Linköping University | Department of Computer and Information Science
Master's thesis, 30 ECTS | Datateknik
202019 | LIU-IDA/LITH-EX-A–19/028–SE

Word Clustering in an Interactive Text Analysis Tool

Klustring av ord i ett interaktivt textanalysverktyg

Gustav Gränsbo

Supervisor : Marco Kuhlmann
Examiner : Arne Jönsson

External supervisor : Jussi Karlgren

LINKÖPING UNIVERSITY

Linköpings universitet
SE–581 83 Linköping
+46 13 28 10 00 , www.liu.se

Abstract

A central operation of users of the text analysis tool Gavagai Explorer is to look through a list of words and arrange them in groups. This thesis explores the use of word clustering to automatically arrange the words in groups intended to help users. A new word clustering algorithm is introduced, which attempts to produce word clusters tailored to be small enough for a user to quickly grasp the common theme of the words. The proposed algorithm computes similarities among words using word embeddings, and clusters them using hierarchical graph clustering. Multiple variants of the algorithm are evaluated in an unsupervised manner by analysing the clusters they produce when applied to 110 data sets previously analysed by users of Gavagai Explorer. A supervised evaluation is performed to compare clusters to the groups of words previously created by users of Gavagai Explorer. Results show that it was possible to choose a set of hyperparameters deemed to perform well across most data sets in the unsupervised evaluation. These hyperparameters also performed among the best on the supervised evaluation. It was concluded that the choice of word embedding and graph clustering algorithm had little impact on the behaviour of the algorithm. Rather, limiting the maximum size of clusters and filtering out similarities between words had a much larger impact on behaviour.

Acknowledgments

I would like to direct a special thank you to some people who have taken time out of their otherwise busy schedules to help me with this thesis. Jussi Karlgren, my mentor at Gavagai, who has provided invaluable insights on language in general, and distributional semantics in particular. Marco Kuhlmann, my supervisor, who has provided continuous and reliable support throughout the work on my thesis, and who got me hooked on natural language processing in the first place. Arne Jönsson, my examiner, who provided valuable feedback on my thesis. Finally, all the wonderful people at Gavagai, who not only made the process of writing this thesis very enjoyable, but also helped mold the thesis into something that can provide real value to their product.

Contents

Abstract iii

Acknowledgments iv

Contents v

List of Figures vi

List of Tables viii

1 Introduction 1
- 1.1 Motivation . . . 2
- 1.2 Aim . . . 3
- 1.3 Research Questions . . . 4
- 1.4 Delimitations . . . 4
- 1.5 Thesis Outline . . . 5

2 Theory 6
- 2.1 Word Clustering . . . 6
- 2.2 Word Embeddings . . . 8
- 2.3 Graph clustering . . . 11
- 2.4 Relative Neighbourhood Graph . . . 12
- 2.5 Set Similarity Measures . . . 12
- 2.6 Topic Modeling . . . 13

3 Method 15
- 3.1 Algorithm Outline . . . 15
- 3.2 Unsupervised Evaluation . . . 18
- 3.3 Supervised Evaluation Scheme . . . 20

4 Results 22
- 4.1 Unsupervised Evaluation . . . 22
- 4.2 Supervised Evaluation . . . 31

5 Discussion 35
- 5.1 Results . . . 35
- 5.2 Method . . . 41
- 5.3 The Work in a Wider Context . . . 44

6 Conclusion 45
- 6.1 Research Questions . . . 45
- 6.2 Future Work . . . 46

Bibliography 47

List of Figures

1.1 Grouping topics in the Gavagai Explorer. a) The initial list of suggested topics, in this case 100 topics, ranked by the topics document frequencies. b) The topics from the initial list have been arranged in groups of similar topics, which are ranked based on the groups combined document frequencies. 2

