Hochschule für Angewandte Wissenschaften Hamburg
Hamburg University of Applied Sciences

Bachelorarbeit

Matthias Nitsche

Continuous Clustering for a Daily News Summarization System

Fakultät Technik und Informatik
Studiendepartment Informatik

Faculty of Engineering and Computer Science
Department of Computer Science

Matthias Nitsche

Continuous Clustering for a Daily News Summarization System

Bachelorarbeit eingereicht im Rahmen der Bachelorprüfung

im Studiengang Bachelor of Science Angewandte Informatik
am Department Informatik
der Fakultät Technik und Informatik
der Hochschule für Angewandte Wissenschaften Hamburg

Betreuender Prüfer: Prof. Dr. Michael Neitzke
Zweitgutachter: Prof. Dr. Olaf Zukunft

Eingereicht am: 06. February 2016

Matthias Nitsche

Thema der Arbeit
Continuous Clustering for a Daily News Summarization System

Stichworte
Clustering, Cluster Analyse, Dokument Clustering, Vector Space Model, Partitionelles Clustering, Hierarchisches Clustering, Probabilistic Topic Modelling, Text Zusammenfassung, Text Mining, Data Mining, Machinelles Lernen, Unüberwachtes Lernen, Information Retrieval

Kurzzusammenfassung
Für eine Maschine ist es schwer, ohne die Supervision eines menschlichen Expertens text zu interpretieren. Techniken des Text Minings und clustern als Ansatz von unsupervisierten Lernen, um Text aus Zeitungen in Kategorien und zu Ereignissen in der realen Welt zu gruppieren, ist zentral in dieser Arbeit. Zusätzlich, wurde ein funktionierendes Datenverarbeitungssystem, zum herunterladen und verarbeiten von Zeitungsartikeln, entwickelt, um clustering Algorithmen eine Grundlage zu geben. In kurz, die präsentierten Selektionsstrategien und clustering Algorithmen haben eine ähnliche Wirkung.
Matthias Nitsche

Title of the paper
Continuous Clustering for a Daily News Summarization System

Keywords
Clustering, Cluster Analysis, Document Clustering, Vector Space Model, Partitional Clustering, Hierarchical Clustering, Probabilistic Topic Modelling, Summarization, Text Mining, Data Mining, Machine Learning, Unsupervised Learning, Information Retrieval

Abstract
Interpreting and summarizing textual content without the supervision of human experts is subject of this thesis. Using techniques of text mining and document clustering as an approach of unsupervised machine learning, grouping textual content of online newspaper articles, into coherent categories and real world events is subject of this thesis. Additionally, building a functioning data pipeline for scraping and preprocessing newspaper articles, feeding clustering algorithms, shows promising results. In short, the presented feature selection and clustering strategies yield similar effects.

Contents

List of Tables

List of Figures

1 Introduction

"Where there is matter, there is geometry."

Johannes Kepler

Text mining and natural language processing have long been studied in the field of artificial intelligence. Within the last decade models that were computationally expensive were reopened for discussion. Machine learning aids in this task by approximating models for structured and unstructured data. Recognizing patterns, retrieving information or classifying content of interest for huge datasets, often scaled over thousands of machines. This process is called knowledge discovery. By continuously clustering daily news articles to summarize them the task at hand is simple. Reduce the noise surrounding textutal documents by presenting the useful fraction of information. Connect documents in their topical logics to enhance the understanding and reasoning about them. Mining documents, preprocessing them into suitable representations and grouping their form to detect underlying patterns that connect documents is the primary goal of this thesis.

It goes without saying that newspaper articles are created by humans. While automation and easy retrieval can be of utmost importance, news are created by authors, journalists, generally professional writers. The whole work of text mining in that sense, is to scale suitable algorithms to datasets, that are too enormous to be comprehended by a single person. Without authors there is no text, therefore the highest good at hand are the documents by the authors.

In the following we will examine the field of unsupervised learning with respect to clustering examplified by a news clustering system. The objective is to build parts of an automatic summarization system that scrapes newspapers, preprocesses the content and groups them into clusters for summarization purposes.
Also in the name of Johannes Keppler "Where there is matter, there is geometry.", lining up perfectly that documents and words have matter and geometry is there for the rescue. More on this, later.

1.1 Machine Learning

Artifical intelligence is the field of study asking the question: Are computers capable of intelligent behaviour? By intelligent it is refered to how an agent can be programmed to react and act to an environment by maximizing the chances of success in a particular task. Machine learning is a subfield of artifical intelligence.

A computer program is said to learn in the context of performing a task if its performance with respect to some measure improves with experience. In the context of this thesis machine learning is closely related to *pattern recognition* - the act of teaching a program to react to or recognize patterns. It can be split into three broader categories namely supervised learning, unsupervised learning and reinforcement learning. *Supervised learning* is the machine learning task of inferring a function from labeled training data. Those labels are typically sorted into classes defined by expert human knowledge. *Unsupervised learning* on the opposite is the task of finding patterns in unstructured data. That means, there is no prior knowledge involved and the algorithms approximate solutions that show underlying patterns and connections. *Reinforcement learning* is learning what to do - how to map situations to actions - so as to maximize a numerical reward signal.

In this thesis we are mainly confronted with unsupervised learning. Unsupervised learning is special in a sense that we do not know what we want to find. Finding patterns can mean anything and most often only human beings are capable of interpreting the quality of a result. As a result most algorithms found in unsupervised learning are *NP-Hard* problems that approximate solutions given some objective maximizing or minimizing function.

1.2 Structure of the thesis

In *chapter 2* theoretical foundations and basics are examined. Helpful but not needed is prior knowledge in linear algebra, statistics and algorithms.
In *chapter 3* we introduce the news clustering system “News-Clusty”, a data pipeline, in comparisson to the *Columbia Newsblaster system*.
In *chapter 4* feature selection strategies for clustering algorithms are formally described.
In *chapter 5* experiments and evaluation of different clustering routines are presented.
In *chapter 6* we will discuss results, problems and chances.
The final *chapter 7* sums up the thesis, gives future directions and provides additional material for reading.

2 Basics

"If I have seen further it is by standing on the shoulders of giants."

Isaac Newton

The goal of this section is to give some intuition and the necessary theoretical background for the following chapters. The areas where clustering problems arise are huge. It provides solutions to problems like market segmentation, classification, document organization or indexing.
Firstly we will have a look at the definition of clustering and summarization. How they are related and the variety of possibilities this imposes.
Secondly the *vector space model* (VSM) is introduced. It contains all information about how to represent documents in a vectorized form. Of special interest are enhanced models which reduce the dimensionality of documents by *Singular Value Decomposition* (*SVD*).
Thirdly traditional clustering algorithms from the hierarchical (Ward, Birch) and partitional (K-Means, Expectation Maximization) family will be presented. Closely related are the generative models. These methods can be used as a kind of clustering algorithm and are highly useful in several steps of traditional clustering. They can be used as dimensionality reduction techniques as well.
At last we will have a glance at, how to measure the quality of clusters based on internal measures (without ground truth labels) and external measures (with explicit labelling of the ground truth).
The reader is assumed to have prior knowledge on linear algebra, statistics and linguistics. This is by no means a complete reference but should cover the topics fairly well.

2.1 Clustering and Summarization

Clustering as defined by Aggarwal and Zhai (2012) is finding groups of similar objects in the data with a defined similarity function between objects. The granularity of the features can vary:

- *Sentence based* - A document d is split into sentences, clustering reveals the most coherent groups of sentences that are closely related.
- *Collection of documents* - A collection of documents d (corpus) is clustered by similarity between all pairs of documents.
- *Stream of documents* - The same as clustering a corpus with the constraint that over time the size of the corpus grows.

Document clustering on large corpora can be seen as a summarization of the underlying concepts. The representation of documents as feature vectors is described with the vector space model in the next section. Data clustering is a computationally expensive *NP-hard* problem. That means there is currently no efficient way to group objectcs in an optimal way. Therefore heuristics are applied such that algorithms converge at a *local minimum*. Theoretically a local minimum can vary vastly compared to a global minimum, in practice however a close to 80% solution seems reasonable.
In this context a distinction has to be made between online and offline learning. Where we have a continuous stream of documents it is often reasonable to run a clustering algorithm continuously. This is called *online learning*. Most clustering algorithms need all documents at once, as a batch, to generate clusters. This is refered to as *offline learning*. The property of an algorithm that can partially cluster documents is desirable. In continuous clustering, several things are much more difficult compared to batch clustering. How can we assign or create new clusters for unseen documents? How do we choose a good representative seed to start off from? Specifically to newspapers, how do we track that a topic is not fresh anymore? On the contrary to this, batch algorithms can be used to cluster a full day of newspaper articles. We then connect clusters or in this case events with the day before and after. Then, clustering are groupings of events, that were written about at the same time. Events of two days can be merged when the distance between both is low e.g. the similarity is high.

Automatic text summarization on the other hand, is the process of reducing textual content to the most important concepts in a readable, formatted form to the user. Mani (2001)

We have to distinct between summarization of single documents and a collection of documents as well. In multi-summarization the problem lies in identifying topics accross documents, finding the most relevant sentences, reducing and removing redundant/duplicate information. Moreover syntactically it is often inappropriate to merge sentences of different documents. The Columbia Newsblaster and Google News are prime examples of summarization engines. Both continuously scrape news, cluster them and summarize the content into cohesive summaries. This is by no means an easy task and involves several components. These topics will be evaluated later and will take a major part of this thesis. See McKeown et al. (2002); Schiffman et al. (2002).
The text summarization and clustering task are closely related. But how can clustering help in improving or creating textual summaries over documents or single articles?

First Cluster documents that contain a *high density of information* so summarizers can work on groups of documents instead of all documents or single ones.

Second Detect the *latent topics* accross and within documents to create a meta concept of closely related documents, helping to improve summarizations.

Third Classify documents into *categories* in a semi supervised way to construct hierarchies of relationships

Fourth Removing *outliers* that will not highly contribute to textual summaries

Clustering itself can be seen as a summarization as well. Stefanowski and Weiss (2003) defined a well versed graphical user interface called Carrot Search boosted by their Lingo3G clustering algorithms [1]. One can see that the proposed clustering techniques can form well defined topical browsers. The user can easily interact with the underlying data in a connected and semantic way.

Supervision As opposed to unsupervised learning strategies such as clustering, supervised learning classifies some input based on a provided ground truth. That is, for an input x there are labels y that describe the class they are in. Supervision can be done by explicitly classifying the documents before the clustering. The input is then split into n classes. Then each class can be individually clustered. Often however this is no option. We need to manually label all documents. This can be time consuming and error prone. Often several labellers are needed to crossvalidate human bias. With this in mind there are two options on how to label unseen or new data:

[1] See http://search.carrotsearch.com/carrot2-webapp/search

- Use a supervised classification algorithm to automatically label unlabelled data. A prerequisite is to have a labelled training set and to have a lot of data. To name a few candidates: *Multinominal/Gaussian) Naive Bayes (NB)*, *Multivariate Logistic/Linear Regression*, *Neural Networks (ANN)*, *Support Vector Machines (SVM)* or *Random Forests (RF)*. Bishop (2006)

- Use an unsupervised clustering algorithm to automatically label unlabelled data. This can be done by first forming clusters and then merging the nearest clusters until k distinct categories remain. Usually the merging criterion can be controlled by some threshold and high variance documents are sorted out into an outlier cluster.

Unfortunately, as it turns out, clustering is good at preclassifying documents into groups, but in total is not accurate enough finding the ground truth. Opposed to this, supervised classification with well defined testing sets, perform extraordinary well on unseen documents. Clustering algorithms do not lead to very good groupings based on hard distinct labelling. The problem ultimately lies in the text domain itself. A lot of documents have different proportions of classes. A document about politics can equally have something to do with economics. This results in mixed clusters, where each cluster represents proportions of different classes. This is too fuzzy to classify by hard distinct labels.

A greater problem lies in the fact that no matter how well tuned a supervised classifier or an unsupervised clustering algorithm is, words change over time. A document about politics in the 1980s might resemble different word proportions today and so accuracy will diminish over time. This time drift can happen much faster. Suppose you originally trained a classifier for putting an article into politics and economics. Where do you put stories on the financial crisis 2008/2009? Lots of models try to overcome this gap by statistical assumptions.

2.2 Vector Space Model (VSM)

The vector space model is directly derived from the vector space subject to linear algebra. If we talk about vector space we often refer to the euclidean vector space where examples are in 2 or 3 dimensions. Linear algebra concerns itself with all dimension in $\mathbf{R}^n$. All dimensions higher than 3 are hard to imagine. From the view of linear algebra the vector space consists of linear combinations that are solved by $Ax = b$ by some matrix decomposition step. In the context of document clustering the vector spaces typically far exceed 3 dimensions up to $\mathbf{R}^n$. For a proper introduction to linear algebra see Strang (2009).

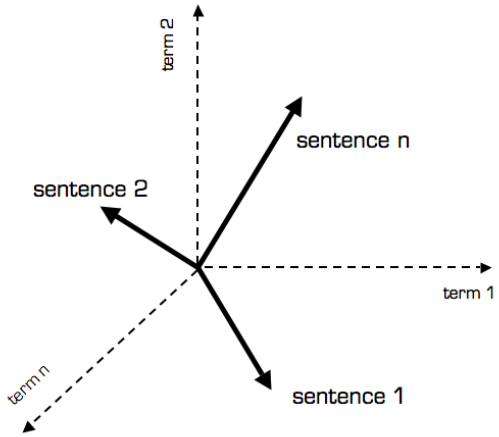

Figure 2.1: "Vector space model"

The vector space model in the text domain has the meaning that each word is a component of a vector resembling documents. If a document has 100 distinct words, the resulting document vector is in 100th dimensional space. If a second document has 50 distinct words, independent of the first document, both vectors are now in 150th dimensional space. That is, every new word will be concatenated to the existing document sets. This is the “bag of words” model which is detailed in the next section.

2.2.1 Notation

Before moving on we have to denote some notations and definitions. A corpus $C = \{d_1, d_2..d_m\}$ is defined as a collection of *documents* where $m = |C|$. A document $d = (w_1, w_2..w_i)$ contains words w where $i = |d|$. Note that each document is not a text but a sequence of words. Often

special stopwords are filtered out of these documents as well. A dictionary $D = \{w_1, w_2 .. w_n\}$ contains all the distinct words from each document where $n = |D|$. A cooccurence matrix M is

$$M = \begin{array}{c} \\ d_1 \\ d_2 \\ .. \\ d_m \end{array} \begin{array}{c} \begin{array}{cccc} w_1 & w_2 & .. & w_n \end{array} \\ \begin{bmatrix} c_{11} & c_{12} & .. & c_{1n} \\ c_{21} & c_{22} & .. & c_{2n} \\ .. & .. & .. & .. \\ c_{m1} & c_{m2} & .. & c_{mn} \end{bmatrix} \end{array}$$

Table 2.1: "Document-term (m x n) matrix"

where rows are the documents $d_1..d_m$ of a *corpus* and columns are the cooccurence counts $w_1, w_2.., w_n$ of the corpus's dictionary. The next section 2.2.2 explains this in more detail. Often the transpose M^T is taken into account as well, setting rows to columns and columns to rows. Often the words are selected based on different feature selection strategies as well, see section 4.2. A distance matrix M_{dist} is defined as

$$M_{dist} = \begin{array}{c} \\ d_1 \\ d_2 \\ .. \\ d_m \end{array} \begin{array}{c} \begin{array}{cccc} d_1 & d_2 & .. & d_m \end{array} \\ \begin{bmatrix} v_{11} & v_{12} & .. & v_{1m} \\ .. & v_{22} & .. & v_{2m} \\ .. & .. & .. & .. \\ .. & .. & .. & v_{mm} \end{bmatrix} \end{array}$$

Table 2.2: "Document-document (m x m) matrix"

where each document is compared to another document by some similarity measure. See section 2.2.3 for more information. v is any value that the distance function can incorporate.

2.2.2 Bag of words

The bag of words assumption says that a corpus can be represented as a count or word occurence matrix. That means *m* documents form a subspace in an *m x n* matrix where *m* denotes the corpus size and *n* the dictionary of the words. Typically the assumption is binary, that is an occurence of word *i* in a document *j* is set to 1 else set to 0. Bag of words is also another way of saying that words are exchangable in order. That means the joint probability of N following words is equal to any of its permutations $p(w_1..w_N) = p(w_N..w_1)$, thusly leading to the assumption that the probability of a sequence of words can be viewed as a mixture over

words. Later we will see this applied to the *expectation maximization* algorithm.
In reality this is not true, but without this assumption most of the models would be computationally expensive (up to unsolvable) and hard to reason about. In real world examples this is typically no problem but one has to account for the fact that indepedence of word sequences is a strong axiom that in reality is not true.

