Topic Detection and Tracking System

Research Project Master Artificial Intelligence Group 5

Daniel Brüggemann
Yannik Hermey
Carsten Orth
Darius Schneider
Stefan Selzer

June 22, 2016

Table of Contents

List of Figures

List of Tables

1. Introduction

Growth of internet came along with an increasingly complex amount of text data from emails, news sources, forums, etc. As a consequence, it is impossible for a single person to keep track of all relevant text data in most cases and moreover to detect changes in trends or topics.

Every company (and every person) would be interested to harness this amount of free and cheap data in order to develop intelligent algorithms, which are able to react to emerging topics as fast as possible and at the same time track existing topics over long time spans. There are many techniques about topic extraction (like Nonnegative Matrix Factorization (NMF) or Latent Dirichlet Allocation (LDA) [Blei et al., 2003]) but there are not many extensions to dynamic data handling. The goal of this project is to explore LDA (or other techniques) as a technique to detect topics as they appear and track them through time. Corpus can be the (fully annotated and immediately available) RCV1 Reuters corpus (810.000 documents) and/or the actual Reuters archive.

This report is organized as follows: following the introduction in Section 1, we describe the state of the art and explain the LDA and NMF algorithm in Section 2. Section 3 describes the text data set, the Reuters Corpus, that was processed by our algorithms to generate topics. Section 4 elaborates on the pre-processing that was performed on the text data as preparation for both the LDA and the NMF approach. In Section 5, the implementations of the LDA and the NMF approach are presented in detail, as well as the results that were achieved with these approaches. Section 6 describes the visualization tools that we implemented for the purpose of visualizing a topic river and getting access to corresponding topics, terms and text files. Finally, Section 7 concludes on the projects' results and suggests future work for improvement. The appendix contains short manuals on how to use the implemented software.

2. State Of The Art

This section gives an overview of the technical approaches currently used in the field of topic detection and tracking. First, the general tasks that have to be performed to detect and track topics over time are presented. Finally the two major approaches, namely Latent Dirichlet Allocation and non-negative matrix factorization, are described in theory.

2.1. Topic Detection and Tracking Tasks

The topic detection and tracking tasks according to [Fiscus and Doddington, 2002] are as follows:

- Topic Tracking: Track an identified topic over a period of time.
- Link Detection: Detect links between topics at different time points.
- Topic Detection: Identify different topics.
- First Story Detection: Identify the first occurence of a certain topic in the time line.
- Story Segmentation: Identify splitting of topics into new ones and the corresponding points in time.

Each of these tasks are covered in both the dynamic LDA and the dynamic NMF approach, which are explained in Section 5.

2.2. Latent Dirichlet Allocation

Latent Dirichlet Allocation (LDA) [Blei et al., 2003] is a generative probabilistic mixture model for topic detection. In such a model, text is represented as a set of words (bag-of-words). Grammar and word order are disregarded while multiplicity, i.e. the frequency of occurrences of single words in the text, is being kept and can be used as feature for topic detection. Words define a vocabulary and topics are represented by a probabilistic distribution of words from this vocabulary. Each word may be part of several topic representations. LDA assumes that each document from a collection of documents is generated from a probabilistic distribution of topics. The model tries to infer these probabilistic distributions by maximizing the probabilities of the words given the topic distribution. This means, that the words in a certain document are generated by first choosing a topic from the topic distribution and then choosing a word from that topic such that the probability of that word given that topic is maximized. As the actual computation of the probabilities is intractable, the posterior probability of a word given a topic is computed iteratively using sampling methods. Blei et al. [Blei et al., 2003] use Bayes' Theorem in combination with a Dirichlet distribution as prior distribution to approximate the true posterior distribution. The probability space defined by the probabilities of the words and topics is multidimensional which is represented by a multinomial distribution. For the a priori estimation the conjugate distribution is needed, which corresponds to a Dirichlet distribution in this case. Information gain is used as measure for the difference between two iterated probability distributions and thereby acts as convergence criterion. Figure 1 shows a graphical representation of the standard LDA model.

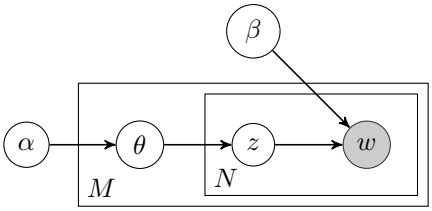


Figure 1: Plate diagram representing LDA as a Bayesian network. The outer box represents M documents, the inner box represents the N times repeated choice of topics and words within a document. θ represents the topic distribution per document. α and β represent the concentration parameters for the per-document and per-topic Dirichlet distributions. The variable w is observable while the other variables are latent. Edges denote dependencies among variables.[Blei et al., 2003]

There are extensions of the LDA model towards topic tracking over time, as [Wang and McCallum, 2006] and [Wei et al., 2007]. But according to [Wang et al., 2008], these methods deal with constant topics and the timestamps are used for better discovery. Opposed to that, in [Blei and Lafferty, 2006] a model for detection of evolving topics in a discrete time space is presented. Here LDA is used on topics aggregated in time epochs and a state space model handles transitions of the topics from one epoch to another. A gaussian probabilistic model to obtain the posterior probabilities on the evolving topics along the time line is added as additional dimension. Figure 2 shows a graphical representation of the dynamic topic model.

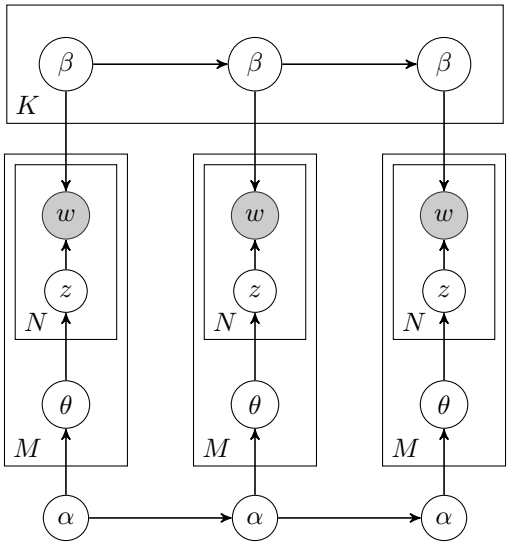


Figure 2: Plate diagram representing the dynamic topic model (for three time slices) as a Bayesian network. The model for each time slice corresponds to figure 1. Additionally, each topic's parameters α and β evolve over time.[Blei and Lafferty, 2006]

2.3. Non-Negative Matrix Factorization

2.3.1. Algorithm

Non-negative matrix factorization is an algorithm that can be used in the field of text mining to factorize and thereby decrease the dimension of a large matrix [Lee and Seung, 1999]. For topic detection, the original matrix can be composed of terms represented in the rows and documents represented in the columns. The cell values can represent the number of occurrences of a word in a document. However, as this would give an advantage to frequently used words that appear in many documents and thus do not have a great expressiveness, another metric called the TF-IDF value can be used. TF-IDF is composed of the term frequency, which is the number of times a term appears in a document, and the Inverse Document Frequency, which defines in how many documents of the data collection the term appears. As a term is usually most specific and expressive when used a lot of times in a low number of documents, the usual formula shown in Equation 1 takes this into account (N = Number of documents). The log function dampens the effect on terms being in a very high number of documents.

$$TFIDF = TF * log_{10}(N/DF) \tag{1}$$

As a TF-IDF value cannot be negative, the algorithm's requirement of a matrix with only non-negative values is fulfilled. This matrix is then factorized into two positive matrices which have significantly lower dimensions, one representing a term-topic relation (how much a term is relevant to a topic), the other one representing a topic-document relation (how much a topic is relevant in a document). When multiplied with each other, these matrices form an approximation of the original matrix and fill in zero-values, acting as predictors for the cell values.

Ranking the terms of a topic by their matrix value reveals the most relevant terms that can make up the description of this topic. In a similar way, ranking a topic's documents by their

matrix values shows the most relevant documents covering this topic. Also, summing up the document's matrix values over the whole topic allows for the ranking of the topics themselves. This makes it possible to visualize topics according to their importance amongst all topics.

Advantages of this algorithm are storage savings due to the sparsity of factorization and the good scalability for increasing matrix dimensions. Disadvantages are that the factorization is not unique and convergence is not always guaranteed, both negligable problems in the domain of topic detection. Although the original data size can be too large for matrix factorization, there already exist variants of the algorithm using an dynamic approach, processing data in chunks [Wang et al., 2011a]. A dynamic and online processing of web articles using NMF is thus feasible.

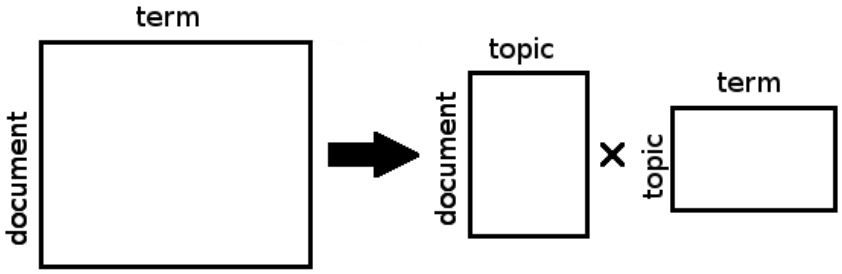


Figure 3: Non-negative Matrix Factorization

2.3.2. Related Work

There exist only few approaches so far that applied NMF for dynamically changing text data, i.e. when detecting and tracking topics over time. They usually focus on detecting topic changes within the algorithm's factorization process, e.g. by introducing new variables that can reflect emerging or evolving topics.

[Cao et al., 2007] and [Wang et al., 2011b] use an online NMF algorithm that applies the factorization to the data of each time step and then updates the matrix bases from the previous calculations accordingly by some metric. However, their algorithms are not able to detect emerging topics.

[Saha and Sindhwani, 2012] defines an evolving set and an emerging set of topics within the NMF algorithm and appends the matrices accordingly in both dimensions whenever a new time step is considered. Topics are only detected when they emerge rapidly, and removing topics that are not relevant anymore is not discussed (the matrices increase gradually).

[Tannenbaum et al., 2015] introduces a sliding window over the time steps. First, NMF is applied on a certain time step, and then the found topics are assigned to the topic model defined by the previous time steps, if possible. If they do not fit into the model, they are added to the emerging set of topics, which are added to the model as soon as there are enough documents that cover this new topic. Within the emerging set, the texts are categorized into new topics using hierarchical clustering.

[Smeets, 2015] adds a topic-transition matrix to the factorization algorithm, which allows the observation of emerging, evolving and fading topics.