2.1 The RNG for 100 points on the unit square where distances were measured using Euclidean distance. 12

4.1 Boxplot of number of single word clusters, average cluster size and average coherence, and barplot of average number of new clusters in the top 30, for all unsupervised experiments. 23
4.2 Boxplots of a) number of single word clusters, b) average cluster size and c) average coherence, for the hierarchical clustering algorithm using Random Index word embeddings. 25
4.3 Average number of new clusters in top 30 for the hierarchical clustering algorithm using Random Index word embeddings. 25
4.4 Boxplots of a) number of single word clusters, b) average cluster size and c) average coherence, for the hierarchical clustering algorithm using Skipgram word embeddings. 26
4.5 Average number of new clusters in top 30 for the hierarchical clustering algorithm using Skipgram word embeddings. 26
4.6 Boxplot of number of single word clusters, average cluster size and average coherence, and barplot of average number of new clusters in the top 30 for different graph types in combination with the GN-algorithm using a) Random Index and b) Skipgram word embeddings. 27
4.7 Boxplots of a) number of single word clusters, b) average cluster size and c) average coherence, for the GN-algorithm, applied to an unweighted MST constructed from Random Index word embeddings. 28
4.8 Average number of new clusters in top 30 for the GN-algorithm applied to an unweighted MST constructed from Skipgram word embeddings. 29
4.9 Boxplots of a) number of single word clusters, b) average cluster size and c) average coherence, for the GN-algorithm, applied to an unweighted MST constructed from Skipgram word embeddings. 29
4.10 Average number of new clusters in top 30 for the GN-algorithm applied to an unweighted MST constructed from Skipgram word embeddings. 30
4.11 Boxplots of a) number of single word clusters, b) average cluster size and c) average coherence, for the best performing parameter compositions presented in Table 4.3. 32
4.12 Average number of new clusters in top 30 for the best performing parameter compositions presented in Table 4.3. 33

4.13 Precision and recall for all evaluated hyperparameter compositions on the three data sets. The compositions ($G_{cluster} = hierarchical, n = 4, p = 2$) combined with Random Index and Skipgram embeddings are marked separately. 34

List of Tables

3.1 Experiment configurations. 18
3.2 Statistics of the silver standard data sets. 21

4.1 Pearson correlation of metrics in unsupervised experiments when using Random Index embeddings. 23
4.2 Pearson correlation of metrics in unsupervised experiments when using Skipgram embeddings. 23
4.3 Best performing hyperparameter compositions. 31
4.4 Select percentiles of the four unsupervised metrics for the best performing parameter composition ($G_{cluster} = hierarchical, n = 4, p = 2$), with Random Index word embeddings. 31
4.5 Select percentiles of the four unsupervised metrics for the best performing parameter composition ($G_{cluster} = hierarchical, n = 4, p = 2$), with Skipgram word embeddings. 31

5.1 The top 30 words before and after clustering, for two data sets randomly sampled among the 110 data sets used in the unsupervised evaluation. Words appear in order of document frequency, with clustered words separated by a wider margin. To avoid displaying partial clusters at the bottom of the top 30, the clustered top list have been extended past 30 words when necessary. Words in the new top 30 that do not appear in the original top 30 are bold. Clustering was performed with hierarchical clustering, max cluster size of 4 and filtering similarities outside of the top 2%. Results from using both Random Index and Skipgram word embeddings are shown. 39

1 Introduction

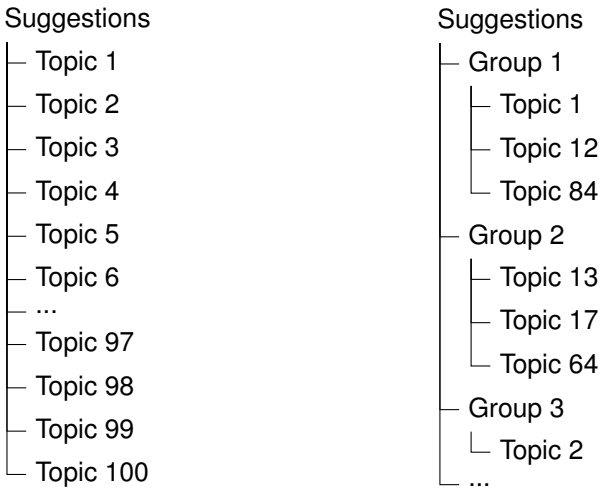
Open-ended free-text questions can be a powerful tool for eliciting unexpected information from a survey respondent. In contrast to closed-ended questions, they do not require a preconceived notion about what information is to be gathered. Instead, they leave it up to the respondent to choose what kind of information they wish to share. When asked about their experience during a recent hotel visit, one respondent might want to share their fond memories of the delicious breakfast scones, and someone else, their bad experience with the full parking lot. Perhaps this flexibility is the reason that open-ended questions have become ubiquitous in online reviews, which often consist of just a numerical grade and an open-ended comment.

However, analysing a large volume of open-ended questions can be problematic. A classical approach to analysing open-ended answers is to have one or more analysts go through the texts, and label them according to an agreed upon coding scheme. Of course, going through all responses manually is time-consuming and expensive. Labels applied by human analysts are also subject to human error, potentially caused by a fatigued analyst or inter-analyst inconsistencies [22, 29].

