have at least discovered some topics with a clear meaning, we can use them and see if we cannot find connections between press releases and the LSA topics.

To do this, we need to first find a suitable document to use as analysis target. Since Press2go has a database full of those, there are many documents to choose from. When looking over the topics, I see repeated mentions of topics like geography (#1, #4, #6, #10, #15, #16, #29), sports/football (#2, #3, #6, #9, #20, #21, #33, #35), space (#5), flowers (#7, #8), politics (#12, #20), music (#6, #14), motor (#17), agriculture (#18) and film/theater (#22, #14). Some of these topics contain combinations of these labels. This is why some of them are repeated. In addition to these broad topics, the last two topics in the list are about Donald Duck (#38) and the Danish TV series "Huset på Christianshavn" (#39).
To find a good match for testing the LSA's ability to home in on the correct topics, I have chosen a press release, about the release of rock, pop and country stats that have performed in the venue "Tinghallen" in Viborg, DK. The press release describes how the book contains stories about the various musicians, athletes and politicians that have been guests at the venue. This press release is typical, while also containing broadly the same topics as in the above list. The press release is included in Appendix G.

The following code loads the dictionary, the tf-idf representation of the Term-Document matrix and the LSA representation of our corpus.

```
>>> dictionary = gensim.corpora.Dictionary.load_from_text('data/wiki_da_wordids.txt')
>>> tfidf_mm = gensim.corpora.MmCorpus('data/wiki_da_tfidf.mm')
>>> lsi = gensim.models.lsimodel.LsiModel.load('data/wiki_da_40t.lsi')
```

The next couple of lines loads the raw text file and converts the text to lower space, then splitting each word into tokens before converting as many of the tokens as possible to BOW representation. We do not need to remove any stop words, since they will be removed anyway when they cannot be found in the dictionary. The last operation is the most interesting one. Here, the BOW representation of the press release will be transformed onto the LSA model, resulting in a list of topics sorted by most significant.

```
>>> text = open('data/pm1.txt').read()
>>> q_bow = dictionary.doc2bow(text.lower().split())
>>> q_lsi = lsi[q_bow]
>>> print sorted(q_lsi, reverse=True, key=lambda item: math.fabs(item[1]))
```

The result is as follows:

```
1: ( 0,  3.4188),
2: ( 2,  0.6430), (10, -0.6395), (16,  0.5484), (17,  0.5327), (18, -0.5070),
3: ( 9,  0.4388), (25,  0.3994), ( 5, -0.3598), (39, -0.3506), (19, -0.3490), (35, -0.3201),
   (37,  0.3015), (27, -0.2994), (20, -0.2953),
```

```
4: ( 7, -0.2400), (30, -0.2146), (33, -0.2065), (12, -0.1997), (38,  0.1957), (26,  0.1762),
   (24,  0.1678), (36, -0.1519), ( 3, -0.1500), (21,  0.1391), (28, -0.1307), ( 1,  0.1220),
   (31,  0.1121), (34, -0.1059), (15, -0.0996), ( 6, -0.0906), (13,  0.0889), (11, -0.0870),
   ( 4,  0.0846), (14,  0.0807), ( 8, -0.0726), (22,  0.0562), (29, -0.0293), (23,  0.0147),
   (32, -0.0029)
```

The scores in the right column of each element show the dot product of the vectors and not the cosine similarity. This is why some of the results will exceed 1. Since the greater the values are, the more the related topic seems to look like the query document (press release) (mangler der ikke noget i sætningen?). When looking at the values, it is clear that some of the topics are closer to each other. I have tried to depict this by placing numbers in the left margin above. This is not a part of the Python print output.

The thing that stands out here is that topic #0 is the most dominant topic in the result. When looking at the words that make up that topic, it is very hard to find any consistency. This is most likely due to topic #0 being a topic of no relation and all relation. To be frank, I do not know exactly why this happens, but after performing transformations on other texts (I did a test of 10 press releases), I can see that this goes for all of them. I then thought that this could be an indication of there not being enough topics represented. So I tried with 80, 120 and 400 topics, but the result is the same. Topic #0 remains the dominant topic in all searches above.

Going forward to the other topics, LSA seems to categorize the press release as something mostly (by looking at the numbers in the column) to do with football (topic #2), German geography (#10, #16), motor (#17) and agriculture (#18). The part about the football is true. The press release does include something about sports, but German geography is not in there. Neither is something to do with cars etc. for the motor part. The motor and agriculture are probably more correct, since the venue has presumably housed some conferences about the two topics. The press release, however, does not say anything about this. The next group of topics suggests “construction” (by topic #25), “space” (#5) and “Huset på Christianshavn” (#39). I will stop here, since none of those topics are suggested by the press release. It seems this one did not work the way I had hoped it would. It was, however, not a complete waste of effort, since the topics largely contain words that often have relations with each other. On top of this, the press release used could also prove not to be the best one for the test. We could try to conduct more experiments, but unless we were to adapt some kind of corpus and topics with guidelines, I fear it would take up too much time when I have to manually go through each of the experiments and conduct some kind of subjective status to each of them.

Since the problem could originate from an inconclusive and noisy dataset (Wikipedia), I would like to perform the same test on the data that was scraped. To do this, the process of picking out the right media outlets to scrape will have to be explained first.

Scraped data

Unlike the Wikipedia data, this chapter is a little fleshier, since it also describes the process of getting articles from the right media outlets. The approach of using gensim is the same as in the previous chapter.

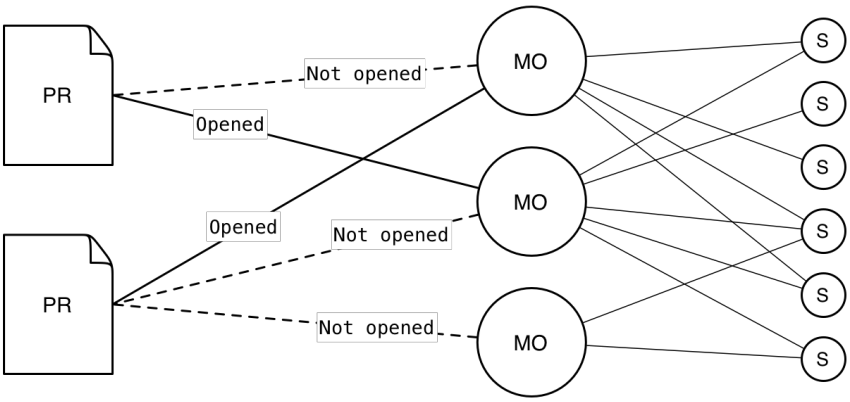
Press2go

The matter of the subjects has been discussed earlier in this report, but I think it is important before starting to analyze what media outlets could prove to add value to a dataset of scraped data. The following is therefore a brief introduction to the parameters Press2go has collected and that are available.

The archives of Press2go contain a lot of existing information about how the system has been used over the last seven years. Since 2009, the system has sent out almost 2 million press releases to media outlets all over the world. The archives contain all the details of those messages. Details like text, recipients and open rates[7]. Combining these factors could prove to come in handy to get a picture of how the software is used.

Apart from textual information like title and story, each press release holds information about whom the press release has been sent to and which recipients have read it. This is done by tracking a clickable link inside the email that takes the recipient to the full story. The recipient receives only an email containing the title, the lead, some meta information and the link to the full story. If the recipient chooses to click on the news story, the event is recorded and used to generate statistics. This means that, over the years, Press2go has accumulated a lot of potential information about what the media outlets are interested in based on what they have opened.
This brings us back to the media outlets and their subjects. Each media outlet in our database is updated by a number of researchers that basically calls each media outlet to make sure the information in the database is correct. In this process, a number of subjects are attached to the outlet, indicating what kind of topics it writes about.

[7] The number of times a press release has been opened by unique recipients.



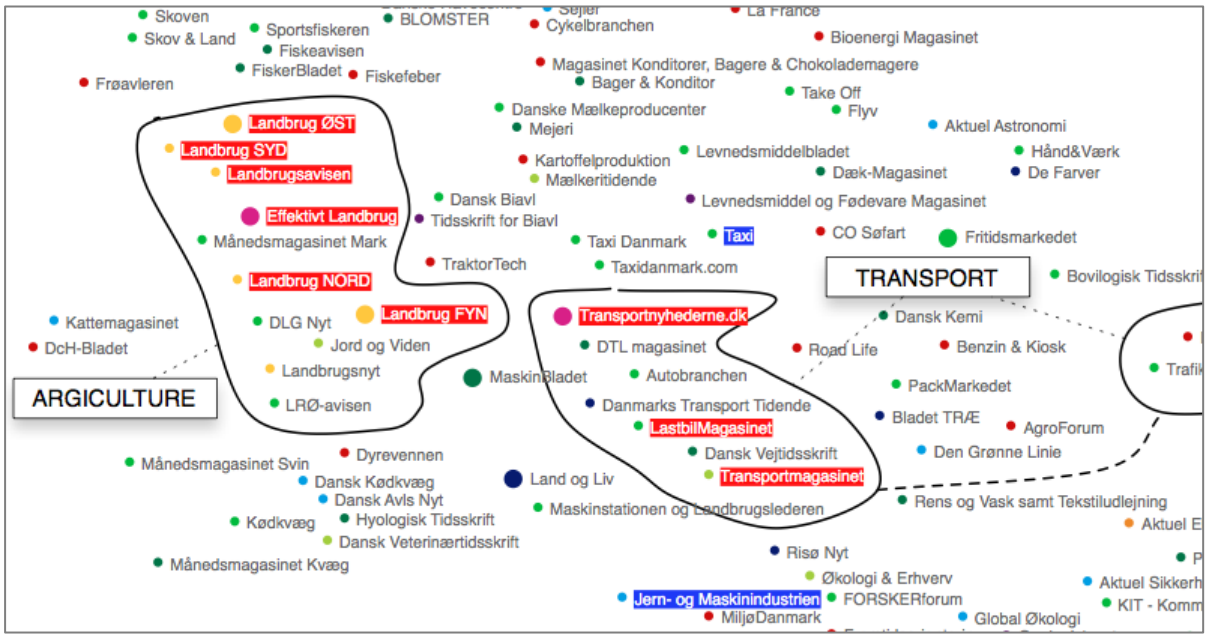
[Fig. 5] *Depiction of the existing resources at Press2go. PR is the press releases, MO is the media outlets and S the subjects. The lines between the press releases and the media are connections, where the solid lines mean that the media outlet has opened the press release, while dotted lines indicate unopened press releases.*

By using the texts from our sent press releases and their relation to the outlet statistics, it could perhaps prove to be possible to close in on what subjects are in tune with certain topics. There is one problem with this approach though; the opening of a story does not necessary mean that the outlet is interested in the story. Since the email only holds information about the sender and a short teaser of what the story is about, they basically have very little knowledge of the relevance of the story. This entails a risk of misinterpretation, which leaves the open rates to only say something about which outlets have received the message (and clicked it), and which maybe have not. This means that we cannot exactly use this as basis for what is relevant for the outlets, since it could consist of an uncertain amount of false positives. On top of this, the click rates are surprisingly low in many cases. This is thought to be because of the overload of non-targeted messages sent to the media outlets.
A better solution would be to somehow combine the monitoring product from Infomedia directly with Press2go's archives. This way, the matching of subjects would be redundant, since the subjects are there simply to guide the users towards relevant media outlets. The present subjects would merely become an indicator of the content of the press release and not a definitive categorization technique.

Searching the outlets

The most optimal solution in discovering LSA topics using actual news articles would be to have access to a dataset containing all media outlets articles. This is, however, not possible, so the next best thing would be to go out and scrape that information on my own. This would, however, take a long time, so instead a targeted search with focus on a cross section of relevant outlets will be as close as I can get at this time.
To accomplish this, I have chosen trade journal outlets, since they bring stories about a very narrow field of observation. Other outlet types like daily newspapers tend to bring the same content, thus making it hard to distinguish one outlet from the other. There could be a question of focus and political views, but from what we have seen in the Wikipedia example,

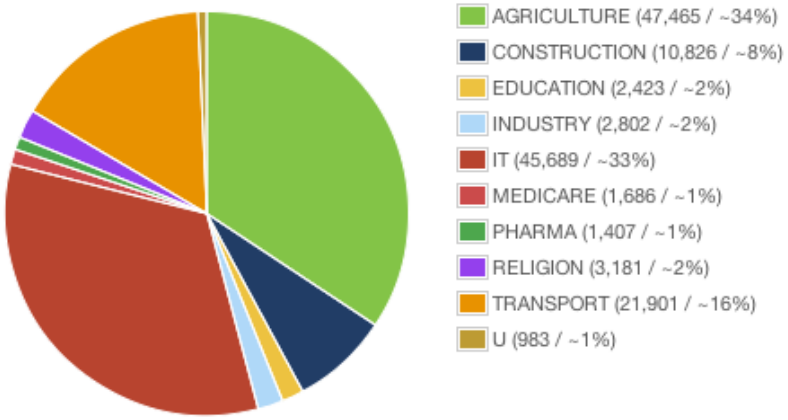
I think it would be better to broaden the gap between the outlets or at least between the trade groups.
To be able to pick out relevant media outlets, we need to first figure out what subjects the media outlets are focused on. Fortunately, we have a guide in already existing Press2go subjects. We also know what each media outlet's issue frequency is (not that we need it for online use) and the amount of news stories they have clicked on. To make it simpler, I have chosen to focus on the 55 main subjects and simply group all underlying subjects into these. Using the open source Excel package NodeXL, I have rendered a map of all the trade journals clustered together with their equals. Each of the journals is plotted as a dot with the size of the amount of press releases they have clicked on. Many of the magazines have never clicked on any of the press releases and remain a small dot. The following is a small cut-out of the map.



The entire map can be found in Appendix H.
The media outlets with a red background are instances where the amount of scraped articles are enough for a group to have an impact on the corpus, while making the corpus big enough for it to match the Wikipedia dataset. Although some of the outlets deviate from the rest, most of them are nicely grouped together. Enough so that I can outline a border around those of the outlets that both have high click rates (larger dots) and have articles for scraping on their website. It was also important to pick outlet groups that were not too close to each other. This was to prevent overlap, but because not many of today's media outlets publicize their articles online (in an open format), this proved harder to do. This is why some of the outlets are marked with a blue background. This is to signify that while the data is relevant and available, the amount of collectable data is insufficient to make the group strong enough to match the other groups. They do, however, still bring value as textual material in a big

corpus of news articles.

The following is a graph of articles collected from the various sites. They have been grouped together to provide an overview. Since it is unlikely that it is possible to gather enough articles from a single outlet for topic comparison, grouping related outlets together could become a better solution to the quantity problem.



The following are all the scraped outlets and the amount of articles they provide. Since scraping is an unofficial way of getting access to a media outlet's articles, most of the time it is not easy to predict how many articles a scraping will provide before building the scraper. The category marked with U is an uncategorized group of various media outlets.

AGRICULTURE	
Effektivt Landbrug	2,713
Landbrugnet	9,769
Landbrugsavisen	34,983
CONSTRUCTION	
Byggeplads.dk	859
ByggeTeknik	4,678
Licitationen	4,494
Mester Tidende	795
EDUCATION	
Folkeskolen	1,879
Friskolebladet	425
Frie Skoler	119
INDUSTRY	
Jern- og Maskinindustri	2,802
IT	
Alt om Data	976
ComON	12,906
ComputerWorld	18,939
Version2	12,868
MEDICARE	
Dangens Medicin	1,085
Fysio	601
PHARMA	
Farmaci	636
Farmakonom	428
Pharma	343
RELIGION	
Indre Missions Tidende	3,181
TRANSPORT	
LastbilMagasinet	5,424
Motormagasinet	2,023
Transportmagasinet	990
Transportnyhederne	13,464
U	
Teknisk Nyt	173
Taxi	136
Nyhedsinformation	674

It should be plain to see that the four groups marked with red media outlets are the groups with the most news articles in them. They are Agriculture, Transport, Construction and IT. When looking at the map, they are, however, not placed in each corner of the map as would be optimal, since that would indicate that the four groups had very little to do with each

other. It is, however, the best I could do with the tools at hand. The number of articles do sum up to 138,363 articles scraped, which is about the same size as the Wikipedia data. This data should however be more accurate since it is more relevant to the media industry.

Analyzing topics

As mentioned in the introduction of this segment, now that we have some interesting and very relevant data, I will use the same approach to analyze the text as in the previous segment.

The creation of the dictionary, the Term-Document matrix and LSA representation is all done in the same way, but with the exception of pulling the data from a database instead of an XML file. For the import of the articles, I will be using the same stop words as can be found in Appendix E. The script used can be found in Appendix I.