As a simpler, intuitive approach, our algorithm applies NMF on intervals of the text data and introduces a similarity metric that is able to connect topics of different time steps to a topic wave, which also enables emerging and fading topics at different time steps and yields coherent

results even for naive similarity measures. The algorithm and its results are described in detail in section 5.2.

3. Reuters Corpus

The main task is to perform topic detection on news articles. For this purpose, there are various data sources. The most well known source for such a task is the Reuters Corpus RCV1, but it is also possible to fetch current news articles from the Reuters homepage and build up a more up-to-date news corpus.

3.1. RCV1

The large document collection, that will be used on this project, is the Reuters corpus RCV1 [Lewis et al., 2004]. It consists of roughly 800.000 news articles from the time of August 1996 to August 1997. The articles are provided in XML-format and in addition to the article text contain various annotations, including the headline, the author, the country code, the date and the ID as well as manually assigned topics. The topic assignments for the articles make use of 103 out of 126 originally provided topics [Lewis et al., 2004].

3.2. Webcrawler

Aside from the RCV1, there is an RCV2 available, that contains articles in 13 different languages for the same period as the RCV1. Furthermore, there is a TRC2, containing the articles between 2008-01-01 and 2009-02-28[1]. Since that data is at least 7 years old (as of March 2016), an option for the project is to generate a custom Corpus for research purposes. Reuters offers a news archive, where all news articles can be viewed sorted by date[2]. A Webcrawler was programmed, that crawls through the articles of a certain day, month, or year and automatically downloads the HTML file associated with each article. The tool furthermore offers to format each article and generate a TXT-document, that contains all important information about the article, which is the URL, time, headline, description, contents and tags.

4. Preprocessing

This section explains the preprocessing steps applied to the corpus prior to the topic extraction. The different topic detection approaches have different requirements for their input. We first describe the preprocessing steps necessary for the LDA approach and then the ones for the NMF approach.

4.1. Preprocessing for LDA

The preprocessing steps for the LDA topic detection are separated into two major parts. First the article text is extracted from the documents and then the text is analyzed using natural language processing techniques to generate a meaningful vocabulary for the topic extraction. [Scholtes and van Cann, 2013] describes the importance of applying natural language processing techniques on vocabulary generation for topic extraction. The procedures used here are derived

[1] http://trec.nist.gov/data/reuters/reuters.html
[2] http://www.reuters.com/resources/archive/us/

from the approaches presented in [Scholtes and van Cann, 2013] and described in detail in the following sections.

4.1.1. XML-Extraction

The articles from the Reuters RCV1 corpus are formatted in xml, so the first step is to extract the text from the xml-files for each article. The relevant text, that is extracted, includes the article text itself from the text element and the headline text contained in the headline element of the xml-structure. Additionally, meta information like the annotated topics for each article is extracted for later evaluation. The text extraction is combined with a first cleaning step by identifying and removing editorial parts of the article, usually consisting of the location of the writer and a number connected to that location. The keyword “newsroom” is used to detect and eliminate the corresponding paragraph.

4.1.2. Natural Language Processing

The main natural language processing of the article text, namely the tokenization, named entity recognition (NER) and lemmatization, is done using the Stanford CoreNLP [Manning et al., 2014]. The text is first split into single tokens. The tokens are filtered according to the categories, that the named entity recognition assigned to them. As numbers are not very descriptive for topics, the named entity recognition is used to exclude all tokens categorized in number-related categories, precisely those of the categories "DATE", "DURATION", "MONEY", "NUMBER", "ORDINAL", "PERCENT", "TIME" and "SET".
Lemmatization is used to normalize the tokens without loosing informational detail. Standard stemming algorithms are not applicable in topic extraction as they reduce words aggressively to a common base even if these words are actually of different meaning. Consider the difference between “marketing" and “markets” for example. Lemmatization only removes inflectional endings and returns a dictionary form of the token.
The Stanford parser is highly context dependent and does not always categorize words correctly. As there are a lot of number-related words left, an additional step of removing such words is performed by regex-cleaning. This step also removes any words containing special characters human language words normally do not contain.
The next normalization step involves removing dashes and concatenating combined words as well as spell correction. As the news articles contain a lot of proper names and improperly resolving ambiguities can lead to loss of information, spell correction is done very carefully. The Levenshtein distance is used [Navarro, 2001] to correct those words of distance one who do not reveal ambiguities when compared to the entries of the official Hunspell dictionary. Named identities are excluded, as they cannot be correctly processed automatically. Even the lowest correcting distance of 1 for example would change “Ashton” to “Aston”, as the latter despite of the former is known in the dictionary. Both names do define completely different entities, of course. As the spell correction is computationally expensive, care is also taken, that it is only performed, when it makes sense. A preliminary, much faster test for existing equal words of the same length in the dictionary is preformed beforehand. A comparison is only done when the length of the strings differs by no more than the tested distance, which is 1 in this case. Last but not least, the spell correction is done as almost final step, after all the other refinements are applied.
Finally the remaining list is filtered using a stopword list, that contains the most common words like “the” and “and”. Such words do not contribute to reasonable meaning of the article and are not useful to identify topic content.

4.1.3. Vocabulary Generation

The process described in section 4.1.2 is applied to the text content of the news articles from August and September 1996 of the RCV1 corpus described in section 3.1 to obtain a vocabulary for the topic detection containing terms, that are as meaningful and descriptive as possible while eliminating as much noise, consisting of not descriptive or ambiguous terms, as possible. The first two months of the RCV1 corpus contain 83.650 documents, which is about 10% of the corpus documents overall. Table 1 shows the results of the reduction of the number of terms from 308.854 distinct terms of about 16 million words overall to a final vocabulary size of 131.202. This corresponds to a reduction to 42% of the number of distinct tokens.

Terms overall	16.467.261
Distinct tokens	308.854
NER category removals	133.976
Lemmatization removals	17.372
Regex cleanup removals	18.962
Spellcheck cleanup removals	6.768
Stopword removals	574
Final vocabulary size	131.202

Table 1: Terms and vocabulary for documents of August and September 1996 of RCV1

Performing topic extraction on the preprocessed data reveals the limits of natural language processing. The algorithms are context dependent and cannot always detect entities correctly. Therefore, it is necessary to apply the preprocessing and the topic detection iteratively to refine the results. Major problems specific to the Reuters news articles for example appeared with some special character sequences denoting some kind of identification for article contents. Such article ids are for example given by sequences of characters like “lrb” and “rrb”, mostly combined with some number but partially inconsistently standing alone. These terms appeared just in the right frequencies to have a major impact on the topic detection marking them as most important terms for even most of the topics.
Identifying and taking care of such problems though can produce good results, as will be shown in section 5.1.

4.2. Preprocessing for NMF

The algorithm of non-negative matrix factorization can be very computationally expensive in terms of time and memory. Especially the original matrix can be very large using the complete vocabulary and thus needs a lot of memory. Executing the algorithm for the first time we ran into those memory issues and even increasing the heap size of Java’s virtual machine (to 4GB) did not solve the problem.
Among several algorithm specific parameters, the algorithm’s performance mainly depends on three values: the number of terms (vocabulary), the number of topics and the number of documents. Also the computational time for NMF depends on these parameters. Looking at some benchmark tests concerning time for the number of topics and the size of vocabulary, a linear dependency for these parameters can be observed.

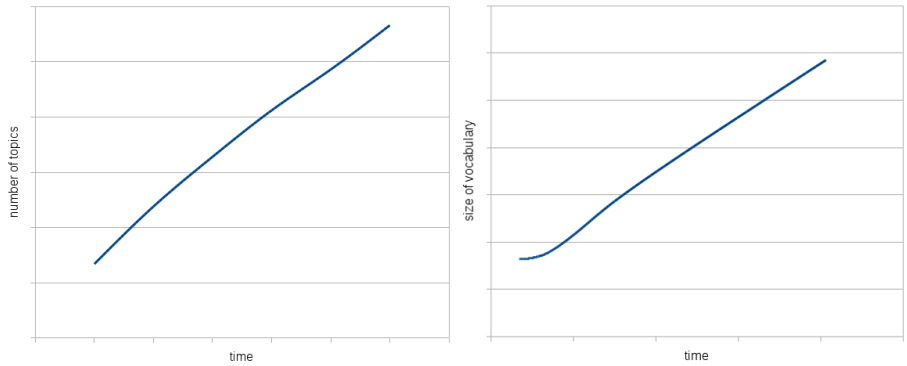


Figure 4: Benchmark tests for the size of vocabulary and the number of topics

Since the number of topics is flexible and the number of documents could only be reduced by sampling, the size of the vocabulary should be reduced to handle memory issues, preferably without loss of quality.
Therefore we applied the following pre-processing steps to reduce the number of terms for our vocabulary.

Porters Stemmer
The whole vocabulary reduction is based on the java implementation of Porter's Stemmer. The algorithm traverses a text file character wise and buffers the current word. This way numbers and punctuations are removed automatically and everything is casted to lower case. If a word delimiter is found, the buffers content can be stemmed applying some simple syntactical rules ([Porter, 1980]). This stemming process reduces multiple forms of a term to the same stem. The resulting character sequence is not a proper word in contrast to lemmatization algorithms, and it also stems entity names. In order to retranslate the stemmed terms after the NMF process, an appropriate mapping has to be stored as well.

Stopword Removal
Within the articles there are a lot of terms which occur in almost every text and do not contribute meaningful content to the topic. These words are called stopwords and are stored in a separate text file. Traversing the articles, these words are filtered and will not appear within the vocabulary. This is also a common text mining technique and several stopword lists are provided online [3]. These lists usually contain pronouns, prepositions and articles. After the first run, the NMF algorithm extracted some topics that only contained time specifications like "January" or "jul". To tackle this problem, we manually added these terms to the stopword list.

Americanization
The reuters corpus is a collection of articles written by many different authors. These authors did not use the same writing style for the articles, thus same words may be spelled in different dialects (American/British English). To reduce the vocabulary and map semantically same words to a common term, we used an algorithm which parses everything according to the American spelling rules. This works very similar to the Porter's Stemmer, applying some rules to the original word.

[3] http://www.lextek.com/manuals/onix/stopwords1.html

Here we sparsely modified the java implementation of Christopher Manning, which is part of the stanford core nlp and acquirable in a public github repository [4].

Low Occurrence Removal

Since the term-document matrix for NMF uses TF-IDF (1), values become very low if a tern either has a low occurrence in general (tf - term frequency), or the term occurs in most of the documents (idf - inverse document frequency). We can assume that a term mentioned only in one article is very specific to this document and does not contribute much to a more abstract topic. Consequently, terms with a low occurrence in general or mentioned only by a few different documents can be deleted from the vocabulary to lower the matrix dimensions and improve performance. Deleting terms named by less than five documents reduced the vocabulary considerably without lowering the quality of the extracted topics. Furthermore, typo errors or misspelled words will be (mostly) removed from the vocabulary as a useful side effect.