	Words		
Documents	politics	corruption	policy
document1	1 (2)	1 (1)	0 (0)
document2	1 (4)	0 (0)	1 (2)
document3	1 (1)	1 (6)	0 (0)

Table 2.3: "Document-term (m x n) matrix"

Normally we have a lower *m* and a much higher *n* resulting in highly sparse vectors with a lot of zeros, typically 99%. As stated before we state that words are indepedant from each other. That means highly correlating words like *New* and *York* are not accounted for. In this case they can actually refer to the verb *new* and *York* as a city in *Great Britain*. A bigram e.g. *(New, York)* would capture the concept *New York*. Bigrams, trigrams or generally ngrams are not taken into account. Ngrams transform a sequence of words by *n* such that

$$f(n = 2, (w_1, w_2, w_3) \in d) \rightarrow [(w_1, w_2), (w_2, w_3)] \tag{2.1}$$

Adding ngrams to a document term matrix can greatly enhance similiarity between two documents. Keeping the dictionary and adding ngrams to the vector space results in *n+(n-1)* memory requirements. If the data is already extremely sparse this will not help much but can increase the semantic effect on documents that share similar word combinations. There are other much more complex models e.g. *noun phrases* or *named entities* that can grasp this intuition as well. From a statistical point of view this is captured by collocations stating that certain word combinations occur more often than they would by chance occur. Therefore it is reasonable to assume that collocations are found in other documents as well strengthening the connection between two documents.

The document term matrix can be enhanced by taking the count of the occuring words instead of just labelling it by occurence. This is called the raw frequency and can be normalized in a couple of ways by the term frequency (tf) model

$$f(t, d) = c \tag{2.2}$$

where c is the total count of term t occuring in document d. And a general term frequency function that helps against long document bias by normalizing with the maximum frequency of any occuring word in d.

$$tf(t,d) = 0.5 + \frac{0.5 * f(t,d)}{max(f(w,d)\,\forall w \in d)} \tag{2.3}$$

The term frequency model can be further advanced by caculating the inverse document frequency (idf).

$$idf(t,C) = log(\frac{|C|}{|\forall d \in C : t \in d|}) \tag{2.4}$$

Where |C| is the size of the corpus and we check for every document in the corpus if the term occurs in the document. The intuition is: How often does a term occur in other documents. If a term appears more often then it is a common word such as “the”, whereas “super-symmetry” might be a rare word. It is therefore some measure of importance. Because idf and tf only measure either importance accross all documents or importance of one document, we need to find words that are not rare and not common either. Generalizing the term-frequency and inverse-document-frequency we obtain the tf-idf:

$$tfidf(t,d,D) = tf(t,d) * idf(t,D) \tag{2.5}$$

The tf-idf has a high score, if a term occurs often in a single document and less often in other documents. This translates to the notion that a term represents the current document better than other terms and are therefore highly discriminative words. There are other models such as graph based or tree based approaches. For the porpuse of this thesis these models are left out. (Manning et al., 2008a, chp. 6)

One of the big problems with the vector space model is that feature inflation or feature explosion arises quickly. If documents have a high variance between other documents and the connections between two documents are small the sparsity can go up to 100%. That means that no document has overlapping words and each document only accounts for the words that were originally in the document. Think of it another way, suppose one has 3 documents in 3 dimensional space with each axis set to zero but one (1,0,0), (0,1,0), (0,0,1) we have an independent basis. This means that all documents are orthogonal to each other, meeting at no point except at the null point (0,0,0). It is therefore not really possible to derive any connections

between those 3 points. Thus each document represents its own topic. These kind of problems need a well thought out solution which is presented later in this section.

2.2.3 Similarity and Distances

Partitional clustering algorithms commonly work through some objective distance or similarity function between two objects. After lifting the documents into vector space, we are faced with the problem of distance between two documents. Several measures were proprosed that can be easily interpreted by geometry or linear algebra. Our goal is to have an *m x m* matrix for *document x document* distances. A *document x terms* matrix on the other hand is highly relevant when clustering under the assumption of topical classification. That is, compare documents not to each other but to all the occuring words as topics. *Latent Semantic Analysis* (*LSA*) is a method where words are projected to lower dimensions by singular value decomposition. This resembles partial categorization to topics of a document by the most distinct words. Then a clustering can be done by documents that highly correlate to the same topics. See 2.2.4.1 for more information.

The Cosine similarity is a measure of orientation, not the magnitude. The angles between documents are compared and thus if the angle is 0 both documents are equal in size and word occurences. Given two documents d_1 and d_2 the classical cosine similarity is defined by

$$cosine(d_1, d_2) = \frac{d_1 * d_2}{||d_1|| * ||d_2||} \tag{2.6}$$

If the documents are already defined unit vectors cosine similarity is just $cos(d_1, d_2) = d_1 * d_2$. Often we would like to consider the geometrical distance between two points taking the maginute of a vector into account as well.

The Euclidean distance ($l^2\ norm$) is the geometrical distance between two vectors in $\mathbf{R}^n$. Again given two documents d_1 and d_2 it is defined by

$$euclidean(d_1, d_2) = \sqrt{\sum_{i=1}^{M}(d_{1,i} - d_{i,2})} \tag{2.7}$$

This results in the fact that documents running into different directions, e.g. have different angles might as well be very close. It is a geometric measure and accounts for magnitude rather than direction.

The Manhattan distance ($l^1\ norm$) is commonly known under the city block distance measure. The system is divided up in equally squared brackets comparing dimensions in sequential order and taking the absolute squared difference between two documents. Given two documents d_1 and d_2 it is defined by

$$manhattan(d_1, d_2) = \sum_{i=1}^{M} |d_{1,i} - d_{2,i}| \tag{2.8}$$

There are more distance functions such as the *Jaccard coefficient* that works on intersections of sets and the *Chebyshev distance* which is a maximizing greedy strategy of the manhattan distance. The underlying concepts of similarity should be clear though. Each coefficient works good on a particular set, however cosine and euclidean distances are the commonly used choices.

Hierarchical algorithms use different metrics. The goal here is not to find the closest points but to find the closest intersecting clusters. Hierarchical algorithms merge clusters based on linkage strategies. Those merges are seen as a split in the resulting directed acyclic graph. We will review them later as their functionality strays away from the vector space.

2.2.4 Enhancing the Vector Space Model

The vector space model comes with a variety of problems. First we have extremely high dimensions. As we see later this is particularly true for newspapers. Unlike Twitter the content of a newspaper article can be substantial and range over a variety of topics. This can lead to extremely poor clustering results. Several methods have been proposed to tackle this high dimensionality problem. One of the most strickening comes from linear algebra and was initially proposed in Deerwester et al. (1990) called *Latent Semantic Analysis* (*LSA*). It is based on *Singular Value Decomposition* (*SVD*). The *Principal Component Analysis* (*PCA*) on the other hand uses *SVD* in a different manner. Apart from these there are also much more elaborate techniques in the domain of topic modelling which will be reviewed later.

2.2.4.1 Latent Semantic Analysis

Latent Semantic Analysis is a progress to reduce as much noise as possible from a given term x document matrix to expose the so called latent variables. It is connected to topic modelling but seems to be more relevant in the context of dimensionality reduction. Dimensionality reduction is used for dealing with large sparse data to find the underlying connecting relations. *LSA* is a truncated *Singular Value Decomposition* (*SVD*) on a term x document matrix. *SVD* is a

matrix decomposition where an n x m matrix A is decomposed into three matrices U, Σ and V^T. *LSA* is constrained by the number of dimensions k:

$$svd(A) = U_{mxm}\Sigma_{mxn}V^T_{nxn} \rightarrow lsa(A,k) = U_{mxk}\Sigma_{kxk}V^T_{kxn} \tag{2.9}$$

While in the original *SVD* we keep all components decomposing their respective singular values, *LSA* only keeps k components reducing the dimensions from n words to k. Σ holds the k best singular values in sorted order. V^T is a k x document matrix, where k are the reduced topics from the words. While we can do *LSA* over documents, we can also use it for summarizations of single documents.

$$\sigma_1 \geq \sigma_2 ... \geq \sigma_r > \sigma_{r+1} = ... = \sigma_n = 0\,.$$

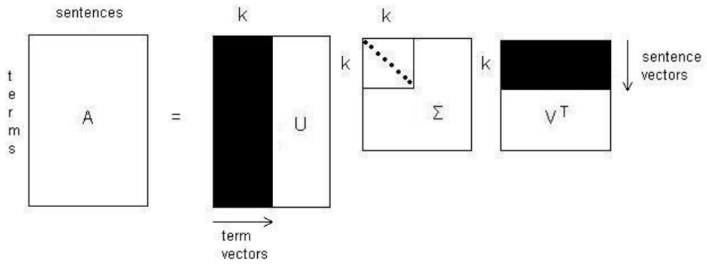

Figure 2.2: "Singular Value Decomposition"

The $\sigma_1 \geq \sigma_2..\sigma_k$ stands for k preceeding singular values sorted by magnitude. *LSA* gives us a mapping from the top k topics of the document to the most important sentences of the document. In descending order the document presents less information per sentence. Thusly taking the top k sentences to create summarization can lead to significat results in a summarization processor. See figure 2.2 by Steinberger and Ježek (2004). The intuition behind *SVD* is to expose as much variance as possible at the same time reducing noise from the data. *SVD* then is a natural progress to reduce highly sparse data projecting it to a lower dimensional space. This corresponce to a notion of topic overlap between different documents. Documents that share a particular topic are more similar. This results in connections between documents even if a lot of the words share no common meaning. Effectively *LSA* tackles problems of synonymy and polysemy. Synonymy means that several words often share the same meaning such as “big” and “large”. While polysemy refers to the fact that one word can have several meanings such as “bank” as a fincancial institution and “bank” as an object to sit on.

2.2.4.2 Principal Component Analysis (PCA)

The *Principal Component Analysis* (*PCA*) is a multivariate technique separating out correlating variables into principal components. *PCA* in general is a dimensionality reduction technique reducing the number of features to a lesser set of features. The difference between *PCA* and *LSA* is the input. *PCA* takes a covariance matrix $A * A^T$ which means that either only word x word or document x document matrices are taken as input. This makes it perfect for reducing dimensions after creating a cosine similarity matrix from the documents to visualize results.

$$\mathbf{PX} = \begin{pmatrix} \mathbf{Px_1} & \mathbf{Px_2} & \dots & \mathbf{Px_n} \end{pmatrix} = \begin{pmatrix} \mathbf{p_1.x_1} & \mathbf{p_1.x_2} & \cdots & \mathbf{p_1.x_n} \\ \mathbf{p_2.x_1} & \mathbf{p_2.x_2} & \cdots & \mathbf{p_2.x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{p_m.x_1} & \mathbf{p_m.x_2} & \cdots & \mathbf{p_m.x_n} \end{pmatrix} = \mathbf{Y}$$

Figure 2.3: "Principal Component Analysis"

In figure 2.3, by Richardson (2009), we find the *n* most likely components of a document by decomposing an original document matrix X by factoring out P principal components. The input is no $n\ x\ m$ matrix but rather a covariance matrix $n\ x\ n$ or $m\ x\ m$. These components capture the variance throughout each document individually and reduce the dimensions down to a specified *k*. The factorization then tries to solve for a principal component that bestly resembles the original data. In practice *PCA* is conversion of a document by term matrix into a covariance matrix followed by *SVD* where principal components resemble the highest singular values up to k. Primarily *PCA* can be described as a dimensionality reduction technique that resembles the original structure well. It can be used for visualization, reducing down $n + 3$ dimensions to 2D or 3D. See more in Richardson (2009).

2.3 Clustering algorithms

Clustering algorithms are the procedures, describing, how data can be automatically assigned to groups by similarity metrics. They are unsupervised methods and can be combined by explicit knowledge engineering. They come in shape of *partitional* models, that flatten the structure, in *hierarchical* models that build tree like structures, *spectral* algorithms that are *graph based* or *density* based algorithms. There is also a fifth category namely *generative models* and closely related in terms of the text domain *probabilistic topic models*. These models try to find the underlying latent semantics by generating the model that could have created the documents in the first place. First let us setup a few definitions that all clustering algorithms incorporate.

- *Hard and Soft clustering* - A clustering algorithm is *hard* if documents can only be assigned to one distinct cluster. On the contrary they are *soft* (often called *fuzzy/overlapping*) if a document can have multiple assignments that resemble topical proportions of the document.

- *Exhaustive and non-exhaustive* - A clustering is exhaustive if every document has an assignment after the run whereas it is non-exhaustive if documents might not have any assignments at all (assigned to a null cluster).

- *Cost functions* - Most clustering algorithms incorporate some sort of cost function that needs to be minimized or maximized after the algorithm completed. The clustering can be rerun several times finding a maximum between different runs by varying the parameters. This should not be confused with the similarity measure between objects during a clustering run.

- *Local minima/global optimum* - A local minimum is when the objective cost functions between objects do not yield better results after several iterations. This does not mean that the best state is found. Similarity metrics between objects are inherently heuristic and have no view of a global optimum. A global optimum can be approximated by objective cost functions rerunning the algorithm several times. There is currently no known algorithm that can solve clustering in a deterministic globally optimized way. However it is possible to use approximative approaches, that near the global optimum quite well.

2.3.1 Partitional clustering

Partitional clustering algorithms build up centroids. They resemble typical cluster centers. After documents are converted into vector space and some sort of similarity measure is applied, cluster centers are incrementally created. We briefly review two partitional clustering algorithms namely *K-Means* and *Expectation Maximization (EM)*. Literature in this area is rigorous, see Aggarwal and Reddy (2013); Anastasiu et al. (2013); Manning et al. (2008b).

Figure 2.4 is a typical clustering result. The crosses denote unassigned single clusters. Note: to display 2D models one has to use dimensionality reduction techniques reducing the feature size. The original dataset consisted of over 100k words so this is a rather unfair approximation of the original dataset. The larger points correspond to cluster centers while the smaller points correspond to individual documents adhering to the cluster centers colors. The axis of the plots do not play any significant role, as they are scaled by the mean of the data. The axis

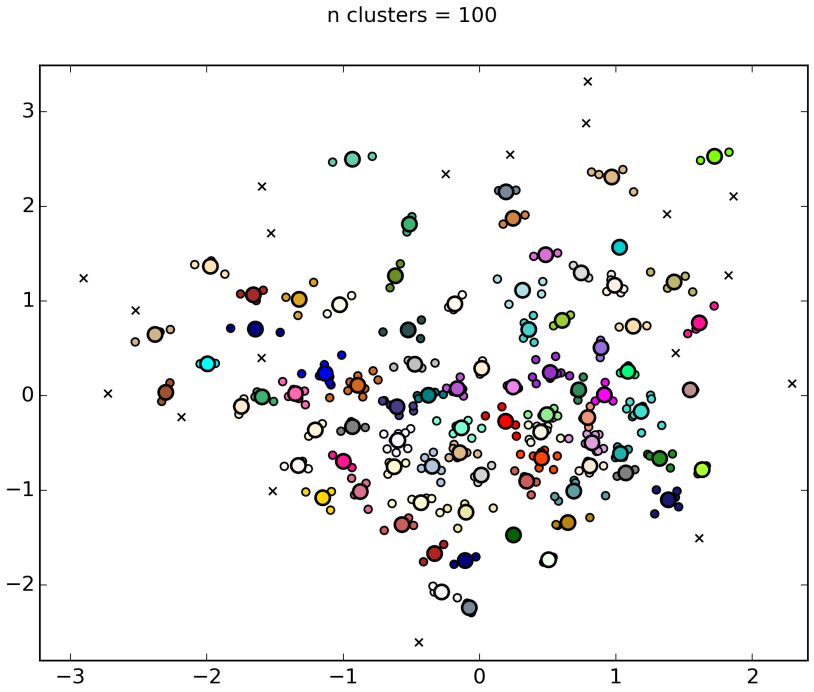

Figure 2.4: "K-means visualisation with applied dimensionality reduction"

represent term frequencies of the documents and reduction by *PCA*.

Partitional clustering algorithms typically have linear running times and thus are often used in practical real world applications. We will see later that this comes with a price mostly in accuracy and by providing the number of clusters. Most of the time we do not know how many clusters will be in the final result so we need algorithms that can find an approximately optimal clustering. In particular partitional algorithms can be globally approximated given a numerical cost function $f : D \rightarrow \mathbf{R}$ such that it maximizes $f(d_i) \geq f(d_j), \forall d \in D$ or minimizes $f(d_i) \leq f(d_j), \forall d \in D$. Often they are much easier to implement because constraints are kept low and requirements are kept to a minimum. d_i, d_j are all pairs of documents in a collection of documents D. There are also online versions available which make it possible to scale algorithms such as K-means with *map reduce* on several processors. This is not true for hierarchical algorithms that build up a hierarchy bottom-up or top-down. The difference is that partitonal algorithms assign to cluster centers, while hierarichal algorithms build up pairs

of merges, resembling a merge tree. This makes it difficult to scale hierarchical algorithms as they need a global picture of distances to be generated.