The result is a Term-Document matrix with 138,363 documents and 16,939 unique words. It has a density of 0.621%. Putting the corpus though the LSA, we get this:

#0		#1		#2		#3	
0.0655	nye	-0.1310	microsoft	-0.1423	procent	0.2111	cvr
0.0645	danske	-0.1280	windows	-0.1021	selskabet	0.1947	boets
0.0644	kommer	0.1167	landbrug	-0.1006	stigning	0.1942	skyldneren
0.0640	godt	-0.1146	version	-0.0992	omsætning	0.1940	fordring
0.0636	får	-0.1062	brugere	-0.0967	kvartal	0.1939	kreditorudvalg
0.0626	procent	-0.1008	google	-0.0949	steg	0.1939	kurator
0.0622	ifølge	-0.0985	microsofts	-0.0900	året	0.1939	konkursdagen
0.0621	helt	-0.0972	software	-0.0895	omsætningen	0.1899	skifteretten
0.0618	danmark	0.0908	landmænd	-0.0894	steget	0.1887	fremsættes
0.0618	sidste	-0.0883	computerworld	-0.0868	marked	0.1849	anmelde
0.0615	hele	0.0882	landbruget	-0.0848	kroner	0.1803	skriftligt
0.0606	dag	-0.0860	applikationer	-0.0831	millioner	0.1784	bekendtgørelse
...		...		...		...	
#4		**#5**		**#6**		**#7**	
-0.0995	hektar	0.0906	skriver	0.0951	landbrug	0.2154	svovlfri
0.0977	direktør	0.0899	sagen	0.0870	landmænd	0.2127	blyfri
0.0950	virksomheder	-0.0819	samarbejde	0.0801	krav	0.2115	fyringsolie
-0.0796	liter	-0.0710	udstyret	0.0784	landbruget	0.2091	dieselprodukter
0.0768	regeringen	-0.0702	nye	-0.0735	gud	0.2065	listepriserne
-0.0725	høst	0.0693	ifølge	0.0733	regeringen	0.1971	energiselskabet
0.0712	vækst	-0.0693	transport	-0.0724	direktør	0.1849	benzin
-0.0708	lidt	0.0683	mener	0.0723	fødevarer	0.1834	diesel
-0.0700	marken	-0.0646	mola	-0.0713	medarbejdere	0.1710	moms
0.0673	offentlige	-0.0646	motor	-0.0691	selskabet	0.1595	liter
0.0669	administrerende	0.0644	landmænd	0.0683	liter	0.1547	basis
0.0649	formand	0.0641	sag	0.0672	hektar	0.1520	standersalg
...		...		...		...	

When looking at the topics, #0 again stands out to be an all-included topic of no real content. The same goes for #5, while #1 seems to be about IT, #2 is about corporate finance, #3 is about property law, #4 and #6 are about agriculture, and #7 is about fuel sources. The whole list can be found in Appendix J and seems to be distributed better. This could be because the content is written in smaller articles with a more narrow perspective. A news article is often about something very specific, while an encyclopedia article tends to cover many topics in a very broad way. It could also be because the dataset is picked up from about 10

narrow groups of different professions. When looking at the amount of articles from each group, about 33% and 34% of all the articles originate from media outlets that write about IT and Agriculture, respectively. The predominance of the two topics could result in the topics being skewed. When looking over the topics, about 52% of the topics are in fact about Agriculture (the source is about 34%), while 27.5% is about transportation (the source is about 16%) and construction is about 7.5% (the source is about 8%). This matches well with the source distribution; the only topic that does not correlate with the source distribution is IT with only 10% compared to ~33% on the source distribution.
It is hard to come up with an explanation for why this is, but it could have something to do with IT (and transportation) being a support industry, where agriculture and construction are more stand-alone trades. What I mean is that while someone can do agriculture without construction and vice versa, transportation and IT cannot be done without having something to transport or to manage/model. The two trades would simply not be there if it was not for the other industry types.

Like the previous analysis, I would like to see if the LSA transformation is able analyze what a press release is about. Since we have an existing structure in place for categorization, I can take advantage of that and find a press release that has only been sent to agricultural media outlets. If the analysis works as expected, it should return a result of mainly the topics where agriculture is weighted highest.

The press release that I have chosen is about tax cuts, corporate economics, politics, exports, consumer demand and energy taxation (Appendix K. While it has strictly been sent to agricultural media outlets, the press release itself does not contain any mention of farming or the like. By running it through the LSA model, we get the following result:

```
1: ( 0,  1.9771),
2: (27,  0.7317),
3: (11,  0.6442), ( 1,  0.6130), (15, -0.5928), ( 4,  0.5465), (32,  0.4967), (16, -0.4875),
   (14,  0.4822),
4: ( 8, -0.3871), (21,  0.3231), (39, -0.3147), (20, -0.3061),
5: (23,  0.2826), (24, -0.2795), (31,  0.2652), (22,  0.2558), (28, -0.2507), (29, -0.2473),
   (19, -0.2431), (12, -0.2386), ( 3, -0.2366), (26, -0.2329), ( 2,  0.1994), ( 7, -0.1930),
   ( 6,  0.1918), ( 5,  0.1537), (13,  0.1359), (17,  0.1189), (30,  0.1160), (36,  0.1044),
   ( 9, -0.0967), (35, -0.0844), (33, -0.0623), (38, -0.0471), (34, -0.0463), (10,  0.0411),
   (25,  0.0272), (37,  0.0215), (18, -0.0036)
```

Topic #0 is again predominant in the result set. Like in the last experiment, I will ignore it, since it clearly does not bring any value. The above result roughly translates into the topics of agriculture and construction (#27), and with a clear bump down in the scores, IT, agriculture (again) (#11, #15), IT (#1), agriculture (#4), (#32), transportation, agriculture (#16, #14). While the LSA result point in the direction of the press release being about agriculture and transportation, the relations to construction and IT are simply not a part of

the press release.
I had hoped for a better hit rate on topics like fuel sources (#7, #8) due to the mention of energy taxation in the press release. The other topics in the press release were not covered by the topics thus making it impossible to hit them.
One thing we did not do in the previous experiment was to make an analysis by reversing the order. By looking at the topics that scored the lowest, we should be able to see what the press release is clearly not about. The last three topics translate into religion (#18), agriculture (#37) and agriculture, transportation (#25). While religion is correctly not a part of the text, the other two counter the high scores by indicating that the press release is not about agriculture and transportation, while this is in fact quite the opposite.

While the result of the above experiment does look better than the one I did on Wikipedia, we still need some kind of guideline for detecting how close the LSA gets to the empirical measurements. For the Wikipedia data, we had no way of doing that, since Wikipedia is a little bit about everything, but the data that was scraped is picked out from a very narrow segment of media outlets that mainly write about Agriculture, Construction, IT and Transportation.
What we got here is in fact an existing structure to the dataset. By using the structure as a guide, we should be able to validate the topics from the LSA. To do this, I will start by feeding back all the scraped articles onto the LSA and recording which topics get activated the most. If I take all of the returned topics, I will get a usage of 100%, since all topics are always represented in the result. This is why only the top-most topics should be picked. The total number of words in each topic is 20, so the first eight topics in each result should suffice. The result can be seen in the "LSA" columns in figure 6.

AGRICULTURE

empirical		*LSA*	
~~#0~~	~~0.9954~~	~~#0~~	~~0.9955~~
#1	0.4341	#5	0.4677
#4	0.3422	#2	0.4098
#2	0.3382	#4	0.4031
#9	0.2880 x	#10	0.3608
#5	0.2677	#12	0.3541 x
#6	0.2509 x	#1	0.2762
#10	0.2351	#15	0.2004 x
Sample total: 47,465		Sample total: 449	

IT

empirical		*LSA*	
~~#0~~	~~0.9991~~	~~#0~~	~~0.9983~~
#1	0.5359	#5	0.4868
#2	0.3124	#4	0.4769 x
#15	0.2949	#1	0.3234
#5	0.2684	#10	0.3119
#10	0.2437	#15	0.2574
#12	0.2323 x	#2	0.2558
#9	0.2273	#9	0.2343
Sample total: 45,689		Sample total: 606	

CONSTRUCTION

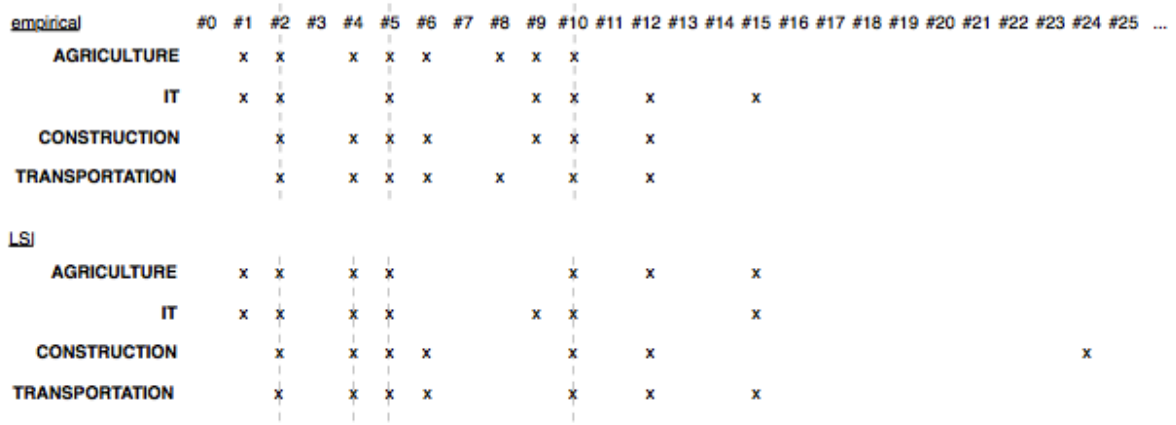
empirical		*LSA*	
~~#0~~	~~0.9985~~	~~#0~~	~~1.0000~~
#5	0.5525	#5	0.7015
#4	0.3964	#4	0.4608
#2	0.2832	#12	0.2789
#12	0.2782	#10	0.2462
#24	0.2379 x	#2	0.2418
#6	0.2308	#6	0.1993
#10	0.2193	#9	0.1819 x
Sample total: 10,826		Sample total: 918	

TRANSPORT

empirical		*LSA*	
~~#0~~	~~0.9252~~	~~#0~~	~~0.9901~~
#5	0.3626	#5	0.5446
#2	0.2977	#10	0.3837
#4	0.2846	#4	0.3465
#8	0.2571 x	#6	0.2599
#10	0.2388	#12	0.2574
#6	0.2031	#2	0.2525
#12	0.1972	#15	0.2426 x
Sample total: 21,901		Sample total: 404	

[Fig. 6] *The empirical and generated topics are sent through the LSA transformation to return which topics they are closest to. The x in the right margin marks topics that are not present in the counter column. The lines that are crossed out are discarded topics.*

If the content in each of the groups, were in fact different from an LSA point of view. The topics above would be spread out a lot more, but it seems that topics #2, #4, #5 and #10 cover all areas, while most of the other topics are activated in a lesser degree. This implies that the topics are not broad enough for this kind of operation. I could include more topics in the overview, but that would bring the hit rate below 20% which I assess is too low. This is unfortunate, since the articles scraped are indeed picked up from very dissimilar media outlets.



[Fig. 7] *The result from above listed as each topic, with the hit rates marked with an x for each of groups. Here it is very easy to see that there is not a correlation between the topics and the groups.*

To finish up this experiment, the same calculations are done for press releases from Press2go's database. Since we know what trade group each of the origin media outlets resides in, we are able to take press releases sent and opened by those media outlets and feed them onto the LSA model. The result of this is shown in the column labeled "empirical". The result matches those of LSA-generated origin, but they are still too broad, since they also match the other groups.

In figure 7 it is very easy to see that there is not a correlation between the topics and the groups. This could be because there are not enough documents to support this kind of analysis. It can also be because the parameters in gensim have not been set properly and lastly it can simply be because the four groups share too many common words.

Conclusion

The goal of this thesis has been to analyze the use of LSA to find a more dynamic way of connecting content from press releases with potential recipient media outlets. In this process I have tried to get useable results by applying LSA in analyzing 121,350 articles from the Danish version of Wikipedia. On top of this, I also scraped 138,363 articles from 28 Danish online news outlets. The approach of the thesis was to try to see if by applying LSA to the articles, usable topics would emerge that could be used to categorize existing press releases.

The result of the two approaches turned out not to have the wanted result. This could however be caused by a lack of relevant articles to perform the LSA on. Typically the operation needs close to a million documents, but for the lack of a solid data source the scraping was as close as I could get.

This leaves me to conclude that even with my best effort, I could not get LSA to predict which media outlets would have an interest in a given press release. If the amount of data could be extended, that would perhaps change the outcome for the better, but for now LSA remains unsuitable for heightening the consumer experience for Press2go customers. The search will have to continue in other areas.

Bibliography

(Deerwester et al., 1990) Deerwester, S., Dumais, S. T., Furnas, G. W., Landauer, T. K., and Harshman, R. (1990). Indexing by latent semantic analysis. *Journal of the American Society for Information Science*, 41(6):391–407.

(Lund et al., 2009) Lund, A. 2009. Hvor kommer nyhederne fra? Den journalistiske fødekæde I Danmark før og nu. Forlaget Ajour.

(gensim, 2012) The gensim Python toolkit: http://radimrehurek.com/gensim/ .

(Landauer and Dumais, 1997) Landauer, T. K. and Dumais, S. T. (1997). A solution to plato's problem: The latent semantic analysis theory of acquisition, induction, and representation of knowledge. *Psychological Review*, 104(2):211–240.

(Manning et al. 2008) Manning, C, Raghavan, P and Schütze H. 2008. Introduction to Information Retrieval. *Cambridge University Press*.

(Griffiths et. al., 2007) Griffiths, T. L., Steyvers M. and Tenenbaum, J. B. (2007). Topics in Semantic Representation. *Psychological Review*, 2007, Vol. 114, No. 2, 211–244

(Wikipedia, 2012) Wikipedia, 2012-06-23. http://en.wikipedia.org/wiki/Polysemy

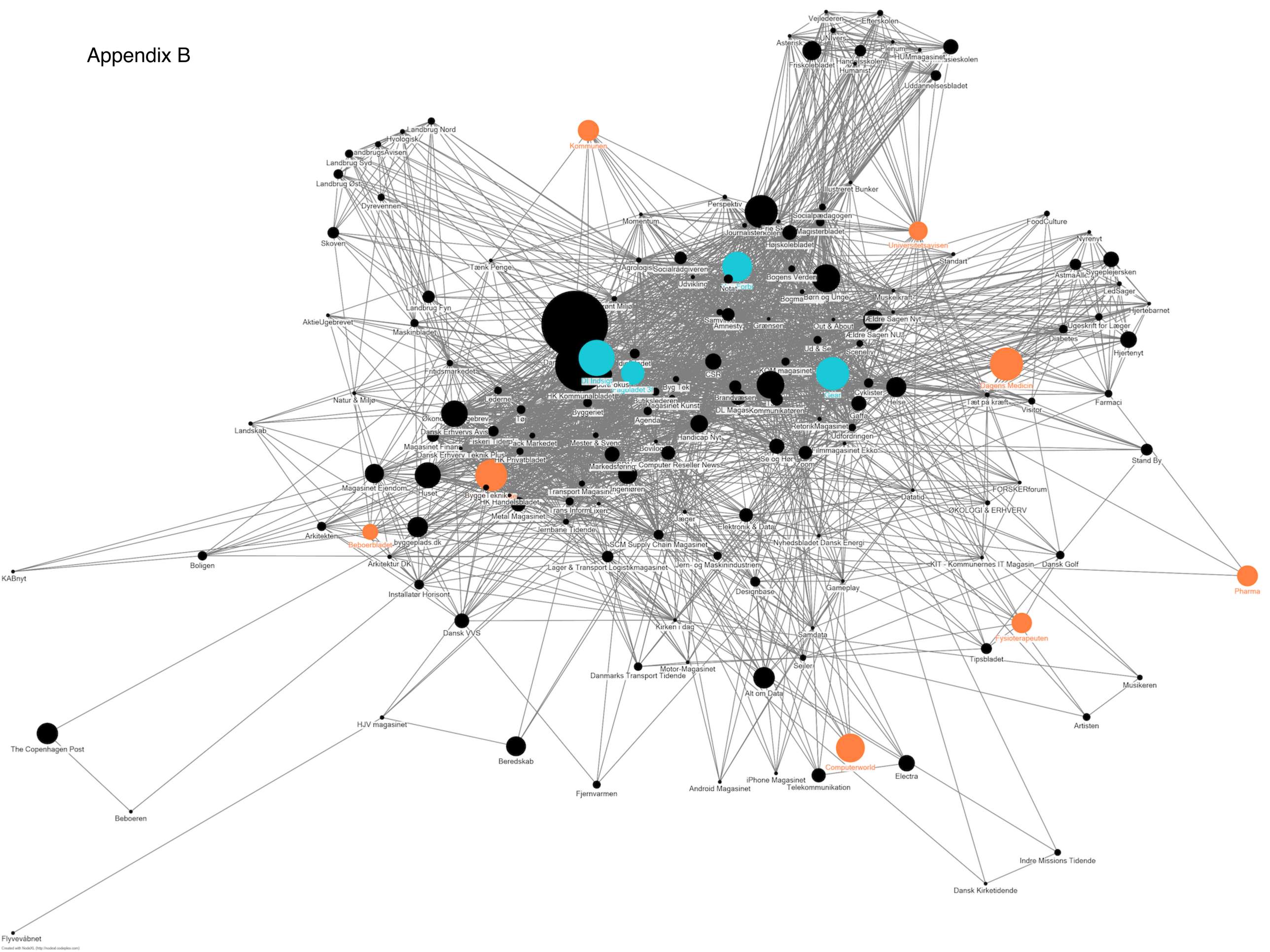
(tfidfmodel, 2012) Website containing the documentation for models.tfidfmodel – TF-IDF model, 2012-06-23. http://radimrehurek.com/gensim/models/tfidfmodel.html

Appendix A

Taken from the Word document "Projekt Nyhedsuge: Deltagende observation" (DA) on the website of the book "Hvor kommer nyhederne fra?": cbs.dk/fbc_Nyhedsugen