Vocabulary Reduction

Vocabulary	Number of Terms	Ratio
Initial Vocabulary	109.259	100.00%
Porters Stemmer	84.769	77.59%
Stopword Removal	84.618	77.44%
Americanizer	84.578	77.41%
Low Occurrence removal (<5)	12.524	11.46%

Table 2: Vocabulary Reduction for NMF

5. Topic Extraction

The topic extraction itself is performed using two different approaches. Latent Dirichlet Allocation, as explained in section 2.2, produces promising results in comparison with other topic extraction techniques [Blei et al., 2003] and the results produced using this approach are presented as well as the results produced by an implementation of NMF over time, in 5.2. Finally, in Section 5.3, we compare both approaches to each other.

5.1. LDA Dynamic Topic Model Approach

The topic detection using LDA is done in a two folded approach. First, the dynamic topic model is applied to the word count vectors of the corpus documents as produced from the preprocessing steps described in section 4.1. The resulting word distributions for each topic and time step are then further compared for similarity to identify evolvement inside the previously identified topics as well as between different topics along the time line. This comparison possibly finds time steps marking the turning points, where changes exceed a given threshold and thereby topics emerge into new ones or different topics aggregate to a common one.

[4] https://github.com/evandrix/stanford-corenlp/blob/master/src/edu/stanford/nlp/process/Americanize.java

5.1.1. Topic Detection

LDA needs to know the number of topics in advance. As the RCV1 corpus has 103 actually used annotated topics, as explained in section 3.1, plus a large amount of unlabeled documents, the parameter for the extraction is set to 104 topics. This corresponds to the 103 annotated topics and one additional "ERROR" topic for the unlabeled documents. As mentioned in section 4.1.3, the first two months of the RCV1 corpus are used as data for the topic extraction. They include 42 days which makes exactly 6 weeks of time. The dynamic topic model is accordingly applied to 6 time steps corresponding to the 6 weeks of the data set. Table 3 shows, that the documents are almost evenly distributed among the weeks.

Aug 20th - 26th	Aug 27th - Sept 2nd	Sept 3rd - 9th	Sept 10th - 16th	Sept 17th - 23rd	Sept 24th - 30th
12807	12800	13953	14606	14487	14997

Table 3: Number of documents per week in August and September 1996 of RVC1

The dynamic topic model partially identifies and reveals events of late summer 1996. Table 4 shows some of the best identified events. The top 10 words of the topics' word distributions already give a precise overview of the topics' contents.

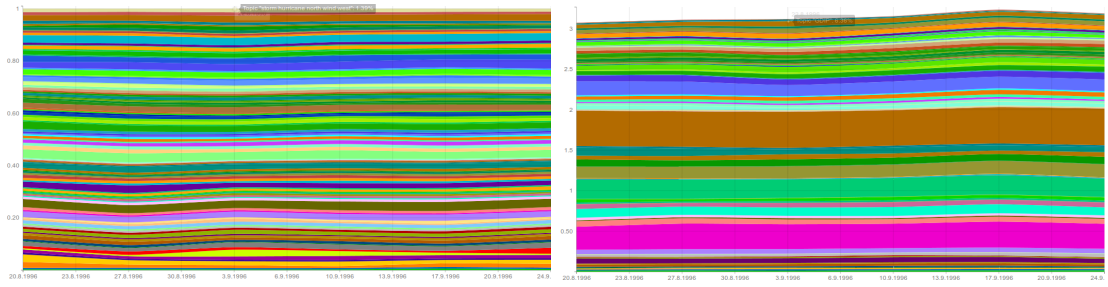
Child abuse in Belgium	Tropical storm Edouard	Peace talks in Palestina	Kurdish war in Iraq
child	storm	israel	iraq
police	hurricane	peace	iraqi
woman	north	israeli	iran
death	wind	netanyahu	kurdish
family	west	minister	turkey
girl	mph	palestinian	northern
murder	mile	arafat	arbil
dutroux	coast	talk	baghdad
body	move	government	force
sex	flood	west	united

Table 4: Extracted topics reveal events from August and September 1996

5.1.2. Comparison to RCV1 Annotated Topics

Matching the number of topics extracted by the dynamic topic model with the number of annotated topics from RCV1 makes a direct comparison of the topics possible. Both topic classifications, the annotations and the LDA extractions, lead to multi-labeled instances, as each document is assigned to multiple topics, which in turn leads to difficulties computing the performance measures. The problem is simplified by treating each label for each document as separate instance. To achieve an even distribution of topics on both sides, the threshold for the LDA topic score is chosen in such a way, that the average number of topics extracted from the LDA topic assignments matches the average number of annotated topics, which is 3.25. A confusion matrix is set up for all classes, respectively topics, interpreting the annotations as given classes and the extracted topics as predictions. Precision and recall are then computed for each pair of LDA

and annotated topics based on the topic assignments per document, treating the current entry as true positives and all remaining entries of the current row, respectively column, as false negatives, respectively false positives. The harmonic mean finally is used to assign matches between topics from both sides, by sequentially assigning and removing the topic pair with the highest value. The results are disappointing. The highest F-score is 0.288, derived from a precision of 0.432 and recall of 0.216. It suggests a matching of the RCV1 topic GSPO (Sports) with the LDA topic defined by the top words "party minister election government political opposition parliament president prime leader". As indicated by the low values for the harmonic mean, no further matching among the remaining assignments can be identified either.
Figure 5 presents a graphical comparison of the topic evolvements for the RCV1 annotated topics and the dynamic topic model detections. The topic rivers give an overview of the complete 104 topics detected by the dynamic topic model and their evolvement along the time line of the 6 weeks. The relative size is based on the overall score of all documents for each topic in the documents' topic distributions. The topic river of the annotated topics from RCV1 based on the document count for each topic is shown for comparison.

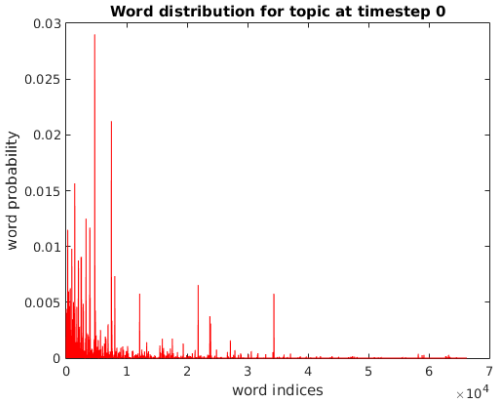


(a) Dynamic topic model topics (b) RCV1 annotated topics

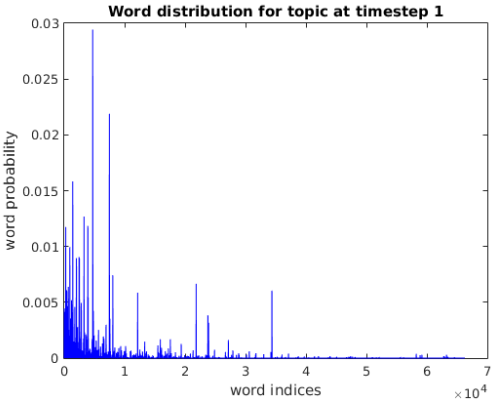
Figure 5: Topic rivers for August and September 1996 of RCV1 corpus

5.1.3. Topic Emergence

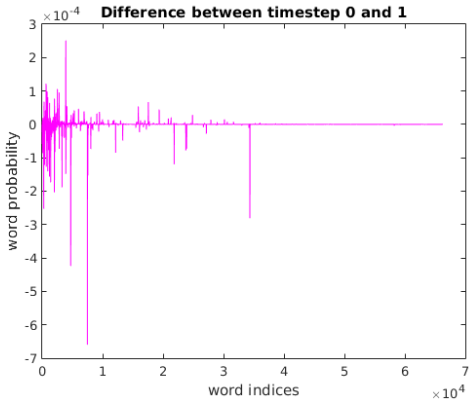
The fact, that there is little evolvement of the topics, is reflected by the word distributions. Inspecting them in detail reveals little difference among the word distributions for the time steps of each topic. Figure 6 shows the word distribution scores for the time steps 0 and 1 and the difference between them for the topic dealing with the tensions in Iraq during that time. The number of topics in the dynamic topic model is fixed and given initially and the computation of the topics infers the topics through a probabilistic distribution. This does not produce topics, that appear or disappear. Instead, the word distributions for one topic changes gradually over time. At some point, these changes might become significant enough to change the topics actual topic.
To identify such changes inside the word distributions, the second step of the two folded approach consists of applying a similarity measure to identify time steps, where the word distributions change enough to declare the topic as a new one. The measurement used in this case is the cosine similarity. Figure 6d shows the distances between the word distributions among each pair of neighboring time steps for the previously mentioned Iraq topic.



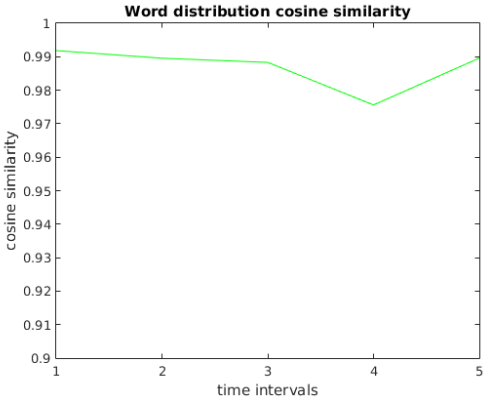
(a) Word distribution at time step 0



(b) Word distribution at time step 1



(c) Difference of time steps 0 and 1



(d) Cosine similarity between each pair of neighboring time steps

Figure 6: Example word distributions for neighboring time steps of the iraq topic

As can be seen from the figures, the changes for this topic are very low, although it is the one with highest changes among the 104 extracted topics. The differences between time steps 3 and 4 and between 4 and 5 are used to divide the topic at these steps, changing the topic into a new one at time step 4 and again at time step 5. Table 5 shows the differences in the top 20 words of the word distributions for the changing topic. Inspecting the top articles for this topic even reveals an evolvement of the story behind the topic, as the main articles in the first weeks talk about the threat imposed by Iraqi forces and air strike battles, while the last weeks talk about concrete U.S. troop deployment in Kuweit. Table 6 presents the headlines of the corresponding articles for verification. On a small scale, this can be seen as different, more specialized topics underneath a more general one on Iraq war.
For the given data of news articles from 6 weeks, the changes inside the extracted topics are very low. Overall, the second step of the topic detection applied with an artificially low threshold of 0.01 reveals just enough differences between the topics to add 20 more topics to the original 104 topics for a total number of 124 topics in the 6 weeks of August and September 1996 of the RCV1 corpus. The number of topics per time step though still remains the same, 104. This second step of topic detection makes some topics disappear at some time step and exchange them by new ones, emerging out of the corresponding topic from the previous time step. Figure 7 shows

the topic rivers with emerging topics. The Iraq topic is the 6th line from the bottom with the yellow peak at the third timestep.