2.3.1.1 K-Means

The *K-Means* clustering algorithm is a centroid based algorithm. Each document is assigned to a cluster center in each iteration. In order to do this k centroids are drawn as a initial cluster centers from the documents. Then interdistance similarity measures, typically $l_2\ norm$, decide if a document is closer to a certain centroid. Centroids will be moved to the new location in vector space by averaging over the assigned documents. In the next iteration a document then might be reconsidered for another cluster. *K-Means* typically forms clusters of equal sizes and is a hard clustering algorithm Manning et al. (2008b).
Typically a *K-Means* algorithm will be run with respect to an objective cost function to be minimized. Globally this can be used for chosing a suitable k for the underlying data. The objective goal is to minimize the residual sum of squares of within cluster distances Manning et al. (2008b):

$$RSS_k = \sum_{\vec{y}_k \in w_k} |\vec{y} - \vec{f}(w_k)|^2 \tag{2.10}$$

If we now sum over all RSS_k the total amount must be minimized. Formula 2.10 is a vectorized form, where y is a prediction column and $f(w_k)$ is the prediction given the assignments w_k where f is some kind of distance function. The algorithm converges either after not improving above a *threshold* or by limiting the iterations by a paramter *max iter* after which the algorithm stops. In this case both criterias are modelled which resembles the implementation by Pedregosa et al. (2011) in Sklearn.

Algorithm 1 X is a document term matrix, μ is a matrix of centroid vectors, c a mapping between X and μ

function KMEANS($X = x_1, ..x_m, k, maxiter$)
  $\mu \leftarrow select\ k\ initial\ centroids\ from\ X$
  $c \leftarrow init\ assignment\ matrix$
  $cost \leftarrow \infty$
  for $i \leftarrow 1, maxiter$ **do**
    $c \leftarrow argmin|\vec{\mu} - \vec{X}|$ ▷ assign X to k nearest centroids
    $\mu \leftarrow average\ X\ over\ assigned\ \mu\ by\ c$
    $cost \leftarrow \sum_{n=1}^{k} RSS_k$
    if $cost\ lead\ to\ convergence$ **then**
      Stop
    end if
  end for
  return $labels, centroids, cost$
end function

Now we are left with one problem: How do we choose the initial clusters? Several strategies have been proposed. The most used is random sampling where *k* random documents are chosen as a centroid. The problem is, when all documents are skewed to one direction or close together the *K-means* converges at a local minimum that seems to suboptimally divide the documents into clusters. Another strategy has been proposed by Arthur and Vassilvitskii (2007) that maximizes the intercluster distance by spreading the clusters as far from each other as possible. As the initial clusters are often picked randomly *K-Means* is a non deterministic approach that will most likely result in different clusters after each run.

2.3.1.2 Expectation Maximization (EM)

The *Expecation Maximization (EM)* algorithm is a generalization of *K-Means* and a soft clustering algorithm with fuzzy associations between clusters and documents. The underlying methodology is that we would like to find a model that generated the given documents. The way this works is first by assuming that the clusters are represented as random distributions over terms. They can be assumed as a normal gaussian distribution. Note: the kind of distribution can vary, gaussian distributions can tend to zero if no assignment was found and so EM could assign no label at all. If that is not desired and a soft and exhaustive clustering is prefered one can add additive smoothing, that other models like the Dirichlet model inheret.

What we would like to know is: How does a data point x relate to the different k guassian distributions? Formally we need to estimate a model Θ that is maximized using a cost function such as *maximum likelihood estimation (MLE)* given some data. The intution is, by estimating the likelihood of a given incomplete dataset D, that is assumed to be normally distributed, we want to maximize the connection between Θ and D.

$$\Theta = argmax \sum_{n=1}^{N} log\, P(d_n, \Theta) \tag{2.11}$$

Where P is the fractional probability that a document d_n was generated by Θ. In the text domain this means: for all words in each D can we generate assignments to distributions that maximize Θ. The *EM* algorithm tries to maximize this function by assigning a probability that a data point x is likely to be in the cluster $c_1..c_k$ by calculating the joint probabilities of occuring words given a prior of cluster $c_1..c_k$. This is also called the expectation step. In the second step each probability of a document d being part of a cluster $c_1..c_k$ is weighted into the average and variance of the defined clusters. The clusters are then recalculated using the beforementioned assignments. This step is repeated until some kind of convergence criteria has been met. The main intuition is if we have a model Θ we can compute the fractional probabilities of a document d to be in a cluster k:

$$P(d|\Theta) = \sum_{k=1}^{K} \alpha_k \left(\prod_{t_m \in d} P(t_m = 1|c_k) \right) \left(\prod_{t_m \notin d} (1 - P(t_m = 1|c_k)) \right) \tag{2.12}$$

In the maximization step we then need to compute the probability that a term t_m has a high probability to be in c_k that is $P(t_m = 1|c_k)$ and the prior α_k that is the probability that any document is in $c_1..c_k$ given no evidence. In the expectation step we compute r_{nk} that is the soft assignment of a document d_n to a cluster k. r_{nk} is computed with the soft assignments of the prior α_k and $P(t_m = 1|c_k)$.
The inherent problem remains: How do we choose α_k and $P(t_m = 1|c_k)$? α_k and $P(t_m = 1|c_k)$ are needed to recompute the soft assignments. At the beginning assignments are assumed to be randomly mixtured. After several iterations documents are weighted into the initial random selection until convergence. So to speak the documents contribute to the model over time up until no change has been seen to the iteration before. For further information see Manning et al. (2008b). Later we will discuss the generative models more indepthly. The *EM* algorithm can work with a lot of distribution. Like the Gaussian distribution, we can work with other multinominal distributions assuming different priors.

2.3.2 Hierarchical / Agglomerative clustering

Hierarchical agglomerative clustering (HAC) algorithms build up hierarchies of clusters with documents on their leaf nodes. There are two approaches available. By bottom-up, assuming all documents to be clusters, agglomerating them upwards until a root node has been found. Or by top-down, starting with all documents in a cluster, splitting them down until the leafs are documents. Often we can constrain this process by limiting the cluster size as well. Then *HAC* first clusters document to create a tree and then later cutting the tree by heuristics such as a given cluster size k or by a distance threshold. It is a big discussion in the scientific community if hierarchical clustering as opposed to partitional clustering algorithms give better results. (Manning et al., 2008a, chp. 17)

HACs typically have higher runtimes due to the fact that most of them infer the count of clusters and build up a hierarchy until convergence. The runtimes are often in $O(N^2)$ or $O(N^3)$. As seen below in figure 2.5 hierarchical clusterings can be easily visualized as dendograms. Dendograms show a history of the merges of different clusters up to the root nodes.

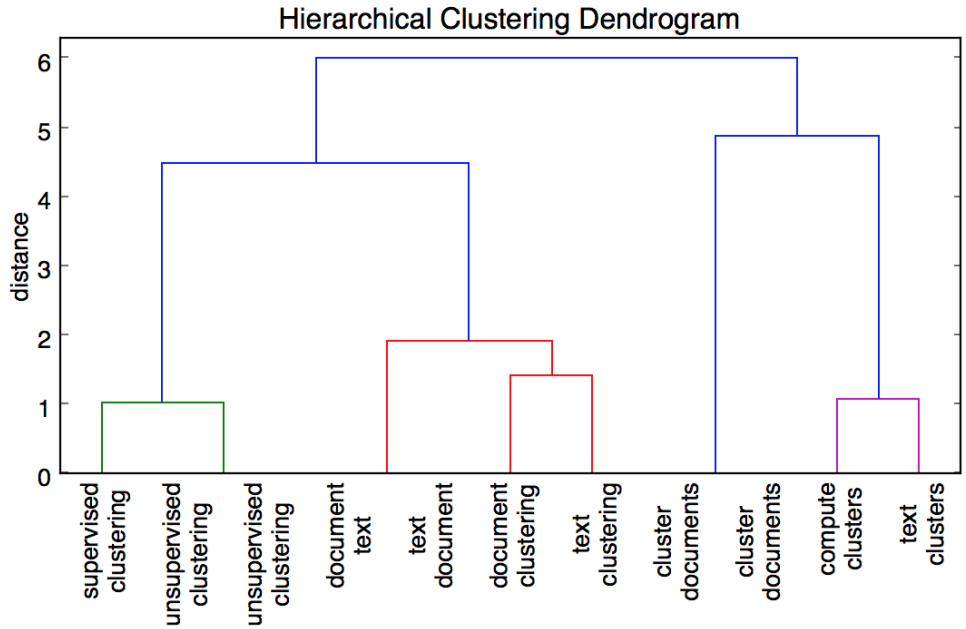

Figure 2.5: "Ward linkage dendogram"

On the y label there are decreasing distances. The x label presents the leaf nodes where at most two documents are paired to each other. For instance “unsupervised clustering” occurs twice so the distance between these two is 0. While at the same time “supervised clustering”

seems to be close. The merge one hierarchy up could resemble a cluster where the 3 documents resemble one bigger cluster. The distance at the y level is monotonically increasing bottom up. Each successive merge must be at a higher distance than the merge before. By threshold, distance density or a fixed cluster size k, we can form clusters at different granularities. To start hierarchical clustering a $m\ x\ m$ distance matrix has to be created. The closest two documents will be iteratively merged until no unassigned document remains. Keeping track of the merging processes in each iteration can be seen as overlapping clustering centers.

A simple *HAC* scheme first finds the maximizing next distance between two documents that have not yet been merged. Then these maximized document pairs d_1 and d_2 will be merged resulting in a new higher level merge. The similarity in *HACs* are stated by linkage criterias. The question then is: given two merges c_1, c_2 how can we compare them?

Algorithm 2 D is a document term matrix

function HAC($D = d_1, ..d_m$)
  $C \leftarrow initialize\ m\ by\ m\ distance\ matrix\ over\ D$
  $I \leftarrow list\ of\ length\ m\ I[m] = 1$ ▷ List of unmerged documents
  $H \leftarrow \{\}$ ▷ History of merges
  for $k \leftarrow 1, m$ **do**
    $g \leftarrow \{i \neq l \wedge I[i] = 1 \wedge I[l] = 1\}$
    $i, l \leftarrow argmax\ \{i, l : g\} C[i][l]$ ▷ Return maximized indices over C
    $H \leftarrow (i, l)$ ▷ Keep track of merges
    for $j \leftarrow 1, m$ **do** ▷ Update C with respect to all other clusters
      $C[i][j] \leftarrow LinkageSim(C[i][l], C[l][j])$
      $C[j][i] \leftarrow LinkageSim(C[i][l], C[l][j])$
    end for
    $I[l] \leftarrow 0$ ▷ Deactive from active clusters
  end for
  return H, C
end function

2.3.2.1 Linkage strategies

In the following we will explore some of these different linkage clustering strategies. The refered to distance function will be one of the conventional distances such as *cosine* or *euclidean*.

In figure 2.6 we can see 3 basic strategies. These strategies can be combined into more complex ones like the *Ward linkage*.

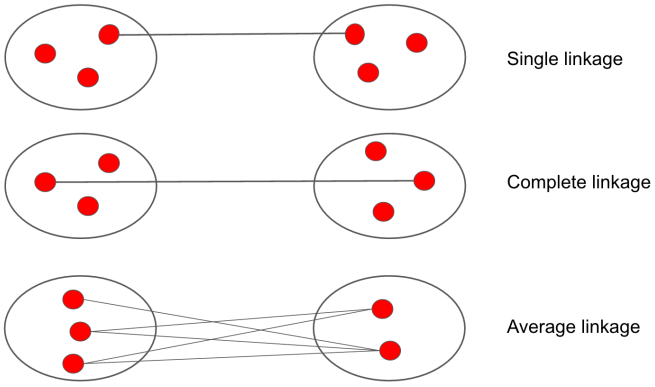

Figure 2.6: "Single, complete and average linkage"

Single linkage takes two documents of the corresponding clusters c_1, c_2 that minimizes the distance between these two. The objective function corresponds to:

$$\forall d_1 \in c_1 \wedge \forall d_2 \in c_1 : min(distance(d_1, d_2)) \tag{2.13}$$

In figure 2.6 we can see that single linkage takes, geometrically, the two clostest points of c_1, c_2. It is single in a sense that no other documents are considered but the closest two.

Complete linkage takes two documents of the corresponding clusters c_1, c_2 that maximizies the distance between these two. The objective function corresponds to:

$$\forall d_1 \in c_1 \wedge \forall d_2 \in c_1 : max(distance(d_1, d_2)) \tag{2.14}$$

In figure 2.6 we can see that complete linkage takes, geometrically, the two farthest points of c_1, c_2. It is complete in saying that it is not underestimated by taking the high variance into account.

Average linkage takes all documents of the two corresponding clusters c_1, c_2 and averages them with all corresponding clusters. That is we take all documents into account and link them by their average. The objective function is:

$$\frac{1}{|c_1| * |c_2|} \sum_{d_1 \in c_1} \sum_{d_2 \in c_2} distance(d_1, d_2) \tag{2.15}$$

In figure 2.6 we can see that average linkage takes the average of all documents in a cluster and calculates the distance between both averages.

There are other strategies such as centroid based, Wards method and minimum energy based. All strategies have different purposes and have their place. Single linkage might be best suited for problems where the data points are dense and not scattered throughout the vector space. Complete linkage might work well if data is scattered widely in high maximas so to approximate an ideal solution. Average linkage would overestimate these cases by too many outliers. On the other hand average linkage is the best choice if documents are assumed to participate equally.

BIRCH (balanced iterative reducing and clustering using hierarchies) is a specialized version of *HACs*. It scales well to large datasets and is an incremental algorithm. It operates in different phases and can be constrained by an optional cluster size *k*. One can provide parameters for maximum nodes per cluster and minimum density threshold as to when a document should be assigned to a cluster. See Zhang et al. (1996) for more information. As with the partitional algorithms this was studied extensively by Aggarwal and Reddy (2013); Aggarwal and Zhai (2012); Manning et al. (2008a). Top down approaches are not of special interest here. They are primarily suitable for large scale topic browsers where linear time constraints play a role.

2.3.3 Generative Models

A generative model is a model for randomly generating observable data values, typically given some hidden parameters. It specifies a joint probability distribution over observations and label sequences. In the area of document clustering and text summarization they play a role in modelling documents by word distributions assigned to topics. The *EM* algorithm is a typical scheme of a generative model, found in the partitional clustering section. In the text domain, topic modelling plays an increasing role for identifying latent topics from a distribution of documents and words. By *bayesian inference* we can draw conditional probabilities from word to document to topic proportions. Making sense of these quantities is part of probabilistic topic modelling. As with the *EM* algorithm we draw partial probabilities that a document

was generated by a mixture of topics. Several widely used methods namely *Latent Dirichlet Allocation* (*LDA*), *Hierarchical Dirichlet Process* (*HDP*) as well as the beforementioned *Latent Semantic Analysis* (*LSA*) are found in the generative realm. Usually, generative models are too expensive for large datasets. That is why most of the algorithms use variational inference resulting in the possibility to use online algorithms. See Bishop (2006) for a good source.

2.3.3.1 Topic modelling

Probabilistic topic modelling is a generative approach where documents are assumed to be mixtures of topics. A topic is a mixture of words describing probabilities how much a word constitutes to a topic. The underlying question then: Given a set of documents D and given random topic mixtures $Z = \{z_1..z_k\}$ what words $w_1..w_n$ of a particular document d constitutes to which topic. Given a document d with words $W = \{w_1..w_n\}$ and some mixtures over words Z how often does a particular word from W occur in topic Z. Further how common is topic Z in the current document d. In the following we will briefly discuss topic modelling and its applications. See Blei (2012) for a good inroduction. The very basic form of topic models are drawn from multinominal mixtures over word proportions. We would like to know, given a document d and some topics Z how likely is it that a specific topic z generated d?

$$p(d) = \sum_{Z} p(z) \prod_{w \in d} p(w|z) \tag{2.16}$$

$p(z)$ is the probability that a random document is part of that topic distribution and $p(w|z)$ is the conditional probability that given a word w evidenced by z, how likely is it to be drawn? This tones down to bayesian inference. We quickly see that in this process each document is drawn with the priors $p(z)$. Further each document gets hard assignments, a document is matched with a probability for a topic z. Statistical learning via *EM* is required for all of these algorithms by drawing probabilities that estimate the maximum likelihood that a document d is assigned to a topic z. The topic proportions need to be learned and so each document increments the parameter size to be estimated.

