UNIVERSITY OF TARTU
FACULTY OF SCIENCE AND TECHNOLOGY
Institute of Computer Science
Computer Science Curriculum

Robert Roosalu

Context embeddings for natural language clustering

Master's Thesis (30 ECTS)

Supervisor: Sven Laur

Tartu 2017

Context Embeddings for Natural Language Clustering

Abstract:

Semantic awareness of natural language is an important step towards general artificial intelligence. A part of which could be embedding words and documents into vector space. We selected most of the common methods for doing so and ran a vast selection of different clustering experiments on word contexts extracted from the Estonian reference corpus. After a total of 20 thousand different experiments, we found that the skip-gram word vector model combined with Spectral clustering yields the best results. The word vectors could simply be averaged, or they could be used as input to recurrent autoencoders. The latter achieved best results overall and hint towards future work of employing more complex sequence to sequence recurrent models. The newly found knowledge is implemented into our custom built application, named *PatternExaminer*, which is used in the pipeline of extracting factual data from medical records. This brings us closer to achievements such as advanced personal medicine and automated clinical trials.

Keywords:

Natural language processing, context based clustering, document embedding, recurrent autoencoders

CERCS: P170 Computer science, numerical analysis, systems, control

Konteksti teisendused loomuliku keele klasterdamiseks

Lühikokkuvõte:

Loomulikust keelest sisuline arusaam on oluline samm üldise tehisintellekti suunas. Osa sellest võib olla sõnade ja dokumentide teisendusel vektorruumi. Võtsime kasutusele põhilised meetodid selles vallas ja implementeerisime suure hulga erinevaid klasterdamise katseid eesti keele koondkorpusest eraldatud sõnade kontekstidel. Peale 20 tuhande katse analüüsimist leidsime, et *skip-gram* sõnavektorid koos spektraalklasterdusega annavad parimaid tulemusi. Seda nii sõnavektorite keskmistamisel, kui kasutades neid sisendina rekurrentsetesse autoenkooderitesse. Viimased saavutasid parimaid üldiseid tulemusi ning viitavad eelseisvale tööle keerulisemate analoogsete närvivõrkude mudelitega. Uued teadmised on lisatud töö käigus valminud rakendusse, nimega *PatternExaminer*, mis on kasutusel meditsiinilistest vabatekstidest faktide eraldamisel. Seega käesolev töö viib meid lähemale näiteks automaatsetele kliinilistele uuringutele ning uuendustele personaalmeditsiinis.

Võtmesõnad:

Loomuliku keele töötlus, kontekstipõhine klasterdamine, dokumentide teisendus vektorruumi, rekurrentsed autoenkooderid

CERCS: P170 Arvutiteadus, arvutusmeetodid, süsteemid, juhtimine (automaatjuhtimisteooria)

Table of Contents

1. Introduction

Estonian *Geenivaramu* has built a large corpora of patient epicrises. These include most of the medical information about patients: diagnosis, treatment, analysis, conclusions, descriptions, etc. A large portion of which are free form natural language. Our goal is to extract factual data from the natural text. One chosen approach is via patterns, such as regular expressions, or context free grammars. A human user defines the patterns, which are then used to extract matches from the raw data. The outputs can range in hundreds of thousands, but the user would still like to evaluate the quality of the chosen patterns. The process can be viewed on Figure 1.

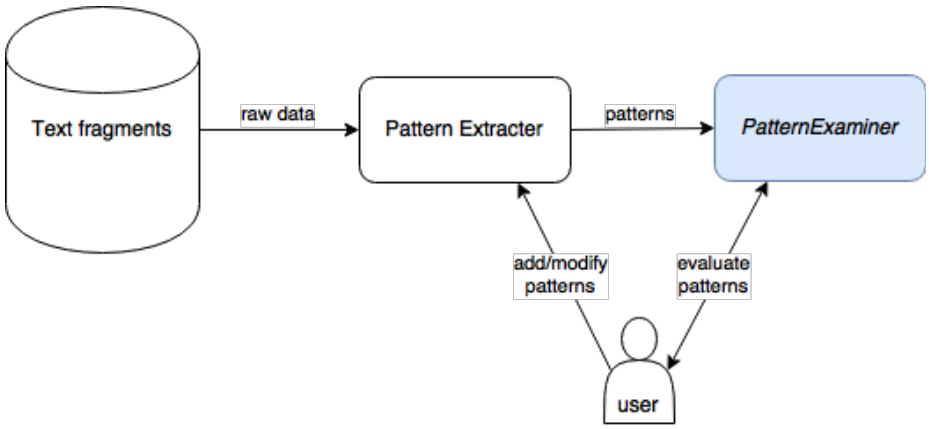


Figure 1. Workflow for extracting factual data from natural medical texts.

To solve this task, we have built a tool, named *PatternExaminer*. The tool allows to find subgroups within the extracted patterns via clustering and filtering. The user samples the clusters and *PatternExaminer* gives an estimation on the evaluation of the whole cluster and thus the whole input pattern set.

The thesis is set into following parts:

1. Description of the capabilities of *PatternExaminer*, after which we have arrived at the problem of requiring highest quality context clustering capabilities.
2. A quick survey into our formulated dataset and required methods for studying this problem.
3. Description of the experiment setup. The result is 20000 various combinations of clusterings
4. Going through all the experiments to find which methods work best.
5. Conclusion.

1.1 PatternExaminer

Custom software for examining patterns within large corpora of text has been developed. The user interface is seen on Figure 2. It features the following capabilities:

- Managing the data, preprocessing and clustering pipelines.
- Overview of experiment specifics. Real-time status reporting on errors and success.
- Clustering – various algorithms. Clusters can be recursively clustered.
- Caching distance matrices.
- Filtering results.
- Sampling results – random or homogenous.
- Evaluating the clusters via sampling and selecting OK/NOK on each line.
- Estimating Bernoulli trial probability from the manual evaluations
- **Receiving an estimation on the contents of the large input, with little effort.**

The software could be easily extended to suit different tasks. The primary aim is to work through patterns extracted using regular expressions or context free grammars from raw texts in health data.

The Estonian Gene Fund epicrisis dataset contains millions of entries. A particular pattern may yield up to multiple hundred thousand occurrences. To gain insight into the nature and correctness of the resulting body of text, we have created PatternExaminer. For this purpose, PatternExaminer clusters the contexts of the extracted medical facts. The clusters can be browsed through, filtered and sampled. Sampling may be either random, or heterogenous. We implemented heterogeneous sampling based on the distance matrix, so it would display maximum variation in the sample. The first element is chosen at random. The next elements are chosen, so they would be the farthest from all the previously chosen points.

The clusters can then be evaluated. Each element in the sample can be observed by a user and labelled either positive – element is suitable and expected to represent the current pattern, or negative – meaning an element the user did not expect to be found by the pattern. We can look at the cluster evaluation as a Bernoulli process. That is, we have a sample of n evaluations, of which m are evaluated suitable and we wish to estimate the unknown probability p of suitable elements

over the entire cluster, and it's confidence intervals. Simple methods exist to solve this task, but they are not suitable if sample sizes are small or the sample fraction is near 0 or 1. (Megill & Pavicic, 2011) solve this task for the edge cases, which we have implemented into our software.

$$E(p)=\frac{m+1}{n+2}$$

$$c_{lower}=I^{-1}_{\frac{1}{2}(1-c)}(m+1,n-m+1)$$

$$c_{upper}=I^{-1}_{\frac{1}{2}(1+c)}(m+1,n-m+1),$$

where $E(p)$ is the expected probability, c is the confidence interval and $c_{\{lower,upper\}}$ confidence interval values. I^{-1} is the inverse incomplete beta function.

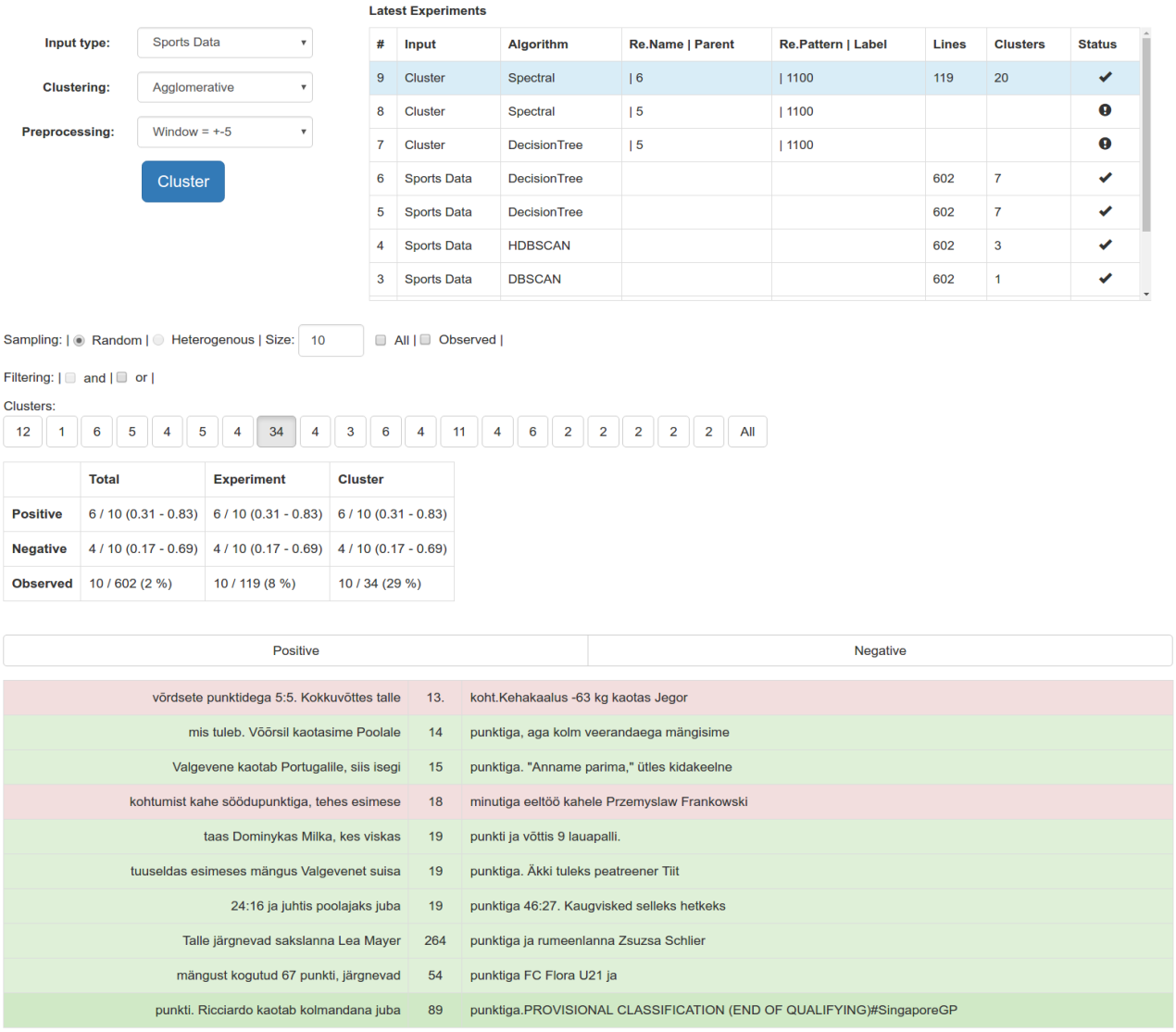


Figure 2. The user interface of PatternExaminer.

2. Methods and materials

In the coming section we will go through all the points relevant to our context clustering experiments.

2.1 Data

Context of a word is n words to the left and right. In this project we consider contexts of sizes 2, 3, 4. Also we look at symmetric and non-symmetric contexts. That is, for example, a context window of size 3, but one context side may have less than three words. The sentence datasets thus have 6 subsets, for combinations of contexts and symmetricity. Punctuation and conjunctions (sidesõnad) were discarded.

The experiments for studying clustering were conducted on 8 different word pairs. Sentences containing the words were grepped from the Estonian Reference Corpus. The corpus 12.6 million sentences from journalistic, fictional, scientific and political texts.

- Road-tea (**tee-tee**) – they present the homonymous relationship in Estonian. That is the same word meaning different things.
- Bucket-bucket (**ämber-pang**) – the synonymous relationship.
- **Countries-capitals** – Sentences which talk about ten European countries: Estonia, Germany, France, England, Finland, Spain, Latvia, Sweden, Netherlands, Switzerland, and sentences which contain their capitals.
- Going deeper with the semantic relationship of a country and it's predominant cities, we made a dataset from **Estonia, Tallinn and Tartu**.
- Apple-rock (**õun-kivi**) – This tuple could either be viewed as different small objects, or just random different objects. Depending on how big rocks are usually discussed in the Estonian Corpus and if word embeddings carry meaning about size.
- Apple-banana (**õun-banaan**) – different fruits.
- Apple-porridge (**õun-puder**) – different foods.
- Good-bad (**hea-halb**) – opposition.

Table 1 lists the count of sentences from each word extraction.

Table 1. Sentence counts for the extracted words.

Word	Count	Word	Count	Word	Count	Word	Count
countries	1076440	Tartu	147778	õun	2789	sõidutee	993
Eesti	767495	hea	147707	banaan	1174	puder	847
cities	519700	halb	24608	ämber	1160	pirn	773
Tallinn	279172	kivi	8745	joogitee	993	pang	492

All the dataset tuples were clipped to equal size classes. For example, in ämber-pang, we clipped ämber size to 492. Sentences more numerous than 5k were clipped due to computational time constraints. Countries and cities lists contained 500 sentences of each element. The tee-tee dataset was manually split after extracting "tee" from the corpus.