- Altinget.dk
- B.T.
- Berlingske
- Berlingske.dk
- Bornholms Tidende
- BT.dk
- Børsen + Børsen.dk
- Dagbladet (Ringsted)
- DR Nyheder
- DR Radio Bornholm
- DR Radio Fyn
- DR Radio Jylland-Ålborg
- DR Radio Jylland-Århus
- DR Radio Næstved
- DR UPDATE
- EB.dk
- Ekstra Bladet
- Fyens Stiftstidende
- Information
- JP.dk
- Jyllands-Posten
- Kristelig Dagblad
- MetroXpress/24timer
- Newspaq+MediaWatch
- Nordjydske Medier
- Politiken
- Politiken.dk
- Ritzau
- TV 2 BORNHOLM
- TV 2 FYN
- TV 2 NEWS
- TV 2 NORD (Ålborg)
- TV 2 Odense
- TV 2 ØST (Vordingborg)
- TV 2 ØSTJYLLAND (Århus)
- TV LORRY

Appendix B


Vejlederen
Efterskolen
Asterisk
UNIvers
Plenum
Friskolebladet
Handelsskolen
HUMmagasinet
Humanist
Uddannelsesbladet
Landbrug Nord
Hvologisk
LandbrugsAvisen
Landbrug Syd
Landbrug Øst
Dyrevennen
Kommunen
Illustreret Bunker
Perspektiv
Socialpædagogen
Momentum
Journalisterkolen
Magisterbladet
Skoven
Højskolebladet
Universitetsavisen
FoodCulture
Nyrenyt
Standart
Tænk Penge
Agrologisk
Socialrådgiveren
Udvikling
Notat
Bogens Verden
Sygeplejersken
LedSager
Bogmagasinet
Børn og Unge
Muskelkraft
Landbrug Fyn
Hjertebarnet
AktieUgebrevet
Maskinbladet
Samvirke
Amnesty
Grænsen
Out & About
Ældre Sagen Nyt
Ældre Sagen NU
Ugeskrift for Læger
Diabetes
Ud & Se
Scenekunst
Hjertenyt
Fritidsmarkedet
CSR
KOM magasinet
Dagens Medicin
DI Indsigt
Byg Tek
HK Kommunalbladet
Gear
Cyklister
Natur & Miljø
Lederne
Brandvæsen
Helse
Tæt på kræft
Visitor
Farmaci
Butiksleder
Magasinet Kunst
DL Magasinet
Kommunikatøren
Byggeriet
Landskab
Agenda
Gaffa
RetorikMagasinet
Handicap Nyt
Udfordringen
Dansk Erhvervs Avis
Fiskeri Tidende
Pack Markedet
Mester & Svend
Magasinet Finans
Filmmagasinet Ekko
Dansk Erhverv Teknik Plus
HK Privatbladet
Se og Hør
Stand By
Markedsføring
Computer Reseller News
Magasinet Ejendom
Huset
Transport Magasinet
Ingeniøren
Datatid
FORSKERforum
HK Handelsbladet
ØKOLOGI & ERHVERV
Trans InformLixen
Metal Magasinet
Jæger
Arkitekten
Jernbane Tidende
Elektronik & Data
Beboerbladet
byggeplads.dk
SCM Supply Chain Magasinet
Nyhedsbladet Dansk Energi
Boligen
Arkitektur DK
Lager & Transport Logistikmagasinet
Jern- og Maskinindustrien
KIT - Kommunernes IT Magasin
Dansk Golf
KABnyt
Pharma
Designbase
Gameplay
Installatør Horisont
Dansk VVS
Kirken i dag
Samdata
Fysioterapeuten
Tipsbladet
Sejler
Motor-Magasinet
Danmarks Transport Tidende
Musikeren
Alt om Data
HJV magasinet
Artisten
The Copenhagen Post
Beredskab
Computerworld
Electra
iPhone Magasinet
Telekommunikation
Fjernvarmen
Android Magasinet
Beboeren
Indre Missions Tidende
Dansk Kirketidende
Flyvevåbnet
Created with NodeXL (http://nodexl.codeplex.com)

Appendix C

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-

"""
USAGE: %(program)s WIKI_XML_DUMP OUTPUT_PREFIX

    Convert articles from a Wikipedia dump to (sparse) vectors.
    The input is a bz2-compressed dump of Wikipedia articles, in XML format.

This actually creates three files:

* `OUTPUT_PREFIX_wordids.txt`: mapping between words and their integer ids
* `OUTPUT_PREFIX_bow.mm`: bag-of-words (word counts) representation, in Matrix
Matrix format
* `OUTPUT_PREFIX_tfidf.mm`: TF-IDF representation

The output Matrix Market files can then be compressed (e.g., by bzip2) to save \
disk space; gensim's corpus iterators can work with compressed input, too.

Example: ./%(program)s data/enwiki-latest-pages-articles.xml.bz2 data/wiki_en
"""



import logging, sys, re, bz2, codecs, os
from hashlib import md5
from gensim import interfaces, matutils, utils, parsing

# cannot import whole gensim.corpora, because that imports wikicorpus...
from gensim.corpora.dictionary import Dictionary
from gensim.corpora.textcorpus import TextCorpus
from gensim.corpora.mmcorpus import MmCorpus

logger = logging.getLogger('gensim.corpora.wikicorpus')

# Ignore articles shorter than ARTICLE_MIN_CHARS characters (after preprocessing).
ARTICLE_MIN_CHARS = 500



RE_P0 = re.compile('<!--.*?-->', re.DOTALL | re.UNICODE) # comments
RE_P1 = re.compile('<ref([> ].*?)(</ref>|/>)', re.DOTALL | re.UNICODE) # footnotes
RE_P2 = re.compile("(\n\[\[[a-z][a-z][\w-]*:[^:\]]+\]\])+$", re.UNICODE) # links to
languages
RE_P3 = re.compile("{{([^}{]*)}}", re.DOTALL | re.UNICODE) # template
RE_P4 = re.compile("{{([^}]*)}}", re.DOTALL | re.UNICODE) # template
RE_P5 = re.compile('\[(\w+):\/\/(.*?)(( (.*?))|())\]', re.UNICODE) # remove URL,
keep description
RE_P6 = re.compile("\[([^][]*)\|([^][]*)\]", re.DOTALL | re.UNICODE) # simplify
```

```
links, keep description
RE_P7 = re.compile('\n\[\[[iI]mage(.*?)(\|.*?)*\|(.*?)\]\]', re.UNICODE) # keep
description of images
RE_P8 = re.compile('\n\[\[[fF]ile(.*?)(\|.*?)*\|(.*?)\]\]', re.UNICODE) # keep
description of files
RE_P9 = re.compile('<nowiki([> ].*?)(</nowiki>|/>)', re.DOTALL | re.UNICODE) #
outside links
RE_P10 = re.compile('<math([> ].*?)(</math>|/>)', re.DOTALL | re.UNICODE) # math
content
RE_P11 = re.compile('<(.*?)>', re.DOTALL | re.UNICODE) # all other tags
RE_P12 = re.compile('\n({\|)|(\|-)|(\|}))(.*?)(?=\n)', re.UNICODE) # table
formatting
RE_P13 = re.compile('\n(\||\!)(.*?\|)*([^|]*?)', re.UNICODE) # table cell formatting
RE_P14 = re.compile('\[\[Category:[^][]*\]\]', re.UNICODE) # categories
RE_P15 = re.compile('=.*=', re.UNICODE) # categories
RE_P16 = re.compile('\[\[.*:.*\]\]', re.UNICODE)

##RE_P50 = re.compile('\[\[Kategori:[^].*\]\]', re.UNICODE) # categories
##RE_P51 = re.compile('^\[\[[a-z]{2}:.*\]\]$', re.UNICODE) # refs to other languages




def filter_wiki(raw):
    """
    Filter out wiki mark-up from `raw`, leaving only text. `raw` is either unicode
    or utf-8 encoded string.
    """
    # parsing of the wiki markup is not perfect, but sufficient for our purposes
    # contributions to improving this code are welcome :)
    text = utils.decode_htmlentities(utils.to_unicode(raw, 'utf8', errors='ignore'))
    text = utils.decode_htmlentities(text) # '&amp;nbsp;' --> '\xa0'
    return remove_markup(text)


def remove_markup(text):
    text = re.sub(RE_P2, "", text) # remove the last list (=languages)
    # the wiki markup is recursive (markup inside markup etc)
    # instead of writing a recursive grammar, here we deal with that by removing
    # markup in a loop, starting with inner-most expressions and working outwards,
    # for as long as something changes.
    iters = 0
    while True:
        old, iters = text, iters + 1
        text = re.sub(RE_P0, "", text) # remove comments
        text = re.sub(RE_P1, '', text) # remove footnotes
        text = re.sub(RE_P9, "", text) # remove outside links
```

```
        text = re.sub(RE_P10, "", text) # remove math content
        text = re.sub(RE_P11, "", text) # remove all remaining tags
        # remove templates (no recursion)
        text = re.sub(RE_P3, '', text)
        text = re.sub(RE_P4, '', text)
        text = re.sub(RE_P14, '', text) # remove categories
        text = re.sub(RE_P5, '\\3', text) # remove urls, keep description
        text = re.sub(RE_P7, '\n\\3', text) # simplify images, keep description only
        text = re.sub(RE_P8, '\n\\3', text) # simplify files, keep description only
        text = re.sub(RE_P6, '\\2', text) # simplify links, keep description only
        # remove table markup
        text = text.replace('||', '\n|') # each table cell on a separate line
        text = re.sub(RE_P12, '\n', text) # remove formatting lines
        text = re.sub(RE_P13, '\n\\3', text) # leave only cell content

        text = re.sub(RE_P16, '', text)



        # remove empty mark-up
        text = text.replace('[]', '')
        # stop if nothing changed between two iterations or after a fixed
        # number of iterations.
        if old == text or iters > 2:
            break

    # the following is needed to make the tokenizer see '[[socialist]]s'
    # as a single word 'socialists'
    # TODO is this really desirable?
    # promote all remaining markup to plain text.
    text = text.replace('[', '').replace(']', '')
    return text


def tokenize(content):
    """
    Tokenize a piece of text from wikipedia. The input string `content` is assumed
    to be mark-up free (see `filter_wiki()`).

    Return list of tokens as utf8 bytestrings. Ignore words shorted than 2 or longer
    that 15 characters (not bytes!).
    """
    # TODO maybe ignore tokens with non-latin characters? (no chinese, arabic,
russian etc.)
    return [token.encode('utf8') for token in utils.tokenize(content, lower=True,
errors='ignore')
            if 3 <= len(token) <= 15 and not token.startswith('_')]
```

```
def write_to_disk(text):
    filename = md5.new(text).hexdigest()
    f = open('data/' + filename[:8] + '.txt', 'w')
    f.write(text)
    f.close()

class MyWikiCorpus(TextCorpus):
    """
    Treat a wikipedia articles dump (*articles.xml.bz2) as a (read-only) corpus.

    The documents are extracted on-the-fly, so that the whole (massive) dump
    can stay compressed on disk.
    """
    def __init__(self, fname):
        self.fname = fname
        self.dictionary = Dictionary(self.get_texts())
        self.dictionary.filter_extremes(no_below=50, no_above=0.5, keep_n=100000)

    def get_texts(self):
        """
        Iterate over the dump, returning text version of each article.

        Only articles of sufficient length are returned (short articles & redirects
        etc are ignored).

        Note that this iterates over the **texts**; if you want vectors, just use
        the standard corpus interface instead of this function::

        >>> for vec in wiki_corpus:
        >>>     print vec
        """
        articles, articles_all, 0, 0
        intext, positions = False, 0
        for lineno, line in enumerate(bz2.BZ2File(self.fname)):

            if line.startswith('      <text'):
                intext = True
                line = line[line.find('>') + 1 : ]
                lines = [line]
            elif intext:
                lines.append(line)

            pos = line.find('</text>') # can be on the same line as <text>
            if pos < 0:
                continue

            articles_all += 1
```

```
            intext = False

            if not lines:
                continue

            lines[-1] = line[:pos]
            text = filter_wiki(' '.join(lines))
            if len(text) <= ARTICLE_MIN_CHARS: # article redirects are pruned here
                continue

            articles += 1
            result = tokenize(text) # text into tokens here
            positions += len(result)
            yield result

        logger.info("finished iterating over the Wikipedia corpus "
            "of %i documents with %i positions "
            "(total %i articles before pruning)." % (articles, positions,
articles_all))
        self.length = articles # cache corpus length


if __name__ == '__main__':
    logging.basicConfig(filename = OUTPUT_FILE + '.log',
        filemode='a',
        format='%(asctime)s : %(levelname)s : %(message)s',
        level=logging.INFO)

    program = os.path.basename(sys.argv[0])


    # check and process input arguments
    if len(sys.argv) < 3:
        print globals()['__doc__'] % locals()
        sys.exit(1)
    INPUT_FILE, OUTPUT_FILE = sys.argv[1:3]

    wiki = MyWikiCorpus(INPUT_FILE)

    # save dictionary and bag-of-words (term-document frequency matrix)
    wiki.dictionary.save_as_text(OUTPUT_FILE + '_wordids.txt')
    MmCorpus.serialize(OUTPUT_FILE + '_bow.mm', wiki, progress_cnt=10000)
    del wiki


    # initialize corpus reader and word->id mapping
    dictionary = Dictionary.load_from_text(OUTPUT_FILE + '_wordids.txt')
```

```
    mm_corpus = MmCorpus(OUTPUT_FILE + '_bow.mm')

    # build tfidf,
    from gensim.models import TfidfModel
    tfidf = TfidfModel(mm_corpus, id2word=dictionary, normalize=True)