week 1	week 4	week 5
iraq	iraq	iraq
missile	missile	gulf
attack	gulf	kuwait
saudi	iraqi	military
iraqi	military	missile
military	kuwait	iraqi
gulf	attack	united
force	united	force
united	force	saudi
war	zone	zone
defense	saddam	troops
air	defense	war
kuwait	war	attack
zone	saudi	defense
arab	air	washington
official	strike	arab
arabia	southern	official
strike	official	air
saddam	troops	saddam
southern	washington	arabia

Table 5: Word distribution top word differences for Iraq topic

week 1 - 3	week 4	week 5 - 6
Perry cites two incidents in Iraq no-fly zone.	Iraq fires at U.S. jets, U.S. bombers move closer.	U.S. boosts Kuwait defence by deploying Patriots.
U.S. warns it will protect pilots over Iraq.	U.S. gets Kuwaiti approval for troops deployment.	U.S. ground troops set to fly to Gulf.
Defiant Saddam urges his warplanes to resist U.S.	Kuwait agrees new troop U.S. deployment.	U.S. carrier enters Gulf, troops land in Kuwait.
Saddam urges his warplanes and gunners to resist.	Iraq says fired missiles at US and allied planes.	U.S. sends last of 3,000 ground troops to Gulf.
U.S. launches new attack on Iraq - officials.	Iraq fires at U.S. jets, U.S. bombers move closer.	U.S. declines to rule out Iraq strikes.

Table 6: Article headlines for top documents of Iraq topic

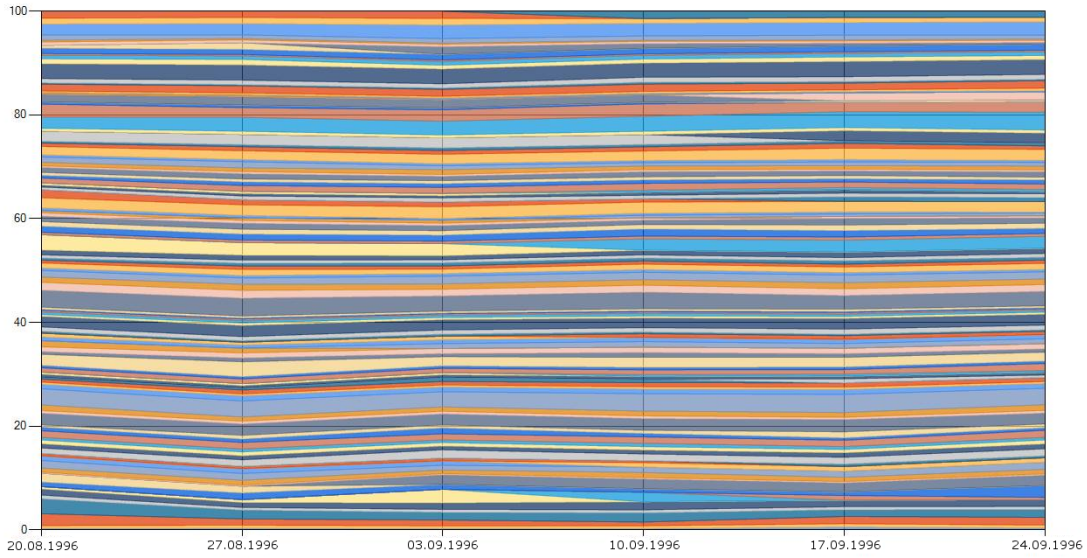


Figure 7: Topic rivers for August and September 1996 for emerging topics

5.1.4. Topic Aggregation

Topic evolvement is not limited to emerging topics. It also includes topic aggregation. This is done in a similar way, using the cosine similarity to identify topics and time steps, where the similarity is above a certain threshold. Points of aggregation, where previously separate topics should become one, are computed this way.
As LDA once more does a good job in clustering, the distance between different topics is rather high. The highest value for the cosine similarity, namely 0.613, can be found at time step 3 for two topics talking about the conflicts, the Iraq was involved in late 1996. Table 7 shows the cosine similarity between both topics along the time line, while table 8 gives an overview of the topic contents, represented by the top 20 words for each topic, at the time point with the highest similarity. One of the topics was already presented in section 5.1.3, as it was also one of the topics with the highest differences among the topic's time steps. The other topic talks about the Kurdish Civil War. Both topics could be clustered further to a more general topic about Iraq politics. But in fact they do present completely different events.

week 1	week 2	week 3	week 4	week 5	week 6
0.559	0.594	0.613	0.568	0.472	0.397

Table 7: Topic cosine similarities for both topics, Iraq and Kurdish Civil War, for each time step

Iraq topic	Kurdish civil war topic
iraq	iraq
missile	iraqi
attack	kurdish
iraqi	iran
military	northern
gulf	turkey
united	turkish
saddam	arbil
force	baghdad
zone	kdp
strike	united
kuwait	kurdistan
air	saddam
saudi	iranian
defense	puk
war	force
southern	official
baghdad	troops
action	border
official	kurds

Table 8: Word distribution top word similarities for both topics, Iraq and Kurdish Civil War, at week 3

5.2. Topic Detection over Time using NMF

In Section 2.3.2, different approaches were described to apply NMF on a dynamic data set that changes over time. Instead of modifying the NMF algorithm, this section elaborates on an algorithm that applies NMF on time steps with arbitrary granularity, and then uses a metric to evaluate the similarity between topics generated for these time steps. This way, topics can also fade away if they do not have a representative at a certain time step, and new topics can emerge if a topic is not similar enough to any existing topic. The similarity metric is intuitively easy to understand and less complex than the related work. It could for example be chosen by an external specialist without an understanding of the NMF algorithm.
First, the algorithm to generate and track the topics is explained, followed up by a description and analysis of the results achieved with this algorithm.

5.2.1. NMF over Time

To perform the basic NMF algorithm, an external library was used. The *Linear Algebra and Machine Learning* library (LAML) covers multiple algorithms and techniques in the field of machine learning, including both LDA and NMF [Qian, 2016]. Documentation and examples how to use the algorithms can be found directly on sourceforge.net/projects/lamal/files/LAML/. LAML makes use of explicitly defined dense and sparse matrices as well as in-place matrix and vector operations, and thus reduces costs on memory and computation time. Because this library is written in Java, we integrated it into a Java project which preprocesses the data, calls the library on the corresponding data of a time step and applies all post-processing steps to generate a topic river over a time span. Figure 8 shows the work flow of this Java project. In the following

paragraphs, this work flow is explained in detail.

Applying NMF
The text files are processed according to the specified interval; for example, if each week is considered a time step, the text files from seven days are processed for one NMF execution. Thus, for each week, a vocabulary is created that contains all unique terms (after pre-processing), and a term-document matrix is created, containing the TF-IDF values of each term for each document. This matrix serves as input for the NMF execution. The number of topics to be generated per time step can be configured. If the number of topics is set too low, important topics might not get caught by the algorithm and the found topics could in fact be a mix of multiple topics with insufficient separation. If it is set too high, topics might get too specific, and the computational time increases as well. Tests found that for weekly processing, 30 topics result in a decent separation of the content without many irrelevant topics.

Generating XML files
The term-topic matrix and the topic-document matrix are converted into Topic instances which store the term and document list, ranked by the TF-IDF values. Usually, the number of available text documents for a day is not always similar for each day - e.g. in the Reuters Corpus, Sundays and Mondays contain far less documents than other days. Thus, it is imported that these values are normalized over all terms/documents to get relative values that can be compared over different time steps without giving advantage to days with more data. All topics of a time step are stored in a Java class and serialized as an XML file. This way, the most time-intensive part of the algorithm can be separated from generating visualization data. The XML files can be loaded into the program as soon as all desired data files are processed.

Topic River
Given all topics for all time steps, a metric has to be defined to identify a time step's topic in the following time steps. A topic that exists over several time steps is defined as a topic wave. The relevance of all topic waves at a given time step sum up to 100%, and the sum of all topic waves over all time steps defines the topic river.

A topic is defined by its terms, and following Zipf's law, only a low number of terms have a significant relevance for a topic. Thus, comparing the top-ranked terms of two topics, and defining a threshold for the similarity to be enough to define one of the topic as successor of the other, creates a wave over time for a certain topic. New topic waves begin at a time step if no previous topic is similar enough. Topic waves fade away if no topic is found at a time step that is similar enough to the topic wave's most relevant terms. This does not mean this topic wave cannot reappear; topics of a time step are always compared against all existing waves, not just the ones that appeared in the last time step. This intuitively makes sense, e.g. for recurrent topics like the FIFA World Cup, that are barely relevant outside of their time frame.

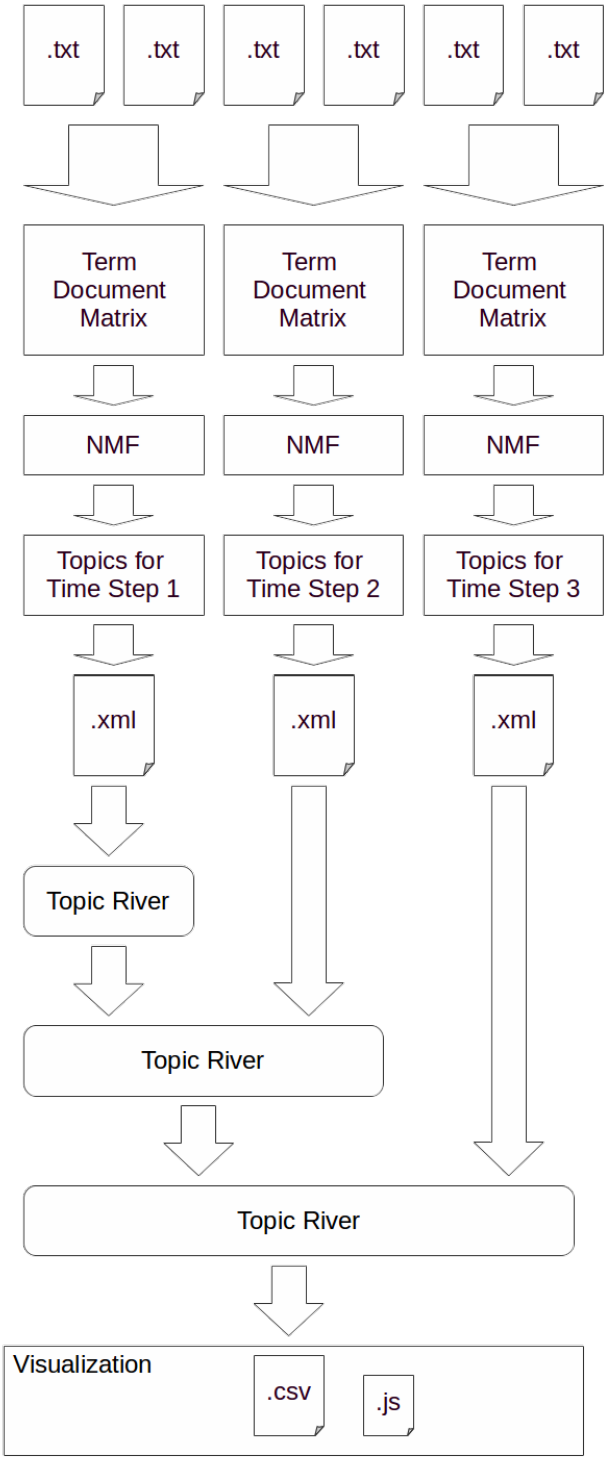


Figure 8: Work flow of the NMF Code. Example with 6 text files which would be processed into 3 time steps.