2.2 Document Embeddings

Term-frequency – inverese document frequency

Tf-idf is a common feature extraction method in text mining. The features are built over a corpus. Each feature corresponds to the frequency of a word, or term found in the corpus (term frequency - tf), multiplied by the inverse document frequency – idf. In practice, multiple versions of both tf and idf exist, with slight variations in the definition. In the following, the implementation found in sklearn will be described. Looking at an example:

corpus: {A cat on a mat, Cat is sat}

terms: {a, cat, sat, on, mat, is}

Term frequencies (tf vectors):

- 1st sentence: [2, 1, 1, 1, 0]
- 2nd sentence: [0, 1, 1, 0, 0, 1]

If we take just the achieved tf vectors, the method would be called Bag of Words. The inverse document frequency is constant for each term in the corpus:

$$idf = \log\left(\frac{1+n_d}{1+df(d,t)}\right)+1,$$

where n_d is count of documents and $df(d,t)$ is the count of documents containing the term. Reaching the final tf-idf formula:

$$tf-idf = tf * idf$$

Latent Semantic Indexing

Tf-idf provides little reduction in the description length and reveals little in the way of inter- or intradocument statistical structure (Blei et al., 2003). A step forward from tf-idf is Latent Semantic Indexing (LSI), sometimes also called Latent Semantic Analysis (LSA). It takes the tf-idf matrix and performs Singular Value Decomposition (SVD) on it. SVD takes as input a matrix M and factorizes it into three matrices:

$$M = U \Sigma V^T ,$$

where Σ is a diagonal matrix with the square roots of eigenvalues of MM^T sorted in descending order. Matrix U with the shape $|T| \times |T|$ where each column represents the eigenvector of MM^T that corresponds to each eigenvalue in Σ and V^T, the transpose of a square matrix with dimensions $|D| \times |D|$ with each column containing the eigenvalue of $M^T M$ corresponding to each eigenvalue in Σ. For any matrix M there exists at least one factorization via SVD. (Chen, Martin, Daimon, & Maudsley, 2013) The components are visible on Figure 3.

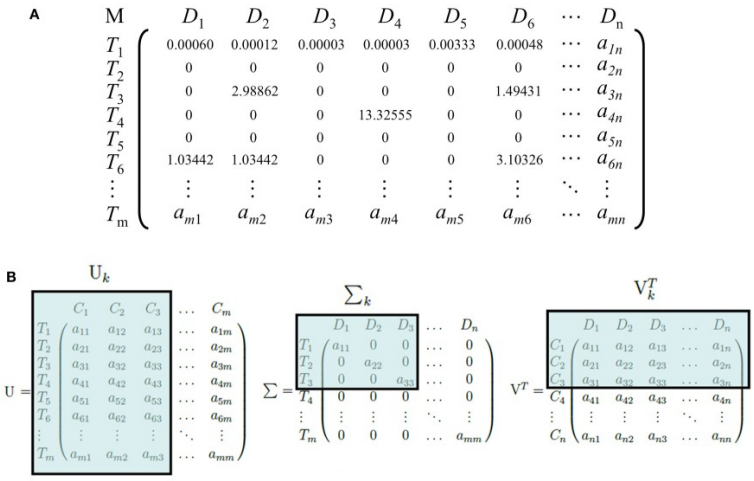


Figure 3. (A) is the originl term frequency matrix. (B) displays the SVD factorized matrices. (Chen et al., 2013)

It might be beneficial to discard singular values which carry less meaning, for a simpler feature matrix. This is called Truncated SVD. We can calculate the cosine distance on the newly acquired features, or use the features themselves.

Word Mover's Distance

Word Mover's Distance (WMD) is a distance function to compare text documents. It can be based on any word embedding, word2vec being most commonly used. Distance is measured as the minimum amount the words in one document have to move in the embedded vector space to reach the words of the other document. An example of this can be seen on Figure 4. This metric was described in (Kusner, Sun, Kolkin, & Weinberger, 2015) and demonstrated to outperform the common metrics such as tfidf, Latent Semantic Indexing, Latent Dirichlet Allocation, etc. in k-Nearest-Neighbours classification benchmarks. The WMD can be cast as an instance of the Earth Mover's Distance (EMD). EMD measures the distance between two probability distributions. It is also known as the Wasserstein metric. In layman's terms it can be thought of as the cost of turning one pile of dirt into another. EMD is a well studied problem and has many efficient solvers. An example being (Pele & Werman, 2009), which uses the successive shortest path algorithm for finding minimum cost flow on a graph. During the first iteration of our research we implemented our own versions of the WMD, a greedy and a brute force solution. The WMD metric can be found in the gensim library, which also handles word embeddings.

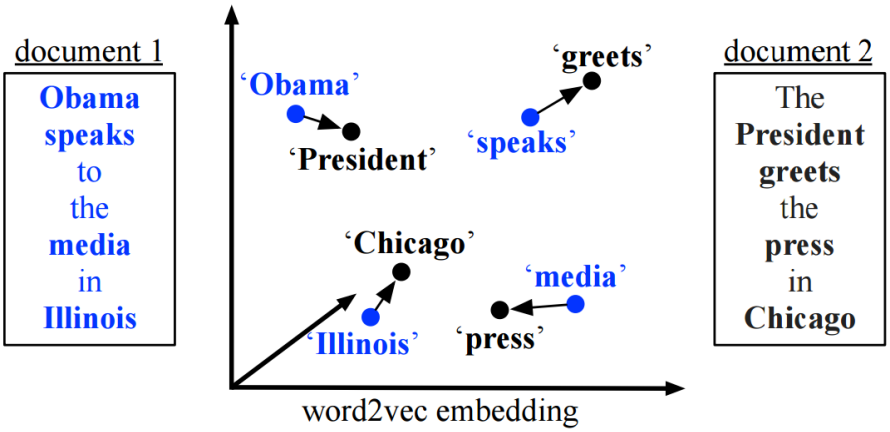


Figure 4. A visual example of the WMD using sentences about the ex-president of USA. (Kusner et al., 2015)

Although delivering state-of-the-art performance in (Kusner et al., 2015), the metric is $O(n^3 \log(n))$ in time, making it's practical usage questionable for slightly larger datasets. However, approximate solvers delivering $O(n)$ time with small errors have been developed, for example (Jang, Kim, Faloutsos, & Park, n.d.).

2.3 Neural networks

At the heart of modern artificial intelligence research, are artificial neural networks. Originating as a concept from mid 20th century, computational capabilities and dataset sizes have allowed these complex models to excel at most machine learning tasks.

Inspired from biological neural cells, neural networks consist of computational units called neurons. Neurons run a weighted sum operation, followed by a nonlinear activation function. The outputs are then fed to the next layer. Various activations are used, such as the sigmoid function, ReLU, hyperbolic tangent, etc. The aim of trianing a neural network is to find a set of neuron parameters (weights and biases), so that an (unseen) input would yield a correct output. For this, a loss function is defined. After each training step, the gradient at which is applied to all the neuron parameters using what's called the back-propagation algorithm.

Autoencoders

Constraining a neural network's output to match the amount of neurons in the input, we arrive at autoencoders. The hidden layers can then be thought of as a compressed version of the input. Or in our case, and embedding of the context. Figure 5 shows an simple, one layer deep autoencoder. It also displays the fully connected property between regular feedforward neural networks, with each neuron being connected to each neuron in the next layer. On the figure, x is input, w and b are the trained parameters, f is the activation function, s the number of samples and p the embedding dimension. (Tammeveski, Zafra, Parts, Matiisen, & Tampuu, 2016)

Recurrent neural networks

Recurrent neural networks (RNNs) are used for modeling sequenctial relations. Its input and/or output are sequentioal. The nature of which may be temporal, or just sequential. Building from regular feedforward neural networks, recurrent networks are augmented by connections, that are connected along the timesteps. At time t, nodes with recurrent edges receive input from the current data point $\boldsymbol{x}^{(t)}$ and also from hidden node values $\boldsymbol{h}^{(t-1)}$ in the network's previous state. The output at each time t is calculated given the hidden node values at time t. Input $\boldsymbol{x}^{(t-1)}$ at time $t-1$ can

influence the output at time *t* and later by way of the recurrent connections (Lipton, Berkowitz, & Elkan, 2015). The descibed process is seen on Figure 6.

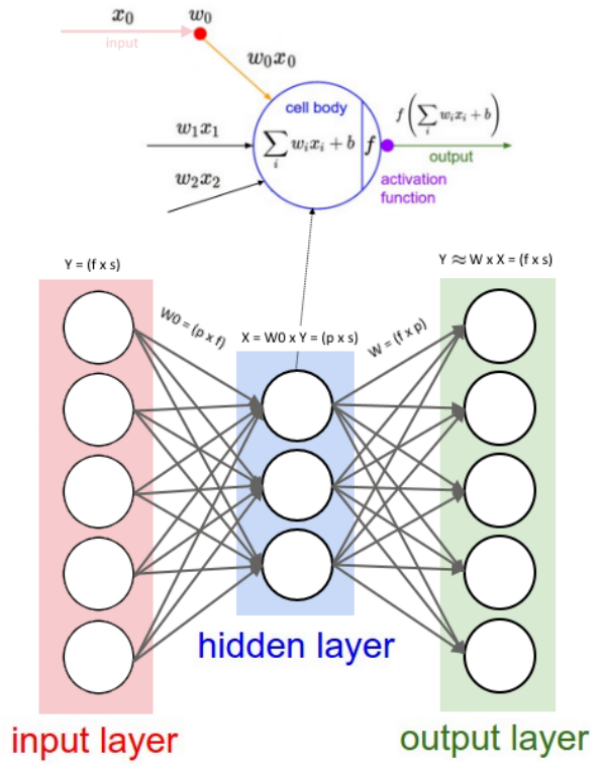


Figure 5. Autoencoder with a single hidden layer. (Tammeveski, Zafra, Parts, Matiisen, & Tampuu, 2016)

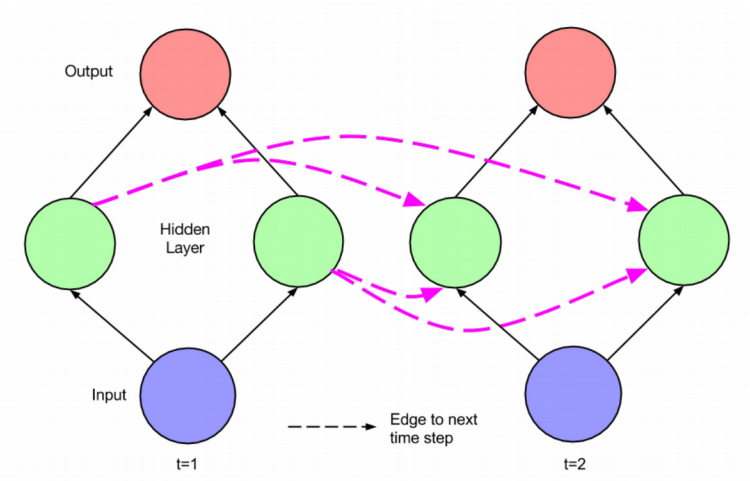


Figure 6. Simple RNN.

The simple self-connected recurrent nodes are not too capable of carrying longer term temporal data. RNNs typically employ more complex nodes in the recurrent layer, for example long short-term memory cells (LSTM). LSTM cells introduce additional parameters, gates and nodes. A good resource on them is for example (Lipton, Berkowitz, & Elkan, 2015) and the current thesis will not dwell deeper into them. In the experimental part we used LSTM based autoencoders.

2.4 Language models

A language model is a statistical model, which assigns a probability to either a sequence of words $P(w_1, w_{2,}\ldots, w_n)$ or a word following a sequence of words $P(w_n|w_1, w_2 \ldots w_{n-1})$.

Such an assignment is useful for various tasks in natural language processing, for example aiding text-to-speech and machine translation models. The quality of a language model can be assessed either extrinsically or intrinsically (Jurafsky, 2017).

- Extrinsic evaluation: use the language model as part of the end task (speech recognizer, MT, spelling correction, etc.) and assess the achieved final task accuracy. The downside is the added complexity and time requirement.
- Intrinsic evaluation: in rapid model prototyping, a quick way to calculate the goodness of a model is desired. The most common method for this is perplexity:

$$perplexity = e^{-\frac{1}{N}\sum_{i=1}^{N} \log p(w_i)}$$

Intuitively, perplexity asks, how well can we predict the next word? The best language model is one that best predicts an unseen test set (Jurafsky, 2017).

A simple yet common language model is the n-gram model, with a general form as follows:

$$P(w_{1,}\ldots, w_m) = \prod_{i=1}^{m} P(w_i, |w_{i-(n-1)}, \ldots, w_{i-1})$$

The probabilities are gained by gathering all the n-grams in the training corpus and calculating the frequencies. In practice, this yields zeroes for many possible n-grams, possibly nullifying many evaluations by the model. To counter-effect this, various distribution smoothing schemes can be utilized, so that even the never before seen n-grams would have some small frequency.