    # save tfidf vectors in matrix market format
    MmCorpus.serialize(OUTPUT_FILE + '_tfidf.mm', tfidf[mm_corpus],
progress_cnt=10000)
```

Appendix D

wiki_da_40t_lsi (d74a7f)

#0	#1	#2	#3
0.3841 han	-0.7662 bebyggelse	0.4041 jeg	-0.3919 han
0.1288 hun	-0.3845 sogn	-0.3744 han	-0.3034 jeg
0.1231 hans	-0.3396 ejerlav	0.3397 cest	-0.2874 cest
0.1149 ikke	-0.1308 kommune	0.2489 cet	-0.1950 cet
0.1098 the	-0.1239 areal	0.1715 kan	0.1576 ligger
0.1071 ved	-0.1066 amt	0.1361 ikke	0.1514 byen
0.0997 jeg	-0.0966 herred	0.1220 man	-0.0980 bebyggelse
0.0995 kan	-0.0872 ligger	0.1058 eller	0.0967 eller
0.0961 sig	-0.0847 sognet	0.1055 skal	0.0948 kommune
0.0952 sin	-0.0845 stednavne	0.0989 artiklen	0.0864 jpg
0.0947 men	-0.0833 sogneportalen	0.0960 vil	0.0827 månen
0.0935 efter	-0.0821 autoriserede	0.0939 wikipedia	0.0807 indbyggere
0.0922 havde	-0.0805 provsti	0.0870 artikler	0.0776 nord
0.0916 dansk	-0.0804 flg	-0.0861 født	-0.0747 hans
0.0903 man	-0.0774 kirke	-0.0860 hans	0.0730 syd
0.0869 eller	-0.0721 stift	0.0851 hvis	0.0711 del
0.0808 hvor	-0.0624 indtil	0.0825 artikel	-0.0709 artiklen
0.0799 født	0.0583 han	0.0824 være	0.0707 kratere
0.0770 også	-0.0380 findes	0.0733 jun	0.0703 mod
0.0742 under	-0.0367 vandareal	0.0685 okt	0.0701 kommunen

#4	#5	#6	#7
-0.7474 hun	0.5237 the	-0.2147 landshold	-0.2615 the
-0.2960 the	-0.4629 hun	-0.2055 klubben	-0.1969 månen
0.2560 han	0.1315 and	-0.1901 kampe	-0.1748 kratere
-0.1617 hendes	0.1077 album	-0.1600 hold	-0.1618 cest
-0.0919 film	-0.0976 kommune	-0.1489 spillede	-0.1545 ligger
-0.0770 hende	0.0936 bebyggelse	-0.1391 cest	-0.1422 hovedkrateret
-0.0726 and	0.0918 bandet	-0.1379 fodboldspiller	0.1375 kan
-0.0670 skuespiller	0.0891 you	-0.1369 spillerinfo	-0.1335 kommune
-0.0669 album	0.0874 albummet	-0.1242 cup	-0.1226 månens
-0.0608 filmen	-0.0846 hendes	-0.1210 mål	0.1215 bebyggelse
0.0584 ligger	0.0825 guitar	-0.1203 kamp	0.1210 eller
-0.0569 født	-0.0801 kirke	-0.1182 spiller	-0.1168 jeg
0.0556 byen	-0.0703 dansk	-0.1139 league	0.1078 hun
-0.0553 you	-0.0694 ligger	-0.1131 ligger	0.1073 man
0.0532 hans	-0.0691 byen	-0.1078 spillet	-0.1068 sogn
-0.0530 bebyggelse	0.0684 udgivet	0.1032 hans	0.1060 landshold
-0.0521 albummet	0.0660 live	-0.1014 hun	-0.1036 byen
-0.0503 amerikansk	-0.0649 københavn	-0.1011 byen	0.1009 kampe
-0.0495 teater	0.0610 rock	0.0980 han	0.0997 klubben
0.0462 mod	-0.0604 sogn	-0.0951 jeg	0.0965 hold

#8	#9	#10	#11
0.3166 månen	0.2556 sogn	-0.6408 cet	0.3709 sogn
0.2786 kratere	0.2512 dansk	0.5689 cest	0.2805 jpg
0.2276 hovedkrateret	-0.2037 han	-0.1848 feb	0.2607 kirke
0.1968 månens	-0.1950 byen	-0.1559 dec	-0.2141 dansk
-0.1846 kommune	-0.1434 hun	0.1542 jun	-0.2014 bebyggelse
-0.1649 sogn	-0.1348 landkreis	-0.1538 jan	0.1999 han
0.1540 bebyggelse	-0.1338 bebyggelse	-0.1247 nov	0.1974 image
0.1526 satellitkratere	0.1225 født	0.1221 apr	-0.1418 død
-0.1506 byen	0.1037 danske	0.1038 sep	-0.1389 cet
-0.1439 kirke	0.1025 københavn	0.0993 aug	0.1255 hun
0.1398 rand	-0.1015 cest	-0.0908 mar	-0.1163 født
0.1374 bogstav	0.1015 skuespiller	0.0853 jul	-0.1126 skuespiller
-0.1339 jpg	0.1002 danmark	-0.0758 sogn	0.1120 kirken
0.1178 liste	-0.0993 ligger	-0.0718 han	0.0954 herred
0.1125 iau	0.0950 død	0.0637 dansk	-0.0916 amerikansk
0.1076 han	-0.0937 indbyggere	0.0623 maj	0.0858 landshold
-0.1034 the	-0.0934 floden	0.0552 okt	0.0851 kommune
0.0976 nedslagskrater	-0.0911 ham	-0.0505 jeg	-0.0846 byen
0.0972 månekratere	0.0908 kirke	0.0445 bebyggelse	-0.0829 forfatter
0.0933 hun	0.0905 hansen	-0.0428 kommune	0.0808 pastorat

#12	#13	#14	#15
-0.5307 jpg	0.2876 arter	0.2234 arter	0.3574 insee
0.4289 sogn	0.1865 almindelig	0.2165 sogn	0.2243 switch
-0.3565 image	-0.1777 sogn	0.1480 hold	0.2095 metadata
-0.2525 kirke	-0.1322 hold	0.1407 almindelig	0.1694 giver
0.2159 kommune	0.1260 bladene	-0.1237 klubben	0.1600 jpg
-0.2106 bebyggelse	0.1214 blomsterne	-0.1220 insee	0.1599 bruges

-0.1687 fil	0.1189 han	0.1210 kamp	-0.1504 meter
-0.1208 kirken	0.1177 landkreis	0.1133 sverige	0.1501 indsætte
0.1107 herred	0.1073 planten	0.1008 cet	0.1379 och
-0.1009 cet	-0.1043 kamp	-0.1007 landshold	0.1277 donnees
0.0926 pastorat	0.1015 blomster	-0.0992 kan	0.1241 timestamp
-0.0904 file	0.1000 frugterne	0.0992 gruppe	0.1222 uformateret
-0.0783 ejerlav	0.0944 slægt	0.0980 bladene	0.1207 skabelon
-0.0672 och	0.0940 landshold	0.0954 blomsterne	0.1197 maritimes
0.0670 kommunen	0.0918 blade	0.0910 norge	0.1193 ugyldig
-0.0625 billede	-0.0911 gruppe	-0.0904 man	0.1185 alpes
0.0606 han	0.0910 art	0.0899 von	0.1177 default
0.0493 amt	0.0907 ligger	0.0877 danmark	0.1173 resume
-0.0484 korttilkirken	-0.0898 sverige	0.0874 frankrig	0.1127 image
-0.0470 kirkens	0.0877 født	0.0844 point	0.1123 kilden

#16	**#17**	**#18**	**#19**
-0.2544 død	-0.6276 och	0.4514 meter	-0.5022 meter
-0.2292 født	-0.2389 till	-0.3240 och	0.1908 jpg
-0.2280 sogn	0.1973 kirke	-0.1754 kirke	-0.1791 och
-0.2271 skuespiller	-0.1913 att	0.1735 jpg	0.1573 medlem
-0.2016 amerikansk	-0.1554 meter	0.1501 sogn	0.1555 image
0.1553 meter	-0.1536 för	-0.1498 von	-0.1439 kirke
-0.1540 dansk	-0.1396 text	0.1395 han	0.1434 formand
0.1248 medlem	-0.1210 ett	0.1238 image	-0.1044 film
-0.1243 forfatter	-0.1181 från	-0.1220 till	0.1004 partiet
-0.1127 film	-0.1058 jpg	-0.1035 klubben	-0.1001 insee
0.1084 hold	0.1020 von	-0.1015 christian	0.0943 parti
0.1062 insee	0.0997 insee	-0.0984 frederik	0.0919 folketinget
0.0988 københavn	-0.0971 sogn	-0.0982 att	0.0916 universitet
-0.0882 dødsfald	0.0936 kirken	-0.0920 landshold	-0.0893 filmen
-0.0866 jpg	-0.0889 bildförhållande	0.0834 medlem	0.0877 kommune
0.0845 han	-0.0889 bildtext	0.0818 fil	-0.0869 kirken
0.0843 formand	-0.0867 bild	-0.0790 för	-0.0789 øen
0.0842 københavns	-0.0855 medlem	-0.0773 guitar	-0.0772 mesterskaber
-0.0839 landshold	0.0803 død	-0.0758 kirken	-0.0741 hæk
-0.0838 filmen	-0.0773 han	-0.0748 album	0.0737 valgt

#20	**#21**	**#22**	**#23**
-0.3035 landkreis	-0.2960 meter	-0.3491 the	0.4112 kirke
-0.2192 bayern	0.1871 film	-0.1934 von	-0.2083 jpg
-0.1987 kommunen	0.1790 kirke	0.1526 bandet	0.1744 the
0.1749 byen	-0.1501 bandet	0.1455 guitar	0.1599 kirken
-0.1528 delstat	-0.1452 album	0.1348 han	0.1538 død
-0.1360 geografi	0.1424 byen	0.1333 album	0.1455 meter
0.1296 insee	0.1411 the	-0.1285 klubben	0.1362 født
-0.1183 hold	-0.1403 guitar	-0.1245 and	-0.1276 film
-0.1105 kan	-0.1365 sogn	-0.1110 frederik	-0.1274 image
-0.1065 jpg	-0.1266 albummet	0.1105 albummet	-0.1182 von
0.0969 kirke	0.1184 hold	0.1064 sangen	-0.1181 byen
-0.0945 landsbyer	0.1158 filmen	-0.1045 christian	-0.1084 fil
0.0934 och	-0.1046 landkreis	0.1039 byen	-0.1033 filmen
-0.0933 bebyggelser	-0.1017 landshold	0.1010 født	-0.0977 klubben
-0.0926 kommune	-0.1002 jpg	0.0998 skuespiller	0.0926 medlem
0.0923 cet	0.0987 teater	-0.0971 gift	-0.0919 teater
-0.0917 image	-0.0986 tyske	0.0946 musik	-0.0876 sogn
-0.0913 film	-0.0975 sangen	0.0928 sang	-0.0867 instruktør
-0.0872 oberbayern	0.0965 kamp	0.0925 kirke	-0.0804 amtet
-0.0860 kamp	-0.0955 von	0.0847 komponist	0.0798 politiker

#24	**#25**	**#26**	**#27**
-0.2542 xiàn	0.3199 xiàn	0.2391 kommune	0.3408 kommune
-0.2382 amtet	0.3133 amtet	0.2299 align	-0.1766 kirke
-0.1765 county	0.2686 county	0.2205 style	0.1603 død
-0.1657 style	-0.2532 byen	0.1808 border	-0.1589 landkreis
-0.1621 align	0.1517 kirke	-0.1336 the	-0.1531 sogn
-0.1424 von	0.1182 landkreis	-0.1331 sogn	-0.1399 teater
-0.1276 landshold	0.1140 husstande	0.1313 stub	-0.1389 bayern
-0.1231 indbyggere	0.1049 amter	0.1281 left	-0.1237 film
-0.1190 border	-0.1007 landshold	0.1248 colspan	0.1168 the
-0.1160 areal	0.0922 amt	0.1136 klasse	0.1155 fredede
0.1086 kirke	0.0898 delstat	0.1093 margin	0.1122 byen
0.1006 landkreis	0.0863 shì	-0.1074 landshold	-0.1099 style
-0.0964 spillerinfo	-0.0830 teater	0.1067 start	-0.1066 delstat
-0.0953 left	0.0781 kinas	0.1026 skabelon	-0.1031 pastorat
-0.0896 amter	-0.0781 spillerinfo	0.1021 wikiprojekt	-0.1030 tyske
-0.0888 byen	0.0771 areal	0.0970 background	-0.1030 instruktør
-0.0881 colspan	0.0749 distrikter	0.0962 text	-0.1023 erne
-0.0871 stub	0.0737 division	0.0936 right	-0.0919 align
0.0857 film	0.0702 enheder	-0.0934 xiàn	-0.0909 county
0.0856 kommune	0.0701 rigsvej	0.0900 kategori	0.0902 jpg

#28		#29		#30		#31	
-0.2842	kommune	-0.4703	county	0.2569	byen	-0.2064	side
0.2810	byen	0.3592	xiàn	-0.2255	kommune	0.1805	station
-0.1844	instruktør	-0.2178	husstande	-0.1980	arter	-0.1394	arter
-0.1737	landshold	0.1745	district	-0.1631	slægt	-0.1392	slægt
-0.1570	xiàn	-0.1484	amtet	0.1627	instruktør	-0.1324	anders
-0.1475	teater	0.1337	distrikter	-0.1359	fredede	-0.1158	meter
-0.1414	film	-0.1136	boede	-0.1079	kommunen	0.1150	stationen
-0.1356	spillerinfo	-0.1133	familier	0.1063	kan	0.1140	klassen
0.1286	død	-0.1088	årligt	0.1033	teater	-0.1068	venstre
0.1227	dansk	0.1057	amter	0.0962	medlem	-0.1051	partiet
-0.1177	norge	0.1007	administrative	-0.0958	hans	-0.1041	parti
0.1135	county	-0.0984	kommune	-0.0951	klubben	-0.1010	klubben
-0.1114	fredede	0.0960	shì	-0.0945	sprog	0.0971	slam
-0.1081	historik	0.0948	development	-0.0918	fortidsminder	0.0932	københavn
0.1029	sogn	-0.0889	hustand	0.0917	man	0.0910	plads
0.0998	pastorat	0.0883	kinas	0.0888	floden	-0.0898	øen
0.0988	klubben	-0.0864	amtets	-0.0887	county	-0.0886	folkeparti
-0.0960	filmen	0.0794	enheder	0.0872	folketinget	0.0886	grand
-0.0943	diskussion	0.0778	rigsvej	0.0846	formand	-0.0852	folketinget
-0.0939	kommunen	-0.0772	afroamerikanere	0.0826	venstre	-0.0846	joakim

#32		#33		#34		#35	
-0.4887	district	0.3737	side	-0.4987	side	-0.2452	teater
-0.2655	development	-0.3514	arter	-0.2596	anders	-0.1605	district
0.2011	xiàn	-0.2883	slægt	-0.2328	arter	0.1368	slam
-0.1656	nepals	0.1997	anders	-0.2039	joakim	0.1288	open
-0.1603	distrikter	0.1853	district	-0.1903	slægt	0.1259	byen
-0.1559	arter	0.1553	joakim	-0.1732	onkel	-0.1236	arter
-0.1436	nepal	0.1366	and	-0.1681	and	0.1148	vandt
-0.1421	committee	0.1324	onkel	-0.1390	stålanden	0.1143	grand
-0.1313	slægt	-0.1103	xiàn	-0.1347	jumbobog	-0.1114	station
-0.1279	distrikt	-0.1093	byen	-0.1305	serieforlaget	-0.1093	teatret
-0.1164	betegnet	0.1052	stålanden	-0.1217	egmont	-0.1089	slægt
-0.1159	lokalt	-0.1026	slægter	-0.1215	mouse	0.1012	etape
-0.1102	byen	0.1022	jumbobog	-0.1213	mickey	-0.0980	skuespiller
-0.1083	opdelt	0.1010	county	-0.1052	xiàn	0.0935	xiàn
0.0882	amtet	0.1000	development	0.1016	district	0.0932	von
-0.0878	bikas	0.0989	serieforlaget	0.0814	the	-0.0910	københavn
-0.0878	samiti	-0.0979	familie	-0.0794	slam	0.0895	plads
-0.0877	vdc	0.0959	almindelig	-0.0769	indhold	-0.0866	mod
-0.0873	gaun	0.0944	planten	-0.0695	grand	-0.0848	odense
-0.0868	zilla	0.0926	egmont	0.0669	county	-0.0837	division

#36		#37		#38		#39	
0.8085	instruktør	-0.2083	øen	-0.2596	stones	-0.2539	pastorat
0.1758	blom	0.1856	stones	-0.2480	slam	0.2479	herred
0.1027	lau	0.1848	fredede	-0.2469	jagger	0.2226	slam
0.0938	lauritzen	0.1750	richards	-0.2468	richards	-0.2135	historik
0.0927	schnedler	0.1749	jagger	-0.2132	rolling	-0.1916	diskussion
-0.0894	teater	-0.1720	pastorat	-0.1930	grand	-0.1845	flag
0.0867	holger	-0.1641	historik	-0.1882	open	-0.1781	stones
0.0862	dinesen	-0.1641	diskussion	-0.1477	mick	-0.1711	richards
0.0844	madsen	-0.1580	flag	-0.1430	øen	0.1711	grand
0.0807	ukendt	-0.1535	instruktør	0.1376	byen	-0.1710	jagger
0.0744	eduard	0.1521	rolling	-0.1286	atp	0.1678	open
-0.0743	olsen	0.1448	kommune	0.1207	fredede	0.1634	amt
-0.0698	film	0.1409	byen	-0.1191	tennisspiller	-0.1463	rolling
0.0677	davidsen	0.1261	fortidsminder	-0.1175	keith	0.1194	teater
0.0670	fredede	0.1054	mick	-0.1144	sangen	0.1133	atp
0.0656	kommune	-0.0984	slam	-0.1092	plads	0.1068	tennisspiller
0.0644	august	0.0963	jeg	-0.1046	pastorat	-0.1033	mick
0.0643	hjalmar	0.0893	sangen	0.1041	meter	-0.0980	etape
-0.0609	øen	-0.0852	areal	0.0927	metal	-0.0868	norsk
-0.0565	filmen	-0.0844	erne	0.0914	bandet	-0.0833	keith

Appendix E

English stop words from gensim.parsing.preprocessing.STOPWORDS

a, about, above, across, after, afterwards, again, against, all, almost, alone, along, already, also, although, always, am, among, amongst, amoungst, amount, an, and, another, any, anyhow, anyone, anything, anyway, anywhere, are, around, as, at, back, be, became, because, become, becomes, becoming, been, before, beforehand, behind, being, below, beside, besides, between, beyond, bill, both, bottom, but, by, call, can, cannot, cant, co, computer, con, could, couldnt, cry, de, describe, detail, did, do, doesn, done, down, due, during, each, eg, eight, either, eleven, else, elsewhere, empty, enough, etc, even, ever, every, everyone, everything, everywhere, except, few, fifteen, fify, fill, find, fire, first, five, for, former, formerly, forty, found, four, from, front, full, further, get, give, go, had, has, hasnt, have, he, hence, her, here, hereafter, hereby, herein, hereupon, hers, herself, him, himself, his, how, however, hundred, i, ie, if, in, inc, indeed, interest, into, is, it, its, itself, just, keep, kg, km, last, latter, latterly, least, less, ltd, made, many, may, me, meanwhile, might, mill, mine, more, moreover, most, mostly, move, much, must, my, myself, name, namely, neither, never, nevertheless, next, nine, no, nobody, none, noone, nor, not, nothing, now, nowhere, of, off, often, on, once, one, only, onto, or, other, others, otherwise, our, ours, ourselves, out, over, own, part, per, perhaps, please, put, quite, rather, re, really, regarding, same, see, seem, seemed, seeming, seems, serious, several, she, should, show, side, since, sincere, six, sixty, so, some, somehow, someone, something, sometime, sometimes, somewhere, still, such, system, take, ten, than, that, the, their, them, themselves, then, thence, there, thereafter, thereby, therefore, therein, thereupon, these, they, thick, thin, third, this, those, though, three, through, throughout, thru, thus, to, together, too, top, toward, towards, twelve, twenty, two, un, under, unless, until, up, upon, us, used, using, various, very, via, was, we, well, were, what, whatever, when, whence, whenever, where, whereafter, whereas, whereby, wherein, whereupon, wherever, whether, which, while, whither, who, whoever, whole, whom, whose, why, will, with, within, without, would, yet, you, your, yours, yourself, yourselves

Suppliment of english stop words from http://snowball.tartarus.org/algorithms/english/stop.txt

about, above, after, again, against, all, and, any, are, aren, because, been, before, being, below, between, both, but, can, cannot, could, couldn, did, didn, does, doesn, doing, don, down, during, each, few, float, for, from, further, had, hadn, has, hasn, have, haven, having, her, here, here, hers, herself, him, himself, his, how, how, into, isn, its, itself, let, more, most, mustn, myself, nor, not, off, once, only, other, ought, our, ours, ourselves, out, over, own, same, shan, she, she, she, she, should, shouldn, some, such, than, that, that, the, their, theirs, them, themselves, then, there, there, these, they, they, they, they, they, this, those, through, too, under, until, very, was, wasn, were, weren, what, what, when, when, where, where, which, while, who, who, whom, why, why, with, won, would, wouldn, you, your, yours, yourself, yourselves

Danish stop words from http://snowball.tartarus.org/algorithms/danish/stop.txt