There are different approaches how to compare the ranked terms of two topics. We chose to compare the 20 top-ranked terms against each other. If a term is found in both topics, its TF-IDF value in both topics is summed up. The total sum is then checked against a threshold. This considers the fact that terms in topics can change slightly (e.g. the focus in a war might shift towards the cities that are currently contested).
We also implemented and tested a similarity metric using the cosine distance of topics. The resulting topic waves were of roughly equal quality, but had the undesirable effect of a higher number of topic waves that only existed in one time step. Thus, we kept the first approach as our similarity metric.

Finally, with the topic river generated from all topic waves over all time steps, we can generate a data file with the format required to visualize this topic river.

5.2.2. Results

With our algorithm, we were able to calculate topics from the annotated Reuters Corpus of the year 20.08.1996 - 19.08.1997. We generated 30 topics for week-wise intervals (7 days of text data). The topic river was visualized via the implemented Charting tool, see Figure 9.

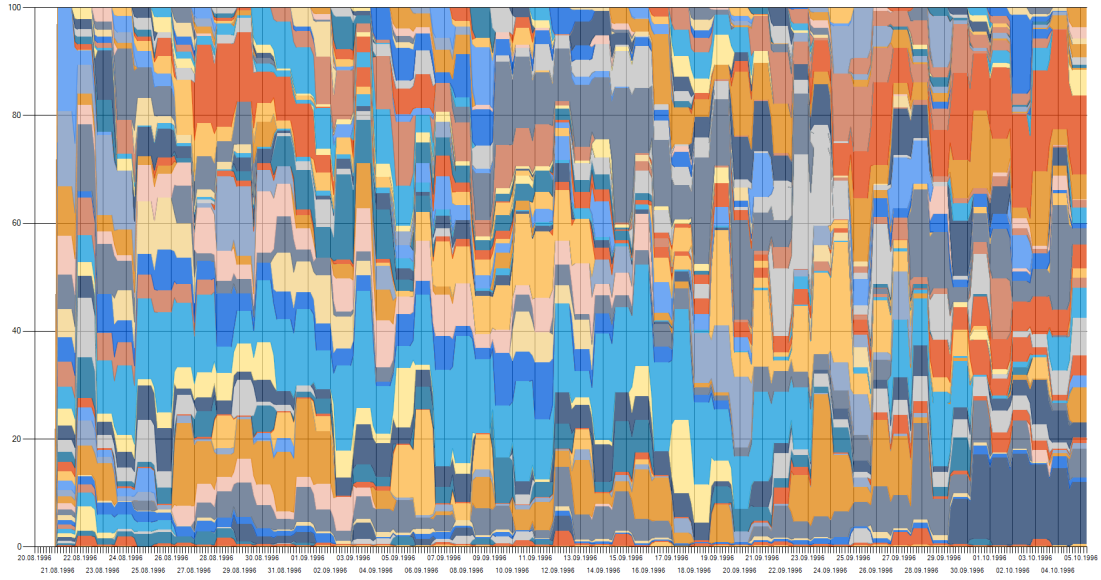


Figure 9: Visualization of topics from 20.08.1996 to 19.08.1997.

Although these tags present a way to evaluate the created topics, it is difficult to decide on a precise method to do so, because tags are defined for each article and represent 103 different rather general topics, that are not easily comparable to the topics generated by the NMF over time algorithm, which are defined by distribution of multiple terms. Additionally, the topic's relative weight is difficult to compare against the weight of tagged documents. Topic descriptions and their importance in a time frame are subjective criteria; an evaluation by external judges would be optimal, but out of the scope for this project. In Section 7, we describe a concept for an evaluation system to measure the quality of this algorithm.
Instead, we researched the most important events of 1996/1997 and their approximate relevance

over time, and compared them against the closest topic found by our algorithm.

We also calculated topics for the whole year of 2015, which is not annotated, but represents more recent data which can be easier checked manually for coverage of all recent events from that year. This text data was crawled from the Reuters web page, see Section 3.2. Again, we generated 30 topics for week-wise intervals and visualized the results, see Figure 10.

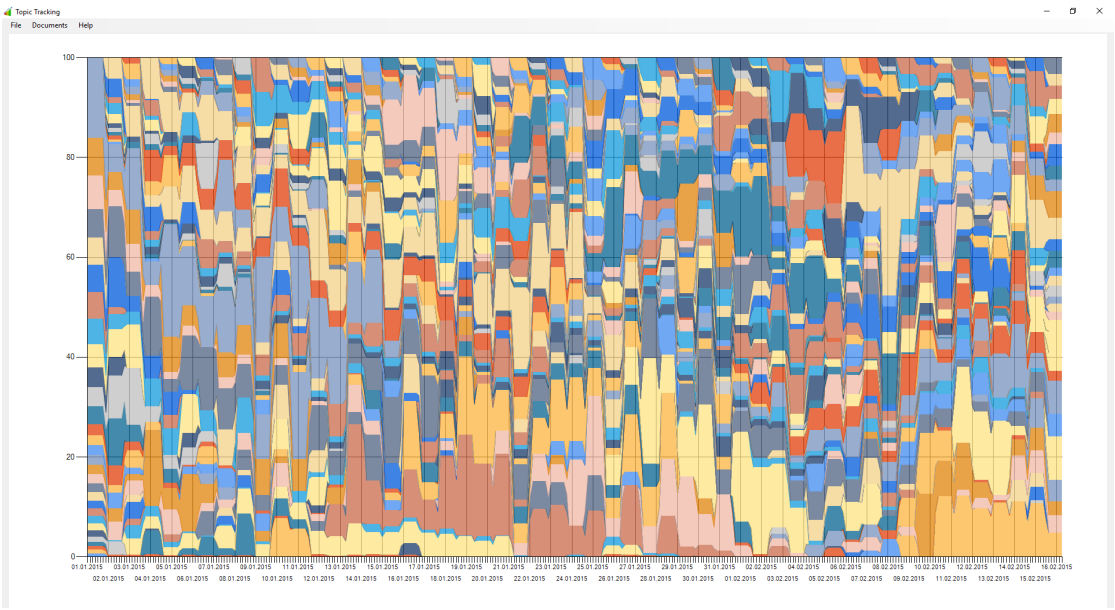


Figure 10: Visualization of topics from the year 2015.

Greece Financial Crisis	Charlie Hebdo Attacks	FiFA Corruption	Ukraine Crisis	VW Emission Scandal
greeks	attacker	fifa	ukraine	vws
greece	paris	soccer	russia	piech
syriza	charlie	cups	russians	winterkorn
tsipras	hebdo	blatter	sanctions	ceos
euros	police	corruption	moscow	porsche

Table 9: Topics and their most relevant terms in 2015

Jan 08	Jan 15	Feb 12	Jun 25
attacker	police	attacker	tunisia
paris	charlie	houthis	attacker
hebdo	french	police	tourists
police	attacker	boko	tunisians
french	hebdo	haram	hotels
islamic	paris	killed	tourism
france	muslims	islamic	beaches
killed	cartoons	yemen	islamists
muslims	islamists	militant	killed
kouachis	killed	libya	sousse

Table 10: Development of a topic: terror attacks in Paris, Jemen, Tunesia

Table 9 shows different prominent topics of 2015, that were reflected in topics generated by our algorithm. Table 10 shows an example for a (short) topic wave, present in four time steps. The first two topics in January follow the terror attack in Paris on the magazine staff of *Charlie Hebdo* at January 7th. The third topic is in February and covers a terror attack of the terror group Houthis, located in Yemen. Finally, the last topic represents a terror attack on tourists in Tunesia in June 2015. The similarity between the topics within the topic wave can thus be confirmed by the actual events.

Looking at the overview of all topics, one can observe that the most relevant topics usually are of business-related nature, made up of general financial terms like "stock market", "import" and "share", and not very specific to the corresponding time period. Normally, TF-IDF values should prevent terms that are used often in general from being that significant. This appears to be an attribute of the test data, the Reuters Corpus, that has a dedicated section of articles that contain financial content, thus creating high TF-IDF values for these terms that are often used in this small section of documents. The same problem on a smaller scale exists for weather forecasts.

5.3. Comparison of LDA and NMF Approaches

We developed dynamic implementations of the two most promising static topic extraction algorithms; LDA and NMF. In LDA, the dynamic topic model implementation was used, while NMF was applied on segments of the data and then joined using a similarity metric. In many cases, both algorithms produced very similar topics with similar word distributions.
The LDA algorithm is much slower than the NMF algorithm, because it is not able to reduce the vocabulary as much as the NMF algorithm using preprocessing techniques. The topics in LDA change only little, and topics do generally not emerge or disappear over the time span, while NMF topics are rather split up into a lot of small waves and could be more generalized. All in all, both algorithms show the powerful ability to develop comprehensive topics out of a large text data set. The figures 7 and 9 can be used for visual comparison of the topic evolvement in general, taking the different time spans and the different number of topics into account.

6. Visualization

The final part of the project and the most important one after the topic detection itself is a good visualization of the result data, so it becomes easier for humans to understand what to make of the results and how to interpret the data.
The easiest and most readable way of visualization we came up with is the use of a normalized stacked area graph. That means, that we have for each topic a value for each day which represents the prevalence of that topic. These values are normalized for each day and then displayed in a graph. The normalization is for the graph height to be independent of the number of documents for each day, as lesser documents would normally result in smaller values for the topics, shrinking the total height of the graph.
The layout of the graph should be as follows: The values on the X-axis are the dates; in intervals of single days. The Y-axis goes from 0 to 100, representing the prevalence of the topics in percentages for the days. Essential features for the graph is to be as interactive as possible. There should be a zoom function, so users can focus on single days to analyze topics in a certain range. There should be tooltips to inform the user about the topic he is looking at and the prevalence of that topic. Furthermore, the topic-document connection should be investigated. To each topic, there is a number of documents for each day, that count towards the prevalence of that topic. There should be a function to view these.