The most successful early experiment to implement a language model using a neural network was by (Bengio, Ducharme, Vincent, & Janvin, 2003). The model consisted of an input embedding layer, one hidden layer for nonlinearity and a softmax layer for the output probability, seen on figure 7

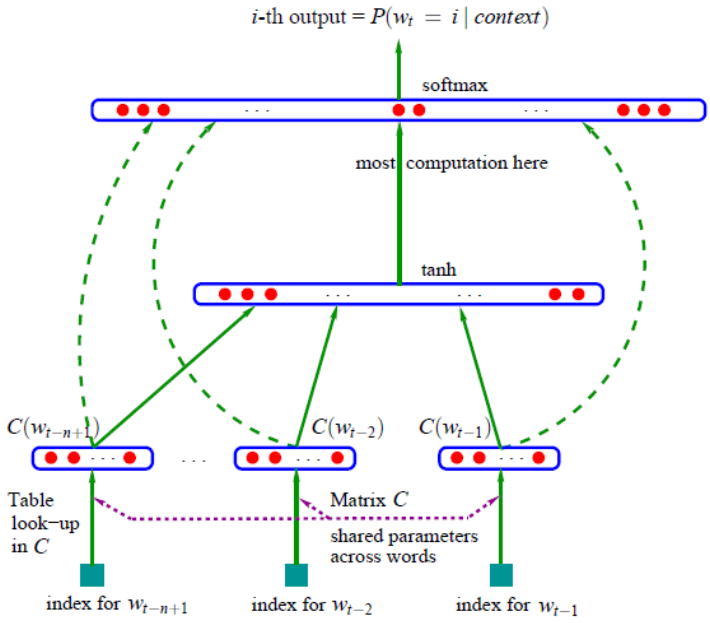


Figure 7. Architecture of the first successful neural language model. (Bengio et al., 2003).

The authors achieved a 20-35% improvement in perplexity over the then state-of-the-art, smoothed tri-gram models. The paper is also one of the first to introduce the concept of word embeddings.
Allowing for some years of development in computing power, another landmark model was developed in (Tomáš Mikolov, Karafiat, Burget, Cernocky, & Khudanpur, 2010). The previously described neural network language model (NNLM) has the limiting factor of only accounting for a fixed, predetermined context of small size (5 to 10 words). Humans have the ability to account for much larger context, but so do recurrent neural networks. Mikolov et al. thus use what is usually called a simple recurrent neural network for language modelling. The network, seen on Figure 8, has an input layer *x*, a hidden layer *s* and an output layer *y*. Input in time is the current word *w* and the concatenation of hidden layer at the previous timestep *s(t-1)*. Output is the predicted word. Input and output were embedded as a *1-of-V* vector, also called a one-hot vector. This is a vector with zeroes everywhere, but one at the index of the specific word.

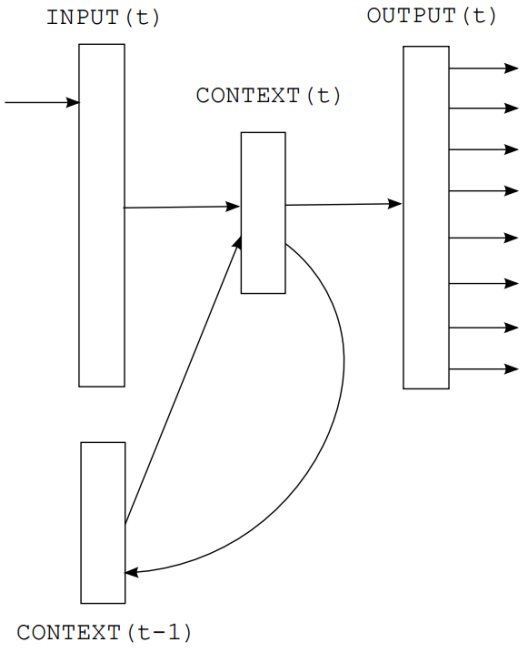


Figure 8. RNNLM as presented in (Tomáš Mikolov et al., 2010)

The authors benchmarked it for speech recognition tasks and the resulting improvement was 50% reduction in perplexity, 18% reduction in word error rate for similar training sets, and 5% reduction in error, if the then state-of-the art backoff n-gram model was trained on a much larger dataset.

The RNNLM presents itself as a basis for more complex developments in NLP, such as (Ahn, Choi, Pärnamaa, & Bengio, 2016). The group embeds a knowledge graph consisting of triplets *(subject, predicate, object)* and the description of the subject from wikipedia into a knowledge memory matrix. This matrix is then queried by the RNN during training and inference.

2.5 Word embeddings

A word embedding, or in computational linguistics known as a distributional semantics model, is a method to represent natural language words as points in a vector space. Most word embeddings build on the idea that semantics of a a word arise simply from it's context.

The most common example to demonstrate the semantic embedding capabilities of word embeddings is

$$vector(\text{"King"}) - vector(\text{"Man"}) + vector(\text{"Woman"}) \approx vector(\text{"Queen"}).$$

In the original word2vec paper, Mikolov et al also illustrate the model's ability to automatically organize with an example about countries and their capital cities (Figure 9).

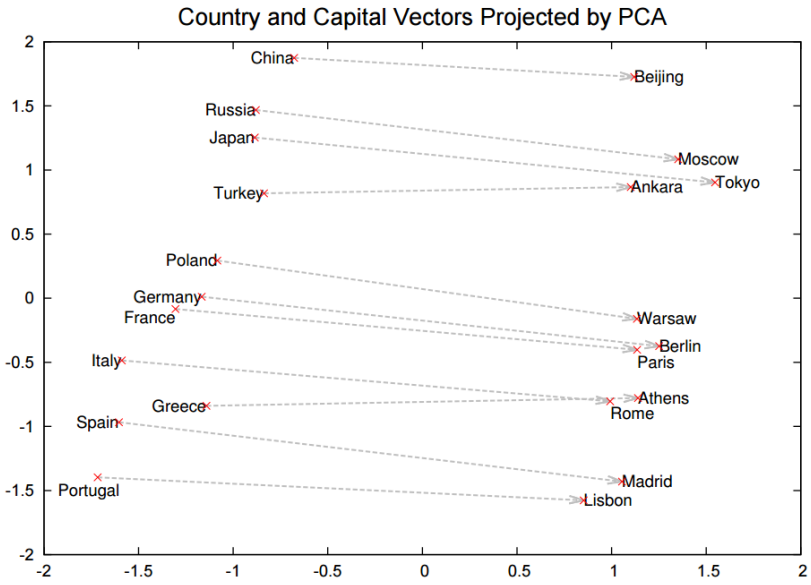


Figure 9. Two-dimensional PCA projection of the 1000-dimensional Skip-gram vectors of countries and their capital cities. (Tomas Mikolov, Chen, Corrado, & Dean, 2013)

Word2vec

The most prominent word embedding architectures are the Continuous Bag-of-Words Model (CBOW) and the Continuous Skip-gram Model (skip-gram, or SG) by (Tomas Mikolov, Corrado, Chen, & Dean, 2013). More generally known as the word2vec models (Figure 10). The models are inspired by (Bengio et al., 2003), but simplified by removing the nonlinear hidden layer and utilizing a symmetric context, instead of a one sided. The CBOW model uses n words before and after the target word w_t to predict it. It seeks to maximize the following objective function:

$$J_\theta = \frac{1}{T}\sum_{t=1}^{T} \log p\left(w_t | w_{t-n}, \cdots, w_{t-1}, w_{t+1}, \cdots, w_{t+n}\right).$$

Where T is the number of words in the training set. Skip-gram takes the opposite approach, of predicting the context from the word w_t. The objective function to maximize thus becomes:

$$J_\theta = \frac{1}{T}\sum_{t=1}^{T} \sum_{-n \le j \le n, \neq 0} \log p\left(w_{t+j} | w_t\right).$$

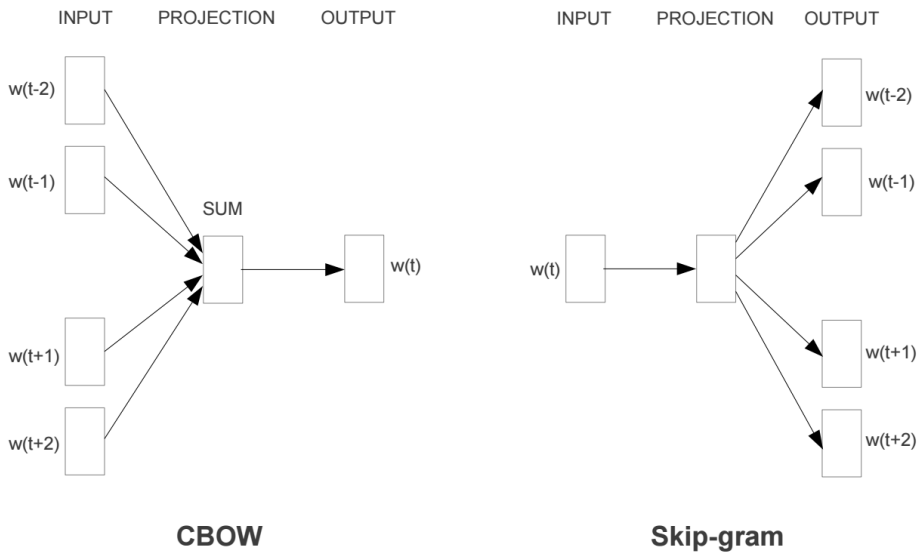


Figure 10. The word2vec model arhchitectures, as presented in (Tomas Mikolov, Corrado, et al., 2013).

GloVe

The authors of Global Vectors (GloVe) (Pennington, Socher, & Manning, 2014) argue that a weakness in word2vec models is that they look at word contexts separately, discarding any global meaning. They propose a global word to word co-occurrence matrix X. Element X_{ij} is the global count of word j being found in the context of word i. They further hypothesize how meaning of words is carried in the ratio of co-occurrences and aim to encode this with a weighted least squares regression loss function, arriving at a loss function:

$$J = \sum_{i,j=1}^{V} f(X_{ij})(w_i^T \tilde{w}_j + b_i + \tilde{b}_j - \log X_{ij})^2 ,$$

where w are the word vectors and b are the bias terms. f is a weighting functions which helps prevent overweighting rare and frequent co-occurrences. The original ratio of co-occurrences gets cancelled out in the derivation of the function.

Although the authors of GloVe claim to beat word2vec models by up to 11% on various benchmarks, several people, including the research duo Levy & Goldberg claim the benchmark experiments were not carried out on equal terms and that word2vec is still the state-of-the-art.

2.6 Distance and similarity

A metric or distance funcion yields a smaller value for more similar elements. By definition the following properties must hold to be a metric:

1. Non-negativity or separation axiom $d(x,y) \geq 0$
2. Identity of indiscernibles $d(x,y) = 0 \Leftrightarrow x = y$
3. Symmetry $d(x,y) = d(y,x)$
4. Triangle inequality $d(x,z) \leq d(x,y) + d(y,z)$

A similarity function yields a larger value for more similar elements. In machine learning, kernel functions are often used as a similarity measure. If x and y are feature vectors, some examples of kernel functions:

- $k_{linear}(x,y) = x^T y$
- $k_{poly}(x,y) = (\gamma x^T y + c_0)^d$
- $k_{sigmoid}(x,y) = \tanh(\gamma x^T y + c_0)$
- $k_{RBF}(x,y) = \exp(-\gamma \|x-y\|^2)$,

where $\|x-y\|^2$ is the squared euclidean distance. It is possible to convert between a similarity and distance function. Various formulas for this exist. The choice depends on the convertible function value range and the specific usecase. With s being similarity, d the distance function and γ being a tunable parameter, some examples include:

$$s = 1 - d \qquad (1)$$
$$s = -d$$
$$s = e^{(-\gamma d)}$$
$$s = \frac{1}{1+d}$$

When converting from a similarity to distance, some of the properties for a true distance metric might break. For example, cosine distance is achieved from cosine similarity following eq (1), but this new metric does not follow the triangle inequality. But it can still be a very useful metric. The tf-idf based cosine distance is among the first methods to try in text mining. Cosine similarity and distance become unintuitive, if the feature space isn't all positive. This is often the case for word embeddings. We can then look at the angular distance and similarity, which is the normalized angle between the points.

$$angular\ distance = \frac{\cos^{-1}(cosine\ similarity)}{\pi}$$

This new distance function can again be turned into angular similarity using eq (1).

In our experiments we used euclidean and angular distance.

2.7 Dimensionality Reduction

The data under study is often very high dimensional. This makes simple observation impossible to the human eye. To aid the aforementioned shortcoming, we can map the high dimensional data to a lower dimensional space. The benefits to these methods are twofold. First, to help visualize high-dimensional data. Second, patterns of interest may lie on some manifold within the data, which can be concentrated onto less dimensions, to aid in learning.

Principle Component Analysis

Dimensionality reduction can be linear or non-linear. The most common linear methods include principle component analysis (PCA) and linear discriminant analysis (LDA). PCA is an unsupervised method, which seeks to map a new orthogonal basis to the data, so that each of the newly formed components display successively highest possible variance. Components, or dimensions with the lowest variance can be discarded, while maintaining the actual important variability in the data. LDA is a supervised method, which tries to find the dimensions along which variance between classes is highest. We can again discard the less informative dimensions. LDA can also be used as a classifier on it's on.

Multidimensional Scaling

Multidimensional Scaling (MDS) is a technique to transform a dissimilarity matrix to points in n-dimensional space, where *n* is chosen by the user, and usually small. The method can be used for dimensionality reduction if a dissimilarity matrix is calculated from the points in high-dimensional space. Essentially, it is a mapping from a dissimilarity matrix, to points in the new space, with the objective of keeping the new distances as close to the original distances, as possible. The difference between the two distances is called stress:

$$\sigma^2 = \sum_{i=2}^{n} \sum_{j=1}^{i-1} w_{ij}(\delta_{ij} - d_{ij})^2,$$

where δ_{ij} is the dissimilarity, d_{ij} is the euclidean distance in the new space, w_{ij} is a user-defined weight, i and j are indexes of datapoints (Groenen & Van De Velden, 2004). The previous objecetive function is usually solved using the SMACOF algorithm.