alle, alt, anden, andet, andre, blandt, blev, blevet, blive, bliver, brug, dem, den, denne, der, deres, derfor, det, dette, dig, din, disse, dog, dvs, efter, eks, eller, end, erne, fik, findes, flere, for, fra, første, gik, ham, han, hans, har, havde, have, hende, hendes, her, hos, hun, hvad, hvis, hvor, ikke, ind, indtil, inkl, jeg, jer, kan, kun, kunne, ligger, man, mange, med, meget, mellem, men, mere, mia, mig, min, mindre, mine, mio, mit, mod, ned, nemt, noget, nogle, når, ofte, også, omkring, over, pct, på, samme, sammen, samt, selv, selvom, sig, sin, sine, sit, skal, skulle, som, store, står, svært, sådan, således, thi, til, under, var, ved, vil, ville, vor, være, været

Swedish stop words from http://snowball.tartarus.org/algorithms/swedish/stop.txt

alla, allt, att, av, blev, bli, blir, blivit, de, dem, den, denna, deras, dess, dessa, det, detta, dig, din, dina, ditt, du, dør, då, efter, ej, eller, en, er, era, ert, ett, från, för ha, hade, han, hans, har, henne, hennes, hon, honom, hur, hør, i, icke, ingen, inom, inte, jag, ju, kan, kunde, man, med, mellan, men, mig, min, mina, mitt, mot, mycket, ni, nu, nør, någon, något, några, och, om, oss, på, samma, sedan, sig, sin, sina, sitta, sjølv, skulle, som, så, sådan, sådana, sådant, till, under, upp, ut, utan, vad, var, vara, varför, varit, varje, vars, vart, vem, vi, vid, vilka, vilkas, vilken, vilket, vår, våra, vårt, øn, ør, åt, över

Norwegian stop words from http://snowball.tartarus.org/algorithms/norwegian/stop.txt

alle, at, av, bare, begge, ble, blei, bli, blir, blitt, både, båe, da, de, deg, dei, deim, deira, deires, dem, den, denne, der, dere, deres, det, dette, di, din, disse, ditt, du, dykk, dykkar, då, eg, ein, eit, eitt, eller, elles, en, enn, er, et, ett, etter, for, fordi, fra, før, ha, hadde, han, hans, har, hennar, henne, hennes, her, hjå, ho, hoe, honom, hoss, hossen, hun, hva, hvem, hver, hvilke, hvilken, hvis, hvor, hvordan, hvorfor, i, ikke, ikkje, ikkje, ingen, ingi, inkje, inn, inni, ja, jeg, kan, kom, korleis, korso, kun, kunne, kva, kvar, kvarhelst, kven, kvi, kvifor, man, mange, me, med, medan, meg, meget, mellom, men, mi, min, mine, mitt, mot, mykje, ned, no, noe, noen, noka, noko, nokon, nokor, nokre, nå, når, og, også, om, opp, oss, over, på, samme, seg, selv, si, si, sia, sidan, siden, sin, sine, sitt, sjøl, skal, skulle, slik, so, som, som, somme, somt, så, sånn, til, um, upp, ut, uten, var, vart, varte, ved, vere, verte, vi, vil, ville, vore, vors, vort, vår, være, vært, å

Appendix F

wiki_da_40t_lsi (92070a)

#0		#1		#2		#3	
0.0838	henvisninger	-0.2664	sogneportalen	-0.1748	fodboldspiller	0.1292	landshold
0.0834	eksterne	-0.2641	stednavne	-0.1647	født	0.1260	kampe
0.0784	dansk	-0.2632	ejerlav	-0.1606	landshold	0.1248	fodboldspiller
0.0776	født	-0.2619	autoriserede	-0.1568	kampe	0.1114	ligger
0.0763	kilder	-0.2569	flg	-0.1335	spillede	0.1066	spillerinfo
0.0709	del	-0.2498	sognet	-0.1299	klubben	0.1018	klubben
0.0707	ligger	-0.2497	bebyggelse	-0.1260	spillerinfo	-0.0981	dansk
0.0699	tidligere	-0.2382	provsti	-0.1257	spiller	-0.0979	søn
0.0683	senere	-0.2365	herred	-0.1144	vandt	0.0960	nord
0.0665	danske	-0.2339	sogn	-0.1056	spillet	-0.0943	gift
0.0650	danmark	-0.2313	stift	-0.0922	debuterede	0.0928	syd
0.0639	sammen	-0.2216	amt	-0.0897	klub	0.0917	spiller
0.0616	store	-0.1982	kommune	-0.0874	titler	0.0895	indbyggere
0.0595	andet	-0.1751	kirke	-0.0853	cup	0.0882	delstat
0.0590	dag	-0.1494	areal	-0.0828	karriere	0.0860	øst
0.0589	findes	-0.1419	ligger	-0.0806	mål	0.0855	vest
0.0587	blevet	-0.1216	findes	-0.0804	league	-0.0848	københavn
0.0585	kendt	-0.1053	kilder	-0.0790	hold	0.0804	landkreis
0.0579	tre	-0.0909	region	-0.0709	scorede	-0.0803	datter
0.0572	københavn	-0.0828	vandareal	-0.0689	noteret	0.0780	byen

#4		#5		#6		#7	
-0.1156	ligger	0.1477	månekratere	-0.1190	album	0.1411	bladene
-0.1116	nord	0.1472	iau	-0.1032	amerikansk	0.1338	blomsterne
-0.1093	delstat	0.1466	nedslagskrater	-0.1031	delstat	0.1228	frugterne
-0.1069	indbyggere	0.1461	kratere	-0.0953	albummet	0.1158	synlige
-0.1038	syd	0.1403	månens	-0.0938	landkreis	0.1133	arter
-0.1037	landkreis	0.1399	navigation	-0.0914	band	0.1070	hjemsted
-0.1008	byen	0.1392	astronomiske	-0.0898	diskografi	0.1067	blomster
-0.1002	kommunen	0.1337	måneatlas	0.0882	fodboldspiller	-0.1038	cest
-0.0958	vest	0.1295	månen	-0.0859	udgivet	0.1006	træk
-0.0956	øst	0.1270	karakteristika	0.0849	landshold	-0.1005	artiklen
-0.0945	kommune	0.1208	omgivelser	-0.0846	bandet	0.1004	udbredt
-0.0919	beliggende	0.1191	hovedkrateret	-0.0834	indbyggere	0.0938	almindelig
-0.0906	geografi	0.1191	satellitkratere	-0.0823	geografi	0.0927	rodnettet
-0.0822	bayern	0.1187	bibliografi	-0.0812	guitar	0.0876	blomstringen
0.0810	sogneportalen	0.1128	satellitter	-0.0802	byen	0.0875	planten
0.0810	autoriserede	0.1121	konvention	-0.0785	kommunen	0.0871	størrelse
0.0787	ejerlav	0.1120	midte	0.0765	kampe	0.0870	blade
0.0785	stednavne	0.1069	bogstav	-0.0753	rock	-0.0867	artikel
-0.0776	tyske	0.1063	tildelt	-0.0748	new	-0.0854	cet
-0.0768	københavn	0.1060	identificere	-0.0745	usa	-0.0846	wikipedia

#8		#9		#10		#11	
0.1044	bladene	-0.1676	style	0.1560	landkreis	0.1513	timestamp
0.0996	dansk	-0.1611	align	0.1484	delstat	0.1490	uformateret
0.0985	blomsterne	0.1577	fodboldspiller	0.1397	bayern	0.1474	metadata
0.0904	frugterne	0.1403	spillerinfo	-0.1376	hjemmeside	0.1460	insee
0.0878	kommune	0.1323	landshold	0.1152	geografi	0.1455	maritimes
0.0845	synlige	-0.1251	border	0.1127	tyske	0.1446	ugyldig
0.0844	arter	-0.1209	margin	-0.0858	beliggende	0.1408	alpes
0.0823	hjemsted	-0.1204	class	0.0834	skabelon	0.1352	nøgle
0.0800	blomster	-0.1201	dato	0.0817	kommunen	0.1340	departement
0.0754	sidder	-0.1198	right	0.0771	default	0.1265	switch
0.0711	danmark	-0.1091	width	0.0770	switch	0.1263	bemærkes
0.0709	almindelig	-0.1036	left	0.0705	søn	0.1232	default
0.0702	består	-0.0995	cellpadding	0.0685	landsbyer	0.1227	angive
0.0697	danske	0.0983	klubben	0.0678	timestamp	0.1142	kommunens
0.0693	hjemmeside	-0.0979	cellspacing	-0.0677	dag	0.1142	kommunerne
0.0688	rodnettet	-0.0942	hold	0.0668	uformateret	0.1119	indsætte
0.0676	udbredt	-0.0917	size	-0.0666	danmarks	0.1099	kode
0.0671	blade	-0.0901	text	-0.0661	grundlagt	0.0954	resultatet
0.0655	planten	-0.0891	font	0.0659	insee	0.0930	eksempler
0.0653	blomstringen	-0.0887	resultater	0.0654	metadata	0.0918	franske

#12		#13		#14		#15	
0.1447	medlem	-0.2121	align	-0.1846	film	0.1806	landkreis
0.1216	formand	-0.2110	border	-0.1800	skuespiller	0.1716	bayern
0.1178	politiker	-0.2003	style	0.1510	album	0.1406	delstat
-0.1118	opført	-0.1690	cellpadding	-0.1382	filmografi	0.1220	geografi
0.1025	universitet	-0.1662	cellspacing	0.1311	albummet	-0.1140	cest
0.0993	delstat	-0.1487	right	0.1248	guitar	0.1089	tyske

0.0965 uddannet	-0.1483 size	-0.1237 filmen	-0.0956 cet
0.0955 landkreis	-0.1461 font	0.1225 band	0.0917 kommunen
0.0951 født	-0.1418 margin	-0.1220 amerikansk	-0.0907 amjaabc
0.0898 valgt	-0.1406 width	0.1211 bandet	-0.0901 artiklen
-0.0873 timestamp	-0.1284 left	0.1188 trommer	-0.0864 indbyggere
-0.0860 uformateret	0.1279 landkreis	-0.1158 instrueret	-0.0861 diskussion
0.0850 cand	-0.1248 background	0.1067 diskografi	0.0831 bebyggelser
-0.0842 insee	-0.1227 class	0.1066 udgivet	-0.0822 historik
-0.0842 metadata	-0.1090 colspan	0.0996 bas	0.0820 landsbyer
-0.0838 maritimes	0.1083 bayern	0.0950 vokal	0.0816 style
-0.0832 ugyldig	-0.1046 text	-0.0905 medvirkende	-0.0787 arne
0.0827 folketinget	0.1041 delstat	-0.0894 serien	-0.0783 pugilist
-0.0826 frederik	-0.1022 spillerinfo	0.0838 metal	0.0777 inddeling
-0.0821 ejere	-0.0985 fodboldspiller	-0.0811 teater	0.0774 oberbayern

#16	**#17**	**#18**	**#19**
0.1237 landkreis	-0.1010 eksterne	-0.1615 atlet	0.1557 atlet
0.1145 bayern	-0.1006 omdr	-0.1479 meter	0.1499 mesterskaber
-0.0882 forfatter	-0.0994 model	-0.1426 mesterskaber	0.1384 daf
0.0874 delstat	-0.0957 cylindre	-0.1418 daf	0.1249 areal
-0.0837 spillerinfo	-0.0941 henvisninger	0.1290 ejere	0.1159 meter
-0.0807 tysk	-0.0922 historik	0.1149 hektar	0.1134 tal
-0.0784 svensk	-0.0920 hjemmeside	0.1035 gods	-0.1010 kommune
-0.0767 indbyggere	-0.0889 modellen	0.1010 hovedbygningen	-0.0998 kirke
-0.0762 areal	-0.0871 ventiler	0.0974 gården	0.0988 enheder
0.0757 kommunen	-0.0850 bygget	-0.0937 vandt	-0.0979 kirken
-0.0753 administrative	-0.0808 benzinmotorer	-0.0914 tal	0.0953 administrative
-0.0736 komponist	-0.0806 jpg	-0.0882 rekorder	-0.0923 beliggende
-0.0716 værker	-0.0748 diskussion	-0.0878 født	-0.0888 korttilkirken
-0.0714 frankrig	0.0746 areal	-0.0825 rekord	0.0852 amter
-0.0700 sverige	0.0733 indbyggere	0.0803 parti	0.0822 rekorder
-0.0695 sprog	-0.0730 motorer	0.0802 trap	0.0771 rekord
-0.0693 fransk	-0.0729 svensk	-0.0758 deltog	0.0763 danske
-0.0692 landshold	-0.0729 tekniske	0.0757 hjemmeside	-0.0754 region
-0.0692 norge	-0.0718 motor	-0.0756 kirke	0.0751 xiàn
0.0681 geografi	-0.0708 specifikationer	-0.0731 profil	-0.0733 kirker

#20	**#21**	**#22**	**#23**
-0.1479 hjemmeside	-0.1855 mesterskaber	0.1785 historik	-0.2460 fortidsminder
0.1272 politiker	-0.1816 atlet	0.1760 norsk	-0.2211 fredede
0.1202 kommune	-0.1593 daf	0.1623 svensk	-0.1710 bygger
-0.1172 grundlagt	-0.1201 tal	0.1263 diskussion	-0.1696 listen
0.1145 spillerinfo	-0.1055 meter	0.1203 norge	-0.1678 data
0.1094 kilder	-0.0870 rekorder	-0.1079 administrative	-0.1478 bygninger
-0.1036 klubbens	0.0852 administrative	0.1043 sverige	-0.1360 liste
-0.0956 fodboldklub	-0.0851 rekord	0.1033 skuespiller	-0.1260 kommune
-0.0956 henvisning	0.0850 omdr	-0.0984 enheder	-0.1230 viser
-0.0937 ekstern	-0.0833 medlem	-0.0913 folketællingen	0.0972 korttilkirken
0.0932 valgt	-0.0813 atletik	-0.0907 distrikter	0.0946 kirken
0.0912 sogn	0.0810 cylindre	0.0821 filmografi	0.0849 kirke
0.0900 fortidsminder	0.0779 enheder	-0.0787 mesterskaber	-0.0808 kilder
-0.0898 klubben	-0.0732 personlige	-0.0786 amtet	0.0695 jpg
-0.0890 officielle	0.0732 ventiler	-0.0779 atlet	0.0686 parti
0.0886 folketinget	0.0723 model	0.0737 teater	-0.0683 kraks
0.0851 medlem	0.0706 modellen	-0.0715 samiti	-0.0664 hektar
0.0836 født	0.0701 københavn	-0.0715 bikas	0.0654 omdr
0.0822 data	-0.0684 vandt	-0.0715 gaun	0.0639 tallet
0.0820 landshold	0.0679 areal	-0.0715 ilakaer	-0.0619 fylke

#24	**#25**	**#26**	**#27**
0.2808 fortidsminder	0.1647 fortidsminder	-0.1422 hustand	0.1776 historik
0.2501 fredede	0.1458 fredede	-0.1361 norsk	-0.1374 fortidsminder
0.1959 bygger	0.1243 liste	-0.1298 folketællingen	0.1363 norsk
0.1951 listen	0.1145 listen	-0.1246 tilfældende	0.1269 diskussion
0.1927 data	0.1103 bygger	-0.1242 afroamerikanere	-0.1193 fredede
0.1899 historik	0.1082 bygninger	-0.1225 husstande	0.1123 svensk
0.1556 norsk	0.1047 projektsiden	-0.1221 ægtepar	-0.1000 amtet
0.1555 bygninger	0.1036 huskeliste	-0.1215 svensk	-0.0992 listen
0.1518 svensk	0.0995 wikiprojekt	-0.1208 enlig	-0.0967 data
0.1498 liste	0.0982 data	-0.1199 beboer	-0.0947 bygger
0.1424 viser	0.0978 vurderingen	-0.1160 historik	0.0942 gaun
0.1321 diskussion	-0.0965 cellpadding	-0.1126 befolkningens	0.0942 ilakaer
0.1214 kommune	-0.0963 cellspacing	-0.1112 boende	0.0942 jilla
-0.0879 spillerinfo	0.0953 koordinere	-0.1097 county	0.0942 samiti
0.0810 cellspacing	0.0882 vurderet	-0.1077 enlige	0.0942 bikas
0.0806 cellpadding	0.0879 redigere	-0.1033 etniske	0.0940 zilla
-0.0717 landshold	-0.0872 county	-0.1017 kvindelig	0.0937 sundhedsprofil
-0.0716 klubplan	-0.0869 historik	-0.0977 demografi	0.0935 vdc
0.0713 statistik	-0.0865 right	-0.0967 familier	0.0915 municipalities
0.0693 kirker	0.0845 diskussionsside	-0.0938 amtet	0.0896 nepals

#28		#29		#30		#31	
0.1543	norsk	-0.1300	sogn	0.1152	amter	0.1260	sogne
0.1476	historik	0.1035	fortidsminder	0.1080	xiàn	-0.1224	arter
0.1059	diskussion	-0.0997	indbyggere	0.1006	amtet	0.1065	kommunen
0.1045	rigsvej	0.0977	hustand	0.0964	filmografi	0.1033	sogn
0.1021	xiàn	0.0967	folketællingen	0.0932	kinas	-0.1032	systema
0.0992	svensk	-0.0941	region	0.0930	rigsvej	-0.1022	naturae
0.0932	kinas	0.0939	fredede	0.0862	hanzi	-0.0965	classification
-0.0921	filmografi	0.0918	bygninger	0.0848	pinyin	-0.0960	taxon
-0.0903	region	0.0902	delstat	-0.0834	medvirkende	-0.0953	slægt
0.0903	amter	0.0861	tilfældende	0.0819	skuespiller	0.0946	sognene
-0.0832	teater	0.0861	afroamerikanere	0.0807	jurisdiktion	-0.0937	klassifikation
0.0821	trafik	0.0846	ægtepar	0.0799	sogn	-0.0927	politiker
0.0813	beliggenhed	0.0837	enlig	-0.0781	byer	0.0834	landshold
0.0809	politiker	0.0836	beboer	-0.0765	fylke	0.0832	historik
0.0798	hanzi	0.0828	husstande	0.0763	amt	0.0818	klubplan
-0.0797	indbyggere	-0.0797	landsby	0.0756	historik	0.0800	norge
-0.0794	mesterskaber	-0.0788	trap	0.0756	indb	0.0798	kommune
0.0785	pinyin	0.0769	befolkningens	0.0748	præfekturet	0.0754	fakta
0.0737	jurisdiktion	0.0766	korttilkirken	-0.0739	øst	0.0743	nørre
0.0733	præfekturet	0.0758	boende	0.0734	sogne	-0.0730	slægten

#32		#33		#34		#35	
0.1482	komponist	-0.1783	olympiske	-0.2089	arter	-0.1179	olympiske
-0.1423	historik	-0.1692	lege	-0.1587	systema	-0.1170	lege
-0.1336	arter	-0.1677	slam	-0.1571	naturae	0.1082	region
-0.1124	norsk	-0.1674	tennisspiller	-0.1490	classification	0.1022	spillerinfo
0.1082	olympiske	-0.1543	atp	-0.1478	taxon	-0.0994	sportsreference
0.1019	lege	-0.1411	sportsreference	-0.1473	slægt	-0.0988	spillede
-0.1016	slægt	-0.1375	open	-0.1339	klassifikation	-0.0987	deltog
-0.0960	diskussion	-0.1309	grand	-0.1149	olympiske	-0.0958	profil
-0.0931	systema	-0.1301	singlerækkerne	0.1101	slam	0.0950	noteret
-0.0923	naturae	-0.1196	bronzemedalje	-0.1061	lege	-0.0902	bronzemedalje
0.0907	tysk	-0.1102	professionel	0.1036	tennisspiller	-0.0859	ejere
-0.0897	norge	-0.1087	sommer	0.1006	atp	-0.0850	scorede
-0.0894	fylke	-0.1076	doubletitler	0.0965	ejere	-0.0837	hektar
-0.0871	classification	-0.1064	vandt	-0.0954	slægter	0.0817	provstiportalen
-0.0870	taxon	-0.1064	deltog	0.0930	hektar	0.0802	pastorater
-0.0828	klassifikation	-0.1060	spillerinfo	-0.0922	sportsreference	0.0800	provstiet
-0.0796	kulturnett	-0.1021	profil	0.0898	open	0.0793	sogne
-0.0791	mesterskaber	-0.1002	placering	-0.0865	danmark	0.0789	klubben
-0.0771	arkiv	0.0971	daf	0.0846	grand	0.0772	tennisspiller
0.0754	sportsreference	-0.0939	single	-0.0843	udbredt	0.0761	pastorat

#36		#37		#38		#39	
-0.1843	slam	0.1718	region	-0.3053	jumbobog	0.1337	historik
-0.1796	tennisspiller	0.1563	historik	-0.3031	stålanden	-0.1211	henvise
-0.1727	atp	-0.1356	grænser	-0.3027	serieforlaget	0.1085	flyttemand
-0.1475	grand	-0.1318	sogne	-0.2812	egmont	0.1033	vicevært
-0.1472	open	-0.1289	fylke	-0.2651	mouse	0.1022	reichhardt
-0.1408	singlerækkerne	-0.1273	kulturnett	-0.2602	joakim	0.1021	virkner
0.1384	olympiske	0.1213	beliggende	-0.2582	mickey	-0.1008	fylke
-0.1356	professionel	0.1212	jumbobog	-0.2221	onkel	-0.0971	norge
0.1312	lege	0.1202	stålanden	-0.1856	indhold	0.0942	løwert
-0.1297	arter	0.1201	serieforlaget	-0.1756	anders	-0.0920	kulturnett
-0.1244	karriere	0.1117	egmont	-0.1196	udgivet	0.0901	region
-0.1188	doubletitler	0.1063	mouse	-0.1157	gearløs	0.0884	christianshavn
0.1164	sportsreference	-0.1051	øst	-0.1152	vims	0.0877	karla
0.1129	profil	0.1032	mickey	-0.1145	fætter	-0.0871	kommunen
0.1041	com	-0.1024	norge	-0.0963	bjørne	0.0854	udsen
-0.1030	højeste	0.1018	joakim	0.0941	historik	0.0851	dyrehandler
-0.1014	opnåede	-0.0980	kommuner	-0.0853	banden	0.0832	rathnov
-0.0992	placering	-0.0942	vest	0.0796	region	0.0816	lege
0.0956	klubben	-0.0936	provstiet	-0.0734	trick	-0.0809	bebyggelse
-0.0924	australian	-0.0927	korttilkirken	-0.0643	georg	0.0806	sportsreference

Appendix G

Press release used for testing the topics of LSA.

40 år med stjernestunder i Tinghallen

30. marts 2009

Tinghallen i Viborg har i år eksisteret i 40 år, og det markeres med udgivelse af en jubilæumsbog med titlen "Her hvor TING'ene sker"