6.1. Rickshaw

Rickshaw[5] is a JavaScript-library that is based on the popular, also JavaScript-library called D3[6], which is a library designed to visualize data using common HTML and CSS techniques as well as scalable vector graphics. It can be used to create graphs and all kinds of diagrams dynamically with a given set of input data. Rickshaw makes use of D3 in order to facilitate the creation of time-series graphs, as required for this project.
Using Rickshaw, it is easy to get an initial graph setup, however it is not as easily customizable as it may seem. Most of the information to work with it must be taken from the Internet or tutorials, since there is no comprehensive documentation to be found.
Furthermore, it became very hard to implement all the features needed. Firstable, the library is obviously bugged when it comes to the tooltip functionality, as the tooltips are only shown for the topmost serieses of a graph, but serieses that lay deeper are ignored, which is a real problem. That problem also extends to hittests, that means clicking somewhere on the graph gives faulty values for the series, that was clicked. This would have been essential to implement the topic-document connection. Zooming is another thing, that is not supported in the library and, inherent to JavaScript, it is not possible to directly access the file system due to security restrictions, which would require to load the entire corpus in order to view connected documents directly.

[5] http://code.shutterstock.com/rickshaw/
[6] https://d3js.org/

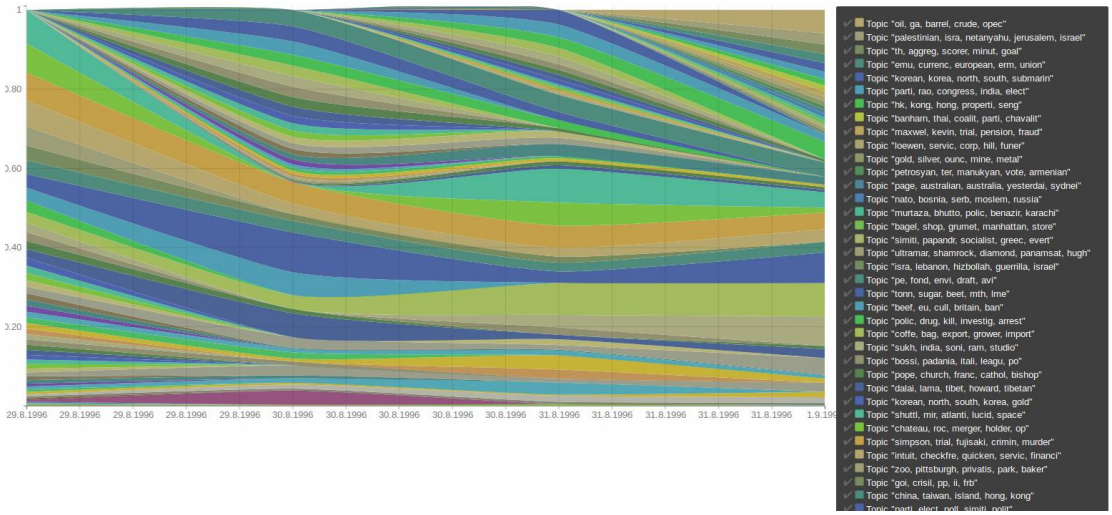
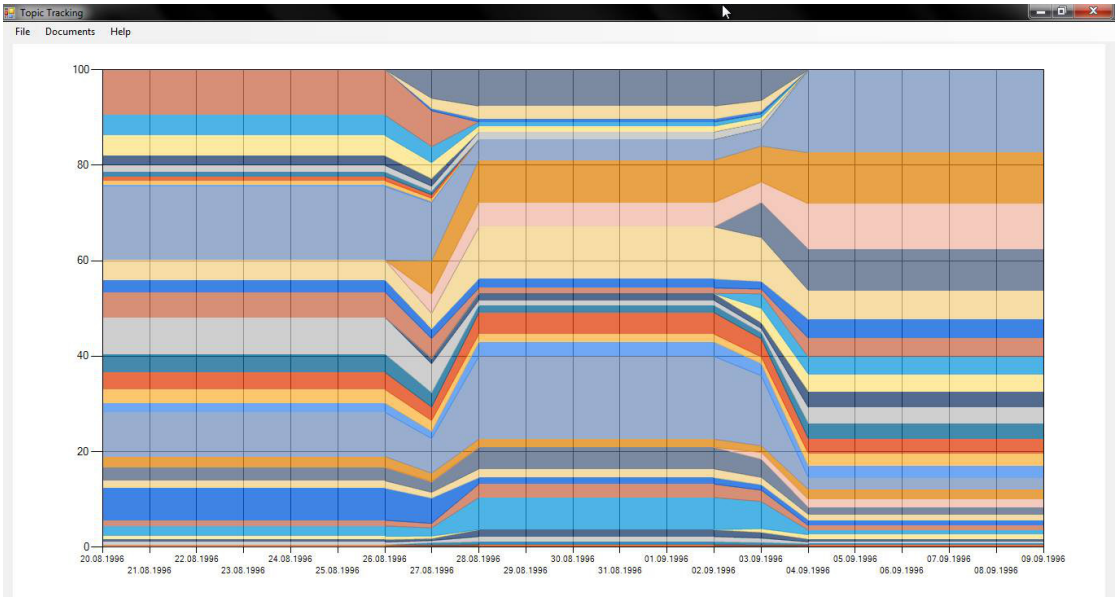


Figure 11: Early Rickshaw chart with data of a few days and 104 topics

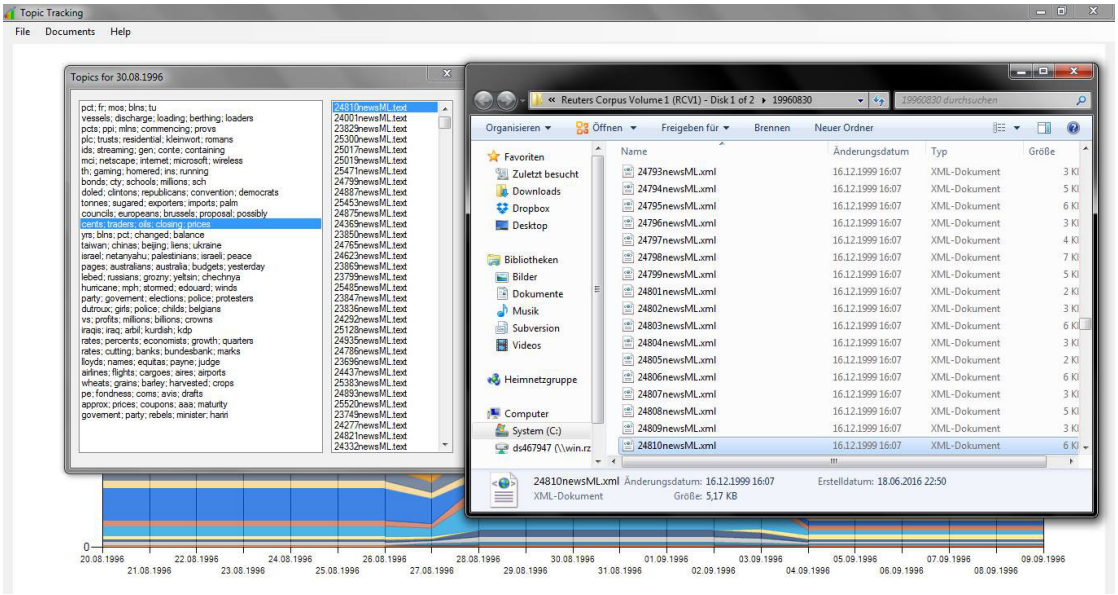
6.2. .NET Charting

Due to the aforementioned restrictions, we moved on to the DataVisualization.Charting library of the .NET framework. Not only is there a comprehensive documentation[7] and more predefined functionality for the chart controls, but it is also widely enough used so that common questions can be found answered on the web. The Charting-library is used in conjunction with Windows Forms[8], where, similar like a dynamic website, it is easy to implement event-driven techniques. In order to start the visualization, a file with topic data and a file with datapoints and documents must be selected. In the topic data file, all the topics that occurr over the dataset must be listed; their order is important. In the datapoints file, the first line is the start date, for all the consecutive lines one day is added. In each line, there is the prevalences of the topics. For each topic, there is one entry, with irrelevant topics having a value of 0. After each value for a topic, there follows a list of all the documents that add to the prevalence of that topic ordered by their importance. The program then builds the chart using the data given. It is interactive in the sense, that the graph can be zoomed to only display certain days; there are tooltips showing the topic names and the prevalence of the current topic. Also, by right-clicking the graph, the user has the opportunity to explore the documents the data was created by. A right-click on the graph opens a window displaying all of the topics of the day clicked with the topic highlighted, that the cursor was hovering over. The user can then scroll through the list in order to see the corresponding documents of that day adding to the prevalence of the selected topic. By selecting a parent folder of the document corpus, the files can be searched on the user's local system and displayed by double-clicking a document name. The file extension is ignored in that case.