2.8 Clustering algorithms

The following section gives an overview into the three used clustering algorithms. They we chosen because of accepting a precomputed distance metric and their relatively different nature.

Agglomerative

Agglomerative clustering is a bottom-up hierarchical clustering. Bottom-up, as in all the points start in their own cluster and clusters are being iteratively merged. The merging is done based on a precomputed distance metric. The most common here being euclidean distance. For clusters of a single element, the inter-cluster distance is simply the element's distance. As the number of elements in a cluster grows, multiple options (Figure 11) arise for calculating the distance (Sch, 2008).

- Single link – choose the distance between points in the clusters that are the closest.
- Complete link – choose the distance between points that are the farthest from each other.
- Average link – average distance between inter-cluster points.
- Ward'd link – Choose the cluster pairing that least increases total intra-cluster variance.

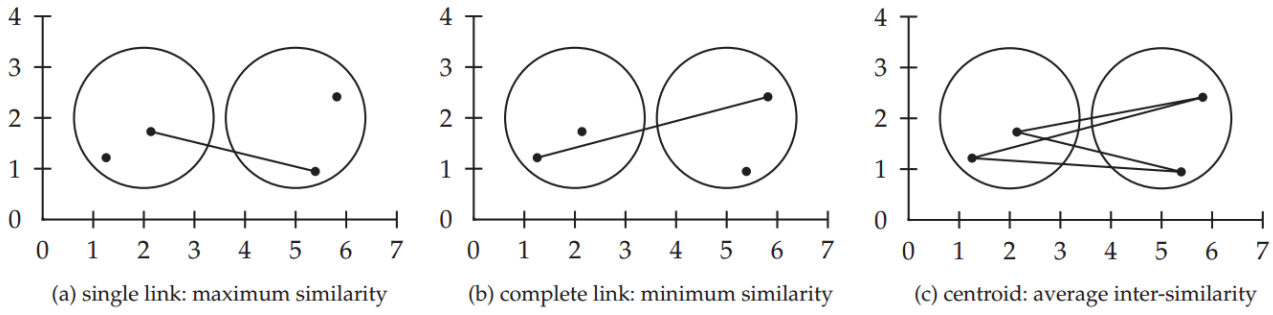


Figure 11. Different types of deciding similarity between clusters in agglomerative clustering. (Sch, 2008).

After forming the hierarchical relationship between datapoints, the final clusters can be achieved by cutting the hierarchic ties at some depth. The user may tweak the amount of clusters in such a manner.

Specral

Spectral clustering comes in multiple, slightly varied flavours. In the following it is described as defined in (Ng, Jordan, & Weiss, 2001). The algorithm begins by building an affinity matrix A between the datapoints. Let D be a diagonal matrix whose i-th diagonal element is the sum of A's i-th row and construct

$$L = D^{-1/2} A D^{-1/2}.$$

Eigenvectors are calculated from L. The eigenvectors are stacked by columns, normalized and the rows are clustered themselves, most commonly via k-means. This matrix contains as many rows as there are datapoints, and the clustering of these rows corresponds to the clustering of the original data.

The common suggestion is to use the Gaussian kernel, as it often best models local neighbourhoods. However, most of out experiments were ran using angular similarity, owing to it's usefulness in comparing word and document embeddings.

HDBSCAN

Extending from the density-based clustering paradigm, aiming to solve limitiations set by the DBSCAN algorithm, (Campello, Moulavi, & Sander, 2013) introduce Density-Based Clustering Based on Hierarchical Density Estimates (HDBSCAN). They build a clustering hierarchy on top of DBSCAN, implement a new cluster stability measure and help maximize the stability of the clusters.

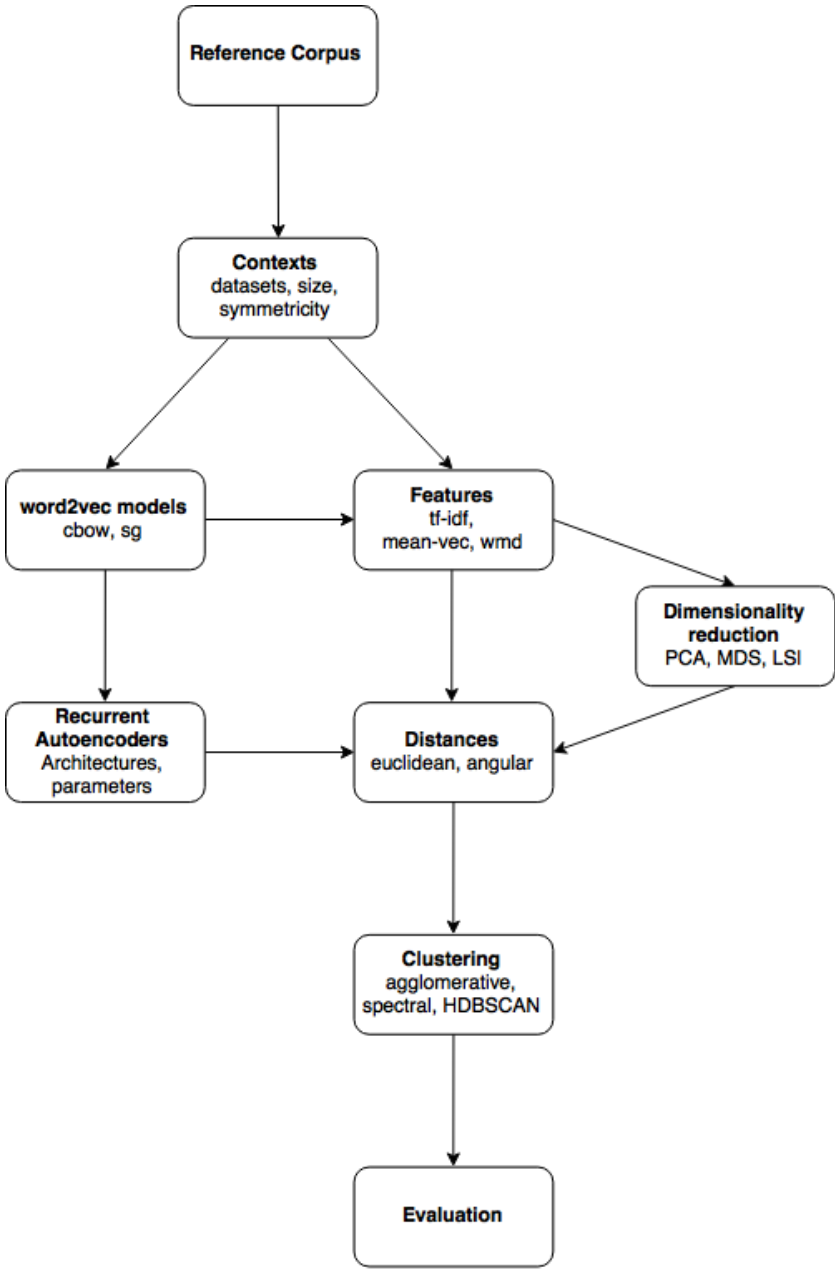
3. Experiments

The experiments in the current thesis follow a sort of breadth-first, grid search style. Previous work in seminars and on *PatternExaminer* had shown, that clusterability of contexts is somewhat obtainable. Thus the aim now was to go through as many as possible of the commonly used methods, on multiple datasets. Most of the classical methods were evaluated. Initial steps with the current state-of-the-art, recurrent autoencoders, were also taken.

3.1 Experiment Pipeline

Schematic representation of the pipeline followed by our experiments. The pipeline does not contain all the various parameters that were tweaked. They will be discussed in the next section.

Figure 12. Experiment pipeline.



3.2 Experiment parameters

In addition to what can be seen on the experiment pipeline figure 1, let's specify which parameters were searched through:

- **Contexts** – window sizes 2, 3 and 4, symmetric and non-symmetric. Each dataset thus had 6 variations.
- **word2vec models** – we used the lemmatized, 100-dimensional models found from (Tkachenko, 2016).
- **Features** – No specific parameters were tuned here. The implementation of WMD is the one found in gensim (Řehůřek & Sojka, 2010)
- **Dimensionality reduction** – for all the algorithms: PCA, MDS, LSI, we tried various dimensions: 2, 10, 20, 40
- **Distances** – Euclidean and angular distance were always normalized to range [0,1]. Converting to similarity was done by $similarity = 1 - distance$. RBF kernel was also tried, as it is most commonly suggested for Spectral clustering. However, it led to problems with Spectral clustering convergence. Possibly, the RBF parameter needs tweaking.
- **Clustering** – for algorithms which enable setting the cluster count manually, we used values 2, 10, 40. HDBSCAN does not accomodate this, but it has parameters *min_samples* and *min_size*. 6 combinations of the latter two parameters were found, which achieved clusters of various sizes in the range [2; 100].
- **Reccurent autoencoders** – 4 different acrhitectures were used. Using the notation [count of layers in encoder; count of layers in decoder], they were as follows: [1,1], [1,2], [2,1], [2,2]. The layer sizes were also searched through, using combinations of values from {16, 32, 64, 128}.

3.3 Evaluation

Standard methods of evaluating clustering quality are not suitable for us. We are interested in increasing the quality of *PatternExaminer*, which is a task with a slighlty lower bound, than strict clustering. We therefore continue by defining our own evaluation metrics, most notably accuracy, which will be examined throughout the results.

Random baseline

Since our clustering usage metrics might be slightly unintuitive, we will also generate them for a random clustering. The simplest way is to assign random cluster elements uniformly *P(x in cluster) = 1/k*, where *k* is amount of clusters. However, our experiment clustering algorithms do not form uniform clusters. Thus, a random assignment would be unjustified.

A discrete distribution over all the experiments was formed. Figure 13 displays the cluster size distributions of agglomerative clustering.

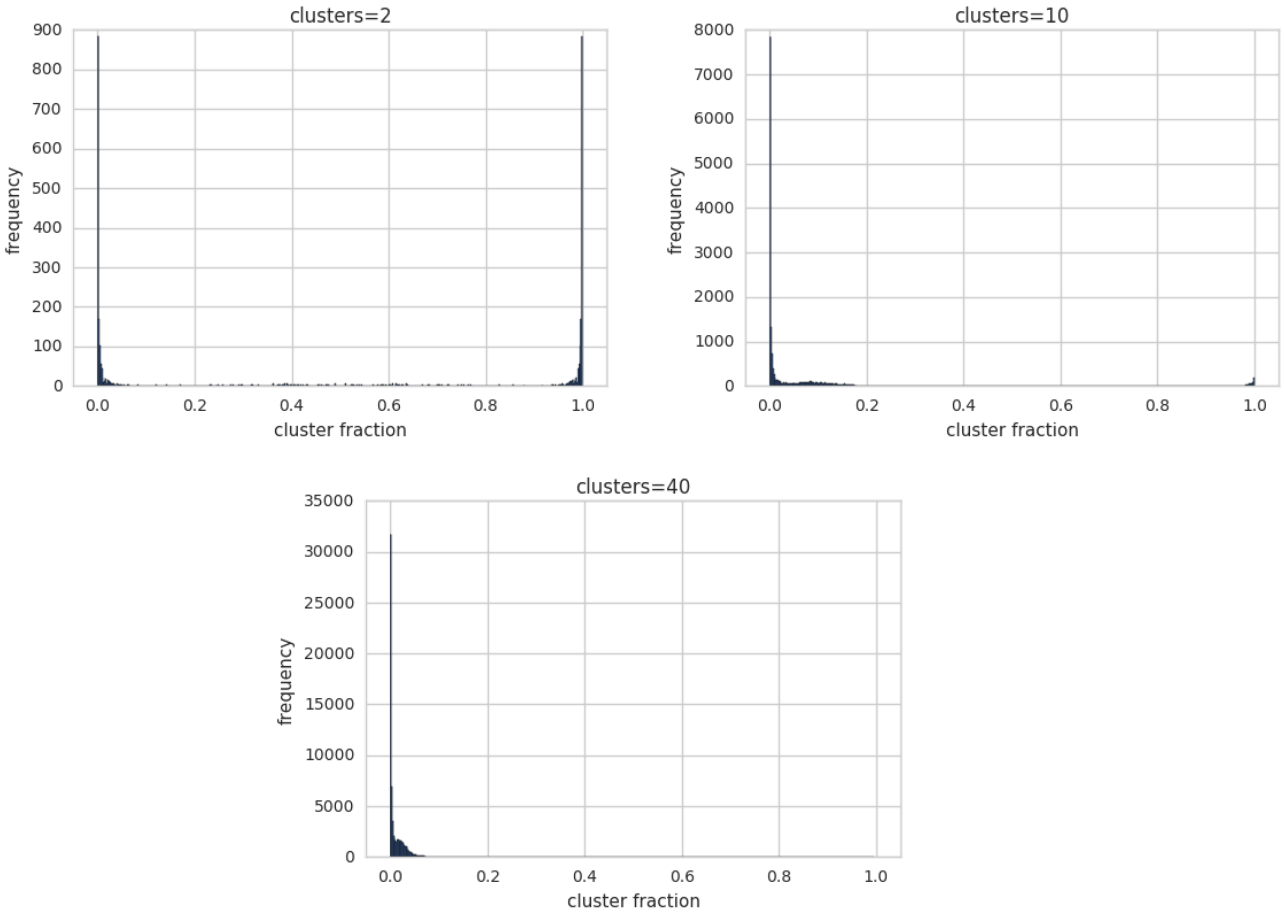


Figure 13. Agglomerative clustering experiments cluster size distributions.