Bayesian inference In bayesian inference one would like to know, given some evidence how likely is it that some event happens? You would like to know given some event A, evidenced by B, how probable is A? In order to do so we have to take the prior of $P(A|B)$ that is the probability that some event A without any evidence is likely, called $P(A)$. Multiply it by the

probability $P(B|A)$ that is the reversed probability of $P(A|B)$ dividing by the probability of evidence B, $P(B)$.

$$P(A|B) = \frac{P(A) * P(B|A)}{P(B)} \tag{2.17}$$

The definition of a prior or prior distribution is important. A prior is a raw distribution over the outcomes of some event A, *without* any evidence B. It is knowledge that needs to be observed. If no observations took place, we have to assume, beforehand, that the prior is randomly distributed. The generative process then weights in the words of each document and approximates the prior distributions. In the context of topic modelling, priors are topic distributions over words or documents.

Probabilistic Latent Semantic Indexing (pLSI)

Advancing on the above statement, Hofmann (2001) created the *probabilistic Latent Semantic Indexing* (*pLSI*).

$$p(d, w_n) = p(d) \sum_{Z} p(w_n|z)\, p(z|d) \tag{2.18}$$

pLSI takes into account that words contribute to a topic. Documents are sequences of words, so a document can contain several topics as well. The assumption: A topic generated a specific word, so a document was generated by several topics. This is done by drawing $p(d)$ as before with the additional probability $p(w_n|z)$ that a word w_n of a document d, evidenced by z and $p(z|d)$ the most likely probability that a topic is drawn evidenced by d. Informally for each word w_n in a document d how likely was it generated by a topic z given a document d.

Algorithm 3 probabilistic Latent Semantic Indexing

1: $Choose\ a\ document\ d_m\ with\ p(d)$
2: **for** $w_n \in d_m$ **do**
3: $\quad z_n \leftarrow multinominal\ from\ d_m\ p(z|d_m)$
4: $\quad w_n \leftarrow multinominal\ from\ z_n\ p(w|z_n)$
5: **end for**

As seen in algorithm 3 the underlying problem is that topic distributions must be estimated for each document d_m and for each topic z_n. This can be mitigated by using priors for topics instead of parameter estimations by *EM*. For this reason *Latent Dirichlet Allocation* (*LDA*) was

proposed by Blei et al. (2003) where a Dirichlet prior is assumed before hand. For completeness the two models are briefly described.

Latent Dirichlet Allocation (LDA)

Latent Dirichlet Allocation created by Blei et al. (2003) is *pLSI* assuming a Dirichlet prior. Dirichlet distributions are often used as prior distributions in Bayesian statistics. The Dirichlet distribution is the conjugate prior of the categorical distribution and multinomial distribution. Its probability density function returns the belief that the probabilities of k rival events are x_i given that each event has been observed $\alpha_i - 1$ times.
This fact comes in handy especially when assuming that *LDA* should generate k distinct topic distributions over words that can be mapped to a topic distribution over documents. Blei (2012) Alternatives to this are more complicated models like the *Hierarchical Dirichlet Process* (*HDP*) by Teh et al. (2006) that can automatically infer the size of the topics or the *Pachinko allocation* creating a hierarchy with directed acyclic graphs (DAGs). Li et al. (2007) The enhanced models are non parametic, which means we do not estimate certain parameters, rather we assume parameters be of some form.

Probabilistic topic models are a smart way of counting proportions between documents. This is done by distributions that estimate probabilities over words to conclude topics. Topics can be seen as soft assigned clusters.

2.3.4 Others

In the literature there are a lot more strategies involved how clustering can be done. Often they are particulary good in specific domains such as images. In the text domain however they have high runtimes and are often not desirable in their output. They often lack good strategies to work with high dimensionality. We briefly note them here.

1. *Spectral* - these are graph based algorithms. Distances are based on graph partitioning problems, where shortest paths, min cuts and graph partitions are used to define clusters. These problems are perfect for small document groups and are often used as subcluster procedures.

2. *Density* - these algorithms try to find documents that are within a certain distance to each other. Often one needs to provide density thresholds or neighboorhood graphs that determine how two documents should be grouped. From a geometric point of view, we would like to cluster documents that are within a certain distance radius. These algorithms are locally aware in that they do not care about a global optimization.

3. *Grid* - the vector space of the documents is split into equally sized grids. We then compute the density of the grids and identify the maximizing grids. Then the problem to cluster each document to a distinct cluster is based on graph traversal techniques. The intuition is that we do not consider single points but rather the surrounding grids.

There are indepth studies on spectral, density and grid based algorithms in Aggarwal and Reddy (2013).

2.4 Cluster evaluation

In order to comprehend if a clustering result is reasonably good or bad, we need to evaluate them. This can be done by purely mathematical formulas that output numbers a human expert can interpret or by displaying intuitive results via plots, topic browsers or text. The second category is highly subjective to the reviwer, while the first, at least, holds some objective criterias. Evaluation is a black art. Clustering itself has no real goal in terms of the domain. The primary goal is to group similar and dissimilar data points. What similarity and dissimilarity actually means is an open topic. All results of evaluation metrics need to be analysed and interpreted, otherwhise its numbers. High numbers can correspond to dense clusters, low numbers can correspond to scattered clusters. The actual numbers only make sense compared to what the domain needs to be solved.
Formally we have two groups of measurements, the internal measures and the external measures. Going ahead a bit for the sake of this thesis we will use the *purity*, *V-measure* and *silhouette coefficient* for evaluation of clustering results.

2.4.1 Internal measures

Internal measures describe how well a clustering result represents the original data. By mathematically evaluating the final intra cluster distances between the documents and their respective clusters and by calculating the dissimilarity between documents and other clusters. What we will find is that internal measures are often not suitable in asking about the quality of clusterings. Higher dimensionalities might lead to much better clustering results in a few cases, because the original data has a high variance. Internal measures on the other hand have better scores if the dimensions are lower. Using *LSA* reducing the dimensions down to 1 will lead to extra ordinary high scores. The results of internal measures can thusly only be compared to other clusterings of the same dimensions. As there are several measures that can be employed for internal clustering evaluation we will briefly look more closely at the silouhette coefficient. There are other measures as well, for completeness they are shortly reviewed.

The silhouette coefficient is a measure for how dissimilar a document d is to its own cluster C_d and how dissimilar a document d is to the closest neighbouring cluster in $C_{d \notin C}$. Often denoted as $a(d)$ it is the average dissimilarity from d to all other documents in C_d and $b(d)$ picks the minimum dissimilarity of d to all clusters $C_{d \notin C}$. Then the silhouette of a document d for a clustering C is defined as

$$s(d, C) = \frac{b(d, C_{d \notin C}) - a(d, C_d)}{max(\ b(d, C_{d \notin C}), a(d, C_d)\ )} \tag{2.19}$$

The coefficient is 1 if the dissimilarity a is low and dissimilarity b is high, meaning d is well clustered into C_d. Near to 0 if both are fairly equal, which means d could fit into the neighbouring cluster almost as good as C_d. Finally -1 if the clustering went bad and d is falsely assigned to C_d whereas it should have been assigned to its neighbour. See Rousseeuw (1987) for detailed explanations.

The Davies–Bouldin index is a measure of distance between assigned intra cluster distances and inter cluster distances. That is a clustering is best if there is low intra cluster distance and high inter cluster distance to all clusters. A smaller value means better clustering. See Davies and Bouldin (1979) for detailed explanations.

The Dunn index minimizes the distance between two clusters c_1 and c_2 with a variety of possible distance metrices such as average, maximum or euclidean and divides it by the maximized intra cluster distances. Thus the Dunn index is like the Davies-Bouldin index a measure of intra cluster compactness and inter cluster variance. See Dunn (1973) for detailed explanations.

2.4.2 External measures

External measures describe how well a clustering performed provided with a ground truth of the documents. We often refer to this as the golden standard where human labellers assigned concrete classes to documents. With newspaper articles this is often the category of the article, like politics or business. Clustering results are measured by their ratios of labels to clusters and clusters to labels. The question that arises is: why use clustering for any kind of grouping in the first place if we already have class labels for the documents? This is adressed by only labelling a few documents of the original dataset and is called the training set. While this tackles the issue of spending time for labelling other problems arise. In clustering for summarization our inherent goal is not to group documents that necessarily share a label. Maybe we are interested

in groups of documents that share proportions of labels as well.
What we are mostly interested in are metrics that faciliates the notion of facts and predicition accuracy. The problem with clustering is that we do not know what the clusters actually mean. Before we can compare them to their respective labels we need to know for each cluster how documents were labeled. By counting all the occurences we can infer if documents were grouped into clusters with the same labels. If this is not the case then the labels either were not accurate enough or were accurate but the structure of the documents could not capture the assumptions. Generally we would like a high true positive score, meaning most of the documents that share the same label were in the same cluster. While at the same time maintaining a false positive score, meaning documents assigned to the wrong clusters. Wrong however seems relative if we are really not interested in a hard labelled clustering.

Purity is a basic score and facilitates the rate between correctly predicted results in relation to false results. Formally:

$$\frac{True\ Positives}{True\ Positives + False\ Positives} \tag{2.20}$$

It is the ratio between all correctly predicted results the examples that were assigned into the correct cluster *true positives* as a ratio to those that were falsely assigned to another cluster *false positives*. A High purity indicates low error ratio in the False positives. This only accounts for the false and true assignments. But what is about labels that are partially in all clusters? We have to account for the variance how labels are spread too. It is actually worse if we have 5 clusters and 5 labels and one label is partially found in all 5 clusters how can we account for that?

V-Measure is an entropy based measure. It relies on the homogeneity score that measures if all of the clusters $c_1..c_k$ contain documents that are labelled with the same class and completeness score that measures if all documents that are members of a class are elements of the same cluster. Completeness and homogenity run in opposite directions. Informally if all points are in a single global cluster, then all classes are part of the same single cluster, thusly generating a high completeness score and a low homegeneity score. If each document is assigned to its own cluster, having as many clusters as documents, the homogeneity score is very high and the completeness score extremely low. The V-Measure then favours solutions where classes

are assigned to a correct cluster (completeness), while keeping the classes distinct between clusters (homgeneity) Rosenberg and Hirschberg (2007):

$$homogeneity = 1 - \frac{H(C|K)}{H(C)} completeness = 1 - \frac{H(K|C)}{H(K)} \tag{2.21}$$

where $H(C|K)$ is the conditional entropy of the classes given the cluster assignments (similary for completeness, change the conditional probability):

$$H(C|K) = -\sum_{c=1}^{|C|}\sum_{k=1}^{|K|}\frac{n_{c,k}}{n} \cdot \log\left(\frac{n_{c,k}}{n_k}\right) \tag{2.22}$$

and $H(C)$ is the entropy of the classes:

$$H(C) = -\sum_{c=1}^{|C|}\frac{n_c}{n} \cdot \log\left(\frac{n_c}{n}\right) \tag{2.23}$$

Rosenberg and Hirschberg (2007) then define the V-measure as the harmonic mean of homogeneity and completeness:

$$v = 2 \cdot \frac{h \cdot c}{h + c} \tag{2.24}$$

Adjusted Rand Index - The Rand Index computes a similarity measure between two clusterings by considering all pairs of samples and counting pairs that are assigned in the same or different clusters in the predicted and true clusterings. The adjusted score than weights in an error rate that accounts for chance, meaning that all pairs could be randomly formed accounting for pure chance. See Rand (1971) for more information.

There are more models such as *adjusted mutual information* or *F-Measure*. All these measures have the underlying assumptions that an expert perfectly labelled the underlying topics of a document. Often, these scores are skewed, revealing information about total failure or an approximatively good solution. As said before documents can be partially assigned to different topics and thusly all metrics measuring the performance by hard labels will fail in actually stating if the desired goal was found. What if we would like to have clusterings where certain topics such as sports and business have an overlap? In summarization we might want to consider that sports can be about business decisions as well. Getting more diversified summarizations by finding connections between different labels can be desirable. Thusly

scoring too high on the *V-Measure* scale might indicate that there are no topic overlaps, while at the same time clusterings obviously fail if the score is near to zero.

3 Data pipeline

"Everything should be built top-down, except the first time."

Alan J. Perlis

This section explains a general dataflow and the necessary steps to get data from an external source into a vectorized form. It will be held short giving intuition about a general setup, necessary preprocessing steps and as a point of reference what the Columbia Newsblaster system did. For the source code used throughout this thesis see the News-Clusty Github repository.

3.1 Pipeline

A data pipeline for text data has various modules of interest. Where do we get the textutal content from? How is it stored? How is it preprocessed? What intermediate representations are useful? How do we handle the ongoing stream of input?

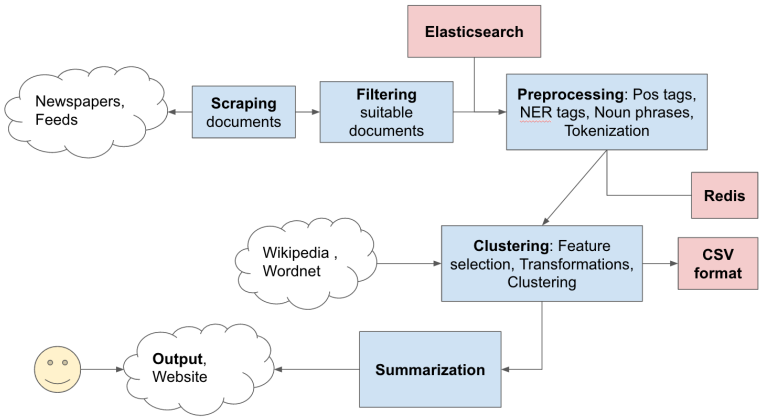

Figure 3.1: "News-Clusty pipeline"

The system that was developed for this thesis which is an early version is currently called "News-Clusty". Scraping, preprocessing and clustering are in place. There is a command line interface to interact with scraping and preprocessing. Currently it is an open question how to incorporate clustering schemes by a command line. The summarization and output generation is yet an open topic. The Columbia Newsblaster System by McKeown et al. (2001) is a very good point of reference. The core components of the Columbia Newsblaster are scraping, preprocessing, routing, clustering, summarizing and output generation. News-Clusty uses most of these components. They are briefly described in the following.

Scraping and Filtering is the process of retrieving documents of interest in its raw form from websites, archieves and other sources. In the news domain we talk about online newspapers such as "The Guardian" or news feeds like google news. The inherent goal is then to persist the found pages. It is crucial to identify correct pages by several heuristics:

1. *encoding* to utf-8 if working in english.
2. *language* to english, as a lot of newspapers have several languages to offer.
3. *content-length* has to be >= 500 chars to be relevant
4. *irrelevant content* like advertisement, not found pages, subscriptions etc.

The more information one can collect about the article the better. Meta information like authors, meta keywords, keywords, categories, publishing dates, tags and links can be used later for more semantic strategies. Often meta keywords can completely contain what articles are about. These keywords can be reasonably used by looking up definitions on *Wikipedia* or *Google*, enhancing the content of an article. Publishing dates can be used to capture freshness by penalizing articles that are several days old.

Preprocessing is the step where raw content is send into a pipeline of filters until all noise and unnecessary information is stripped. In News-Clusty everything is on the internet in form of html pages. So a major task is to extract raw content from html to representative forms.
In figure 3.2 we work on two elastic search indices. The preprocessing module reads articles from the "articles" index, a continuous stream of documents and posts on the "prep" index. In the preprocessing we take all the actions listed and save the intermediate result as its own field. This is helpful later when computing expensive operations such as named-entity tags, noun-phrases or pos-tags. After processing the html we take a second filter step to make sure that the conditions of the scraping steps still hold true. If not the article is removed. Stripping

html is not the only necessary step, we need to clean documents from bad characters, unuseful information we cannot work with like images. If the need arises spelling correction can be applied as well that is if there is sufficient evidence that articles have a lot gross misspellings, due to the central limit theorem this can be mitigated.

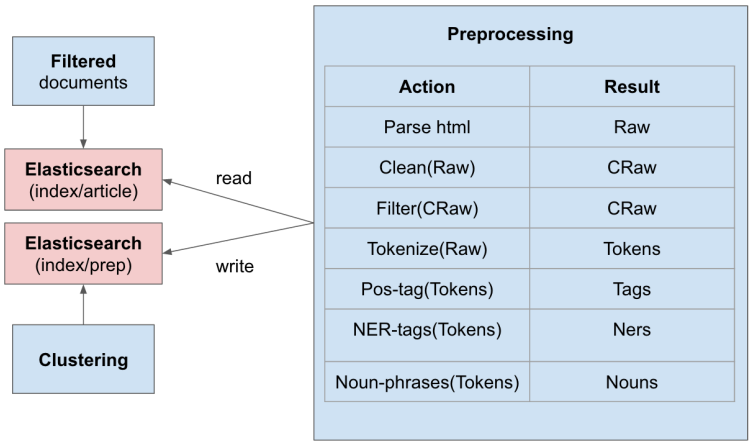

Figure 3.2: "Preprocessing"

In figure 3.2 “Raw” is the parsed text from the html, while “CRaw” represents text that contain no numbers, stopwords, short words, punctuations and special characters. The tokenization is not a whitespace splitter, it is a fully trained statistical parser that detects sentences and boundaries, as well as sentence tokens. The same goes for noun phrase extractors, pos-taggers and ner-taggers. “CRaw” is used as a representation of the text, while “Raw” is used to generate the beforementioned concepts.