Dolly Parton, Bryan Adams, Chuck Berry, Kris Kristofferson, Kenny Rogers. Listen er lang over de rock-, pop- og countrystjerner, som inden for de senere år har optrådt i eller ved Tinghallen. Men det er langt fra noget nyt, at internationalt kendte bands og solister gæster kongres- og musikhuset i Viborg, fremgår det af den jubilæumsbog, som Tinghallen udsender i forbindelse med sit 40 års jubilæum.
"Her hvor TING'ene sker" hedder bogen, som bl.a. beretter om de mange store navne, der i de første år efter åbningen i februar 1969 gæstede Tinghallen. Ofte som det eneste sted i provinsen, nogle gange som det eneste sted i Danmark. Det var således ikke uden grund, at Viborg i disse år blev kendt som provinsens måske førende koncertby. Steppenwolf, The Move, Nice, Procol Harum, Ten Years After, The Small Faces og John Mayall er rocknavne, der stadig bringer søde minder frem hos mange. Men også popdrenge og –piger i den mere bløde ende kom på besøg. Fx gav den nyligt afdøde tyske schlagerkonge Freddy Breck sin første danske koncert i netop Tinghallen.
Bogen nævner også farcen "Stamherren", som i 1970'erne blev omplantet fra ABC Teatret i København til Tinghallen. Her sørgede Dirch Passer, Jørgen Ryg, Ulf Pilgaard og andre af tidens kendteste komikere for ikke færre end 16 fyldte huse. I samme periode lykkedes det at få Nationalmuseet til at flytte sin store udstilling "Kina – Riget i midten" til Tinghallen. I de seks uger, udstillingen var i Tinghallen, blev den set af 56.000 mennesker fra hele landet.

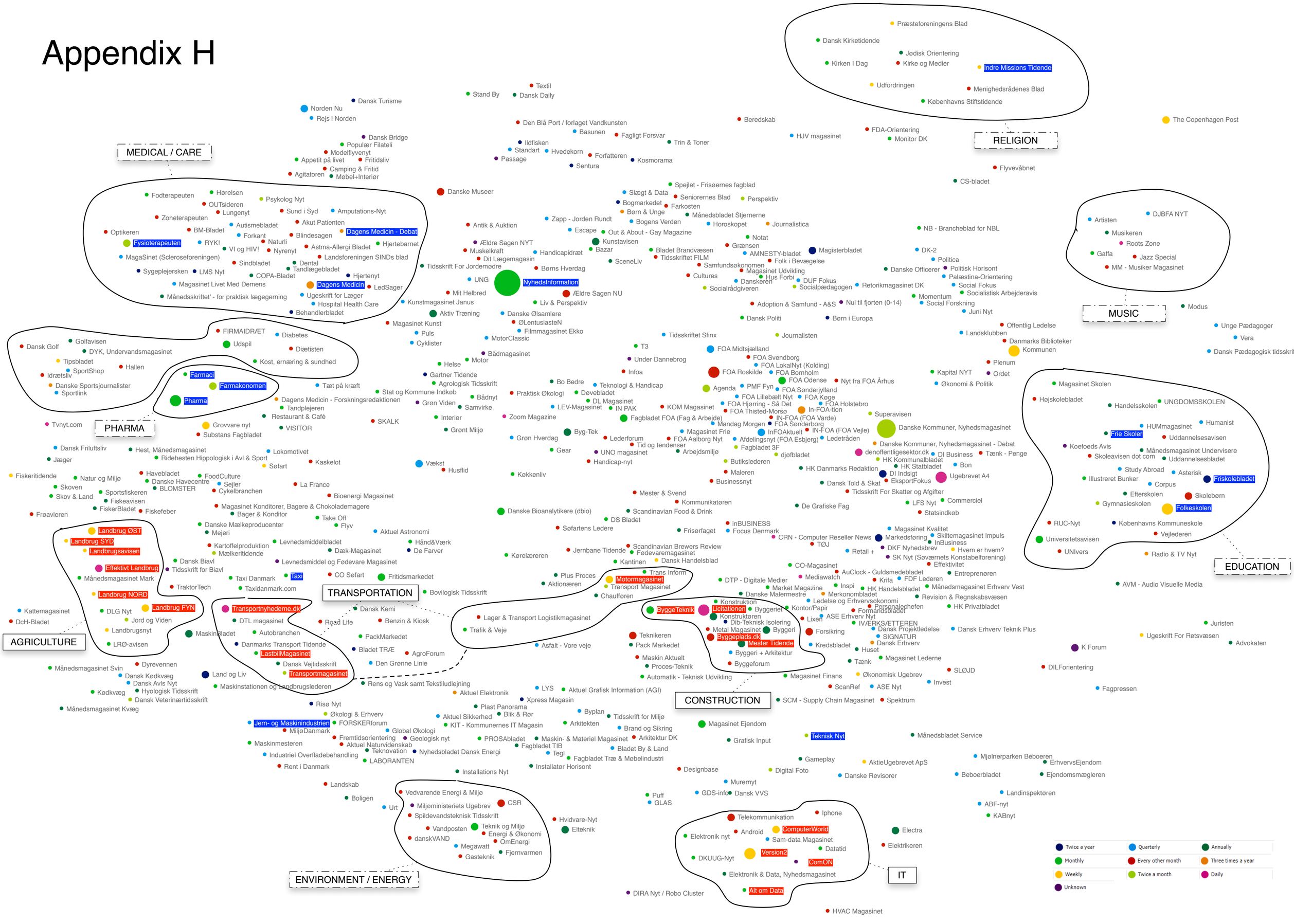
Omdiskuteret fra starten
En del af midlerne til at opføre Tinghallen blev skaffet gennem aktietegning blandt lokale borgere og virksomheder. Mange var således glade for, at Viborg nu endelig havde fået en stor hal, der fra starten blev brugt til både koncerter, udstillinger, fester og sport. Men Tinghallen havde også sine modstandere, der ikke mindst var utilfredse med byrådets tilsagn om at ville dække hallens eventuelle driftsunderskud.
Især måtte erhvervsmanden og lokalpolitikeren Axel Brøndum stå for skud. Brøndum var om nogen initiativtageren til Tinghallen, og han var i de første vanskelige år formand for hallens bestyrelse. Han talte ligefrem om to fronter i Viborg: De, der var for Tinghallen, og de, der var imod. Efter Brøndums mening var viborgenserne ikke flinke nok til at bakke Tinghallen op, og han henviste i den forbindelse bl.a. til Holstebrohallen, " hvor tilsyneladende hele byen er stamgæster, idet man slutter op om hvert eneste arrangement", som Brøndum udtrykte det.

Rivende udvikling
Som det fremgår af jubilæumsbogen, har Tinghallen i den 15-20 år været inde i en rivende udvikling. Både arrangementsmæssigt og med hensyn til hallens fysiske rammer.
I 1989 fik Tinghallen et nyt stort køkken. I 1993 tog man en ny stor foyer samt en større og moderniseret restaurantafdeling i brug. Tre år senere blev siddekomforten i hallen væsentligt forbedret, bl.a. med et nyt system af teleskopsæder. Og i 2004 kom så det store scenetårn med alskens teknisk udstyr.
Bogen fortæller også om de aktuelle planer om en ny multiarena med et tilhørende stort hotel. Et kæmpeprojekt, der tænkes gennemført i samarbejde mellem Tinghallen, Viborg HK, Viborg Kommune og private investorer, og som også indebærer, at den nuværende Tinghal skal bygges om til et decideret musik- og teaterhus med faste stole.
Jubilæumsbogen om Tinghallen er skrevet af journalist Ronald Nielsen, MedieKontoret.

Yderligere oplysninger:
Direktør Tommy S. Pedersen, Tinghallen, tlf. 86 62 61 00.

Til orientering:
Jubilæumsbogen vil blive fremsendt til redaktionen pr. post.

Appendix H


RELIGION
Præsteforeningens Blad
Dansk Kirketidende
Jødisk Orientering
Kirken I Dag
Kirke og Medier
Indre Missions Tidende
Udfordringen
Menighedsrådenes Blad
Københavns Stiftstidende
MEDICAL / CARE
Fodterapeuten
Høreisen
Psykolog Nyt
OUTsideren
Lungenyt
Sund i Syd
Amputations-Nyt
Zoneterapeuten
Akut Patienten
Optikeren
BM-Bladet
Autismebladet
Forkant
Blindesagen
Dagens Medicin - Debat
Fysioterapeuten
RYK!
Naturli
VI og HIV!
Nyrenyt
Astma-Allergi Bladet
Hjertebarnet
MagaSinet (Scleroseforeningen)
Landsforeningen SINDs blad
Sindbladet
Dental
Sygeplejersken
LMS Nyt
Tandlægebladet
COPA-Bladet
Hjertenyt
Magasinet Livet Med Demens
Dagens Medicin
LedSager
Månedsskriftet' - for praktisk lægegerning
Ugeskrift for Læger
Hospital Health Care
Behandlerbladet
Dansk Turisme
Norden Nu
Rejs i Norden
Dansk Bridge
Populær Filateli
Modelflyvenyt
Appetit på livet
Fritidsliv
Camping & Fritid
Agitatoren
Møbel+Interiør
Textil
Stand By
Dansk Daily
Den Blå Port / forlaget Vandkunsten
Basunen
Fagligt Forsvar
Ildfisken
Standart
Hvedekorn
Trin & Toner
Passage
Forfatteren
Kosmorama
Sentura
Beredskab
HJV magasinet
FDA-Orientering
Monitor DK
The Copenhagen Post
Flyvevåbnet
CS-bladet
Danske Museer
Spejlet - Frisøernes fagblad
Slægt & Data
Seniorernes Blad
Perspektiv
Bogmarkedet
Farkosten
Børn & Unge
Månedsbladet Stjernerne
Zapp - Jorden Rundt
Bogens Verden
Horoskopet
Journalistica
Antik & Auktion
Escape
Out & About - Gay Magazine
Ældre Sagen NYT
Kunstavisen
Notat
Muskelkraft
Handicapidræt
Bazar
Grænsen
Dit Lægemagasin
Bladet Brandvæsen
AMNESTY-bladet
Magisterbladet
Tidsskrift For Jordemødre
SceneLiv
Tidsskriftet FILM
Folk i Bevægelse
Børns Hverdag
Samfundsøkonomen
Magasinet Udvikling
UNG
NyhedsInformation
Cultures
Danskeren
Hus Forbi
DUF Fokus
Socialrådgiveren
Socialpædagogen
Retorikmagasinet DK
Mit Helbred
Ældre Sagen NU
Kunstmagasinet Janus
Liv & Perspektiv
Aktiv Træning
Danske Ølsamlere
ØLentusiasteN
Magasinet Kunst
Filmmagasinet Ekko
Puls
MotorClassic
Cyklister
Tidsskriftet Sfinx
Journalisten
T3
FOA Midtsjælland
Bådmagasinet
Under Dannebrog
FOA Svendborg
Motor
FOA LokalNyt (Kolding)
Helse
Infoa
FOA Roskilde
FOA Bornholm
Gartner Tidende
FOA Odense
Nyt fra FOA Århus
Agrologisk Tidsskrift
Bo Bedre
Teknologi & Handicap
Agenda
PMF Fyn
Stat og Kommune Indkøb
Praktisk Økologi
Døvebladet
FOA Søndjylland
Bådnyt
DL Magasinet
FOA Lillebælt Nyt
FOA Køge
Grøn Viden
LEV-Magasinet
IN PAK
KOM Magasinet
FOA Hjørring - Så Det
FOA Holstebro
Samvirke
FOA Thisted-Morsø
In-FOA-tion
Interiør
Zoom Magazine
Fagbladet FOA (Fag & Arbejde)
IN-FOA (FOA Varde)
Grønt Miljø
Mandag Morgen
FOA Sønderborg
Byg-Tek
Magasinet Frie
InFOAktuelt
IN-FOA (FOA Vejle)
Grøn Hverdag
Lederforum
FOA Aalborg Nyt
Afdelingsnyt (FOA Esbjerg)
Ledetråden
Gear
Tid og tendenser
Fagbladet 3F
UNO magasinet
Arbejdsmiljø
djøfbladet
Handicap-nyt
Butiklederen
HK Danmarks Redaktion
HK Statbladet
Bon
Køkkenliv
Maleren
DI Indsigt
Ugebrevet A4
Businessnyt
EksportFokus
Dansk Told & Skat
Tidsskrift For Skatter og Afgifter
Mester & Svend
LFS Nyt
Commerciel
Kommunikatøren
De Grafiske Fag
Danske Bioanalytikere (dbio)
Scandinavian Food & Drink
Statsindkøb
DS Bladet
Søfartens Ledere
inBUSINESS
Focus Denmark
Frisørfaget
Magasinet Kvalitet
CRN - Computer Reseller News
Markedsføring
Skiltemagasinet Impuls
TØJ
InBusiness
Jernbane Tidende
Scandinavian Brewers Review
Retail +
DKF Nyhedsbrev
Hvem er hvem?
Kørelæreren
Fødevaremagasinet
SK Nyt (Søværnets Konstabelforening)
Kantinen
Dansk Handelsblad
CO-Magasinet
Effektivitet
Plus Proces
Trans Inform
Mediawatch
AuClock - Guldsmedebladet
Entreprenøren
Aktionæren
Motormagasinet
DTP - Digitale Medier
Krifa
FDF Lederen
Transport Magasinet
Market Magazine
Inspi
HK Handelsbladet
Månedsmagasinet Erhverv Vest
Chaufføren
Danske Malermestre
Merkonombladet
Revision & Regnskabsvæsen
Konstruktion
Ledelse og Erhvervsøkonomi
HK Privatbladet
Byggeriet
Kontor/Papir
Personalechefen
Formandsbladet
Konstruktøren
Lixen
ASE Erhverv Nyt
Dib-Teknisk Isolering
IVÆRKSÆTTEREN
Metal Magasinet
Byggeri
Forsikring
Dansk Projektledelse
Dansk Erhverv Teknik Plus
Byggepladsdk
SIGNATUR
Mester Tidende
Kredsbladet
Dansk Erhverv
Byggeri + Arkitektur
Huset
Byggeforum
Tænk
Magasinet Lederne
Maskin Aktuelt
Proces-Teknik
Økonomisk Ugebrev
SLØJD
Automatik - Teknisk Udvikling
Magasinet Finans
Invest
ScanRef
ASE Nyt
Aktuel Elektronik
LYS
Aktuel Grafisk Information (AGI)
Fagpressen
Xpress Magasin
SCM - Supply Chain Magasinet
Spektrum
Plast Panorama
Byplan
Tidsskrift for Miljø
Økologi & Erhverv
Aktuel Sikkerhed
Blik & Rør
Arkitekten
Brand og Sikring
Magasinet Ejendom
Grafisk Input
Teknisk Nyt
Månedsbladet Service
Gameplay
AktieUgebrevet ApS
Designbase
Digital Foto
Danske Revisorer
Murernyt
GDS-info
Dansk VVS
Puff
GLAS
Telekommunikation
Iphone
Android
ComputerWorld
Elektronik nyt
Sam-data Magasinet
Electra
Elektrikeren
Datatid
DKUUG-Nyt
Version2
ComON
Elektronik & Data, Nyhedsmagasinet
Alt om Data
DIRA Nyt / Robo Cluster
HVAC Magasinet
Danske Kommuner, Nyhedsmagasinet
Superavisen
Danske Kommuner, Nyhedsmagasinet - Debat
denoffentligesektor.dk
DI Business
Tænk - Penge
HK Kommunalbladet
Danske Officerer
Politisk Horisont
Politica
DK-2
NB - Brancheblad for NBL
Palæstina-Orientering
Social Fokus
Socialistisk Arbejderavis
Momentum
Social Forskning
Juni Nyt
Adoption & Samfund - A&S
Nul til fjorten (0-14)
Dansk Politi
Børn i Europa
Offentlig Ledelse
Landsklubben
Danmarks Biblioteker
Kommunen
Plenum
Ordet
Kapital NYT
Økonomi & Politik
Modus
Unge Pædagoger
Vera
Dansk Pædagogisk tidsskrift
MUSIC
Artisten
DJBFA NYT
Musikeren
Roots Zone
Gaffa
Jazz Special
MM - Musiker Magasinet
EDUCATION
Magasinet Skolen
Højskolebladet
Handelsskolen
UNGDOMSSKOLEN
HUMmagasinet
Humanist
Frie Skoler
Uddannelsesavisen
Koefoeds Avis
Månedsmagasinet Undervisere
Skoleavisen dot com
Uddannelsesbladet
Study Abroad
Asterisk
Illustreret Bunker
Friskolebladet
Corpus
Efterskolen
Skolebørn
Gymnasieskolen
Folkeskolen
RUC-Nyt
Københavns Kommuneskole
Universitetsavisen
Vejlederen
UNIvers
Radio & TV Nyt
AVM - Audio Visuelle Media
Juristen
Ugeskrift For Retsvæsen
Advokaten
K Forum
DILForientering
Fagpressen
PHARMA
FIRMAIDRÆT
Diabetes
Dansk Golf
Golfavisen
Udspil
Diætisten
DYK, Undervandsmagasinet
Tipsbladet
Kost, ernæring & sundhed
Hallen
SportShop
Idrætsliv
Danske Sportsjournalister
Sportlink
Farmaci
Farmakonomen
Pharma
Tæt på kræft
Dagens Medicin - Forskningsredaktionen
Tandplejeren
Restaurant & Café
Tvnyt.com
Grovvare nyt
VISITOR
Substans Fagbladet
SKALK
Dansk Friluftsliv
Hest, Månedsmagasinet
Lokomotivet
Jæger
Ridehesten Hippologisk i Avl & Sport
Kaskelot
Søfart
Vækst
Husflid
Fiskeritidende
Natur og Miljø
Havebladet
FoodCulture
Skoven
Danske Havecentre
Sejler
BLOMSTER
La France
Skov & Land
Cykelbranchen
Sportsfiskeren
Bioenergi Magasinet
Fiskeavisen
FiskerBladet
Fiskefeber
Magasinet Konditorer, Bagere & Chokoladem agere
Frøavleren
Bager & Konditor
Take Off
Danske Mælkeproducenter
Flyv
Mejeri
Aktuel Astronomi
Kartoffelproduktion
Levnedsmiddelbladet
Hånd&Værk
Mælkeritidende
Dæk-Magasinet
De Farver
Landbrug ØST
Landbrug SYD
Landbrugsavisen
Levnedsmiddel og Fødevare Magasinet
Effektivt Landbrug
Dansk Biavl
Tidsskrift for Biavl
Månedsmagasinet Mark
Taxi Danmark
Taxi
CO Søfart
Fritidsmarkedet
TraktorTech
Taxidanmark.com
Landbrug NORD
Bovilogisk Tidsskrift
TRANSPORTATION
Landbrug FYN
Kattemagasinet
Transportnyhederne.dk
Dansk Kemi
DLG Nyt
DTL magasinet
Road Life
Benzin & Kiosk
DcH-Bladet
Jord og Viden
Autobranchen
PackMarkedet
Landbrugsnyt
MaskinBladet
Bladet TRÆ
Danmarks Transport Tidende
AgroForum
LRØ-avisen
LastbilMagasinet
Den Grønne Linie
Dansk Vejtidsskrift
Dyrevennen
Transportmagasinet
Land og Liv
Rens og Vask samt Tekstiludlejning
Månedsmagasinet Svin
Dansk Kødkvæg
Maskinstationen og Landbrugslederen
Dansk Avls Nyt
Kødkvæg
Hyologisk Tidsskrift
Dansk Veterinærtidsskrift
Månedsmagasinet Kvæg
Risø Nyt
Økologi & Erhverv
Jern- og Maskinindustrien
FORSKERforum
MiljøDanmark
Global Økologi
Fremtidsorientering
Geologisk nyt
Maskinmesteren
Aktuel Naturvidenskab
Industriel Overfladebehandling
Teknovation
Nyhedsbladet Dansk Energi
Rent i Danmark
LABORANTEN
Installations Nyt
Landskab
Vedvarende Energi & Miljø
Boligen
Miljøministeriets Ugebrev
CSR
Urt
Spildevandsteknisk Tidsskrift
Hvidvare-Nyt
Vandposten
Teknik og Miljø
Elteknik
Energi & Økonomi
danskVAND
OmEnergi
Megawatt
Fjernvarmen
Gasteknik
ENVIRONMENT / ENERGY
Installatør Horisont
Arkitektur DK
Fagbladet TIB
Bladet By & Land
Tegl
Fagbladet Træ & Møbelindustri
PROSAbladet
Maskin- & Materiel Magasinet
Twice a year
Quarterly
Annually
Monthly
Every other month
Three times a year
Weekly
Twice a month
Daily
Unknown

Appendix I

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-

import logging, MySQLdb, redis, os, sys
import MySQLdb.cursors

from gensim.corpora.dictionary import Dictionary
from gensim.corpora.textcorpus import TextCorpus
from gensim.corpora.mmcorpus import MmCorpus
from gensim import interfaces, matutils, utils

NS = 'index2'

STOP_WORDS = ['all', 'whoever', 'oss', ... 'once']

def tokenize(content):
    return [token.encode('utf8') for token in utils.tokenize(content, lower=True,
errors='ignore')
            if 3 <= len(token) <= 15 and not token.startswith('_')]

def strip_stop_words(content):
    return list(set(content) - set(STOP_WORDS))

class MyCorpus(TextCorpus):
    def __init__(self):
        self.dictionary = Dictionary(self.get_texts())

        #  ignores all words that appear in less than 20 documents, or in more than
50% documents
        self.dictionary.filter_extremes(no_below=100, no_above=0.5, keep_n=100000)

    def get_texts(self):
        i = 0
        j = 0

        while 1:
            logging.info("Getting chunk");
            cursor = conn.cursor(MySQLdb.cursors.DictCursor)
            cursor.execute("SELECT * FROM scraper_data WHERE dataset IN (1,2) AND
page IN (SELECT page FROM scraper_index /*WHERE `group` = 'TRANSPORT'*/) LIMIT %i,
1000" % (1000*j))
            if cursor.rowcount == 0:
                break

            for row in cursor.fetchall():
                r.set(NS + '_gensim_%d' % i, int(row['id']))
                r.set(NS + '_mysql_%s' % row['id'], i)
```

```
                i = i+1
                yield strip_stop_words(tokenize( row["lead"] + " " + row["body"] ))

            cursor.close()
            logging.info("Done %i / %i" % (i, total_count));
            j = j+1

if __name__ == '__main__':

    logging.basicConfig(format='%(asctime)s : %(levelname)s : %(message)s',
level=logging.INFO)

    pool = redis.ConnectionPool()
    r = redis.Redis(connection_pool=pool)
    logging.info("Redis connection established");

    conn = MySQLdb.connect('10.211.55.2', 'root', 'sqlpass', 'bachelor',
        port = 4040) #, cursorclass = MySQLdb.cursors.SSCursor
    conn.set_character_set('utf8')
    logging.info("MySQL connection established");

    cursor = conn.cursor()
    sql = "UPDATE scraper_data SET title = REPLACE(title, '&amp;amp;', '&'), lead =
REPLACE(lead, '&amp;amp;', '&'), body = REPLACE(body, '&amp;amp;', '&'), journalist =
REPLACE(journalist, '&amp;amp;', '&')"
    cursor.execute(sql)

    cursor.execute("SELECT COUNT(*) AS c FROM scraper_data WHERE dataset IN (1,2)
AND page IN (SELECT page FROM scraper_index /*WHERE `group` = 'TRANSPORT'*/)")
    total_count = int(cursor.fetchone()[0])
    logging.info("The database holds a total count of %d documents" % total_count);

    corpus = MyCorpus()

    # save dictionary and bag-of-words (term-document frequency matrix)
    corpus.dictionary.save_as_text('data/' + NS + '_wordids.txt')
    MmCorpus.serialize('data/' + NS + '_bow.mm', corpus, progress_cnt=10000)
    del corpus

    # initialize corpus reader and word->id mapping
    dictionary = Dictionary.load_from_text('data/' + NS + '_wordids.txt')
    mm_corpus = MmCorpus('data/' + NS + '_bow.mm')

    # build tfidf,
    from gensim.models import TfidfModel
    tfidf = TfidfModel(mm_corpus, id2word=dictionary, normalize=True)
```

```
    # save tfidf vectors in matrix market format
    MmCorpus.serialize('data/' + NS + '_tfidf.mm', tfidf[mm_corpus],
progress_cnt=10000)
```