We see how agglomerative clusters are extremely unevenly distributed. Overwhelming majority of the experiments form a single, almost whole dataset containing cluster, and the most other clusters containing just a few elements. We may conclude agglomerative is not well suited to our textual data.

Observing the results of HDBSCAN on Figure 14, a similar pattern emerges.

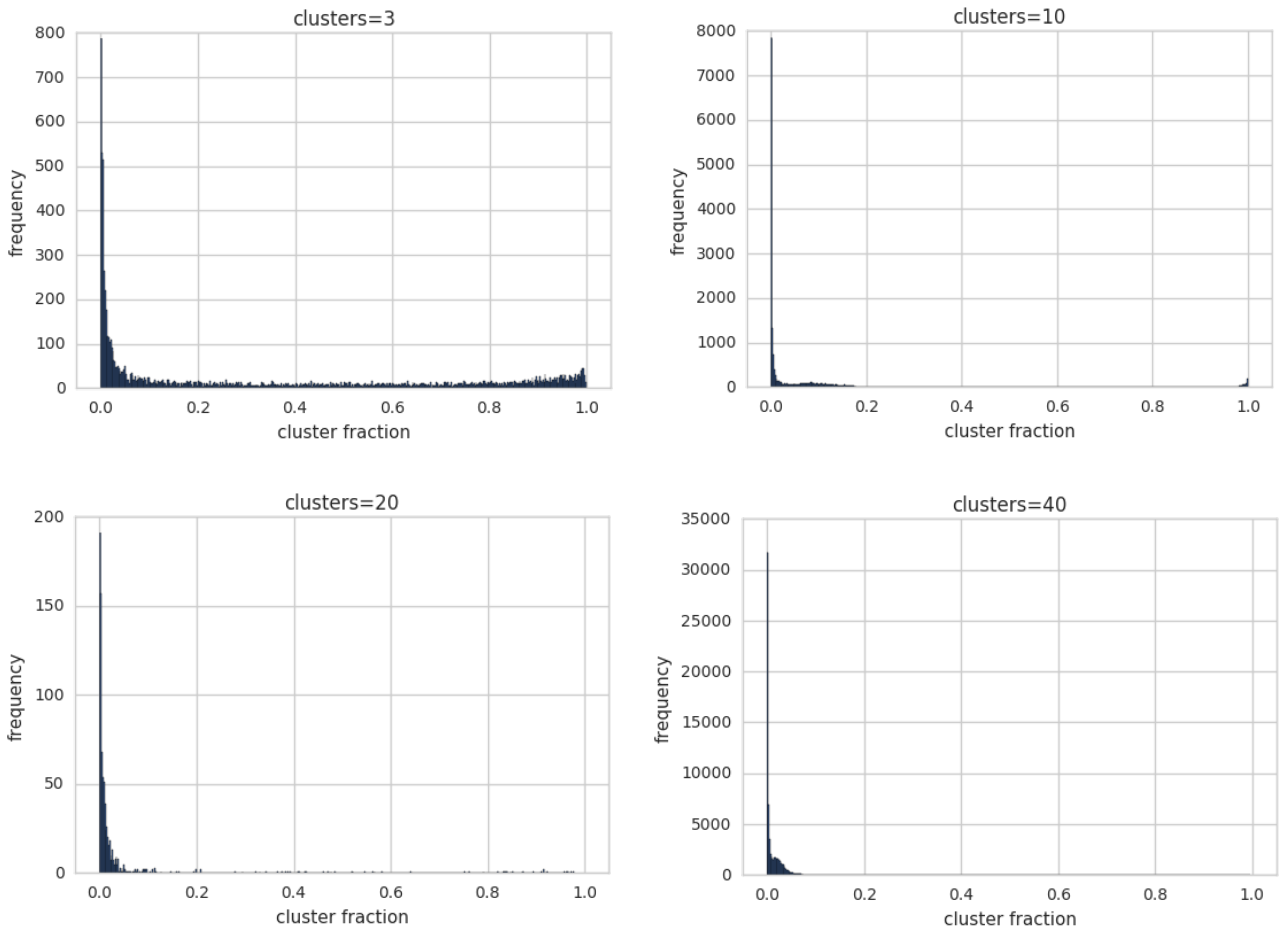


Figure 14. HDBSCAN clustering experiments cluster size distributions.

Large cluster counts are again just chipping away few datapoints from the whole input. Smaller cluster count is better than the small count for Agglomerative, with all fractions being somewhat represented. However it is still too skewed towards small fractions.

Examining the cluster size distribution for Spectral clustering, we observe a totally different pattern.

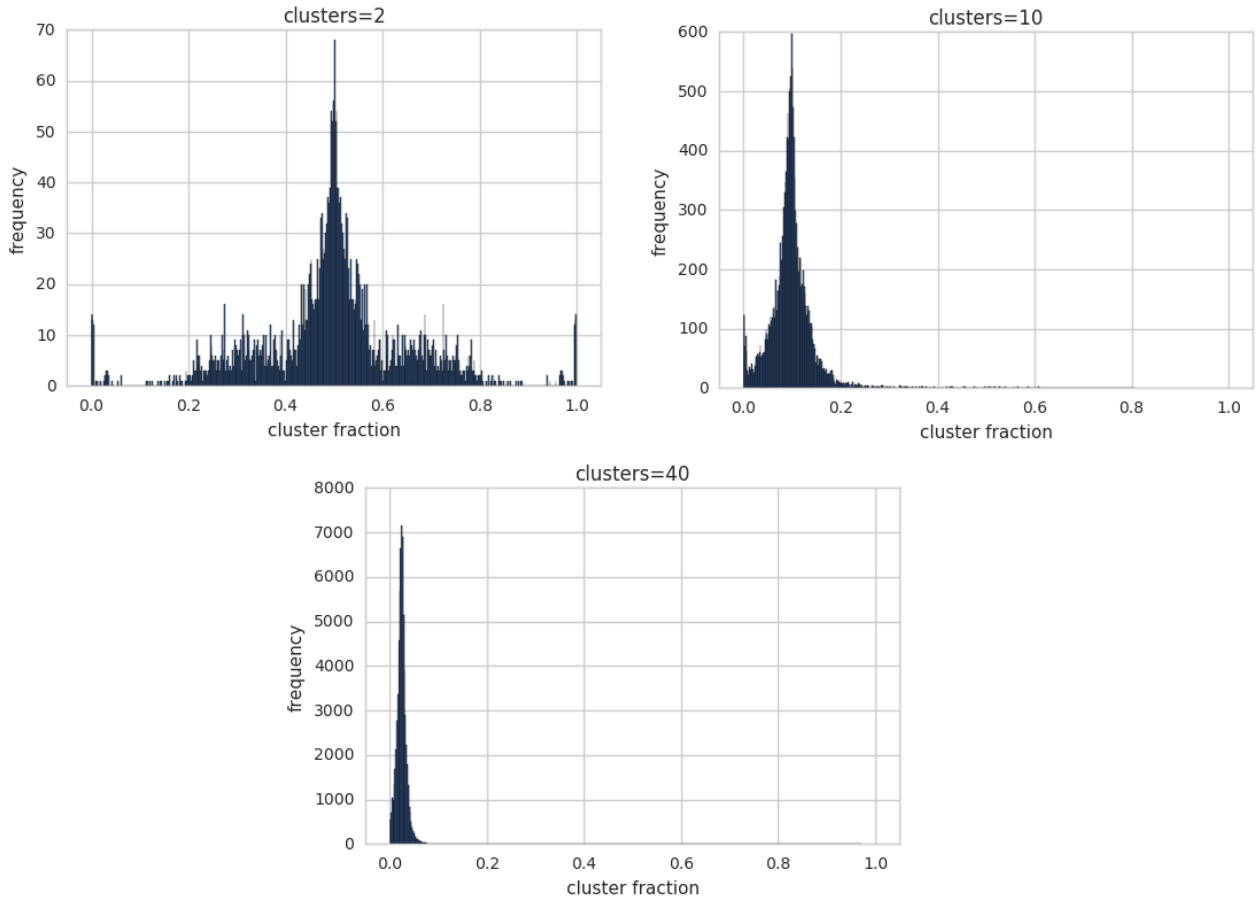


Figure 15. Spectral clustering experiments cluster size distributions.

These plots are nearly perfect in the sense that they display the Spectral clustering's ability to form clusters of equal sizes. This is a desirable property in the context of *PatternExaminer*. Especially, if the clustering are of good quality, as this helps the final evaluation be of most even quality. The plots also flaunt an interesting symmetry prevailing from spectral theory.

Having arrived at the corresponding cluster distributions, we can start sampling from them to arrive at a random clustering. Following is the pseuducode for the corresponding process.

```
for algoritm in algoritms:
    for n in cluster_counts [2,10,40]:
        cluster_sizes ~ cluster_sizes_distribution
        random_labels = labels proportional to from cluster_sizes
        true_labels = half and half split between OK/NOK
        perform_clustering_evaluation
```

The result of the previously described sampling is shown on Figure 16.

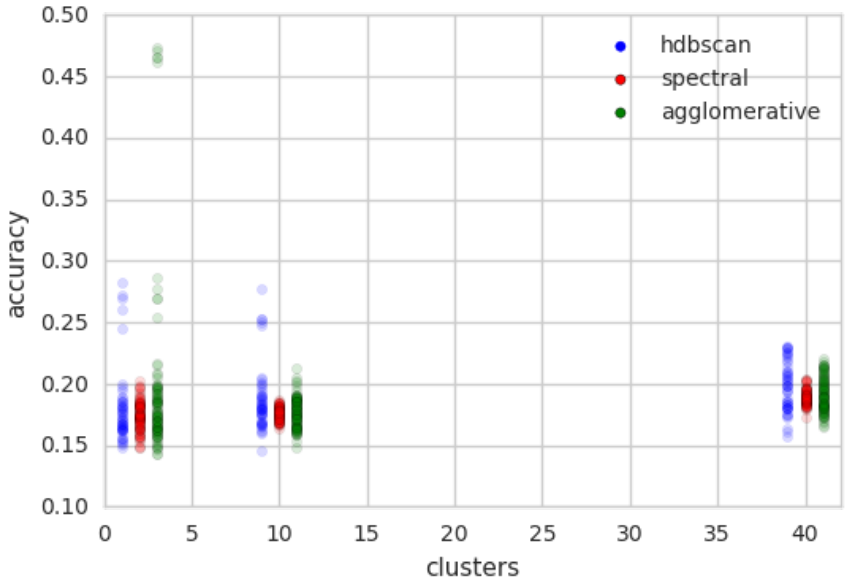


Figure 16. Random clustering accuracy evaluated by our usage simulation metric.

Each algorithm and cluster count tuple were evaluated 100 times. The random clustering does not portray too much variance. With the exception of hdbscan and spectral in the lower cluster count range, all the results are around 0.15 – 0.23 range.

Custom metrics

To evaluate the quality of our clusterings, a custom set of metrics was defined, in line with the usage of *PatternExaminer*.

Let us recall the usage of PatternExaminer. The aim of which is to assign a binary label, OK or NOK to the whole input set. The input is broken into clusters, and we wish to assign OK/NOK to each cluster. For this we iterate over all clusters and on each

1. Sample N elements from a cluster
2. Label them manually:
    1. OK > threshold * N → whole cluster is OK
    2. OK < (1-threshold) * N → whole cluster is NOK
    3. Values in between the two previous are assigned *mixed cluster*

Which leads us to an assigned label for each element in the original input set. As the experiments in the current thesis were conducted on a test set, for which we know the true labels, we arrive at a slightly modified confusion matrix.

Table 2. Custom confusion matrix to evaluate the efficiency of the chosen clustering methods.

True \ Predicted	OK	Mixed	NOK
OK	True Positive (TP)	True Mixed (TM)	False Negative (FN)
NOK	False Positive (FP)	False Mixed (FM)	True Negative (TN)

From the confusion matrix we can derive the metrics, which are slightly modified versions of the classic accuracy, precision, recall. Although *precision* carries the exactly same form, it is originating from the custom confusion matrix, and will be discussed about as a novel metric. The metrics describe clustering efficiency, if evaluated through PatternExaminer.

$$accuracy = \frac{TP + TN}{total\ count}$$

$$recall = \frac{TP}{TP + TM + FN}$$

$$precision = \frac{TP}{TP + FP}$$

$$mixed\ fraction = \frac{TM + FM}{total\ count}$$

Accounting for the randomness in step 1 of the above described algorithm, we run this evaluation process multiple times.

Although calculating all the metrics, we resort to reporting only the accuracy. The other metrics may be used upon fine tuning and further inspection of the next models.

It is also important to note, that when a human expert uses *PatternExaminer* the cycle is completed just once. We sample it multiple times, to get a more general overview of the performance.

4. Results and discussion

The experiments were organized into two batches. The second batch contains the autoencoder experiments, while the first is the remaining full experiment pipeline, discussed in section 4.

The following section goes through the experiment results and the emerging discussion.

4.1 First batch experiments

We begin by assessing the quality of a single combination of preprocessing methods and clustering algorithm using the method described in section 3.2. We plot it against the number of clusters formed. A result is displayed on figure 1. The specific example is of tee-tee, with symmetric window of size 4, Spectral clustering, with the mean of cbow word embeddings.