The Clustering module does several transformations as displayed in figure 3.3. Feature selection is the process of picking features that are of most interest. This often happens mathematically by *TF-IDF* or by formulas describing information density of words. If desired, resulting features can be enhanced by definitions and categories of *Wikipedia* or projected via *WordNet* ontologies, see section 4.3. Before the actual clustering algorithm is run similarity matrices and normalization to unit vectors is done. After this documents can be clustered by any kind of clustering algorithm. Outcomes are evaluated and redone if desired evaluation criterias were not met.

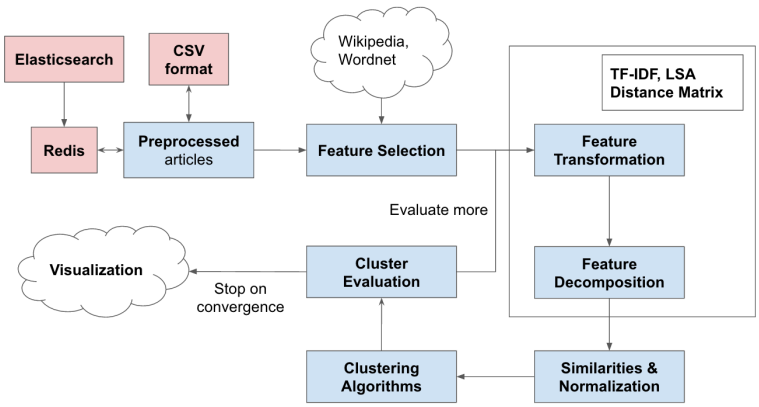

Figure 3.3: "Clustering cycle"

The Summarization module would receive the clustered documents as input. In addition meta information can be used here as well. The summarizer either gets single documents or a collection of documents that where in the same cluster. If assignments are soft, possible cluster overlaps can be taken into account. As this is not further part of the thesis we will leave it at that.

Output generation is a matter of visualizing the results of a summarization engine. For instance a website with rendered html. Currently clustering results can be projected to lower dimensions and to render 2D or 3D visualizations. Additionally the measures of clusterings can be compared and examined.

The Routing engine detects if the input is a type of event. They are categorized into "single-events", "multi-events", "person-centered" and "other". A single event is about the same topic e.g. the earthquake in Nepal 2015. Multi-events take place at different times, locations and with different subjects but with the same content, like terrorist attacks in 5 countries. A person-centered view is often a biography like a profile of Barack Obama running for president. This view makes it possible to make different assumptions about how articles should be summarized in case of the category of events that occured. A router like this could bring in various performance gains and strengthen the quality of a summarization. Due to time constraints this was left out of *News-Clusty*.

3.2 Python and Libraries

News-Clusty is entirely written in the programming language Python and a well versed collection of libraries.

Python is a dynamic, object oriented higher level programming language with a rich environment for scientific computing. Python incorporates many different styles and assumptions, leading to a variety of programming paradigms between object oriented and functional programming. From the Zen of Python: "Beautiful is better than ugly. Explicit is better than implicit. Simple is better than complex.". For a proper introduction to Python see Bird et al. (2009).

The Persistence of articles is held in *Elasticsearch*. *Elasticsearch* is a document store that can be easily distributed accross hundreds of nodes. It is schemaless and can be dynamically adjusted if need be. Its primary use case is for search and anything related to huge chunks of text. *Redis* on the other hand is used as a cache for different feature selection strategies. It is a key value store and commonly chosen for fast execution. The common strategy is to first generate feature sets by querying *Elasticsearch*. Each successful feature selection on a document will be persisted as a numerical feature vector into *Redis*. Multiple accesses to the same documents will be read from *Redis* if possible.
While databases have huge advantages by querying documents, there is an overhead when accessing all files at once. If for a particular reason a whole dataset must be used, it is advised to use some of the serialization tools that output sparse matrix market formats or serializes to raw text files. Input and output on filesystems is faster if you do not need to query the data. This holds true as long as a single node filesystem is involved. *Elasticsearch* for instance has its powers by leveraging clusters of servers (nodes).
The indices for *Elasticsearch* are scoped by the scraping date, $20150701/article/id = document$ or $20150701/prep/id = document$. This is particulary helpful if continuous processing, day by day, is an objective. The reason for choosing both is that we get highly expressive power and fast solutions for document queries and caching. If for several reasons the system should be setup in a distributed system of server nodes, both databases come with well defined abstractions.

The Libraries used throughout *News-Clusty* are well-versed scientific data processing libraries. The library creators and maintainers should be mentioned and accredited for. *News-Clusty* leverages linear algebra, feature selection strategies and clustering algorithms from

Scipy (especially numpy and sklearn) see Pedregosa et al. (2011). It uses natural language analysis tools from the nltk by Bird et al. (2009). Moreover named entitiy recognition by the *StanfordNLP group*, noun phrase extraction by *Conll*, word ontologies by *WordNet*, knowledge and dictionaries by *Wikipedia*. Additionally *newspaper3k* for scraping newspaper articles and *gensim* build by uv rek and Sojka (2010) as high level abstractions for probabilistic topic models. In the future it would be good to rely less on certain libraries where it makes sense, e.g. not using *newspaper3k* for scraping or *Scipy* for clustering algorithms.

4 Feature Selection

"Coming up with features is difficult, time-consuming, requires expert knowledge. "Applied machine learning" is basically feature engineering."

Andrew Ng

Considering the "right" features for clustering is a demanding and error prone process. Currently there is really just one way of describing documents: The vector space model. It breaks down to counting occurences and cooccurences of words and measuring distance by mathematical functions. We could just take all the words of a document, removing stopwords, and put them into a feature vector. This results in dimensionality explosion and extreme noise. Contrary to a document vector $d = \{w_1, w_2..w_n\}$, the feature vector represents a document by concepts $\{c_1, c_2, ..c_j\}$. It is a projection of the original document $d = \{w_1, w_2..w_n\}$ to more general concepts resulting in fewer dimensions. This lifting is bestly described as combining several words of a document, often occuring in the same sentence, extracting a shared meaning. We hope to find fewer words that share enough information with the original word, that the following holds:

$$f : d = \{w_1, w_2, ..w_n\} \rightarrow \{c_1, c_2, ..c_j\} \tag{4.1}$$

The function f transforms a sequence of words $w_1..w_n$ of a document d to a sequence of concepts $c_1..c_j$. The concepts can be derived in a lot of ways.

1. Pruning words of low and high significance.
2. Using syntactic parsing to retrieve noun phrases, named entity tags or part of speech tags.
3. Using ontologies of *WordNet* to derive a shared meaning of words.
4. Mapping documents to *Wikipedia* categories.

5. Using kernel methods (constrained clustering), preselecting initial clusters in a semi-supervised way.

In the end, feature selection is probably the most demanding task. Expert knolwedge needs to be applied and can change over time, called time drift. A computer handles documents in vector space, by counting. A human however perceives content differently. For any sufficiently adavanced algorithm that works with a knowledge base it is still: Garbage in, garbage out. More fancy algorithms will lead to better results, but better features will accelerate the accuracy.

In the following we will briefly explain what *semantics* mean, especially in the *domain* of newspapers. How *feature selection* generally works and how this can be enhanced by *syntactic parsing*. Strategies using *WordNet* and *Wikipedia* are explained. In the experimental chapter we will then present how all these mechanisms come together.

4.1 Semantics

Semantics is the study of meaning. Given some symbols, characters, words or phrases what is their underlying meaning? The question is inherently hard and lots of literature focuses on how computers can get better at this. Most of the concepts depicted here taken from Jurafsky and Martin (2000). Semantics can also be viewed from a statistical point of view. Given a lot of phrases and words, can we infer their underlying structure that generated them? How can statistical patterns reveal what was meant and to what degree?
In computational linguistics we often speak of *word-sense disambiguation (WSD). WSD* is short for identifying sense of a word, if a word can have several meanings, in a sentence or paragraph. The sentence

"The bail out during the financial crisis of the Lehmann brothers bank, was much too late."

makes it obvious that it is about financial institutions "bank" during the financial crisis, political intervention by providing money "bail out" and a specific financial institution or entity "Lehmann brothers". How could we possibliy discern such a sentence so that we can actually reveal all the beforementioned concepts? To successfully find such concepts we have to identify what parts of speech, e.g. nouns, verbs, adjectives etc., each word of a document has:

$$postag(d = [w_1, w_2, ..w_n]) = [(w_1, tag_1), (w_2, tag_2), ..(w_n, tag_n)] \tag{4.2}$$

Part of speech tagging works by parsing a document sentence by sentence. Tagging each word by identifying its position relative to other words and predicting what their tags might look like. See (Jurafsky and Martin, 2000, chp. 5) for an extensive study. For clustering we want to identify these *semantic fields*, a set of words grouped by meaning. We then often analyze *WSD* through *synonymy, polysemy, hyponymy, hypernymy and meronymy*. All of those concepts are important to taxonomies and ontologies. A *taxonomy* is refered to as a simple hierarchical structure of parent-child relationships, that change in granularity per hierarchy level. An *ontology* is much broader and can have complex relations other than parent-child. In that sense both taxonomies and ontologies are structures, showing how to classify words in context to each other. Traversing through these hierarchical structures is typically done by hyponyms and hypernyms.

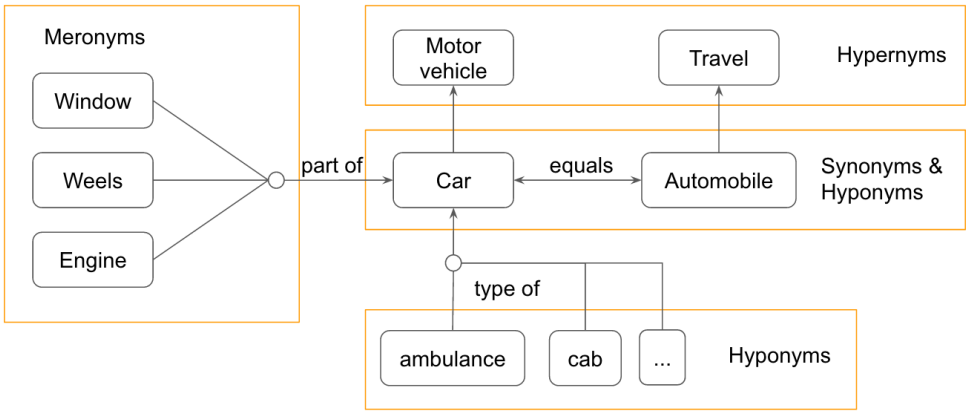

Figure 4.1: "Semantic fields, hierarchies"

Meronyms are “part-of” relations, *hyponyms* have a “type-of” relation to a higher concept, called *hypernyms*. In figure 4.1 we see that from a single concept “car” we can infer a semantic field around it. We will see later how this works, a statistical concept around this is *LSI* and *probabilistic topic modelling*. The symbolic way is to use knowledge bases such as *WordNet* and *Wikipedia*.

In the text *domain* of newspaper articles a few problems arise. Analyzing a long book, a long speech or journal articles from the scientific community, is easier compared to high varying fragments from different authors on different topics. A long book written by one person will use a specific language that is typical of that author. Speeches for a specific person contain similar concepts and often use the same language as well. In the scientific community

rhethorial and anecdotal phrasing is uncommon. Facts, citation and correct formatting is of central importance. The text might be heterogenous but the fact remains that a certain wording / glossary, is shared throughout these examples.

This does not hold true for newspaper articles. *Topics* about different events co-occuring on the world. Different *authors* with different writing styles. Different *newspapers* with different directions of content presentation. *Long* and very *short* articles. And this does not take images, videos or comments into account. When dealing with a vast landscape of different topics, spanning connections between two documents becomes a hard task. In context of such sparse data, that is documents with almost no connecting words, clustering works poorly. This can also be described as a high variance problem, where each document contributes a lot of unseen words to the feature vector.

In order to avoid these variance problems, we need to find a solution to *WSD* and then apply a lifting from the original concept to a hypernym. Going back to figure 4.1 we see that *car* and *motored vehicle* might mean the same thing. Both are about cars, if we project the concept *car* to *motored vehicle* the concept would connect both the documents containing car and motored vehicle. This is not always what we want to achieve but it could drastically improve similarity between documents that would miss each other by synonymy and polysemy.

4.2 Selection

As described before we want to tackle *WSD* and find ways to connect documents that share common meaning but not a lot of common words. To do so we have several strategies at our disposal. First, word pruning is presented, it is probably the most widely used technique for lowering dimensions and removing insignificant words. Second, synctactic parsing is described in by part of speech tagging, noun phrase extraction and named entity recognition. One topic which is left out are the kernels. They are important to retrieve better results in a semi supervised way but could not make it into the thesis experiments.

4.2.1 Word pruning

Before pruning words we have to convert the documents into a suitable vsm such as *counting vectors* or *TF-IDF* described in 2.2.2. The counting alone in its basic form is sufficient in telling if a term has a high or low connection with all other documents. The *TF-IDF* on the other hand is a measure of importance and solely based on the resulting frequencey per word. It is

possible to make significant pruning on both representations. The *TF-IDF* variant is prefered as it normalizes frequencies. Higher or lower counts are weighted into a formula that better represents the significance of a term.

Either way we need to create a cooccurence matrix $M = count(C, D)$ where C is a corpus and D is the dictionary of the corpus. Then we transform by $M = tfidf(M)$ or leave it with the *term frequency*.

Pruning M is done by cutting off the documents with a very low ratio of counts with respect to all documents. The procedures work in the single day clustering case. For multiple days, a caching on the pruned words, need to be implemented. This can be done by threshold in proportion to all terms, a percentage, removing j terms. It can also be achieved by a hard count, cutting of all terms that have no counts higher than that. This means, we remove insignificant terms or terms that do not contribute to any connection. Semantically this means, we cut off words that have a high meaning in a single document and a very low in others. Those words are redundant, or in other words they have no discriminant value to the clustering process.

$$\begin{aligned} m &= [0, 1, 5, 0, 4, 0, 1, 1, 2, 3] \\ m_s &= sort([0, 0, 0, 1, 1, 1, 2, 3, 4, 5]) \\ percentcut(m_s, 0.2) &= [0, 1, 1, 1, 2, 3, 4, 5] \\ totalcut(m_s, 1) &= [2, 3, 4, 5] \end{aligned} \tag{4.3}$$

The min cut on percentage $0.2 = 20\%$ cuts the first 2 samples or removes the first 6 in case of a total count. The parameter has to be varied, depending on the outcome of a cost function. Further we can take off the top j words as well by the same principle. The problem with the top words is, that they highly correlate with a lot of different documents, meaning a high correlation between a term and the corpus. Leaving them out erases a lot of connections, resulting in more discriminant features. This is what we want to achieve, finding the middle

words, that are common in certain documents and uncommon in others. During clustering this will result in much more coherent clusters.

$$\begin{aligned} m &= [0, 1, 5, 0, 4, 0, 1, 1, 2, 3] \\ m_s &= sort([0, 0, 0, 1, 1, 1, 2, 3, 4, 5]) \\ maxcut(m_s, 0.8) &= [0, 0, 0, 1, 1, 1, 2, 3] \end{aligned} \tag{4.4}$$

The max cut works with a ratio that selects from lowest to highest 80% except the last 20%. Note that the samples are not on m dimensional vectors. For this to work we have to aggregate the counts and then prune the most insignificant words. The great thing of this approach is that it can follow any feature selection strategy. By using word pruning on feature vectors, the selection process can be refined. The fine tuning is necessary in gaining percentages in accuracy.

4.2.2 Syntactic parsing

Parsing reduces symbols, to a parsed tree of following expressions. In English parsing, probabilistic shift reduce parsers in combination with trained neural networks is state of the art. Zhu et al. (2013) Valid words and characters are defined within the rules of the English alphabet. In figure 4.2 we can see a parse tree for an English sentence parser. The parsing is syntactical based on English grammar. When it comes to parsing expressions from English language to a meaningful representation for the computer, the problem statement gets a lot more difficult. Too many words, too many forms of sentences - statistics in combination with structural parsers come to the rescue. Solely syntactic parsers that work on symbols work well in a specific domain up to a certain parsing accuracy. Statistical models remove these barriers by likelikhoods and predictions, finding answers accross domains. The most inherent problem is overfitting. Overfitting describes that a probabilistic parser works well on a specific problem domain or data source, because it learned the domain well. Unseen domains and datasets perform poorly due to the missing knowledge. The problem becomes more clear, in the domain of newspaper articles where different language styles and domains are frequent. Statistical parsers like the *Stanford* parser are very accurate in identifying valid syntactic English expressions. In the following we will look more closely at noun phrase extraction and named entity recognition. They can be used as initial seeds for the feature space of a corpus.