[7] https://msdn.microsoft.com/en-us/library/dd456632.aspx
[8] https://msdn.microsoft.com/de-de/library/dd30h2yb(v=vs.110).aspx



(a) Example chart with data for 3 weeks



(b) Finding a document in the file system

Figure 12: Visualizing data that was generated through NMF

6.3. pyLDAvis

While our visualization so far gives an insight into the topic flow and development we can't really gather more information about it. So we decided to take a look into another part of visualization to be able to show the topics in a detailed way.
The library we chose is the called pyLDAvis[9] and is designed to help users interpret the topics in a topic model extracted from a corpus of text data. The package creates an interactive web-based visualization to get an idea of topic and document distribution. A disadvantage for our situation is the fact that it is intended to be used within a tool called Jupyter notebook so it would be exhausting to make the connection between the two parts of the visualization. Due to the occurence of numerous errors there was a lot to overcome but one error was resistent. Even after hours of bugfixing the installation of pyLDAvis wasn't working. Using four different computers in the process, the error was obvious but not expected. The operating system on all of these computers is Windows and pyLDAvis can't work on this operating system. In a side node of a comment by a developer of a subpackage called scikit-bio it is stated that scikit-bio does not currently support Windows. The developer team is planning to add Windows support during the code sprints at the end of the SciPy conference in July.[10] Still we investigated this kind of visualization to complement the already described normalized stacked area graph.
The python library based on the principle "bring your own model". In order to visualize it properly, we need to provide the topic-term distributions, the document-topic distributions, the document length, the vocabulary and the term frequencies.
In the picture provided below you can get a basic understanding on how the visualization works. On the left hand side, there is a so called "Intertopic Distance Map" which shows the different topics (in this case 20 topics) and how they can be related to each other. If two topics are near to each other or even intersect, they are very similar and even partially the same – in case of the intersection – while two topics with a high distance in between are unlike. An interesting fact to notice is the sub topic 3 of 1. Since the topic is completely in the range of the other topic, it is a true sub topic. This means that every term occurring in topic 3 is also occurring in topic 1. On the right hand side, there are more information about the topic distribution. If none topic is selected on the left, it shows the top 30 most salient terms. The saliency of these terms are calculated by a formula created by Carson Sievert and Kenneth E. Shirley.[11] The terms are listed and their overall frequency is shown blue in the graphic. By selecting a topic the list changes to the top 30 of relevant terms for the topic selected. The top 30 terms are chosen via their frequency within the selected topic and the list on the right hand changes in so far that there is another color in contrast to the blue one to display this estimated term frequency. In summary, this second part of the visualization is not able to display topic development over time but the terms of which a topic is built upon. Additionally a similarity measurement between the topics is provided via multidimensional scaling.
This kind of visualization would add up great to the existing graph and should be kept in mind for the future work especially when there will be the much needed support for every operating system.

[9] https://github.com/bmabey/pyLDAvis
[10] https://github.com/biocore/scikit-bio/issues/1382
[11] http://nlp.stanford.edu/events/illvi2014/papers/sievert-illvi2014.pdf

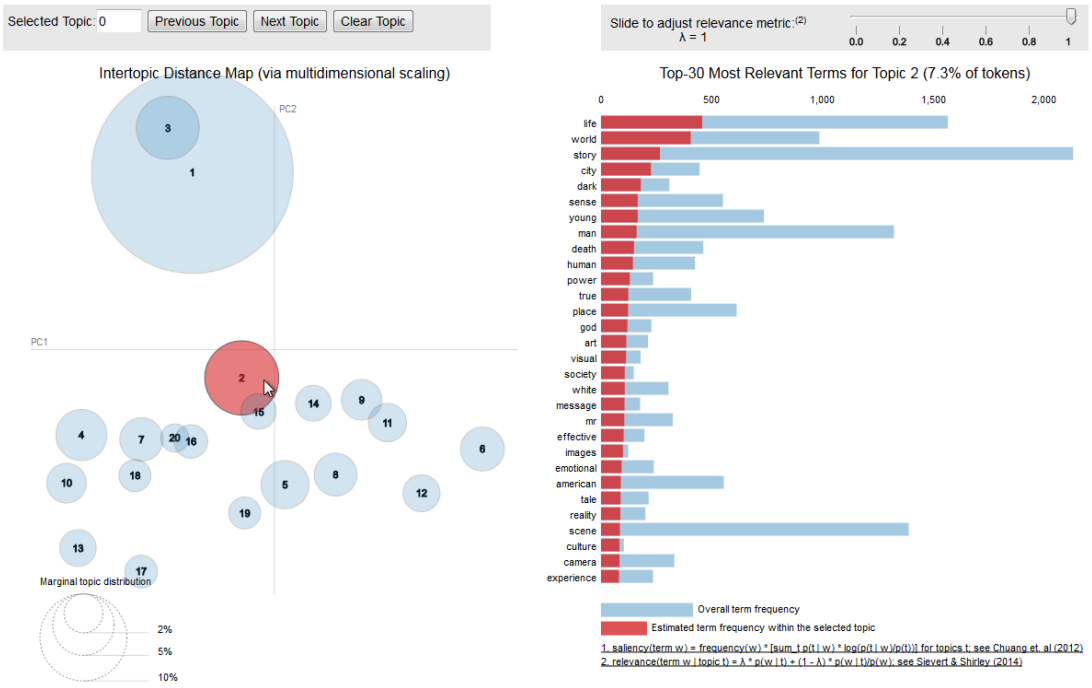


Figure 13: Example visualization with movie review data

7. Conclusion and Future Work

This report presents two approaches to detect evolving and dynamically changing topics in a large document collection, and visualizes them in the form of a topic river, allowing for easy and direct association between topics, terms and documents.

The LDA dynamic topic model was researched and applied to the news articles of the 6 weeks in August and September 1996 of the Reuters corpus RCV1. After applying careful preprocessing, it was possible to identify some of the main events happening at that time. The Kurdish civil war or the horrible crimes in Belgium were clearly reflected in the extracted topics. A matching with the RCV1 annotated topics can not be found, although the topics reveal a similar behavior in their evolvement. The dynamic topic model itself works with a fixed number of topics and does not present much evolvement of topics directly. To identify more details and find possible turning points of topic contents according to the differences among the word distributions of a topic at different time steps, a second step of comparing the word distributions inside each topic at each time step is added. The cosine similarity is used to compute these turning points, where a topic's word distribution is regarded as having changed enough to declare a new topic. As the corresponding differences are very small for the given data, a proof of this concept was implemented using an artificially low difference to produce such an evolvement. The results should be more distinguishable and clearer when applied to the whole corpus. Similar to the topic emergence, topic aggregation is presented. Here the similarities between different topics at all time steps are compared to find points in time, where different topics are similar enough to be aggregated into a common one. Corresponding to the topic emergence, the clustering of

LDA produces clearly separated distances between topics as it produces high similarity inside the topics. Therefore, concrete results impose doubts, if the topics really should become one, or if they clearly present different events. The question, how to combine the emerging topics with the aggregated ones still has to be answered. If a topic is emerging from itself and aggregating with other topics at the same time, as it was seen here, how can this be combined properly ? More open question would be, how to define the aggregated topic in terms of the word distributions with regard to the original ones and how to properly visualize aggregating topics on a 2D canvas, if topics merge multiple times with different topic rivers flowing in completely different parts of the stacked graph. Summarizing the LDA based approach, the dynamic topic model produces topics, that are on a more generalized level, at least when the same number of topics is chosen, similar to the annotated topics. A high frequency of topic evolvement can not be seen here.

For NMF over time, the NMF algorithm was applied on separate time steps of the data and then connected using a similarity metric, thus creating a topic river with evolving and emerging topics. By using extensive cleaning of the vocabulary during pre-processing, a fast data processing algorithm was developed that is able to process a year of data with around 3000 text files per day and 50 topics generated per month in circa 15 hours. The generated topics can easily be identified by their most relevant terms and associated with events happening in the corresponding time period.

The highly significant financial topics that are not very specific to the respective time frame, as mentioned in Section 5.2.2, are a problem and could maybe be prevented by extending the stop word list with these general business terms. A closer examination of the documents making up these topics would therefore be advisable.

So far, the NMF over time implementation was not evaluated by a quality metric, as explained in Section 5.2.2. As a first approach, the tags from the Corpus 96/97 documents that are most relevant for a topic (e.g. the tags of the ten best documents) could be compared against the best terms of the topic. To compensate for the more general tags that are assigned to the documents, a categorizing system like the WordNet ontology could be used to measure the distance between terms and tags horizontally (e.g. synonyms) and vertically (hierarchical distance), allowing a more granular measurement.

The visualization with a stacked graph over time already works well. What can be improved is the performance for large data sets in a way, that for cases when not everything can be displayed at once, approximations to the original series are built, that are less expensive to decrease load time. Other additions can be flexible visualizations with user-defined time spans to display or statistics for single topics (even if some tend to be very short-lived). Other improvements include more colors for the graph palette if there are too many topics to display at once, and, if needed, smoothing of the graph lines.

In this case, the Reuters Corpus was used as test data, but the developed systems are dynamic and reusable and can take an arbitrary corpus of text data to extract a topic river. To categorize these text files into a time frame correctly, the files have to be put in the corresponding format (see Section A for a manual on both the LDA and the NMF approach). Topics so far are mainly identified by their most relevant terms, which already gives a sufficient overview on the topic's content. However, for a more comprehensive and sophisticated description of a topic, it is possible to create story lines or summaries by applying natural language processing techniques on the most relevant documents of a topic.

Appendices

A. Technical Documentation

A.1. LDA Topic Detection

The technical process for the dynamic topic model includes a JAR file `LDAProcessor.jar` for the main pre- and postprocessing tasks and several additional processing scripts for minor tasks. The `LDAProcessor.jar` has different operation modes defined on command line and input parameters defined in a configuration file placed and called `resources/config.properties`. The operations and the parameters are described in the following sections. The JAR depends on the Stanford CoreNLP version 3.6.0 [12].

A.1.1. Corpus Preprocessing

The preprocessing technically is divided into two steps. First the vocabulary is generated from the original documents. For debugging and to better understand the results, the processed article text is written to files in an output directory together with the vocabulary file. The command line call for the main JAR is `java -jar LDAProcessor -generateVocabulary`. The parameters are described in section A.1.4.
Then the word counts are computed from the previously written text files and the vocabulary file. The produced output file contains one line for each document consisting of the word ids from the vocabulary and the word count for this document, if the word is present at all. This output can directly be used as input for the LDA or DTM implementation. The command line call for the main JAR is `java -jar LDAProcessor -generateWordcount`. The parameters are described in section A.1.4.

A.1.2. Topic Extraction

The topic detection itself is performed using the implementation of the dynamic topic model provided by David Blei on his homepage [13]. The source code is written in C and has to be compiled accordingly. To compile successfully, the code has to be fixed. An include statement is missing in the file `main.h`: `include <iostream>`. There is a sample script contained in the main directory explaining the useage. The call from commmand line in this case is `./main -ntopics=104 -mode=fit -rng_seed=0 -initialize_lda=true -corpus_prefix=output/rcv1 -outname=output/out -top_chain_var=0.005 -alpha=0.01 -lda_sequence_min_iter=6 -lda_sequence_max_iter=20 -lda_max_em_iter=10`. This produces output files for the topic and word distributions in the given output directory, which are further processed as explained in the next section.