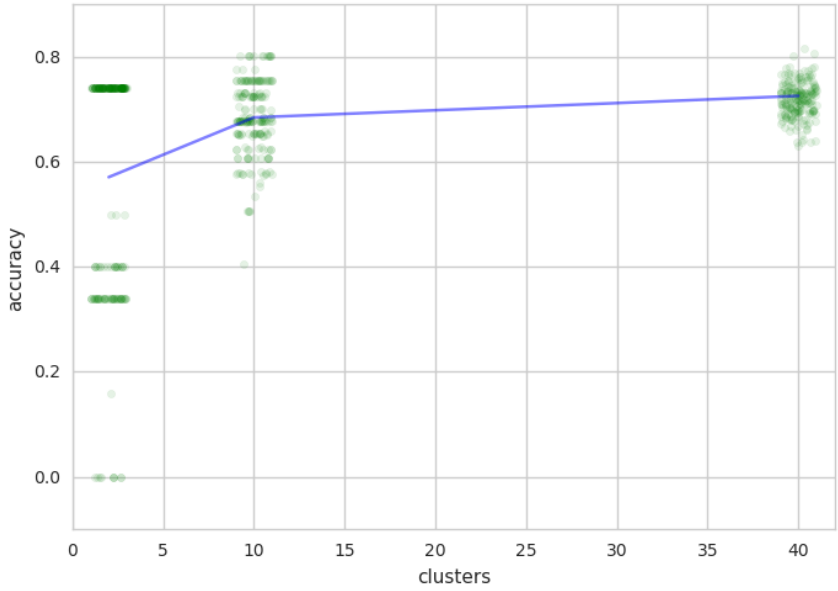


Figure 17. The plot corresponding to a specific preprocessing and clustering method. Accuracy is our custom clustering accuracy.

The count of clusters is important, because it translates into the amount of work required to be done using *PatternExaminer*. We may define a desired accuracy and budget for the whole process. When

they are achieved from our clustering metrics, the actual usage of *PatternExaminer* may commence. Our experiments are clipped at 40 clusters, for it is the ceiling of viability of the manual evaluation.

we continue to plot the results of all the experiments, faceted by the clustering algorithms. This leads us to the most global look on all the experiments. The lines are plotted with high opacity, to be able to see some patterns. Also, the random clustering comparison points are also added. This reveals, how the bulk of both Agglomerative and HDBSCAN are as good as random clustering. Spectral, however is considerably above random clustering.

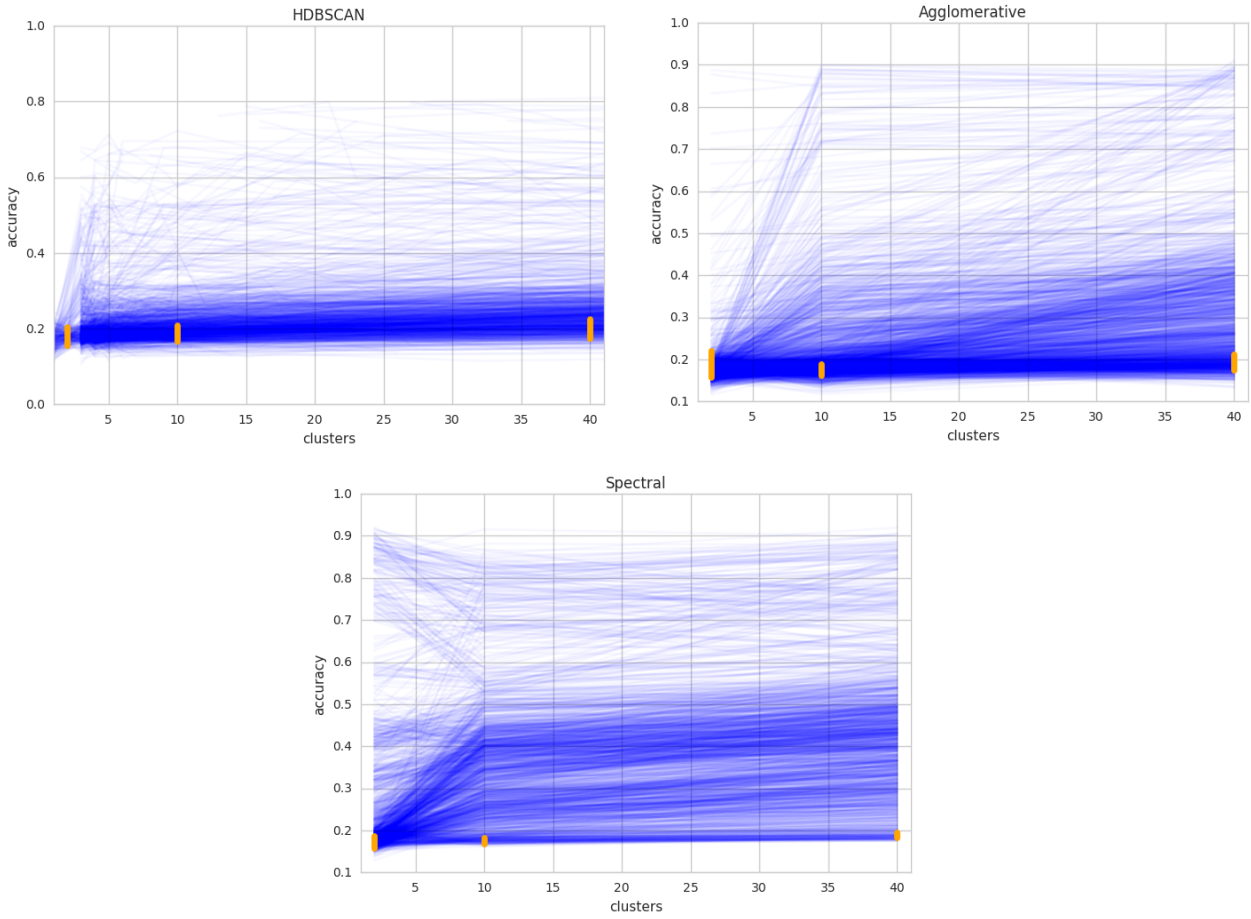


Figure 18. All the first batch experiments. Line point correspond to a specific clustering experiment. The orange bars contain upper and lower deciles of random clustering baseline.

It is evident, that plotting the experiments using lines is not too informative, due to the high amount of experiments and multitude of changing parameters. Let us explore the experiment using categorical violin plots.

On the Figure 18, each experiment is a point on the line. Switching to violin plots, each point becomes part of a distribution, for which the corresponding kernel density estimations are displayed, together with the quartile ranges. The middle, heavy dashed line is the median, dotted lines above and below are the 0.25 and 0.75 quartiles. For comparison, the upper decile of random clustering is added, visible as a red line. Figure 19.

Figure 19. Global overview of alll the experiments excluding autoencoders. Red line is the random clustering baseline for each clustering algorithm.

The large and multifaceted plot proves to be rather informative. Some dominant observations:

- Agglomerative and HDBSCAN are not good for discovering clusters in textual embeddings. Both algorithms are either on par with random or just slightly above it, for most datasets.
- Spectral however seems well suites, as it finds above-random clusterings in all datasets.
- Tf-idf does not lend itself to clustering small contexts. This understandable, as on such small contexts the dataset is extremely sparse.
- Mean-vector tends to outperform WMD. This is probably again owed to the small size of our input datapoints. Intuitievly, the less vectors for calculating mean, the more of semantic meaning in each of the vectors gets carried on to the result. Inversely for WMD, we may hypothesize that it needs more words to compare to see over-all patterns in a dataset.
- Tee-tee is an incredibly separable dataset, achieving notable separation even for when all the other datasets are comparable to random cclustering. Indeed, this word pair is semantically the farthest from the eight in our dataset.

The initial comprehensive plot does not answer all questions. Lets fixate the best overall performance: Spectral clustering with 40 clusters and observe some deeper relations.

Context size and symmetricity

Aggregating our remaining experiments by the window size and symmetricity we find, unsurprisingly, that symmetricity seems to have a little effect towards symmetric being better. However, surprisingly, window size doesn’t seem to matter. This could be due to our models simplicity being unable to grasp longer temporal relations. Seen on Figure 20.

Profiling window size against choice of model, features, metric, dimensionality reduction methods and all the datasets we see that all parameters except for datasets have no visible effect on accuracy. For the datasets, some display no difference, some an increase and some a decrease. Please refer to appendix A for the figures. In conclusion, for tf-idf, mean-vector and WMD, a context larger than 2 has no effect on clustering success.

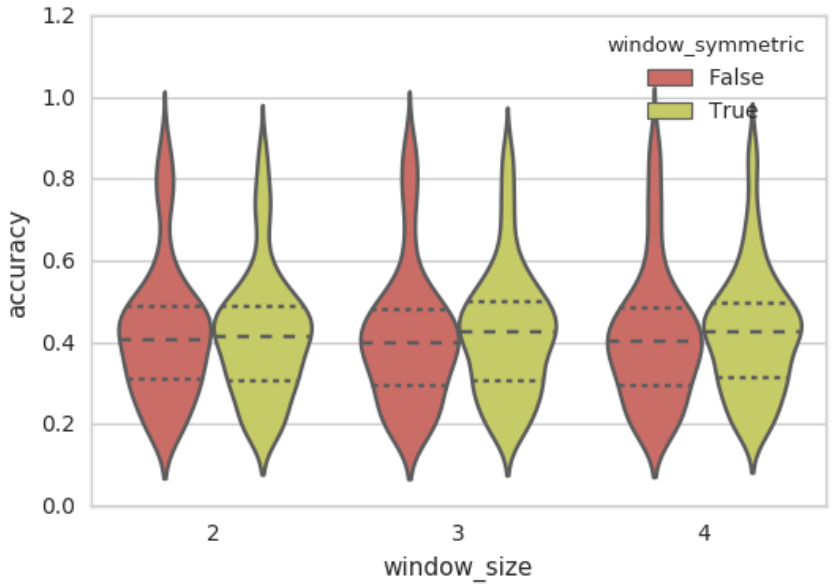


Figure 20. Spectral, 40 clusters. Relationship of the context properties and final accuracy.

Angular and euclidean distance

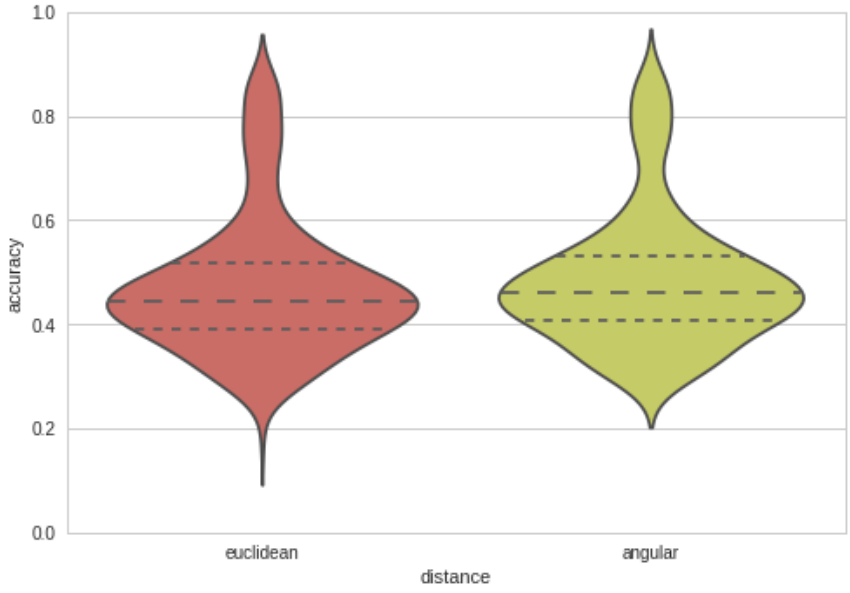


Figure 21. Spectral, 40 clusters. Difference between our two different distance metrics.

Although not by a huge margin, angular is better, as siin on Figure 21. The lower end is higher and the mean and quartiles are as well. This is coherent with the recommendations in literature, to use angular metrics when assigning affinities to textual data.

CBOW and skip-gram

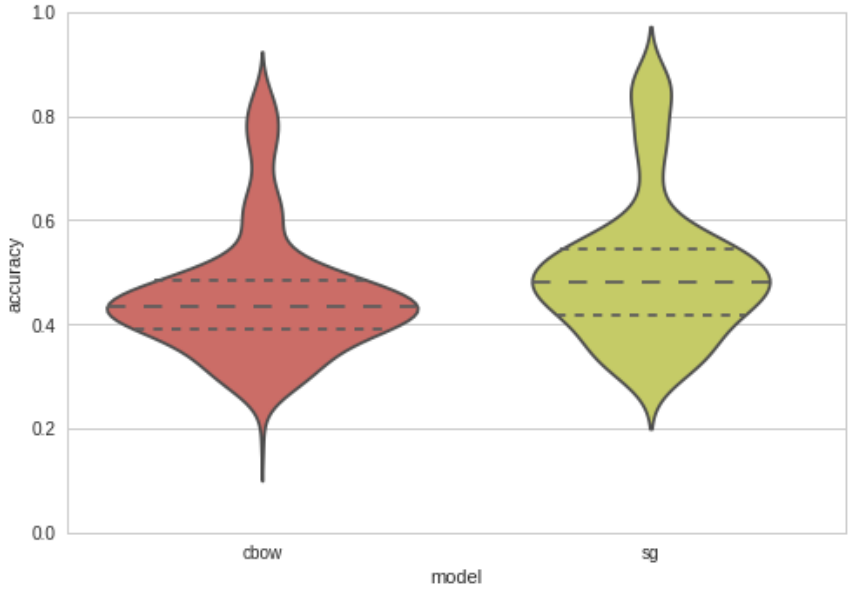


Figure 22. Spectral, 40 clusters. Comparison of word2vec models.