Figure 4.2 is displayed as a projective tree, where each word has exactly one (or none) incoming and outgoing edge. The outgoing edge is refered to as head. Edges contain information

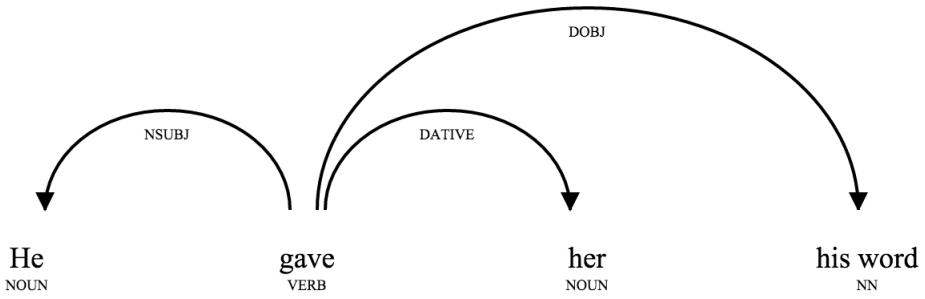

Figure 4.2: "Syntactic parsing"

about the relation between both words. Each word has a part of speech tag described by many corpuses. Current research projects use Googles ngrams over time, with a 3 billion ngram corpus to infer the most likely structures. Goldberg and Orwant (2013) The parsing of English grammatical structures is entirely dependant upon specification of the rules. What part of speech tagging system is used? How is the grammar defined? With the help of the Chomsky hierarchy it was proven that not all natural languages have context-sensitive (type-1) nor turing complete (type-0) grammars. See (Jurafsky and Martin, 2000, chp. 16) for a great introduction. In clustering this is not a huge problem, because the granularity of the task is not that fine tuned. A slightly false parse might still yield good feature results.

Noun phrases

Noun phrases are often referred to as key phrases. Formally a noun phrase is a phrase with a noun as its head word. A head is a word that determines the syntactic type of a phrase. Generally a noun phrase is a part of a sentence that captures meaning of a sentence. Thusly they are good samples to represent a document. However noun phrases are rather bad samples for clustering. They do not connect well to other documents due to their unique occurences. In the *WordNet* section we will see how they are great for projecting to their respective hypernyms. Noun phrase extraction is highly connected with part of speech tagging where certain tag patterns are used to filter nouns or noun phrases. For further information see (Jurafsky and Martin, 2000, chp. 5, 12).

Named entities

Named entity recognition is the task of information retrieval that extracts and classifies names of persons, organizations, locations, expressions of times, quantities, monetary values, percentages, etc. (Jurafsky and Martin, 2000, chp. 22).

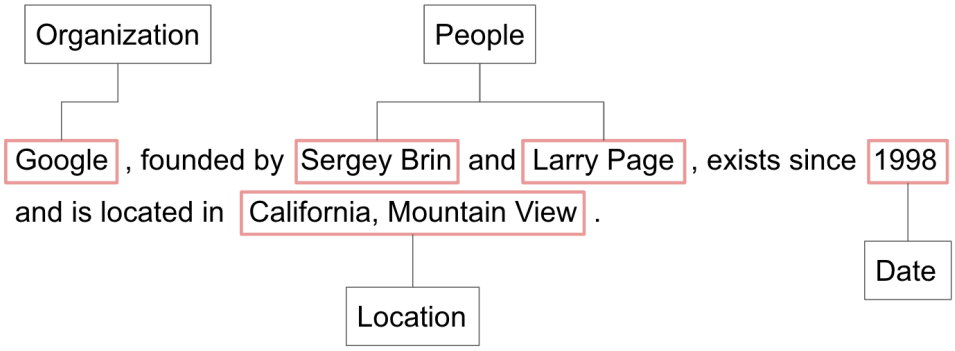

Figure 4.3: "Named entities"

As seen in figure 4.3 we are interested in informative terms. Often occuring named entities tend to give a direction what a document is about. If Google occurs in one document, it most likely will have to do with a lot of other documents that are about Google. We would like to favor documents that are connected by their respective named entities and weight them in more than others. Named entities like noun phrases are very discriminative. In systems like the Columbia Newsblaster system, named entities would be used to determine if a document is biographical, has large shifts in time (dates) or connecting documents based on significant organizations, peoples etc. McKeown et al. (2001)

4.3 External Knowledge

In order to enhance the syntactical selection models we can add a knowledge base such as *WordNet* and *Wikipedia*. The data reresentation is often defined as a typical dictionary. For a definition of a word we can infer semantic fields and additional text describing the words in more detail. Moreover knowledge bases such as *Wikipedia* categorize/classify concepts into ontologies. *WordNet* and *Wikipedia* are great in the sense that human authors around the world add missing information and enhance the models frequently. The knowledge bases are enhanced frequently by writing rules and reviewing processes.

4.3.1 WordNet

WordNet is a lexical database of English. Lexical parts of speech such as nouns, verbs or adjectives are grouped into synonyms (synsets). Each synset is linked by semantical and lexical relations. In parts *WordNet* resembles a thesaurus, grouping words based on meaning. With its onotologies it deals with *WSD*. Further words are interlinked by semantic relations. For more information see Miller (1995); Fellbaum (1998).
WordNet is often used for lemmatization. Lemmatization is the process of removing the inflected forms of a given word to its lemma. A lemma is a dictionary entry or canonical form of a word. The major difference to stemming is that *WordNet* is able to inflect a canonical form that depends on context sensitive pos tags.

$$\begin{aligned} lemmatize(\text{"}savings\text{"}, pos = \text{"}noun\text{"}) &\rightarrow \text{"}saving\text{"} \\ lemmatize(\text{"}savings\text{"}, pos = \text{"}verb\text{"}) &\rightarrow \text{"}save\text{"} \\ stem(\text{"}savings\text{"}) &\rightarrow \text{"}save\text{"} \\ stem(\text{"}save\text{"}) &\rightarrow \text{"}save\text{"} \end{aligned} \tag{4.5}$$

Inflecting the canonical form with pos tags enhances the precision of the context where the words came from. Often this can result in different inflectional forms that would otherwise be equal. In comparisson we can see that stemming treats “savings” and “save” entirely the same. This is a basic strategy to inflect a generalized version of a document, to lower the dimensions. Further we can take the hypernyms of a document.

$$\sum_{i=1}^{|d|} hypernyms_{first}(d_i) \tag{4.6}$$

We iterate over all words in d and take the first of the hypernyms of the words, projecting it to a higher concept. *WordNet* sorts the hypernyms from most likely to less likely, taking the first is a good approximation. Enhancing the above statement we scale this up to the depth d going up the hypernyms of the *WordNet* ontologies.

$$\begin{aligned} closure(seq, d = 0) &= empty \\ closure(seq, d > 0) &= \sum_{w \in seq} closure(hypernyms(w), d - 1) \end{aligned} \tag{4.7}$$

At last, we can infer the most common meaning of a sentence by calculating the lowest common hypernyms.

$$lowest\_common\_hypernyms(w_1, w_2) = hypernyms(w_1) \cap hypernyms(w_2) \tag{4.8}$$

For this we compute for each word in a document the transitive closure with the above statement not restricted by the first hypernym.

Algorithm 4 *WordNet* closure with hypernyms

```
for sent ∈ closure(doc, d) do
    for (w1, w2) ∈ sent : w1 ≠ w2 do
        result ← lowest_common_hypernyms(w1, w2)
    end for
end for
return result
```

In the following we give an example to understand the notion of a lexical closure. Given a document with one sentence, *the dog and the cat*. The transformation of this sentence stripped by stopwords results in

$$\begin{aligned} sents &= [(\text{"}dog\text{"}, \text{"}cat\text{"})] \\ dog &= synset(\text{"}dog\text{"}) \\ cat &= synset(\text{"}cat\text{"}) \end{aligned} \tag{4.9}$$

The closures by hypernyms for each word results in

$$\begin{aligned} clos1 &= closure(hypernyms(dog), depth = 2) \\ &= (\text{"}canine\text{"}, \text{"}domestic\_animal\text{"}, \text{"}carnivore\text{"}, \text{"}animal\text{"}) \\ clos2 &= closure(hypernyms(cat), depth = 2) \\ &= (\text{"}feline\text{"}, \text{"}carnivore\text{"}) \end{aligned} \tag{4.10}$$

Then we compute the lowest common hypernyms that connect hypernyms of the word closures.

$$lowest\_common\_hypernyms(clos1, clos2) \rightarrow "carnivore" \tag{4.11}$$

We then apply word pruning to get the mid vector words, sorting out extremes. It is also possible to greatly enhance the above models with lexical chains, see Wei et al. (2015).

4.3.2 Wikipedia

Using *Wikipedia* in the feature selection process works analogous to the *WordNet* approach. *Wikipedia* will be used as a holder for metadata and not for the content of the articles alone. We use conceptual words that represent a document well, like noun phrases and named entities querying the *Wikipedia* knowledge base. *Wikipedia* has a lot of different knowledge kinds, like ontologies between definitions, that connect their relations similar to *WordNet* ontologies. Each definition is categorized into distinct subgroups of relatedness by broader categories, like politics or economics. Additionally all links referencing external definitions can be found in the documents of a *Wikipedia* concept.

The basic concept is to use a noun phrase and find outlinks and categories from *Wikipedia*. Categories of *Wikipedia* vary widely and are not normalized in any way. For classification purposes a scheme has to be found to classify documents. However using clustering to group documents into similar events, a scheme was proposed by R. et al. (2010) and Hu et al. (2009). In these approaches a document vector is mapped to categories and outlinks. However using noun phrases, one can find a good representation of a document while at the same time lowering the dimensionality of the data.

In another study by Hu et al. (2009) concepts and categories are mapped by relatedness. The two approaches are exact-match and relatednessmatch, where both take documents and map their respective words to the *Wikipedia* onotologies. The exact-match is based on the fact that *Wikipedia* contains concepts, describing a word, the relatednessmatch is based on the article content. For each concept a mapping to categories is build in order to represent documents as a sum of *Wikipedia* categories. Through this dimensions are greatly decreased. The hypothesis is, that a document composed of unique frequencies of categories can represent a document better.

Advancing on this concept, a solution to the semantic problem of a document is proposed by R. et al. (2010) using outlinks and categories of *Wikipedia*. The process is that each document gets a set of concepts by *Wikipedia* based on the document words. A set of outlink concepts and a set of category labels. Each set is then weighted into an accumulated cosine similarity, where parameters control the impact of each set.

In this thesis, no evidence was found, that mapping to category concepts, could greatly enhance feature selection. This it not due to a missing effect but rather due to time constraints. In chapter 7 we will briefly reference the beforementioned steps. The strategies should be investigated further.

5 Clustering experiments

"An algorithm must be seen to be believed."

Donald Knuth

In this section we evaluate some strategies for clustering. At first we will show how a single day of news events are clustered. Then clustering news events over multiple days. Due to time constraints it was not possible to elaborately test multiple days clustering. It is a notoriously difficult topic that greatly relies on the data and time. Originally it was intended to use a scraped dataset over several month, categorized by date, with as much meta data as possible. As this is at the barrier of legallity, a smaller standardized dataset namely the BBC dataset by Greene and Cunningham (2006) was used. In the original dataset it was possible to assume that the data was ordered and sorted by day. Several thousand articles could then be easily clustered on a daily basis and enhanced over the course of several months. Instead we are focusing on how the BBC dataset can be clustered and evaluated. Multiple day clustering will be depicted, but not in any form evaluated.

The BBC dataset had a few limitations in the sense that dates were not provided and meta data not provided by the authors. As such we treat the data as a a bulk of events occuring in the timeframe of 2004-2005. Fortunately unlike real world data the articles are tagged with precise categorical labels, so external evaluation measures were an option. There were limitations in time to find a more adequate set of articles that were consecutively sorted by day in a broader category such as "world news".

The clustering was done in several steps, similar to the Columbia Newsblaster system.McKeown et al. (2002) A hierarchical clustering approach by first clustering events into categories like politics, business or entertainment and second clustering the categories into distinct events. The classification in the first step garantuees some broader topical cohesion. In the BBC dataset a purity of up to 94% could be achieved on the test set. This is unusually high, but is probably due to the normalized dataset that represents the classes exceptionally well. The real world

dataset could go as high as 45% purity and less. The purity on the BBC dataset varies broadly between 79% and 94%. This variability results in overlapping topics where the majority of documents in a cluster are in the same class. Each of the resulting clusters resembling the underlying classes are reclustered. A clustering algorithm that optionally does not take any constraints on the amount of clusters is used. In this second step clusters represent news events on several topics per cluster.

Some final words on the measurement of the two steps. In the first step it is easy to evaluate how well documents were assigned to their respective classes. In the second step however evaluation is not that simple. At this point, we are interested in document similarity based on content and not class labels. The results of the second clustering are informal and not measured.

5.1 Single day clustering

Single day clustering is done by focussing on one specific date. The hypothesis is, as long as there are no major news headlines, documents will have a high variance in vector space. Thusly there will be more clusters. The major news headlines is important, because, during the original dataset scraping there were ground breaking events, that literally absored almost all articles into single clusters. That was expected because roughly 90% of the articles were about the same news event. The angle of the different articles covering the event varied greatly enough that topics that had no underlying connection to the event were clustered into as well.

In the following the two steps are refined. First we want to classify news events by classes, in case of the BBC dataset: politics, business, entertainment, tech and sports. For this it is possible to use algorithms with a hard number of final clusters to classify documents into the classes. Alternatively one could use a supervised classifier, which in practice, is much more accurate. See Chase et al. (2013) for a real world example learning labels in comparison of *K-Means* and a *supervised classifier*. Second the resulting clusters, now assigned by news category are clustered again. This time the clustering algorithm has the constraint of automatically detecting the cluster numbers. For this a hierarchical algorithm with soft thresholds was chosen like BIRCH or Ward linkage. The resulting clusters resemble news events dealing with a similar topic.

5.2 Implementation

The implementation of a more general clustering algorithm in the text domain can vary drastically. A general scheme is given by algorithm 5.

Algorithm 5 General clustering

function GENERAL_CLUSTER($X_{train}, y_{train}, \kappa, \alpha, d$)
    $X_{norm}, y_{norm} = normalize_{text}(X_{train}, y_{train})$
    $X_{project}, y_{project} = project(X_{norm}, y_{norm})$
    $X_{trans}, y_{trans} = transform(X_{project}, y_{project})$
    $X_{lsi} = lsa(X_{trans}, topics = \alpha)$
    $X_{dim} = reduce\_dimensions(X_{lsi}, dimensions = d)$
    $X_{sim} = similarity(X_{dim})$
    $X_{scaled} = scale(X_{sim})$
    $model, centroids, labels, k = cluster(X_{scaled}, clusters = \kappa)$
    return $model, centroids, labels, k$
end function

The general implementation 5 is separated into several steps. As input the function gets a training set X_{train} and a label set y_{train}. κ is an optional parameter for the total amount of cluster centers. α a parameter that sets a desired size of topics for models like *LSA*. And a dimension d to reduce X to for visualization purposes by e.g. *PCA*. In general we assume that the data X_{train} is of any textual sort, the preprocessing needs to do the necessary steps that fits the function. Moreover it is of utmost importance to think in terms of vectorized code. Any procedures gradually transform a document into the vector space. After this all rules by linear algebra hold.

1. *normalize* takes X_{train} and y_{train} to remove special characters, stopwords, numbers. Lower casing any words and removing non english characters or sentences.

2. *project* is any kind of projection strategy via *WordNet*, noun phrases or named entities, lowering word sense disambiguation and dimensions. This step deals with the initial seed of knowledge that is used throughout the algorithm.

3. *transform* takes in the projected data and transforms it via TF-IDF, word pruning, as well as ngram enhancement.

4. *lsa* is an optional step that transforms the resulting data to dense low dimensional matrices, keeping as much variance as possible while reducing noise. In this case *LSA* was used but *LDA* or *HDP* would be an option as well.

5. *reduce_dimensions* is an alternative step that reduces the dimension of the data to 2d or 3d plotting clusters. This is typically done by *PCA*.

6. *similarity* transforms document x term vectors to document to document similarity by cosine or euclidean distance measures.

7. *scale* is a second normalization step, scaling the data by variance and average.