Appendix J

index2_40t_lsi (1694dd)

#0		#1		#2		#3	
0.0655	nye	-0.1310	microsoft	-0.1423	procent	0.2111	cvr
0.0645	danske	-0.1280	windows	-0.1021	selskabet	0.1947	boets
0.0644	kommer	0.1167	landbrug	-0.1006	stigning	0.1942	skyldneren
0.0640	godt	-0.1146	version	-0.0992	omsætning	0.1940	fordring
0.0636	får	-0.1062	brugere	-0.0967	kvartal	0.1939	kreditorudvalg
0.0626	procent	-0.1008	google	-0.0949	steg	0.1939	kurator
0.0622	ifølge	-0.0985	microsofts	-0.0900	året	0.1939	konkursdagen
0.0621	helt	-0.0972	software	-0.0895	omsætningen	0.1899	skifteretten
0.0618	danmark	0.0908	landmænd	-0.0894	steget	0.1887	fremsættes
0.0618	sidste	-0.0883	computerworld	-0.0868	marked	0.1849	anmelde
0.0615	hele	0.0882	landbruget	-0.0848	kroner	0.1803	skriftligt
0.0606	dag	-0.0860	applikationer	-0.0831	millioner	0.1784	bekendtgørelse
0.0604	vores	-0.0854	brugerne	-0.0829	markedet	0.1782	opfordres
0.0602	mener	-0.0831	service	-0.0813	faldet	0.1730	advokat
0.0596	gang	-0.0823	news	-0.0786	salget	0.1654	opgjort
0.0593	stor	0.0820	formand	-0.0776	milliarder	0.1586	anmodning
0.0589	går	-0.0763	oversat	-0.0757	samlede	0.1371	eventuel
0.0583	samtidig	-0.0731	internet	-0.0740	tal	0.1286	enhver
0.0582	del	0.0727	fødevarer	-0.0730	seneste	0.1262	senest
0.0574	inden	-0.0725	data	-0.0718	salg	0.1254	valg

#4		#5		#6		#7	
-0.0995	hektar	0.0906	skriver	0.0951	landbrug	0.2154	svovlfri
0.0977	direktør	0.0899	sagen	0.0870	landmænd	0.2127	blyfri
0.0950	virksomheder	-0.0819	samarbejde	0.0801	krav	0.2115	fyringsolie
-0.0796	liter	-0.0710	udstyret	0.0784	landbruget	0.2091	dieselprodukter
0.0768	regeringen	-0.0702	nye	-0.0735	gud	0.2065	listepriserne
-0.0725	høst	0.0693	ifølge	0.0733	regeringen	0.1971	energiselskabet
0.0712	vækst	-0.0693	transport	-0.0724	direktør	0.1849	benzin
-0.0708	lidt	0.0683	mener	0.0723	fødevarer	0.1834	diesel
-0.0700	marken	-0.0646	mola	-0.0713	medarbejdere	0.1710	moms
0.0673	offentlige	-0.0646	motor	-0.0691	selskabet	0.1595	liter
0.0669	administrerende	0.0644	landmænd	0.0683	liter	0.1547	basis
0.0649	formand	0.0641	sag	0.0672	hektar	0.1520	standersalg
-0.0648	uger	-0.0640	meter	-0.0660	jesus	0.1463	øre
0.0631	sikre	-0.0628	maskiner	-0.0657	administrerende	0.1288	dieselolie
0.0624	dansk	-0.0626	virksomheden	0.0613	grøn	0.1165	øvrige
-0.0598	meter	-0.0624	leveret	-0.0611	omsætning	0.1059	fuelolie
-0.0594	korn	-0.0611	nyt	-0.0610	liv	0.1039	følgende
0.0593	medarbejdere	-0.0599	monteret	-0.0608	ansatte	0.0987	ekskl
0.0592	skabe	-0.0593	medarbejdere	-0.0591	guds	0.0979	prisen
0.0592	udvikling	-0.0580	biler	0.0572	miljø	0.0805	politiet