To alleviate the problems associated with manual coding of open-ended answers, some degree of automation is desirable. One approach to finding themes in a large collection of texts is *topic modeling*. Popular topic models such as *Latent Dirichlet Allocation* (LDA) and *Probabilistic Latent Semantic Analysis* (pLSA) are able to both detect a set of latent topics in a data set, and assign texts a probability of belonging to each topic. Unfortunately, the topics are not always easy to interpret [7], and the number of topics needs to be determined before applying the topic model.

The Swedish company *Gavagai* takes another approach to solve the problem of open-ended answer analysis. Their product *Gavagai Explorer*[1] helps an analyst label and analyse a document collection, and iteratively improve the labels. The labels, representing different topics in the document collection, are represented by a set of keywords. Any text that contains one of the topic keywords will be given that topic label, and potentially many other labels. A user can merge topics, split them, or extend them using a list of words that are similar to the keywords already in the topic. At each step of the process, the documents currently labeled

[1] https://explorer.gavagai.io



(a) Original list of topics

(b) List of grouped topics

Figure 1.1: Grouping topics in the Gavagai Explorer. a) The initial list of suggested topics, in this case 100 topics, ranked by the topics document frequencies. b) The topics from the initial list have been arranged in groups of similar topics, which are ranked based on the groups combined document frequencies.

with a topic can be analysed by viewing the labeled texts, commonly co-occurring topics, as well as sentiment.

This thesis explores how clustering common keywords can help a user more easily detect relevant topics. To accomplish this, a new word clustering algorithm is introduced.

1.1 Motivation

The main task of a user of the Gavagai Explorer is to detect a set of interesting topics and extend their keywords so that they cover all relevant documents. This is accomplished by choosing useful topics among suggested topics, merging them with other similar topics and extending them with suggested synonyms. Topics with a single keyword are initially suggested in a list sorted by the topic's document frequency. Suggestions act as seeds, and if a user thinks the topic is interesting it can be extended by merging it with other topics or by adding new keywords. By displaying similar topics in groups, a user can more easily assess if two topics should be merged. Also, groups can be ranked based on their combined document frequency, allowing topics formed by many less frequent words to appear with higher rank in the list of suggested topics. Figure 1.1 illustrates the concept of arranging the list of topic suggestions in groups.

More generally, clustering topics can be seen as clustering multiple bags-of-words. However, in this thesis, only single word topics are considered, essentially recasting the problem as word clustering. The task of clustering single words, word clustering, has mainly been studied to aid downstream tasks such as language modeling [6], word disambiguation [18] and topic modeling [38]. Clustering words with the intent of displaying clusters to a user is less common, but one example is thesaurus generation. When generating a thesaurus, each word in a large dictionary is given an explanation in the form of a list of the most similar other words in the dictionary [19]. Providing a list of the most similar words from a large vocabulary is exactly what the Gavagai Explorer does when it suggests synonyms. The task at hand, clustering a limited set of words with the intent of organizing them in coherent groups to display to a user, has not been given much attention though.

With the advent of word embeddings, sparked by the seminal publications on *Word2Vec* by Mikolov et al. [23, 24], the interest in word clustering as a means to aid downstream tasks has cooled off. Word embeddings represent words as dense vectors, where similar words have similar vectors. Since these representations are learned through an un-supervised task, they can be created at a low cost given enough text data. This makes word embeddings very useful for representing words in downstream tasks, rendering explicit word clustering redundant. On the flip side, word embeddings open up new possibilities for clustering words for tasks where an explicit clustering is required.

Interpreting word vectors as points in a high-dimensional space allows reasoning about words locations in relation to each other. Much remains unknown about the topology of these high dimensional spaces, but the work of Karlgren et al. [16] as well as Gyllensten and Sahlgren [8] suggests that local structures are much more interesting than global structures. Building on this idea, this thesis explores a new word clustering approach that takes local structures in word embedding into account. By measuring word vector similarities between all words to be clustered, a weighted graph is constructed. Several techniques are evaluated for processing and clustering the graph:

- Two different word embeddings are used to measure word similarities: Random Index and Skipgram embeddings. (Section 3.1.1)
- Several different levels of similarity are considered relevant, all other similarities being treated as irrelevant. (Section 3.1.2)
- Two graph representations: the relative neighbourhood graph (RNG) and the minimum spanning tree (MST). (Section 3.1.3)
- Two edge weighting schemes: weighted edges and unweighted edges. (Section 3.1.4)
- Two graph clustering algorithms: single-linkage hierarchical clustering and the Girvan-Newman (GN) algorithm. (Section 3.1.5)
- Several choices of maximum allowed cluster size. (Section 3.1.5)

Different compositions of these techniques are evaluated across 110 data sets analysed by users of the Gavagai Explorer, measuring their impact on cluster coherence, average cluster size, number of single word clusters produced as well as the number of novel clusters promoted to the top 30 list of topics.