8. *cluster* finally takes the matrix X_{scaled} and clusters based on a hard constrained clustering number. If not supplied the clustering algorithm needs to predict a cluster number automatically. This results in the actual cluster amount k. The resulting trained clustering $model$. The $centroids$ in case they are actively calculated e.g. by *K-Means*. And finally the $labels$, the assignment from a document to a corresponding cluster.

The implementation is a rough scetch of a real clustering scheme. In reality most of the beforementioned steps can be parameterized or constrained by addtional arguments. Either a configuration object is specified or an extensive API used to create clustering scripts. The API used was $scikit-learn$ by Pedregosa et al. (2011) and is part of the huge scientific computing libraries written in Python, such as $numpy$, $scipy$, $pandas$ etc. From this we can generalize for a clustering scheme over a single day.

Algorithm 6 Single day clustering

$X_{train}, y_{train} = get\_data(timestamp)$
$model, centroids, labels, k = general\_cluster(X_{train}, y_{train}, \kappa, \alpha)$
$X_{assigned} = get\_assigned\_documents(X_{train}, labels)$
$news\_labels = []$
for $c \in X_{assigned}$ **do**
    $model', centroids', labels', k' = general\_cluster(c, nil, \alpha)$
    $assign(news\_labels, X_{train}, labels')$
end for
return $news\_labels$

The implementation 6 is a rough scetch of what the *News-Clusty* system does. First get the data by a timestamp e.g. “20150701”, cluster the data by a fixed κ and then use the resulting

cluster assignments $X_{assigned}$ to group news events. Note that $general\_cluster$ in the second run has a nil argument, because we do not want to constrain by cluster amount. The clustering now depends entirely on the variation of the intermediate procedures of $general\_cluster$ and is now an optimization problem.

5.3 Evaluation

In the evaluation section the beforementioned measuring strategies are used. Several different strategies are weighted against each other. It was found, that there are several different ways with a good performance in the categorial clustering approach. The second clustering is not of utmost importance in the evaluation but is depicted later on. A major problem is, that several parts of the proposed algorithm has parameters that need to be estimated. To narrow down this combinatoric explosion of parameters, several assumptions can be made.

1. Limiting the strategies to simple word tokens and their document titles except stopwords, noun phrases and ner tags as well as *WordNet* first hypernym and *WordNet*lemmatization.
2. *Transforming* by *TF-IDF* with a 80% maximum threshold and a hard count 3 for minimum threshold, excluding ngrams.
3. *topic modelling*, whether *LSA* or *LDA* should be used or not. The topic size is supervisedly adjusted.
4. Setting the *distance measure* to *cosine*.
5. *clustering algorithm* constrains, such as density thresholds and maximum *HAC* depths. Classification is constrained by $k = 5$ for the categories of the BBC dataset.

A comparisson of the different strategies with *LSA* enabled, for the classification task, is shown in table 5.3. Note that the *purity*, *v-measure* and *silhouette score* is averaged over 5 consecutive runs.

Concluding from this, word tokens in combination with noun phrases lead to the highest v-measure. While other strategies perform still well enough, the chance for the best classification by clustering stems from purely syntactic means, e.g. word tokens and noun phrases. With *WordNet* lemmatization we have a highly effective approach. That is, from the original 12000 feature dimensions we projected to 6000. Except for lemmatization, *WordNet* approaches performed worse than the mentioned strategies. This is not generally the case as can be seen in Wermter and Hung (2002) for an example on the Reuters Corpus with *WordNet* using a

Strategy	v-measure	purity	silhouette
Word tokens	0.812	0.926	0.118
Syntactic	0.795	0.916	0.109
Word + noun tokens	0.826	0.934	0.118
WordNet $hypernyms_{first}$	0.710	0.884	0.068
WordNet lemmatization	0.726	0.895	0.069

Table 5.1: Comparison of feature selection with *LSA*

Strategy	v-measure	purity	silhouette
Word tokens	0.667	0.831	0.073
Syntactic	0.634	0.794	0.067
Word + noun tokens	0.666	0.831	0.074
WordNet $hypernyms_{first}$	0.585	0.784	0.048
WordNet lemmatization	0.585	0.784	0.048

Table 5.2: Comparison of feature selection without *LSA*

multinominal Naive-Bayes classifier.
We can see in table 5.3, that the same measurement without *LSA* performs in almost all instances worse. Note that *LDA* was explicitly excluded from the possible topic modelling techniques as it promotes a poor pre clustering step. This is due to the fact that *LDA* maps several topics to documents, resulting in documents having assignments to many topics. This is a desired property for the second clustering run.

The discrepancy between v-measure, purity and silouhette is expected. v-measure takes into account that several clusters contain an amount of labels that are not dominant. The higher the variance of spreading labels, even if their occurence count is low, leads to a penalty in v-measure. Purity on the other hand compares how many dominant labels are in one cluster and penalizes for each non dominant label. It does not account for the variety of falsely assigned labels. The silhouette score is extraordinary low due to the high dimensionality. It can not be taken in absolutes but rather in relative terms between different strategies. The higher the score the higher the cohesion between topic segments. In total, the silouhette score is better when the purity and v-measure is relatively higher.

5.4 Experimental

While clustering for classification is a very good first step to preselect documents into isolated clusters, the requirements change in the second run. The results do not need to be constrained

by cluster amount. Selection of similar news events return smaller topical clusters. It is difficult to show that a connection between different documents can be made when no information about time are present. If a document about “taxes in the us” occured at a specific day and a second document about “Obama raises taxes” at the next day occured, we could conclude that it is a current topic in the news that are related. Without knowing when a document was published, it is not possible to assume that two documents about taxes, are about the same event or piece of information. The second event could have happened 9 months later or earlier. The content might be connected but the events are not. There is no sense in trying to show how two consecutive days would relate to each other. That is why visual representations of topic and word proportions in clusters will be shown, instead of timeline data that relates. Two additional clusterings with BIRCH and *LDA* are proposed.

First let us demonstrate the generative model. In this instance *LDA* has shown promising informal results. *HDP* would be an optimal choice due to the automatic topic amount detection, but was left out as it was too complex to adapt. In figure 5.4 we see a clustering by *LDA* over the business cluster.

$topics \backslash words$	w_1	w_2	w_3	w_4	w_5
t_1	$strong$	$growth$	$manufacturing$	$bank$	$economy$
t_2	$city$	$boeing$	$year$	$company$	$industry$
t_3	$year$	$india$	$germany$	$market$	$growth$
t_4	$tobacco$	$government$	$court$	bn	$case$
t_5	$sales$	car	gm	bmw	$year$
t_6	$economy$	$spending$	$growth$	$recession$	$japan$

Table 5.3: "Business topic proportions"

Typical words such as “growth”, “manufacturing” or “bank” resemble this. A second example of a word to topic proportion can be seen in figure **??** for political topics.

As these word clouds have little value by measurement, they are easy to interpret. In reality both the political and business cluster had roughly 50 subclusters, only showing the first 6 and their respective top 5 words. Note that a topic from *LDA* is not exactly the same as centroids of a *K-Means* algorithm. *LDA* topics relate to centroids as the *EM* algorithm to *K-Means*.

The second approach with the BIRCH algorithm yielded some interesting results as well. We can see in figure 5.1 that there are a lot of clusters of different sizes. There are much

$topics \backslash words$	w_1	w_2	w_3	w_4	w_5
t_1	*election*	*new*	*blair*	*minister*	*tax*
t_2	*government*	*home*	*law*	*people*	*police*
t_3	*government*	*defence*	*uk*	*guantanamo*	*evidence*
t_4	*million*	*year*	*education*	*party*	*marketing*
t_5	*uk*	*card*	*community*	*troops*	*fraud*
t_6	*access*	*child*	*parents*	*years*	*care*

Table 5.4: "Political topic proportions"

bigger clusters and smaller clusters. In news events we mainly want smaller coherent clusters. The centroids are the bigger points painted in a distinct color. Smaller points are documents assigned to a centroid in the same distinct color.

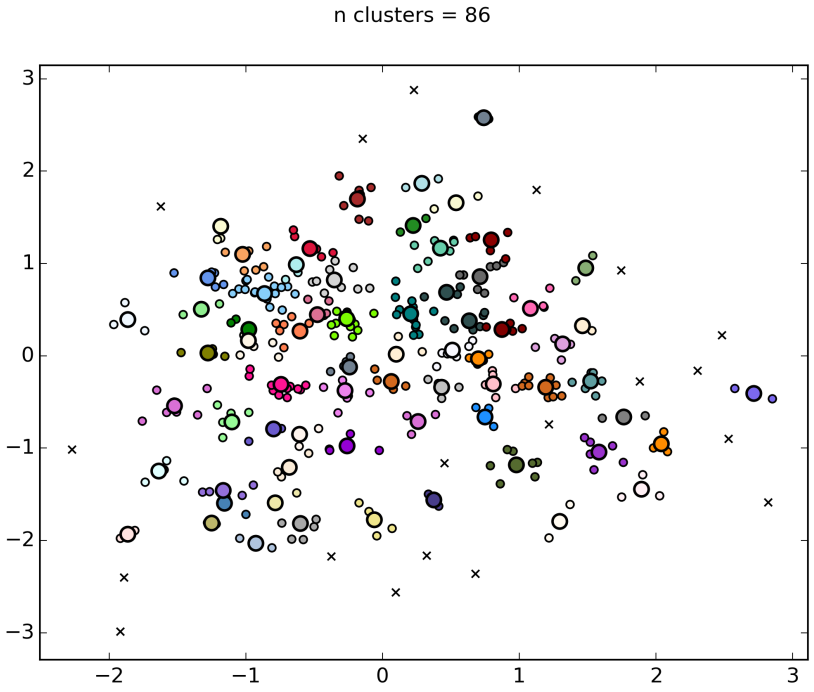

Figure 5.1: "BIRCH clustering of a single day"

The data at hand does not stem from the BBC dataset but from daily scraped newspapers. The papers shall not be named due to legal reasons. It is reasonable to believe that even

less important events have multiple cluster points from different news sources. One can see that the resulting clusters are small and well formed. Some cluster centers are bigger than others. In those cases we can recluster them to further enhance the variance of clusters. The structures resemble that there are news that are covered several times a day and some that are not. Singleton clusters that are presented as x for one time events. Those outliers can be reclustered in a different run and would cover news like “other”. The last figure 5.2 is a clustering with the *Mean Shift* algorithm. It is a partitonal algorithm working with density distributions, automatically infering the size of clusters by a density threshold.

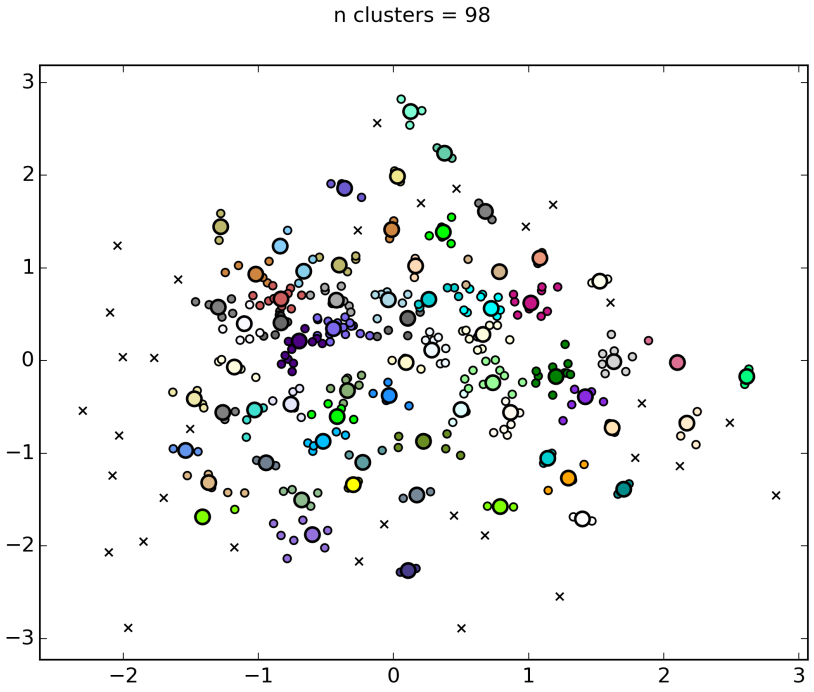

Figure 5.2: "Mean Shift clustering of a single day"

The clusters of the *Mean Shift* are even smaller. The density threshold was carefully fine tuned over several runs. All visualizations were made of data containing roughly 350 documents from the category *world* by two newspapers. Expecting at least 100 events covering different news. The final clusters visualizations are a result of several parameter estimations of the algorithms. All visualizations were done with the help of a Python API called *matplotlib* by Hunter (2007).

5.5 Multiple days clustering

Multiple days clustering was of the greatest interest while starting this thesis. It came to an understanding that it is inherently difficult to cluster and track documents over the course of several weeks. The time ran out to actually cluster multiple days in a row. Also, due to the problem of legallity, the BBC dataset came to the rescue. Multiple days clustering is an advancement on the single day clustering. See section 7.3 for future work.

6 Results and Discussion

"Simple models and a lot of data trump more elaborate models based on less data."

Peter Norvig

In this section the results of this thesis are discussed. The implementation is evaluated. What was problematic and what worked out well? What can we conclude by now?

6.1 Experiment Evaluation

Comparing all implementations from the clustering experiments from chapter 5 we can conclude, that the training set from the BBC is suited for clustering classification. We cannot conclude that it will generally work well. Classification without a ground truth is not possible. The ground truth was explicitly provided by the BBC dataset for all documents. Thusly it was easy to map documents to their respecitve labels and figure out which label had majority in a cluster. This can be mitigated by explicitly creating initial clusters for each class. In multiple days clustering then, the ground truth of classes would be the initial clusters. In that sense it is possible to use supervised algorithms as well.

It was also shown that *LSA* helps, in improving all strategies by a significant bit. It tackles problems of noise and isignificance. We can also conclude that all strategies performed reasonably well, however that pure syntactical measures perform best. That is not to say that this is the best strategy for the second clustering. In classification we need to be sure what each class is, tackling noise with *LSA* seems more reasonable. News article clustering, works with extremly spares data, so smaller and more clusters with more noise are expected. On the other hand providing a higher number of topics for *LSA* seems reasonable to diminish dimensionality inflation.

In the experimental section were some results to different clusterings by topic models and clustering algorithms. We could not draw any mathematical evaluation from the presented images and tables. It is a rather formal conclusion. On the real dataset we could see some

promising results on the clustering that were topically assigned over the course of some days. At least in part the multiple days clustering could be done. However there was not enough evidence to measure the results. This can be studied further in future work.

6.2 Data Pipeline Evaluation

The data pipeline, from chapter 3, performed reasonably well for this task. In all of the latter experiments with the BBC dataset, a data frame was used, to ease the task. This however is rudimentary and does not scale well. Using more sophisticated data persistence techniques would be required to scale the system to much larger datasets. In the following we mainly answer how the system performed in contrast to persistence and overall flexibility in enhancing it.

How is the data stored?

There is reason to believe that pure textfiles are a better way to persist data. Hadoop Distributed File System (HDFS) or more generally virtual distributed file systems, simple matrix market format files or even csv were used in the later approaches for the BBC dataset. This, in combination with a data frame library that abstracts file handling to a linear combination accessible via columns and rows greatly enhanced the workflow. This can be greatly enhanced when working with multiple server nodes when using newly founded concepts like the resilient distributed dataset (RDD) of Spark. It represents a fault-tolerant collection of elements that can be processed in parallell with the map-reduce paradigma. The representation of documents is reasonable for Elasticsearch, concrete data like noun phrases are better kept in the beforementioned form. In that way we gain performance while at the same time eliminating Redis entirely.

How to enhance the data pipeline?

Each component of the data pipeline was separately build and modularized. The facade glued the modules to each other. Dependencies to Redis or Elasticsearch are configured via configuration object. Adding new newspapers is done by creating a classes configured with the correct scraping parameters. That is why only engineers will be able to use the system. By this it is easy to adapt to new patterns. However it gives absolutely no direction. For this project, it turned out to be good to decouple as much as possible. In this way, during later stages it was easier to enhance the system with more functionality. Moreover, investing in a

text based clustering API with feature selection strategies, seems as a good next step to wrap up the system.

On developing the data pipeline

Not accounting for the technical provisioning and clustering algorithms, the problem was incredibly utopic to achieve in one thesis. The problem lies completely in the scraped data and persistence. What articles to keep? What articles to cache and how? Building interfaces to a persistence that can handle all kinds of weird text data. Even after several weeks of running the system, more and more usecases came up that were not accounted for. In the natural language community, especially pos taggers, could run for hours ending in timeouts, because single textual elements were in chinese. Appropriate interrupts and tracking those outliers was important. Different timestamp formats to correctly set them to an appropriate date for the indices of Elasticsearch. Nonsense articles, like subscription pages and HTTP errors. Each new newspaper source added additional complexity to the problem set. In the end, due to possible copyright reasons of scraped data, we cannot actually show any content or liable sources. Most newspapers have APIs to directly access the content. Most of them are paid and do not legally enable you to persist articles for a period of time. The BBC dataset added a second form of complexity that was managable.