From Figure 22 it is evident, that the skip-gram model is better. Skip-gram aims at predicting the context, thus it is more suitable for context based feature engineering.

Dimensionality reduction

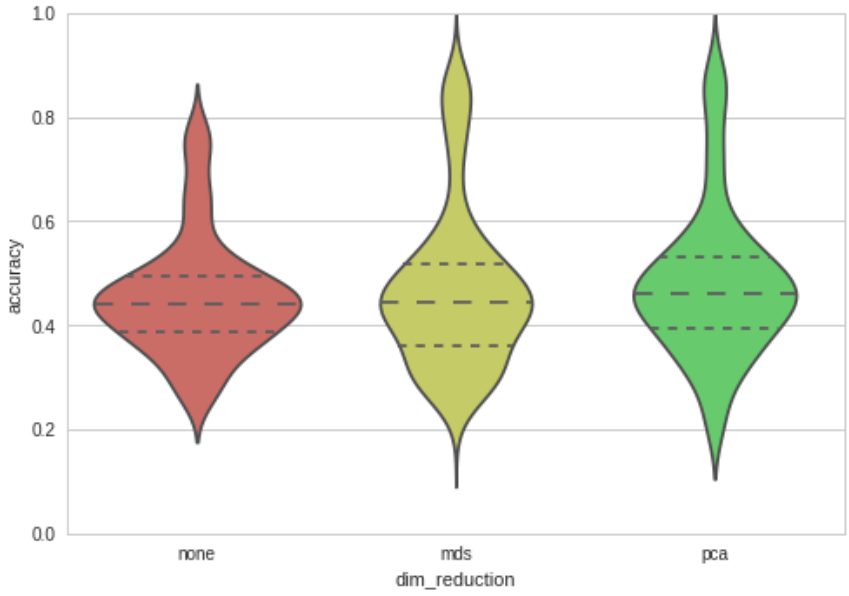


Figure 23. Spectral, 40 clusters. Dimensionality reduction comparison.

As seen on Figure 23, dimensionality reduction is of no help.

4.2 Autoencoder experiments

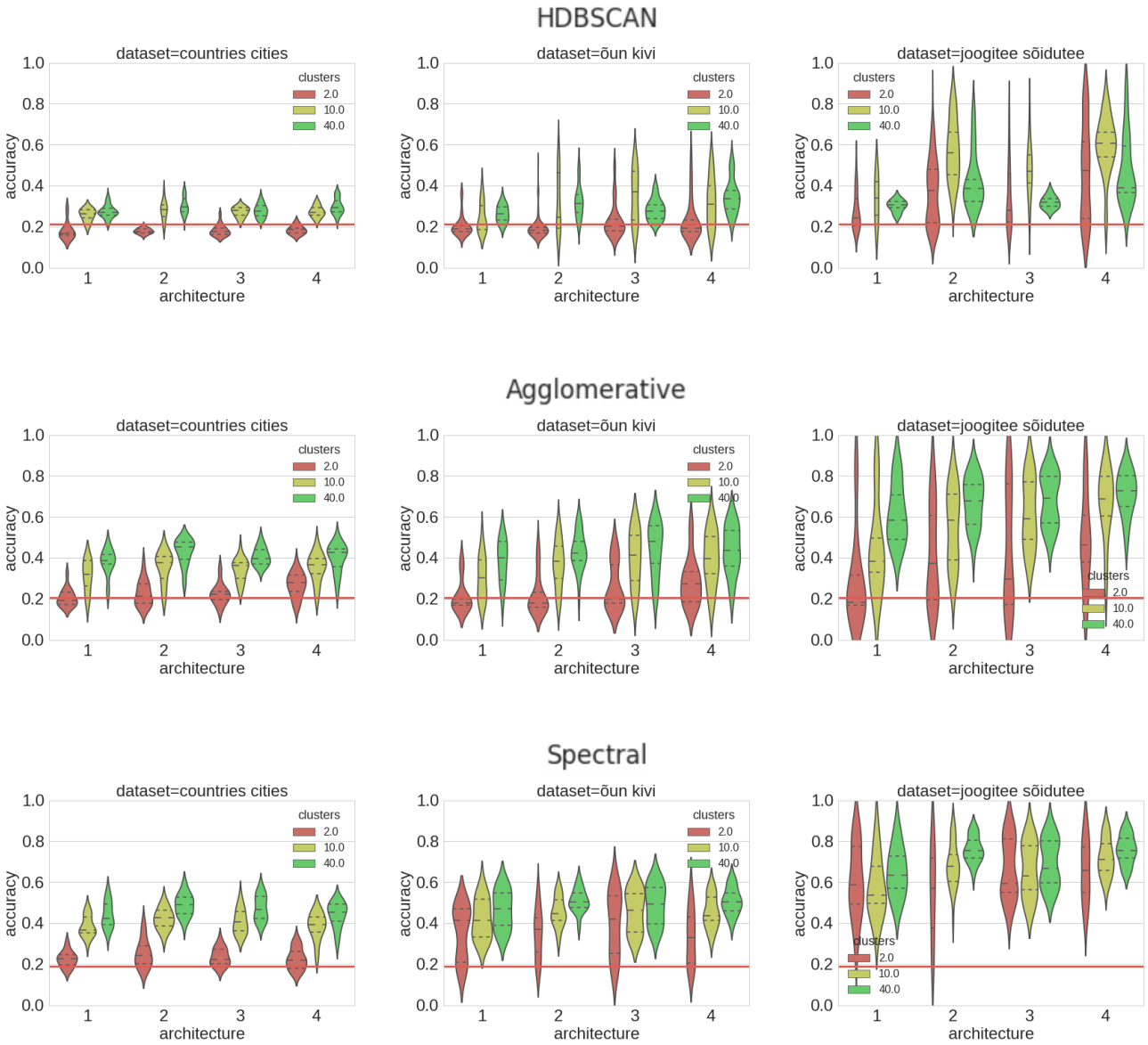


Figure 24. Grand overview of the autoencoder experiments.

On Figure 24 we see that the performance across different architectures is quite consistent, but architecture number 2 seems to yield ever so slightly better performance, than the others. As a reminder, architecture 2 contains two LSTM layers in the encoder and one LSTM layer in the decoder. Five different encoder layers size configurations were evaluated: [[32, 32], [64, 64], [128 , 128], [64, 32], [128, 64]].

The three chosen datasets all show nicely above random results. The important thing here is, how do they compare to the first batch experiments. Lets observe from Figure 25.

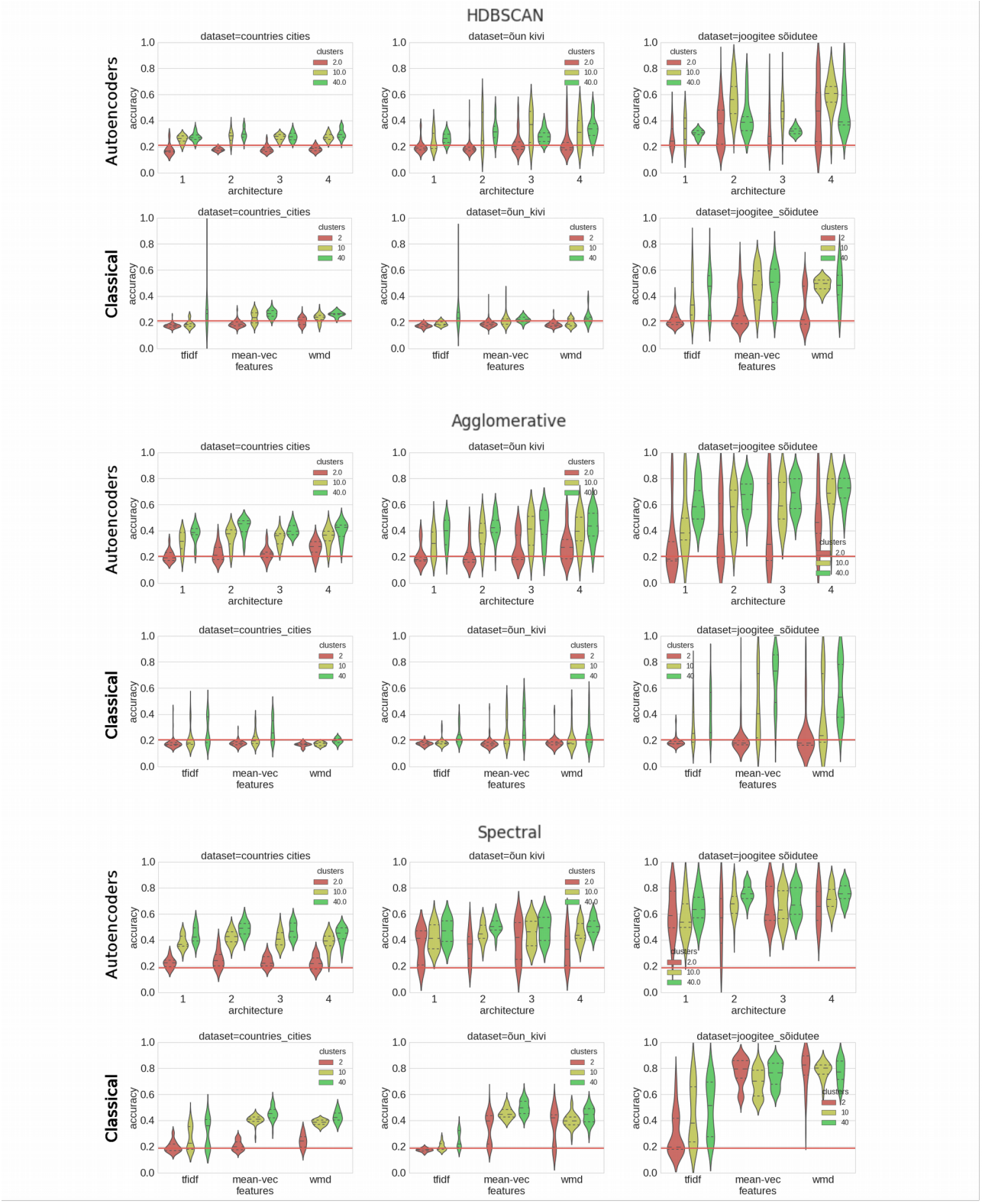


Figure 25. Autoencoder and first batch experiments comparison.

Noteworthy points from Figure 25:

- Autoencoders are somewhat suitable even for the underperformers of first batch experiments: Agglomerative and HDBSCAN.
- Spectral is still the winner. However, with Spectral, mean-vector delivers results comparable to autoencoders. This speaks in favor of mean-vector, because it is a much simpler method, when compared to recurrent autoencoders. Then again, on the flipside, the RNNs have much room for improvement in the architecture department, and a bigger accuracy is expected.

Validation loss and final accuracy

When training neural networks, we may observe the relation between training and validation error. This yields great insight into the training quality of the network. Weather it is fitting or overfitting, etc. On one side, the evaluation loss should indicate the goodness of the final model. In Figure 26 we have plotted all the validation losses of our approximately 3000 autoencoder trianing experiments. A very peculiar pattern emerges. The skip-gram models follow much lower loss values, but they yield better results. Following from this train of thought, we see if validation loss and final output accuracy are correlated. Indeed, they are not. This is somewhat bad news in the unsupervised context of using *PatternExaminer*. Presuming they would be correlated, we could automatically find the best models over some grid search. However, knowing they are not correlated, we may form a test set on the actual data, for which the system developed in this thesis will be used. Ofcourse, the possibility of correlation can not be excluded for more complex models, such that will be explored in future work.