This thesis was carried out at Gavagai AB in Stockholm, Sweden. The new word clustering algorithm proposed in this thesis is applicable outside of the Gavagai Explorer, but this thesis focuses on the algorithm's usefulness in that context.

1.2 Aim

The aim of this thesis is to provide an effective algorithm for arranging a limited set of words in groups deemed meaningful by a user of the Gavagai Explorer. It should be cognitively easy for a user to decide if a group is meaningful, or if it needs to be split up. This means that the words in a group either need to be very coherent, such as a list of cities or colors, or that the group can not be very large. A very coherent group, such as a list of cities, will be cognitively easy to evaluate, since each entry just needs to be verified to be a city. However, if the connection between words in a group is not as apparent, the cognitive load of comparing them pairwise will quickly grow large with an increasing group size.

Arranging topics in the Gavagai Explorer into coherent groups should spare users some of the trouble going through a long list of suggested topics to find topics that should be merged. It should also enable topics represented by a set of less frequent keywords to appear with

higher rank in the list of suggested topics, since the group can be ranked by its combined document frequency.

A scenario in which automatically arranging topics in groups would be useful, could be an analyst who wants to analyse a set of hotel reviews. Common words in hotel reviews might be *room*, *bedroom*, *breakfast*, *scones*, *noise*, *loud* etc. When people use the word *bedroom* in the context of hotel reviews, they are probably referring to the same thing as when they use the word *room*. Thus, the analyst might want to combine these two words into a single topic. The same thing likely also goes for *noise* and *loud*. The words *breakfast* and *scones* will not refer to the same thing, but most likely they are used to talk about the same topic: the hotel breakfast. By automatically grouping these words together, the analyst will not only create the desired topics in fewer actions, she might also find new topics that would not have been found if not suggested.

1.3 Research Questions

To achieve the goal of being able to automatically group a limited set of words in a way that benefits a user of the Gavagai Explorer, this thesis addresses the following research questions:

1. Do words grouped by the proposed word clustering algorithm overlap with groups previously created by users of the Gavagai Explorer?

   If the tested clustering algorithm suggests groups that users have independently created themselves, this means that the method has made suggestions in line with what users find relevant. However, this measure gives no information about whether suggestions help discover previously missing topics.

2. Does grouping keywords affect the rank of some less frequent keywords enough to make novel themes visible in the top list of topics?

   If the rank of a group of words is considerably higher than the rank of any individual word in the group, this enables previously unseen topics to be considered by the user. The default setting in the Gavagai Explorer is to only show the top 30 topics, so it is interesting to measure how many novel themes are promoted to the top 30 list. However, this metric gives no information about the quality of the suggested groups.

3. How reliable is the behaviour of the proposed word clustering algorithm across different data sets?

   Ideally, the word clustering algorithm should produce useful results regardless of the data set that is being analysed. Therefore, it is interesting to measure how the algorithm behaves across multiple data sets.

4. What hyperparameters of the proposed word clustering algorithm have the biggest impact on its behaviour?

   The proposed word clustering algorithm has six hyperparameters: the choice of word embedding, the threshold for filtering out irrelevant similarities, the graph representation, the edge weighting scheme, the graph clustering algorithm and the maximum cluster size. To be able to make a sound choice of hyperparameters it is interesting to know what parameters have the largest impact on its behaviour.

1.4 Delimitations

This thesis makes no attempt to compare the effectiveness of other coding approaches to the Gavagai Explorer. What it does, is to present how a new word clustering can help to improve the usefulness of the Gavagai Explorer.

The word clustering algorithm presented is adapted to the use case of clustering topics in the Gavagai Explorer. Some design decisions are thus made for the sole purpose of fitting the clustering methods into the framework of the explorer. The most notable decision was to tokenize the evaluated data sets using the same tokenizer as the Gavagai Explorer, and combining common collocations into single word tokens.