#8		#9		#10		#11	
-0.1734	svovlfri	0.1852	landbrug	-0.0982	kunder	-0.0950	hektar
-0.1711	blyfri	-0.1356	viser	0.0909	lastbiler	0.0870	markedet
-0.1692	fyringsolie	0.1264	landmænd	0.0866	fødevarer	0.0787	processor
-0.1685	dieselprodukter	0.1222	landbruget	-0.0821	kroner	0.0752	modeller
-0.1662	listepriserne	0.1125	formand	-0.0807	kunderne	0.0728	udstyret
-0.1557	energiselskabet	0.1060	fødevarer	-0.0792	selskabet	0.0709	intel
-0.1403	benzin	-0.0923	undersøgelse	0.0779	viser	-0.0684	microsoft
-0.1295	øre	-0.0903	antallet	-0.0742	penge	0.0665	skærm
-0.1274	basis	0.0837	hektar	0.0737	danmarks	0.0662	regeringen
-0.1251	diesel	-0.0822	undersøgelsen	-0.0718	betale	0.0650	lastbiler
-0.1247	moms	-0.0799	procent	-0.0684	direktør	-0.0649	angreb
-0.1218	standersalg	-0.0788	tal	0.0682	lande	-0.0638	brugere
0.1030	oplyser	0.0761	windows	-0.0681	forklarer	0.0622	euro
-0.0931	dieselolie	0.0750	microsoft	0.0676	lørdag	-0.0616	medarbejdere
-0.0926	prisen	0.0749	landbrugets	0.0656	antallet	0.0616	betale
0.0901	politiet	0.0740	landboforening	0.0656	landbrug	-0.0608	service
-0.0901	liter	-0.0729	stigning	-0.0650	virksomheden	0.0608	model
-0.0857	fuelolie	0.0697	selskabet	0.0646	transport	0.0584	pris
0.0853	lastbiler	0.0693	sagde	-0.0609	løsning	0.0575	bærbare
0.0793	skriver	0.0679	bestyrelsen	0.0605	årets	-0.0574	landbrug

#12		#13		#14		#15	
0.1309	lande	0.1474	hektar	0.1204	lastbiler	0.1326	news
-0.1221	kommune	0.1108	høst	-0.1127	dyr	0.1299	oversat
0.1175	tyskland	-0.1010	landbrug	-0.0915	mælk	-0.1063	nettet
0.0923	europa	-0.0969	danish	-0.0880	køer	0.0977	computerworld
-0.0892	kommuner	0.0966	hvede	0.0869	lastbil	0.0963	intel
0.0868	danmark	-0.0899	crown	0.0852	transport	0.0937	bøndergaard

-0.0863 kommunerne	0.0886 høsten	-0.0842 svin	0.0868 service
0.0859 tyske	-0.0850 formand	-0.0830 kommune	-0.0837 penge
0.0832 sverige	0.0799 usa	0.0767 vækst	-0.0833 brugere
-0.0817 kommunen	0.0789 afgrøder	-0.0759 danish	0.0802 processorer
0.0811 danske	-0.0778 svin	-0.0751 grise	-0.0757 gratis
0.0807 europæiske	0.0776 verdens	0.0749 køre	0.0757 thomas
0.0778 produktion	0.0735 lande	-0.0729 besætninger	0.0747 intels
-0.0754 processor	0.0723 marker	0.0716 ton	-0.0734 landbruget
-0.0745 kroner	0.0718 ton	-0.0710 crown	-0.0732 internet
0.0736 fødevarer	0.0689 amerikanske	-0.0707 kvæg	-0.0724 danskerne
0.0727 usa	-0.0688 grise	-0.0706 økologiske	0.0711 gud
-0.0726 intel	0.0676 jorden	-0.0673 økologisk	0.0706 processor
0.0696 marked	0.0668 korn	0.0666 service	-0.0700 google
0.0693 mælk	0.0662 millioner	0.0666 høst	-0.0685 tdc

#16	**#17**	**#18**	**#19**
-0.1158 lastbiler	-0.1415 landbruget	0.1176 gud	-0.2143 konkursboet
-0.1072 danish	-0.1010 millioner	0.1137 jesus	-0.2032 skiftesamling
-0.1045 crown	0.0976 høst	-0.1068 politiet	-0.1780 eftersyn
-0.0971 køre	-0.0940 skriver	0.1044 guds	-0.1685 fordringer
-0.0912 trafik	-0.0835 milliarder	-0.0881 lastbil	-0.1553 klokken
0.0899 traktorer	-0.0793 grøn	-0.0838 undersøgelse	-0.1547 udlodning
-0.0899 trafikken	-0.0721 energi	0.0831 danish	-0.1546 stadfæstelse
-0.0883 lastbil	0.0717 hvede	0.0780 crown	-0.1544 skiftesamlingen
-0.0763 chauffører	0.0714 høsten	0.0775 version	-0.1534 dividende
-0.0750 vejdirektoratet	-0.0712 kroner	0.0760 kristne	-0.1472 anmeldes
0.0748 motor	-0.0705 landmænd	-0.0753 landbrug	-0.1467 stadfæstelsen
0.0736 udstyret	-0.0704 landbrugets	0.0713 arla	-0.1390 afsluttende
-0.0726 motorvejen	0.0698 medlemmer	-0.0705 viser	-0.1371 cvr
0.0698 landbrug	-0.0698 vækst	-0.0703 undersøgelsen	-0.1343 anke
-0.0692 dtl	-0.0680 politiet	-0.0696 politi	-0.1337 regnskab
-0.0665 transport	0.0678 dtl	0.0690 browseren	-0.1284 udkast
0.0646 traktor	-0.0667 projektet	-0.0688 landbruget	-0.1247 konkurslovens
0.0643 landbruget	0.0661 vognmænd	-0.0686 computerworld	-0.1225 stadfæstet
0.0642 serien	0.0640 chauffører	0.0680 browser	-0.1210 udkastet
-0.0642 kører	0.0605 priser	0.0678 explorer	-0.1188 udbetaling

#20	**#21**	**#22**	**#23**
0.1573 lastbiler	0.1442 danish	0.1210 service	0.1569 mælk
0.1016 transport	0.1362 crown	0.1107 news	0.1451 arla
0.0984 hektar	0.1040 leveret	0.1105 oversat	0.1282 køer
0.0975 økologiske	-0.0992 landmænd	0.1019 traktorer	-0.1273 danish
0.0943 scania	-0.0891 jesus	0.0965 computerworld	-0.1118 crown
0.0935 økologisk	-0.0887 gud	-0.0952 arla	0.1055 foods
-0.0905 traktorer	0.0841 lastas	-0.0922 resultat	0.0963 arlas
0.0854 chauffører	-0.0831 guds	-0.0835 selskabet	-0.0927 leveret
0.0853 leveret	0.0817 motor	-0.0831 adm	-0.0922 kommune
0.0853 vognmand	-0.0751 kunder	-0.0828 direktør	-0.0869 svinekød
0.0778 google	0.0749 crowns	0.0807 bøndergaard	0.0854 køerne
0.0729 køer	-0.0725 lastbiler	-0.0781 danish	-0.0806 priserne
-0.0719 ramt	0.0710 elever	-0.0761 resultatet	0.0769 mejeri
0.0717 vognmænd	-0.0702 landbrug	-0.0760 foods	0.0746 traktorer
0.0700 lastbil	0.0683 monteret	0.0730 dyr	-0.0736 priser
0.0699 mælk	0.0668 hedensted	-0.0706 processor	0.0723 kvæg
-0.0670 traktor	0.0662 konkursboet	-0.0697 bestyrelsen	-0.0718 svin
-0.0669 fødevarer	0.0659 skiftesamling	0.0679 new	-0.0713 prisen
0.0661 android	0.0658 førerhus	-0.0667 omsætning	-0.0698 slagterier
0.0659 biler	0.0652 eleverne	-0.0656 crown	0.0688 mælken

#24	**#25**	**#26**	**#27**
0.1052 news	-0.1176 arla	0.1633 arla	-0.1316 danish
0.1029 computerworld	-0.1061 økologiske	0.1453 mælk	-0.1188 crown
0.1019 oversat	-0.1046 økologisk	0.1258 landmænd	0.1114 byggeri
-0.1006 byggeri	-0.1018 fødevarer	-0.1251 søer	0.1045 køer
0.0993 kroner	-0.1008 mælk	0.1176 foods	0.1029 dyr
-0.0889 firmaet	0.0978 chauffører	-0.1157 grise	0.0953 leveret
0.0889 service	0.0844 kroner	0.1039 lastbil	0.0906 lastas
-0.0829 windows	0.0831 vognmænd	0.1019 økologisk	-0.0876 traktorer
0.0755 bøndergaard	-0.0827 miljøminister	-0.1006 svin	0.0859 dansk
0.0748 prisen	-0.0798 foods	0.1001 arlas	-0.0841 landbrug
-0.0703 apple	-0.0787 tdc	0.0999 økologiske	0.0836 kvæg
-0.0692 microsoft	-0.0774 kunder	-0.0963 slagtesvin	-0.0829 landmænd
0.0682 formand	-0.0744 gud	-0.0923 svineproduktion	-0.0777 lastbiler
0.0681 pris	-0.0739 leveret	0.0903 priserne	-0.0735 fødevarer
-0.0669 medarbejdere	-0.0723 kunderne	0.0862 mejeri	0.0732 hedensted
0.0664 priser	-0.0709 viser	-0.0861 smågrise	0.0723 kel
0.0660 køer	-0.0706 jesus	-0.0845 iphone	0.0712 opbygget
-0.0654 tyskland	0.0705 google	-0.0839 besætninger	0.0695 byggeriet
0.0653 thomas	0.0704 hektar	0.0822 chaufføren	0.0694 regeringen
0.0649 trafik	-0.0703 mejeri	-0.0777 android	-0.0679 crowns

#28		#29		#30		#31	
-0.1478	danish	0.1010	news	0.1595	lastbiler	0.1702	oversat
-0.1371	crown	-0.0999	udstillere	-0.1229	iphone	0.1693	news
0.1016	landmænd	0.0983	oversat	-0.1128	android	-0.1251	lastbil
0.0947	lastas	0.0925	formand	0.1118	intel	0.1209	computerworld
0.0885	landbruget	-0.0923	årets	0.1045	processorer	0.1127	bøndergaard
-0.0881	byggeri	0.0822	internet	-0.1000	apples	0.1100	service
-0.0820	virksomheder	-0.0818	lørdag	-0.0999	økologiske	-0.1038	lastbilen
0.0803	leveret	-0.0810	version	-0.0948	økologisk	-0.1025	chaufføren
0.0755	euro	-0.0797	læs	-0.0947	apple	-0.0973	microsoft
-0.0755	crowns	0.0790	trafik	0.0890	processor	0.0852	thomas
0.0724	hedensted	-0.0784	danish	0.0873	scania	-0.0824	prisen
-0.0713	dansk	-0.0779	skat	0.0849	volvo	-0.0820	politiet
-0.0712	formand	-0.0764	dyr	-0.0838	computerworld	-0.0792	priserne
0.0693	daf	0.0736	mælk	-0.0835	news	-0.0769	microsofts
0.0674	landbrug	-0.0733	messen	0.0823	mercedes	-0.0725	årig
0.0664	passagerer	0.0716	bestyrelsen	-0.0823	oversat	0.0723	dyekjær
-0.0645	dtl	-0.0710	fredag	0.0814	intels	-0.0720	ulykken
-0.0631	politiet	-0.0703	crown	-0.0797	politiet	0.0679	eriksen
-0.0628	kroner	0.0702	service	-0.0796	telefoner	-0.0677	priser
0.0606	rederiet	-0.0688	miljøminister	0.0774	biler	-0.0676	korn

#32		#33		#34		#35	
-0.1144	lastbiler	0.1694	fødevarer	0.1113	landmænd	0.1443	patienter
0.1083	regeringen	0.1302	landbrug	0.0966	danish	0.1349	apotek
-0.0954	kommune	-0.1217	patienter	0.0947	transport	0.1286	apoteker
0.0945	økologiske	-0.1060	lastbiler	0.0944	lastas	0.1260	medicin
0.0839	windows	0.0930	økologiske	-0.0917	volvo	0.1226	apoteket
-0.0831	byggeriet	-0.0884	medicin	-0.0907	søer	0.1134	apotekerne
-0.0824	ton	-0.0858	arla	0.0906	crown	0.1067	maci
0.0811	økologisk	-0.0844	landmænd	-0.0896	lastbil	-0.1036	arla
0.0808	økologi	0.0828	økologisk	-0.0892	biler	0.1023	patienterne
0.0776	ritzau	-0.0823	patienterne	-0.0878	lastbiler	-0.0984	landbrugsavisen
-0.0776	formand	-0.0773	biler	-0.0877	scania	0.0913	læger
0.0766	danish	-0.0748	landmændene	-0.0826	mercedes	-0.0886	lastbiler
-0.0741	arla	-0.0743	læger	-0.0798	news	0.0862	lægemidler
-0.0738	formanden	-0.0724	mercedes	-0.0785	oversat	0.0843	far
0.0733	crown	-0.0721	angreb	0.0767	hedensted	0.0812	behandling
-0.0728	landboforening	-0.0720	foods	-0.0746	mola	-0.0805	computerworld
-0.0727	arkitekter	0.0716	økologi	0.0738	landmændene	0.0779	microsoft
0.0720	regeringens	0.0712	effektivt	-0.0735	økologiske	-0.0778	foods
0.0707	service	-0.0711	volvo	0.0729	kvæg	0.0771	læge
-0.0684	projektet	-0.0702	apoteker	0.0727	kel	-0.0770	læs

#36		#37		#38		#39	
-0.1507	dtl	0.1663	miljøminister	0.1970	effektivt	-0.2157	landbrugsavisen
-0.1251	transport	-0.1365	økologiske	0.1711	landbrug	0.1636	biler
0.1228	bilen	0.1327	lastbil	-0.1578	mle	-0.1510	lastbiler
0.1203	biler	-0.1268	fødevarer	0.1513	artiklen	-0.1409	tlf
0.1149	scania	-0.1162	økologisk	0.1451	helhed	-0.1347	ton
0.1061	byggeri	-0.0995	økologi	0.1085	byggeri	0.1331	bilen
-0.1033	miljøminister	0.0989	miljøministeren	0.1028	tyskland	-0.1261	lastbil
-0.0986	vognmænd	0.0951	lastbilen	0.0935	læs	0.0955	motor
-0.0944	chauffører	-0.0883	lastbiler	0.0924	vejdirektoratet	-0.0935	totalvægt
0.0939	mercedes	-0.0878	dtl	-0.0907	dyr	-0.0911	danish
-0.0919	mle	0.0869	poulsen	0.0877	tyske	-0.0884	kel
0.0863	fødevarer	-0.0857	millioner	-0.0871	passagerer	0.0860	bil
0.0808	motor	0.0848	troels	0.0839	kroner	-0.0829	crown
-0.0771	dtls	0.0785	mola	-0.0832	rederiet	-0.0797	trailere
0.0763	renault	-0.0767	landbrug	-0.0808	skibe	-0.0768	berg
0.0741	bil	0.0759	vandplanerne	0.0803	computerworld	0.0725	vejdirektoratet
-0.0730	kel	-0.0748	vognmænd	-0.0764	skriver	-0.0714	lastas
0.0729	formand	0.0746	vandløb	0.0716	byggeriet	-0.0710	landbruget
-0.0726	østergaard	0.0731	chaufføren	-0.0700	dtl	-0.0704	lastbilen
0.0712	benz	0.0721	danish	0.0638	lande	-0.0670	traileren

Appendix K

Press release used for testing the topics of LSA.

Skatteudspil styrker mindre virksomheder

24. februar 2009

Regeringens skatteudspil er et lys i mørket for små og mellemstore virksomheder. Det mener Håndværksrådet, der er hovedorganisation for mere end 20.000 små og mellemstore virksomheder i Danmark.

”Store lettelser i personskatterne og underfinansiering på 12 mia. kr. er et rigtig godt spark til samfundsøkonomien på det helt rigtige tidspunkt. Mange små og mellemstore virksomheder har fået forværret deres økonomiske situation markant de seneste måneder, og de har virkelig behov for, at danske forbrugere igen tør tage pungen frem og købe ind i danske virksomheder”, siger cheføkonom Søren Nicolaisen fra Håndværksrådet.

Regeringens forslag ligger på mange punkter tæt op af skattekommissionens. Der er lettelser i både top og bund. Vi kunne godt have tænkt os alle Skattekommissions forslag til lettelser, men accepterer at det her er det politisk mulige. Hvis ikke det er nok, har vi tillid til, at regeringen bruger andre midler til at sætte gang i samfundsøkonomien.

”Vores største bekymring her og nu er de mindre produktionsvirksomheder. Mange af dem lever af eksportmarkeder, som er stærkt aftagende. Man skal derfor være meget varsom med at hæve deres omkostninger i disse tider, og derfor er det spørgsmålet, om de kan klare så store stigninger i energiafgifterne. Når regeringen meget rigtigt vil tage særlige hensyn til konkurrenceudsatte virksomheders energiudgifter, er det vigtigt, at den fokuserer bredt og finder løsninger for små og mellemstore virksomheder”, siger han.

Håndværksrådet glæder sig også over, at rentefradraget ikke beskæres helt så meget som i Skattekommissionens forslag.

”Man skal være varsom med at skære på måder, der kan ramme huspriserne. Derfor er det godt, at der er skåret lidt ned på det forslag. Nu må vi i de kommende dage kigge på, om den model, regeringen har skruet sammen, fungerer i praksis”, siger Søren Nicolaisen.