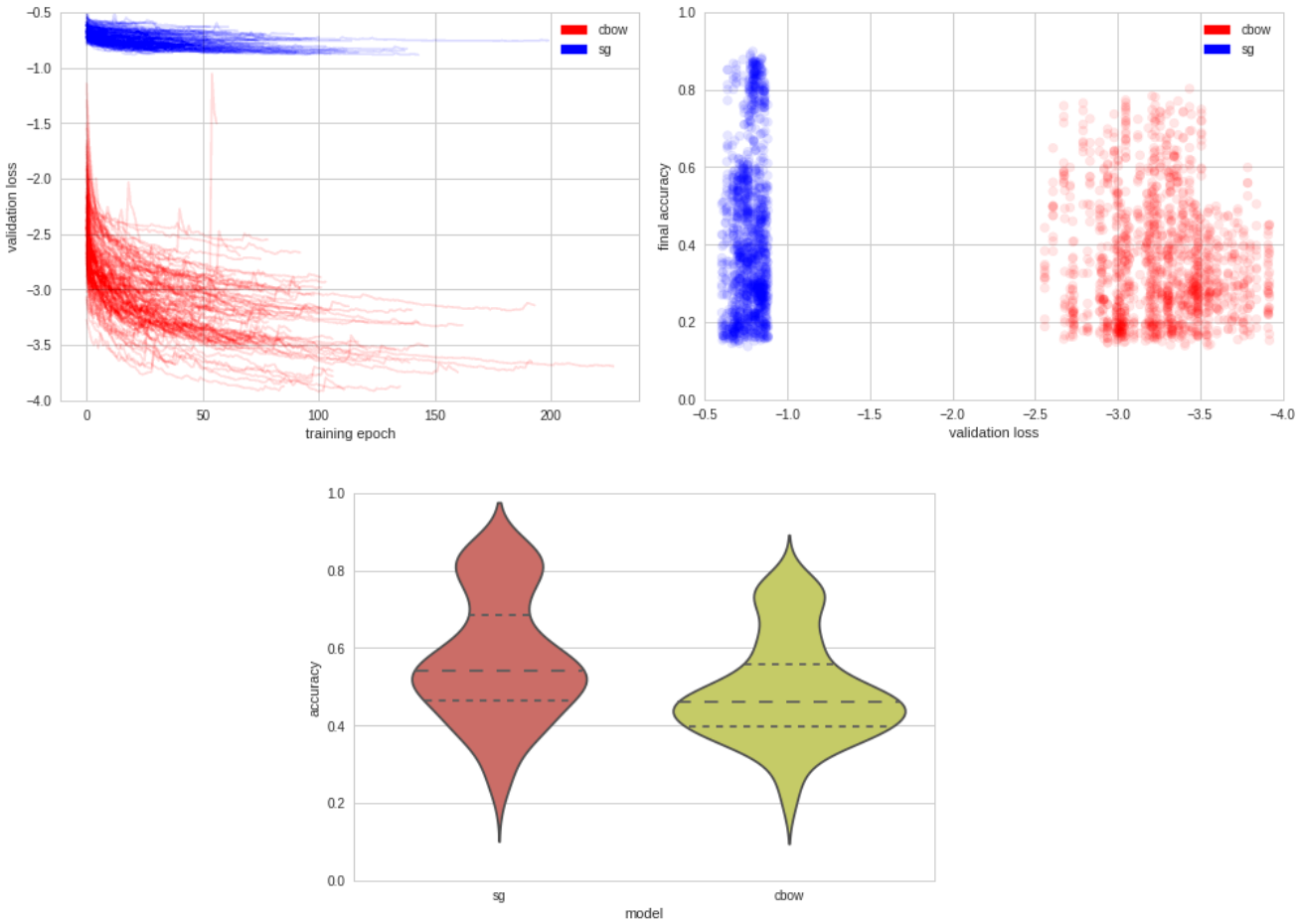


Figure 26. Upper left: all training evaluation losses of our autoencoders. Upper right: lack of correlation between validation loss and final accuracy. Bottom: skip-gram is better than cbow once again.

5. Conclusion

The aim of this thesis has been to study and experiment with various document embedding methods, with the goal of using them as part of a textual pattern examining tool, called *PatternExaminer*. The first use case for *PatternExaminer* will be in developing better fact extraction systems on the Estonian Geenivaramu dataset. This is an important task, because once solved on a sufficient level, is a great leap towards automated clinical trials, personal medicine, etc.

After creating the *PatternExaminer*, a comprehensive batch of experiments was conducted. In total, around 20 000 various experiments. Grid search was employed over all the main parameters of input data, preprocessing methods and clustering algorithms.

We found, that spectral clustering is best for clustering textual data, in comparison with Agglomerative and HDBSCAN. A good context window is symmetric, but the size doesn't matter. Skip-gram word embeddings, averaged to the mean vectors and using angular distance yields the most separable embedding space. Dimensionality reduction did not prove useful in the experiments.

A second batch of experiments was conducted using recurrent autoencoders. A rather simplistic architecture was used. Nonetheless the results were either much better or on-par with the first batch's best performer, described in the last paragraph. Despite good performance by the neural networks, we did not achieve an unsupervised method to select the best method. To overcome this, we will monitor if more advanced models don't have this downside and also create a testing dataset on the corpus on which we are developing the final pattern extracting system.

We also demonstrated how different datasets display extremely different separability. The best separability comes from the most semantically different words – tee-tee. However as the semantics get closer, for example hea-halb, they become increasingly difficult to separate. This leads us to the need for more complex recurrent architectures, which will be experimented with in future work.

The current thesis is a step towards tackling problems facing an aging society, but also towards a semantically aware general artificial intelligence.

6. Bibliography

Ahn, S., Choi, H., Pärnamaa, T., & Bengio, Y. (2016). A Neural Knowledge Language Model. *Arxiv*, 1–12. Retrieved from https://arxiv.org/pdf/1608.00318.pdf

Bengio, Y., Ducharme, R., Vincent, P., & Janvin, C. (2003). A Neural Probabilistic Language Model. *The Journal of Machine Learning Research*, *3*, 1137–1155. https://doi.org/10.1162/153244303322533223

Blei, D. M., Edu, B. B., Ng, A. Y., Edu, A. S., Jordan, M. I., & Edu, J. B. (2003). Latent Dirichlet Allocation. *Journal of Machine Learning Research*, *3*, 993–1022. Retrieved from http://www.cs.columbia.edu/~blei/papers/BleiNgJordan2003.pdf

Campello, R. J. G. B., Moulavi, D., & Sander, J. (2013). Density-Based Clustering Based on Hierarchical Density Estimates. *Advances in Knowledge Discovery and Data Mining*, 160–172. https://doi.org/10.1007/978-3-642-37456-2_14

Chen, H., Martin, B., Daimon, C. M., & Maudsley, S. (2013). Effective use of latent semantic indexing and computational linguistics in biological and biomedical applications. *Frontiers in Physiology*, *4*, 8. https://doi.org/10.3389/fphys.2013.00008

Groenen, P. J. F., & Van De Velden, M. (2004). Multidimensional Scaling. Retrieved from https://core.ac.uk/download/pdf/6515354.pdf

Jang, M.-H., Kim, S.-W., Faloutsos, C., & Park, S. (n.d.). A Linear-Time Approximation of the Earth Mover's Distance. Retrieved from https://arxiv.org/pdf/1106.1521v1.pdf

Jurafsky, D. (2017). Stanford course CS124, Language Modeling. Retrieved May 8, 2017, from https://web.stanford.edu/class/cs124/lec/languagemodeling.pdf

Kusner, M. J., Sun, Y., Kolkin, N. I., & Weinberger, K. Q. (2015). From Word Embeddings To Document Distances. *International Conference on Machine Learning*, *37*. Retrieved from http://proceedings.mlr.press/v37/kusnerb15.pdf

Lipton, Z. C., Berkowitz, J., & Elkan, C. (2015). A Critical Review of Recurrent Neural Networks for Sequence Learning. Retrieved from https://arxiv.org/pdf/1506.00019.pdf

Megill, N. D., & Pavicic, M. (2011). Estimating Bernoulli trial probability from a small sample, (1), 1–4. Retrieved from https://arxiv.org/pdf/1105.1486.pdf

Mikolov, T., Chen, K., Corrado, G., & Dean, J. (2013). Distributed Representations of Words and Phrases and their Compositionality. *Nips*, 1–9. https://doi.org/10.1162/jmlr.2003.3.4-5.951

Mikolov, T., Corrado, G., Chen, K., & Dean, J. (2013). Efficient Estimation of Word Representations in Vector Space. *Proceedings of the International Conference on Learning Representations (ICLR 2013)*, 1–12. https://doi.org/10.1162/153244303322533223

Mikolov, T., Karafiat, M., Burget, L., Cernocky, J., & Khudanpur, S. (2010). Recurrent Neural Network based Language Model. *Interspeech*, (September), 1045–1048. Retrieved from http://www.fit.vutbr.cz/research/groups/speech/publi/2010/mikolov_interspeech2010_IS100722.pdf

Ng, A. Y., Jordan, M. I., & Weiss, Y. (2001). On Spectral Clustering: Analysis and an algorithm. *Advances in Neural Information Processing Systems*, 849–856. https://doi.org/10.1.1.19.8100

Pele, O., & Werman, M. (2009). Fast and robust earth mover's distances. *Proceedings of the IEEE International Conference on Computer Vision*, (Iccv), 460–467. https://doi.org/10.1109/ICCV.2009.5459199

Pennington, J., Socher, R., & Manning, C. D. (2014). GloVe: Global Vectors for Word Representation. *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*, 1532–1543. https://doi.org/10.3115/v1/D14-1162

Řehůřek, R., & Sojka, P. (2010). Software Framework for Topic Modelling with Large Corpora. In *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks* (pp. 45–50). Valletta, Malta: ELRA.

Sch, H. (2008). *Introduction to Information Retrieval*. Retrieved from http://www.math.unipd.it/~aiolli/corsi/0910/IR/irbookprint.pdf

Tammeveski, L., Zafra, R. V., Parts, L., Matiisen, T., & Tampuu, A. (2016). Learning DNA mutational signatures using neural networks.

Tkachenko, A. (2016). Estonian word2vec models. Retrieved from https://github.com/estnltk/word2vec-models/blob/master/README.md

7. Appendix I

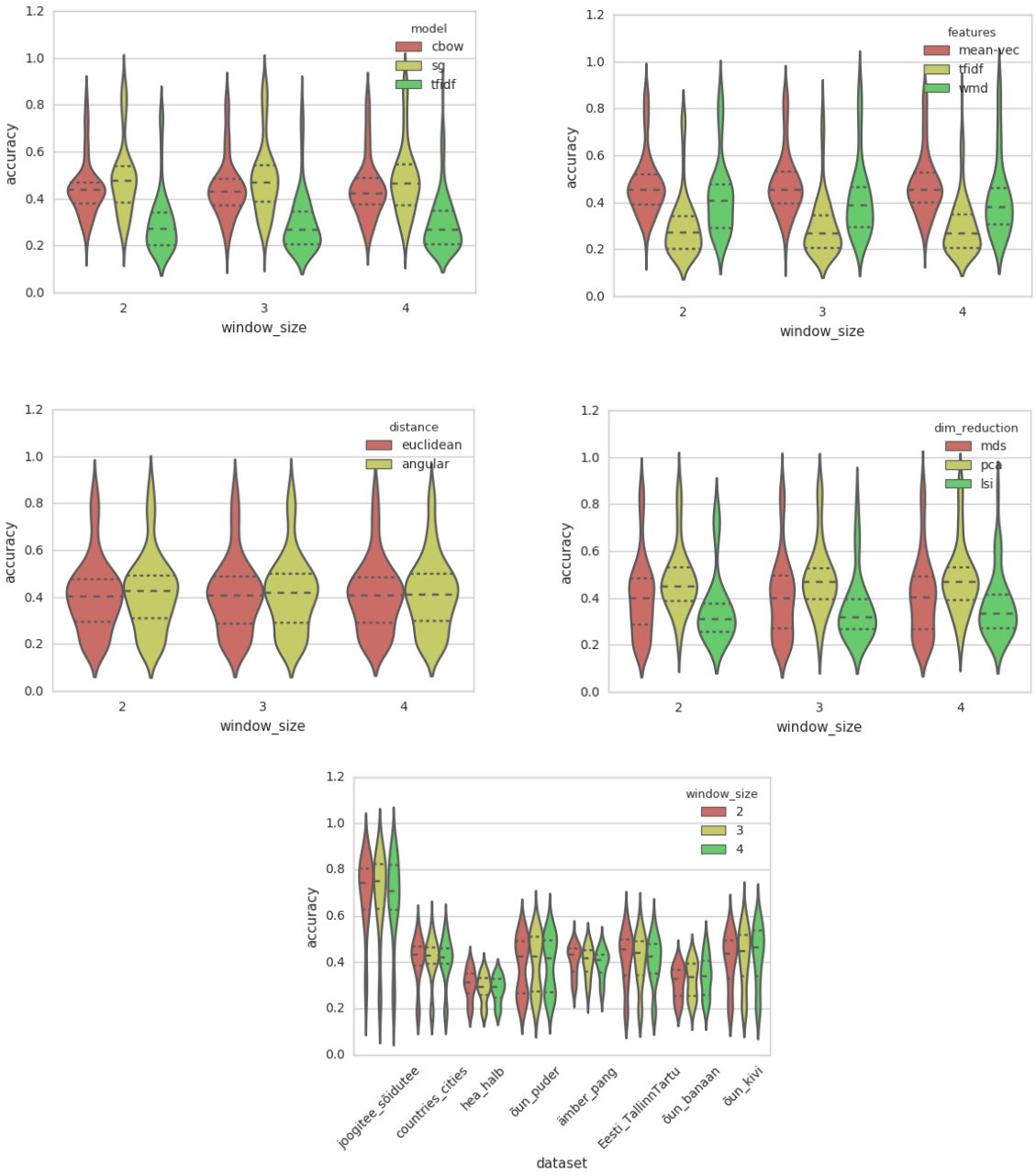


Figure 27. Spectral clustering, n_clusters=40

Profiling the window size against all other parameters. On the first four plots we see the window size does not matter, but on the dataset plot we see how a larger window size can either boost or hurt the performance.

8. Licence

Non-exclusive licence to reproduce thesis and make thesis public

I, Robert Roosalu,

1. herewith grant the University of Tartu a free permit (non-exclusive licence) to:

1.1. reproduce, for the purpose of preservation and making available to the public, including for addition to the DSpace digital archives until expiry of the term of validity of the copyright, and

1.2. make available to the public via the web environment of the University of Tartu, including via the DSpace digital archives until expiry of the term of validity of the copyright,

Context embedding for natural language clustering,
supervised by Sven Laur,

2. I am aware of the fact that the author retains these rights.

3. I certify that granting the non-exclusive licence does not infringe the intellectual property rights or rights arising from the Personal Data Protection Act.

Tartu, 18.05.2017