In the very first iteration of analysis in the Gavagai Explorer, all suggested topics will consist of single words by default. However, some topics will be extended with other words that users have often grouped with that word, based on historic data. This functionality has been disabled for all experiments in this thesis, for two reasons. First, the functionality only affects words that have been analyzed multiple times before, which means that it has no effect on some data sets. Second, it complicates the task of clustering topics by requiring bags-of-words to be clustered instead of words.

1.5 Thesis Outline

The remainder of this thesis is structured as follows. First, chapter 2 introduces related work and the theory relevant for this thesis. Chapter 3 presents the clustering algorithm and its hyperparameters, as well as the unsupervised and supervised evaluation schemes used to evaluate different compositions of the hyper parameters. Chapter 4 presents results from the unsupervised and supervised evaluation. Chapter 5 discusses the results of the evaluation, as well as the method in general. Finally, chapter 6 presents answers to the research questions and outlines possible future work.

2 Theory

This chapter introduces related work, and theory necessary to understand this thesis.

2.1 Word Clustering

Word clustering is the task of arranging words from natural language in groups of words that are in some sense similar. Typically, no correct solutions exist for word clustering and groups to do not correspond to know categories.

Many different problems can be categorized as word clustering, and the purpose of solving these problems vary. This section introduces some tasks that have been solved using word clustering and some ways that word clustering can be evaluated.

2.1.1 Word Clustering as an Upstream Task

Much research on word clustering poses word clustering as an upstream task for some other task. Brown et al. [6] assign words to unlabeled word classes through clustering based on mutual information. They condition a bi-gram language model on these classes in order to help the language model generalize examples of one word to all other words in the class.

Li and Abe [18] extend the clustering algorithm of Brown et al., and use the detected clusters to aid them in a word disambiguation task.

Recently, Viegas et al. [38] introduced a novel document representation for topic modeling, where words are replaced with meta-words consisting of their close neighbours in a word embedding. Finding these local neighbourhoods is a special kind of clustering, where each word is used to seed a cluster, resulting in as many clusters as there are words, most of which overlap with other clusters. They claim to achieve state-of-the-art topic modeling results using this representation.

Common for all these papers is that they evaluate their clustering through the benefit they provide the downstream task.

2.1.2 Word Clustering as a Primary Task

Lin [19] makes more direct use of word clustering, automatically generating a thesaurus listing the closest neighbours of every word measured using a mutual information criterion.

Similar to Viegas et al. [38], they find a cluster centered around each word in the corpus, allowing these clusters to overlap. The clusters are evaluated by comparing them with two manually created thesauri, WordNet1.5 [25] and Roget's Thesaurus.

2.1.3 Approaches to Word Clustering

Brown et al. [6] use a greedy agglomerative clustering algorithm to find a fixed number of clusters. All words begin in different clusters, that are merged in an iterative process until the desired number of clusters remain. In each iteration, a cluster pair is chosen in such a way that their merger results in the least loss of average mutual information across all clusters. Their definition of mutual information is such that it measures the information that seeing a word from class l carries on the likelihood that the next word is from class m, which is fitting for their application in a bi-gram language model.

Lin [19] centered clusters around each word, containing the word's N closest neighbours according to some similarity metric. Their similarity metric of choice is based on mutual information in dependency triplets.

Viegas et al. [38] also find clusters around every word, but do not limit them to a specific size. Instead, they set a threshold on similarity and include all words above that threshold. They measure similarity using cosine similarity in a word embedding, experimenting with both the continuous-bag-of-words (CBOW) implementation of Word2Vec and Fasttext embeddings.

2.1.4 Evaluating Clustering Algorithms Using a Gold Standard

Word clustering is an unsupervised task, and there is not always a clear right or wrong solution. Regardless, comparing word clusters against a set of manually created clusters is an attractive idea since it would allow the use of quantitative evaluation metrics. A gold-standard word clustering might not be the only valid clustering of a given set of words, but if an algorithm produces clusters similar to the gold standard it would indicate that it is doing something right.

The Rand-index, as introduced by Rand [31], evaluates a cluster assignment against a gold standard by pairwise inspecting the clustered elements. Every element pair is classified as either a *true positive* (TP), *true negative* (TN), *false positive* (FP) or *false negative* (FN). If two elements are in the same cluster, the pair is classified as TP if they are also in the same cluster in the gold standard, otherwise they are classified as FP. If two elements are in different clusters, they are classified as TN if they are also in different clusters in the gold standard, otherwise they are classified as FN. The Rand-index is defined as

$$\text{Rand-index} = \frac{|TP| + |TN|}{|TP| + |FP| + |TN| + |FN|}, \tag{2.1}$$

where $|TP|, |TN|, |FP|$ and $|FN|$ are the number of true positives, true negatives, false positives and false negatives respectively. This is exactly the same measurement as *accuracy*, which is common in information retrieval and classification.