6.3 Conclusion

At last, we can conclude that the “News-Clusty” algorithmic scheme performs reasonably well. It is a rather complex approach that takes a lot into account. The complexity is at least quadratic due to the hierarichal algorithm by BIRCH and Ward linkage in both steps. This is not a problem with smaller datasets but rises quickly with more samples. We also did not take into account that we have to do a third clustering on the event clusters. Further splitting clusters that were falsely assigned due to the shared global state to other events. We can conclude that text document clustering works well for unlabelled classification. Further that statistical methods and projections to different word ontologies have an impact on the overall quality and dimensionality of the data.

7 Outlook

"All models are wrong, but some are useful."

George E. P. Box

7.1 Summary

Summing up all the pieces, this thesis provided a starting point of building a robust news article summarization system. The groundwork on different clustering strategies were examined and several techniques of transformation from documents, to words, to coocurence matrices to statistical topical proportions were investigated. We heard a lot about feature selection and semantics, as well as different strategies on how to deal with word sense disambiguation. It was concluded that most of the strategies perform well but that the simplest of all models, namely by using the word tokens of a document, was superior to other approaches. Furthermore the data pipeline "News-Clusty" was evaluated and depicted. Several steps are necessary to get a vector of features from a real world document. Dealing with the noise and with the time drift effects by statistical techniques such as *LSA* or *LDA* showed promising results. At last, we have shown how the clustering algorithm works on the BBC dataset and experimented, without giving any evidence, how the system works on real world data.

7.2 Further Reading / Related Work

In this section it is important to distinguish between the data pipeline and the quality of the algorithmic procedures. A generally well composed article about the different current approaches to clustering large and small documents is Anastasiu et al. (2013).

Several models of topic models were examined. There is more room to study the effects of *LDA* and cerntainly trying *HDP*. For multi document summarization *LDA* can be used as a main approach, see Arora and Ravindran (2008). The concepts of *Hierarchical Dirichlet Process*

(*HDP*) can be found in Li et al. (2007); Teh et al. (2006). Further we could use Non-negative matrix factorization (NMF) as a matrix decomposition technique. See Lee and Seung (1999) for a proper introduction.

While completely using unsupersived learning we could use artifical neural networks as well. A promising approach is the shallow neural network word2vec, that learns a feature representation using skip-gram models. See Goldberg and Levy (2014) for a good introduction.

Enhancing feature selection methods and more semantic relations promising work was done by Zheng et al. (2009) using noun phrases for semantic clustering and **?** with *WordNet* and soft assignment clustering. Further we can investigate external knowledge sources like *Wikipedia* to boost feature selection. Some good approaches were found in Hu et al. (2009) and R. et al. (2010).

Time series clustering could not be tackled in this thesis. Beginning with basic introductions see Warren Liao (2005); Rani and Sikka (2012); Azzopardi and Staff (2012). A good starting point for understanding topic models over time is Lee et al. (2013) analyzing blog posts.

The data pipeline is analogous to the Columbia Newsblaster System by McKeown et al. (2002). The whole literature evolving on the system is a vatange point for further reading. Also everything around the Google News aggregation system examplified by Das et al. (2007).

At last we have to mention the scalability issues. Working with algorithms across several server nodes needs different techniques. Algorithms need to be parallelized by techniques like map reduce, see Dean and Ghemawat (2008). See parallel clustering approaches by Kim (2009).

7.3 Future Work

Future work needs to be done on several things in the data pipeline.

First scaling and enhancement for datasets. This includes adding more newspaper data. In reality the classification system for news categories needs to be normalized accross all newspapers. Finding ways of detecting the label of a document when downloading an article.

Second adding a multi-document summarization system that works with the clustering.

Third a routing system that detects the kind of event by biographical, single and multi events.

Fourth scaling the clustering algorithms to several days by strategies mentioned in the experimental multiple days section.

Fifth adding more feature selection strategies with *WordNet*, *Wikipedia* and word2vec.

Sixth using more advanced algorithms, especially topic modelling and semi supervised kernel methods.

Seventh including external knowledge sources that work during the clustering methods as well.

Eigth using state of the art distributed file systems to ease the task of feature selection and clustering

None of the above tasks is trivial. Each would greatly enhance the systems performance and usability. In theory the accuracy can be enhanced by incorporating more reliable and complex models to the clustering task. In total the system could be further prepared to run on international datasets from official conferences. A compact evaluation of the status quo against other systems was currently not possible. The status of the system is still in alpha stadium and thusly, not production ready.

Multiple days clustering

The proposed model, seen in figure 7.1, is by clustering consecutive days without any overlap. After this, all cluster centers are consecutively merged and reassigned. The merging is based on similarity thresholds. If the similarity is high enough or we find hard evidence, like a named entity, cluster centers can be merged. Otherwise the clusters are kept separate. Going further we can set hard thresholds, how old a cluster should be, before it is not being considered anymore. Deciding when a merge, a reassignment or a new cluster emerges is the goal.

Several important enhancements to single day clustering must be made.

First we need timestamps for each document

Second we need to track documents over the course of several days. As the document size grows, algorithms need to be rethought for online or batch variations. In Lee et al. (2013) is a comprehensive analysis of blogs over time using *LDA*.

Third solutions to overfitting must be found. That is, assigning too many documents to an existing cluster center.

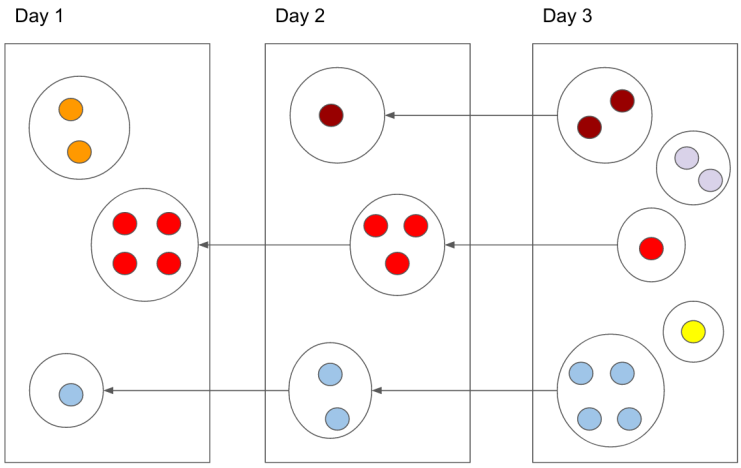

Figure 7.1: "Multiple days clustering consecutive"

Fourth biasing must be avoided by inflating clusters that are of the same topics.

Fifth measures have to be rethought. The silhouette coefficient will grow small with more documents in vector space. Different results are not comparable.

The content of this section is open for future work. Later we will discuss some articles dealing with the problem.

7.4 Final Words

We conclude this thesis with a quote of George E. P. Box, *"All models are wrong, but some are useful."*. I find this especially true for a domain, where text is seen as a system of linear combinations. In this sense, thank you very much for reading.

Bibliography

Aggarwal, C. and Zhai, C. (2012). A Survey of Text Clustering Algorithms. In Aggarwal, C. C. and Zhai, C., editors, *Mining Text Data*, chapter 4, pages 77–128. Springer US, Boston, MA.

Aggarwal, C. C. and Reddy, C. K., editors (2013). *Data Clustering: Algorithms and Applications (Chapman & Hall/CRC Data Mining and Knowledge Discovery Series)*. Chapman and Hall/CRC, 0 edition.

Anastasiu, D. C., Tagarelli, A., and Karypis, G. (2013). Document Clustering: The Next Frontier.

Arora, R. and Ravindran, B. (2008). Latent dirichlet allocation based multi-document summarization. In *Proceedings of the Second Workshop on Analytics for Noisy Unstructured Text Data*, AND '08, pages 91–97, New York, NY, USA. ACM.

Arthur, D. and Vassilvitskii, S. (2007). K-means++: The Advantages of Careful Seeding. In *Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '07, pages 1027–1035, Philadelphia, PA, USA. Society for Industrial and Applied Mathematics.

Azzopardi, J. and Staff, C. (2012). Incremental clustering of news reports. *Algorithms*, 5(4):364–378.

Bird, S., Klein, E., and Loper, E. (2009). *Natural Language Processing with Python*. O'Reilly Media, Inc., 1st edition.

Bishop, C. M. (2006). *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer-Verlag New York, Inc., Secaucus, NJ, USA.

Blei, D. M. (2012). Probabilistic topic models. *Communications of the ACM*, 55(4):77.

Blei, D. M., Ng, A. Y., and Jordan, M. I. (2003). Latent Dirichlet Allocation. *J. Mach. Learn. Res.*, 3(4-5):993–1022.

Chase, Z., Genain, N., and Karniol-Tambour, O. (2013). Learning Multi-Label Topic Classification of News Articles.

Das, A. S., Datar, M., Garg, A., and Rajaram, S. (2007). Google news personalization: Scalable online collaborative filtering. In *Proceedings of the 16th International Conference on World Wide Web*, WWW '07, pages 271–280, New York, NY, USA. ACM.

Davies, D. L. and Bouldin, D. W. (1979). A Cluster Separation Measure. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, PAMI-1(2):224–227.

Dean, J. and Ghemawat, S. (2008). Mapreduce: Simplified data processing on large clusters. *Commun. ACM*, 51(1):107–113.

Deerwester, S., Dumais, S. T., Furnas, G. W., Landauer, T. K., and Harshman, R. (1990). Indexing by latent semantic analysis. In *JOURNAL OF THE AMERICAN SOCIETY FOR INFORMATION SCIENCE*, volume 41, pages 391–407.

Dunn, J. C. (1973). A Fuzzy Relative of the ISODATA Process and Its Use in Detecting Compact Well-Separated Clusters. *Journal of Cybernetics*, 3(3):32–57.

Fellbaum, C. (1998). *WordNet: An Electronic Lexical Database*. Language, Speech and Communication. Mit Press.

Goldberg, Y. and Levy, O. (2014). word2vec Explained: Deriving Mikolov et al.'s Negative-Sampling Word-Embedding Method.

Goldberg, Y. and Orwant, J. (2013). A dataset of syntactic-ngrams over time from a very large corpus of english books. In *Second Joint Conference on Lexical and Computational Semantics (*SEM), Volume 1: Proceedings of the Main Conference and the Shared Task: Semantic Textual Similarity*, pages 241–247, Atlanta, Georgia, USA. Association for Computational Linguistics.

Greene, D. and Cunningham, P. (2006). Practical solutions to the problem of diagonal dominance in kernel document clustering. In *Proc. 23rd International Conference on Machine learning (ICML'06)*, pages 377–384. ACM Press.

Hofmann, T. (2001). Unsupervised Learning by Probabilistic Latent Semantic Analysis. *Mach. Learn.*, 42(1-2):177–196.

Hu, X., Zhang, X., Lu, C., Park, E. K., and Zhou, X. (2009). Exploiting wikipedia as external knowledge for document clustering. In IV, J. F. E., Fogelman-Soulié, F., Flach, P. A., and Zaki, M., editors, *KDD*, pages 389–396. ACM.

Hunter, J. D. (2007). Matplotlib: A 2d graphics environment. *Computing In Science & Engineering*, 9(3):90–95.

Jurafsky, D. and Martin, J. H. (2000). *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics and Speech Recognition (Prentice Hall Series in Artificial Intelligence)*. Prentice Hall, 1 edition.

Kim, W. (2009). Parallel clustering algorithms: survey.

Lee, D. D. and Seung, H. S. (1999). Learning the parts of objects by non-negative matrix factorization. *Nature*, 401(6755):788–791.

Lee, S., Lee, J., Park, C.-Y., and Lee, J.-H. (2013). Blog topic analysis using tf smoothing and lda. In *Proceedings of the 7th International Conference on Ubiquitous Information Management and Communication*, ICUIMC '13, pages 75:1–75:6, New York, NY, USA. ACM.

Li, W., Blei, D., and Mccallum, A. (2007). Nonparametric bayes pachinko allocation. In *In UAI*.

Mani, I. (2001). Summarization evaluation: An overview.

Manning, C. D., Raghavan, P., and Schütze, H. (2008a). *Introduction to Information Retrieval*. Cambridge University Press, New York, NY, USA.

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

McKeown, K. R., Barzilay, R., Evans, D., Hatzivassiloglou, V., Klavans, J. L., Nenkova, A., Sable, C., Schiffman, B., Sigelman, S., and Summarization, M. (2002). Tracking and Summarizing News on a Daily Basis with Columbia's Newsblaster.

McKeown, K. R., Hatzivassiloglou, V., Barzilay, R., Schiffman, B., Evans, D., and Teufel, S. (2001). Columbia Multi-Document Summarization: Approach and Evaluation. In *In Proceedings of the Document Understanding Conference (DUC01*.

Miller, G. A. (1995). Wordnet: A lexical database for english. *Commun. ACM*, 38(11):39–41.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., and Duchesnay, E. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

R., K. G. V., Shankar, R., and Pudi, V. (2010). Frequent itemset based hierarchical document clustering using wikipedia as external knowledge. In Setchi, R., Jordanov, I., Howlett, R. J., and Jain, L. C., editors, *KES (2)*, volume 6277 of *Lecture Notes in Computer Science*, pages 11–20. Springer.

Rand, W. M. (1971). Objective criteria for the evaluation of clustering methods. *Journal of the American Statistical Association*, 66(336):846–850.

Rani, S. and Sikka, G. (2012). Recent techniques of clustering of time series data: A survey. *International Journal of Computer Applications*, 52(15):1–9. Full text available.

Richardson, M. (2009). Principal component analysis.

Rosenberg, A. and Hirschberg, J. (2007). V-Measure: A Conditional Entropy-Based External Cluster Evaluation Measure. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 410–420.

Rousseeuw, P. (1987). Silhouettes: A Graphical Aid to the Interpretation and Validation of Cluster Analysis. *J. Comput. Appl. Math.*, 20(1):53–65.

Schiffman, B., Nenkova, A., and McKeown, K. (2002). Experiments in Multidocument Summarization.

Stefanowski, J. and Weiss, D. (2003). Carrot and Language Properties in Web Search Results Clustering. In *Web Intelligence, First International Atlantic Web Intelligence Conference, AWIC 2003, Madrid, Spain, May 5-6, 2003, Proceedings*, pages 240–249.

Steinberger, J. and Ježek, K. (2004). Using latent semantic analysis in text summarization and summary evaluation. In *In Proc. ISIM '04*, pages 93–100.

Strang, G. (2009). *Introduction to Linear Algebra, Fourth Edition*. Wellesley Cambridge Press, 4 edition.

Teh, Y. W., Jordan, M. I., Beal, M. J., and Blei, D. M. (2006). Hierarchical dirichlet processes. *Journal of the American Statistical Association*, 101(476):1566–1581.

uv rek, R. and Sojka, P. (2010). Software Framework for Topic Modelling with Large Corpora. In *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*, pages 45–50, Valletta, Malta. ELRA.
urlhttp://is.muni.cz/publication/884893/en.

Warren Liao, T. (2005). Clustering of time series data-a survey. *Pattern Recogn.*, 38(11):1857–1874.

Wei, T., Lu, Y., Chang, H., Zhou, Q., and Bao, X. (2015). A semantic approach for text clustering using WordNet and lexical chains. *Expert Systems with Applications*, 42(4):2264–2275.

Wermter, S. and Hung, C. (2002). Selforganizing classification on the reuters news corpus. In *Proceedings of the 19th International Conference on Computational Linguistics - Volume 1*, COLING '02, pages 1–7, Stroudsburg, PA, USA. Association for Computational Linguistics.

Zhang, T., Ramakrishnan, R., and Livny, M. (1996). BIRCH: An Efficient Data Clustering Method for Very Large Databases. In *Proceedings of the 1996 ACM SIGMOD International Conference on Management of Data*, SIGMOD '96, pages 103–114, New York, NY, USA. ACM.

Zheng, H. T., Kang, B. Y., and Kim, H. G. (2009). Exploiting Noun Phrases and Semantic Relationships for Text Document Clustering. *Inf. Sci.*, 179(13):2249–2262.

Zhu, M., Zhang, Y., Chen, W., Zhang, M., and Zhu, J. (2013). Fast and accurate shift-reduce constituent parsing.

Hiermit versichere ich, dass ich die vorliegende Arbeit ohne fremde Hilfe selbständig verfasst und nur die angegebenen Hilfsmittel benutzt habe.

Hamburg, 06. February 2016 Matthias Nitsche