A.1.3. Topic Postprocessing

The output of the DTM software first is consolidated to the regular LDA output consisting of the gamma values, i.e. the topic distributions for the documents, and the beta values, i.e. the word distributions for the topics and furthermore at each time step in the case of the dynamic topic model. These tasks are achieved by some scripts following the descriptions from the original software package.
The python 3 script `topicsFromDTM.py` generates the beta values from the time step files of the

[12] http://stanfordnlp.github.io/CoreNLP/download.html
[13] https://github.com/blei-lab/dtm

dynamic topic model. It takes the output directory of the previous DTM call and the number of time steps as input parameters. The usage from command line is `python3 topicsFromDTM.py output/out 6 > beta.txt`, out would be the directory and 6 is the number of time steps. The script's output is written to the file `beta.txt`.
The gamma values are extracted from the DTM output using the R code example from the description, parameters for the DTM output gam.dat and the number of topics have to be adjusted accordingly :

```
a = scan("out/lda-seq/gam.dat")
b = matrix(a, ncol=104, byrow=TRUE)
rs = rowSums(b)
e.theta = b / rs
write(t(e.theta), "gammas.txt", ncolumns=104)
```

The consolidated word and topic distributions are then further processed using the main JAR to generate output for further inspection of the data and to attain the correctly formatted data for the visualization and additional output of word and topic distributions for further processing. The command line call for the main JAR is `java -jar LDAProcessor -evaluateDTM`. The parameters are described in section A.1.4.
The cosine similarities for topic emergence and aggregation can also be computed separately. The command line call for the main JAR is `java -jar LDAProcessor -evaluateSimilarities`.

A.1.4. Main Configuration File

The application is configured in detail by a configuration file. This configuration file is formatted using a simple syntax of key/value pairs noted in one line separated by an equal sign.
The parameter "CorpusPath" defines to path to the top folder of the document collection, that is going to be processed. All xml files underneath will be processed.
The parameter "NERCategories" defines the named entity recognition categories, whose words should be excluded during the vocabulary generation. It is a list of names according to the Stanford NLP definitions.
The parameter "StopwordsFile" defines the path and name of the file containing the stop wird list, one word per line.
The parameter "DictionaryFile" defines the path and name of the spell correction dictionary containing correct word forms, one word line.
The parameter "ResultDir" defines the path to the output directory of the application. All output files will be written here. intermediate output and input files, like the produced vocabulary files are expected to be present here.
The parameter "VocabularyFilenameBase" defines the base part of the vocabulary file. It will be appended by a filename ending or optional additional file name parts for different debugging vocabularies.
The parameter "MetaDataFilename" defines the name of the intermediate output file containing meta information extracted from the corpus. This is of course specialized for the Reuters Corpus document format.
The parameter "TopicsPerDocsFilename" defines the file name of the topic distributions extracted from LDA, i.e. the gamma values.
The parameter "WordsPerTopicsFilename" defines the file name of the word distributions extracted from LDA, i.e. the beta values.
The paramter "NumTimesteps" defines the number of time steps used in the DTM topic detection step.

The parameter "intraTopicSimilarityTreshold" defines the threshold for the topic emergence.
The parameter "interTopicSimilarityTreshold" defines the threshold for the topic aggregation.
The parameter "NumTopWords" defines the number of top words for each topic, that have to be taken into account for the evaluation and visualization output.
The parameter "NumTopDocs" defines the number of top documents for each topic, that have to be taken into account for the evaluation and visualization output.
The parameter "OutVisDataFilename" defines the output file name for the visualization data. The file is formatted for the .NET Charting visualization.
The parameter "OutDocsPerTopicFilename" defines the output file name for the top documents of each topic. Each line corresponds to one topic, containing the document names in descending order of their score for this topic.
The parameter "OutWordsPerTopicFilename" defines the output file name for the top words for each topic. Each line corresponds to one topic, containing the words in descending order of their score for this topic.
The parameter "OutTopicScoresPerTimestepFilename" defines the output file name for the relative topic scores per time step. Each line corresponds to one topic, containing the normalized topic score for each time step.
The parameter "OutTopicAggregationFilename" defines the output file name for the topic aggregations time step. Each line corresponds to one time step, each entry corresponds to the topic Id of the most similar topic for the topic of the column Id, or -1 if no topic exceeds the interTopicSimilarityTreshold.
The parameter "InterTopicSimilaritiesFilenameBase" defines the output file name base for the inter topic similarities. It is appended by the time step numbers. Each line corresponds to one topic, containing the cosine similarity values for all other topics.
The parameter "IntraTopicSimilaritiesFilename" defines the output file name for the similarities between time steps of topics. Each line corresponds to one topic, containing the cosine similarity values for the word distributions of this topic between all time steps.

A.1.5. Benchmark

The computation time for the main step, the dtm application with the parameters given in section A.1.2, given the data of the first two month of the corpus takes almost 3 full days on an i7 mobile CPU of the 4700 series. Based on this time, the computation of the full corpus can be estimated as taking 30 days on the same machine. A machine designed for heavy computation can certainly do better, but was not available.
One of the reasons for choosing the original C-implementation was based on the higher efficiency and speed of such implementations compared to e.g. Java implementations. There also is a parallelized version of the dtm code. But it took already quite some time in the beginning of the project to fix the single error, mentioned in section A.1.2, in the single core version of the undocumented code. The error list of the parallel version counts for 33 errors. If one would want to use this code, an expert in C programming would probably come in handy to fix all of them in a reasonable amount of time.

A.2. NMF Topic Detection

A.2.1. NMF Code Manual

This section describes how to use this paper's NMF code basis to generate and visualize dynamic topic data from text data. There are two JAR files delivered with this paper for this purpose.

Alternatively, the code is accessible on the public repository github.com/Runner-Runner/dke-topictracking, which also contains the JAR dependencies in the `import` folder for convenience.

The first JAR file, `NMFTopicExtractor.jar`, takes 7 parameters:
`java -jar NMFTopicExtractor <interval> <sequence> <dateformat> <startdate> <input-directory> <output-directory> <topics>`

1. interval (int) - defines the day range of each interval
2. sequence (int) - defines number of intervals which will be extracted
3. dateformat (String) - defines the date format for documents directories, e.g. yyyyMMdd
4. startdate (String) define the start date (in given date format)
5. input-directory (Path) takes a directory containing directories with the text files as input; the inner directories define the time steps (for example, days) to which the contained text files should be assigned.
6. output-directory (Path) directory to store extracted xml files
7. topics (int) - defines the number of topics to be generated in each time step

The parameters allow, for example, to generate topics for each week (if text files are in day-wise directories, interval = 7) and for 4 weeks in total (sequence = 4). Executing this JAR file serializes all TopicTimeStepCollection instances (one for each time step) into XML files and writes them into the specified output directory.

The second JAR file, `TopicTimeTracker.jar`, takes 3 parameters:
`java -jar TopicTimeTracker <inputpath> <writeCSV> <writeJSON>`

1. inputpath (Path) - defines the path to a directory where the XML files generated by NMFTopicExtractor are contained. These XML files, each representing the topics of a certain time span, are then loaded into a topic river.
2. writeCSV (int) - if "1", CSV visualization data will be written. This is the more advanced visualization tool.
3. writeJSON (int) - if "1", JSON visualization data will be written.

The input path can be the output path given to the first JAR file. This JAR file loads the serialized classes, which store the topics for each single time step, and generates a topic river out of them (topic waves over time). It then generates the specified visualization data files and writes them to the JAR file location. These files can now be used as input for the visualization tool.

References

[Blei and Lafferty, 2006] Blei, D. M. and Lafferty, J. D. (2006). Dynamic topic models. In *Proceedings of the 23rd International Conference on Machine Learning*, ICML '06, pages 113–120, New York, NY, USA. ACM.

[Blei et al., 2003] Blei, D. M., Ng, A. Y., and Jordan, M. I. (2003). Latent dirichlet allocation. *J. Mach. Learn. Res.*, 3:993–1022.

[Cao et al., 2007] Cao, B., Shen, D., Sun, J.-T., Wang, X., Yang, Q., and Chen, Z. (2007). Detect and track latent factors with online nonnegative matrix factorization. *IJCAI*, page 2689–2694.

[Fiscus and Doddington, 2002] Fiscus, J. G. and Doddington, G. R. (2002). Topic detection and tracking. chapter Topic Detection and Tracking Evaluation Overview, pages 17–31. Kluwer Academic Publishers, Norwell, MA, USA.

[Lee and Seung, 1999] Lee, D. D. and Seung, H. S. (1999). Learning the parts of objects by non-negative matrix factorization. *Nature*, page 788–791.

[Lewis et al., 2004] Lewis, D. D., Yang, Y., Rose, T. G., and Li, F. (2004). Rcv1: A new benchmark collection for text categorization research. *J. Mach. Learn. Res.*, 5:361–397.

[Manning et al., 2014] Manning, C. D., Surdeanu, M., Bauer, J., Finkel, J., Bethard, S. J., and McClosky, D. (2014). The Stanford CoreNLP natural language processing toolkit. In *Association for Computational Linguistics (ACL) System Demonstrations*, pages 55–60.

[Navarro, 2001] Navarro, G. (2001). A guided tour to approximate string matching. *ACM Comput. Surv.*, 33(1):31–88.

[Porter, 1980] Porter, M. F. (1980). An algorithm for suffix stripping. *Program*, 14(3):130–137.

[Qian, 2016] Qian, M. (2016). Linear algebra and machine learning version 1.6.5. https://sourceforge.net/projects/lamal/files/LAML/.

[Saha and Sindhwani, 2012] Saha, A. and Sindhwani, V. (2012). Learning evolving and emerging topics in social media: a dynamic nmf approach with temporal regularization. *Proceedings of the fifth ACM international conference on Web search and data mining*, page 693–702.

[Scholtes and van Cann, 2013] Scholtes, J. C. and van Cann, T. H. (2013). Improving machine learning input for automatic document classification with natural language processing.

[Smeets, 2015] Smeets, J. (2015). Applying text-mining in the archives of the stedelijk museum amsterdam. Technical report.

[Tannenbaum et al., 2015] Tannenbaum, M., Fischer, A., and Scholtes, J. C. (2015). Dynamic topic detection and tracking using non-negative matrix factorization.

[Wang et al., 2008] Wang, C., Blei, D. M., and Heckerman, D. (2008). Continuous time dynamic topic models. In McAllester, D. A. and Myllymäki, P., editors, *UAI*, pages 579–586. AUAI Press.

[Wang et al., 2011a] Wang, F., Li, P., and König, A. C. (2011a). Efficient document clustering via online nonnegative matrix factorizations. *SDM*, 11.

[Wang et al., 2011b] Wang, F., Tan, C., Li, P., and König, A. C. (2011b). Efficient document clustering via online nonnegative matrix factorizations. *Eleventh SIAM International Conference on Data Mining*.

[Wang and McCallum, 2006] Wang, X. and McCallum, A. (2006). Topics over time: A non-markov continuous-time model of topical trends. In *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '06, pages 424–433, New York, NY, USA. ACM.

[Wei et al., 2007] Wei, X., Sun, J., and Wang, X. (2007). Dynamic mixture models for multiple time-series. In Veloso, M. M., editor, *IJCAI*, pages 2909–2914.