Manning et al. [21] describe how the Rand-index approach of pairwise evaluating elements can be extended to define *precision* and *recall*. Precision is defined as

$$precision = \frac{|TP|}{|TP| + |FP|} \tag{2.2}$$

and recall as

$$recall = \frac{|TP|}{|TP| + |FN|}. \tag{2.3}$$

2.2 Word Embeddings

Word embeddings map words in natural language to vectors of real numbers. These vectors can be interpreted as points in a high.dimensional space called a word space. Useful word embeddings will map similar words to similar vectors, i.e. points that are close in the word space. Similarity is most commonly defined according to the distributional hypothesis, which in the words of Rubenstein and Goodenough [33] states that "words which are similar in meaning occur in similar contexts". What constitutes a context varies with different word embeddings.

This chapter outlines some common ways to construct word embeddings.

2.2.1 Matrix Decomposition

Following the intuition of the distributional hypothesis, a fitting representation of a word would be the contexts in which it has been observed. A matrix where each row represents a word and each column a context, is a type of primitive word embedding. Contexts can be chosen in multiple ways, such as what paragraphs the word has appeared in or what words it has appeared close to. However, these word embedding will have as many dimensions as there are contexts, which leads to two problems with dimensionality: the dimensionality varies with the size of the vocabulary, and it can get very high.

Dimensionality reduction techniques such as Singular Value Decomposition (SVD) can be used to project these high dimensional vectors into a lower dimensional space. Reducing the dimensionality of word vectors might have positive effects beyond making the vectors more manageable. Noisy dimensions can be excluded, and higher-order similarities can captured. A higher-order similarity could be that two words are deemed to be similar not because they appear in the same context, but in similar contexts.

Latent Semantic Analysis (LSA) was one of the earliest topic models, but can also be viewed as a word embedding [9]. In LSA, the context of a word is the paragraph it appears in. A co-occurrence matrix is constructed for words and unique paragraphs. This matrix is decomposed using SVD, and its dimensionality is reduced by removing components with low eigenvalues. The decomposition results in word embeddings where words that often occur in the same paragraphs are similar.

2.2.2 Random Indexing

Decomposing large matrices can be computationally expensive, lending frequent updates to such word embeddings impractical. To deal with this problem, Kanerva et al. [15] introduced the use of Random Indexing to create word embeddings, also studied by Sahlgren [34]. Random Index word embeddings use a set dimensionality D, typically in the orders 1000-2000. Each context is assigned an index vector, which is a vector of D elements, most set to 0 and a few randomly chosen elements set to either -1 or 1. Each word is assigned a word embedding, also of length D, initially with all elements set to 0. The word embeddings are trained iteratively by observing words in different contexts. Any time a word is observed in a context, the index vector of that context is added to the word embedding, eventually leading to words appearing in the same contexts having similar word embeddings. Unlike the matrix decomposition methods, random indexing does not pick up on higher-order similarities.

2.2.3 Neural Word Embeddings

Neural word embeddings are generated by training a neural network to solve some problem, such as language modeling, and then interpreting the networks internal representations of words as word vectors.

An early neural word embedding was the probabilistic feedforward neural language model of Bengio et al. [3]. It learns to predict the next word in a sentence given the pre-

vious N words. The model consists of a projection matrix C, used to project each of the N previous words into a low-dimensional representation, in other words C is the learned word embedding. The representations of the previous N words are concatenated, and then fed to a fully connected layer, which in turn is connected to an output layer with one output neuron per word in the vocabulary. By applying the softmax function to the output layer a probability for each word in the vocabulary V given the context of the previous N words is estimated. The model achieved state-of-the-art language modeling performance, but suffered from long training times due to the many parameters involved in the computation. Even when significantly reducing the cost of the expensive softmax calculation over the $|V|$ output neurons using the hierarchical softmax [27], the cost of calculating the activations of the second hidden layer remains relatively large.

Mikolov et al. [23, 24] introduced the Skipgram neural word embedding model with the goal of efficiently training word embeddings from very large collections of texts. Their architecture consists of a projection matrix C, projecting a single word w into a low-dimensional vector, followed immediately by a an output layer of $|V|$ neurons corresponding to the words in the vocabulary V. The task of the model is to predict a target word, sampled from the N words before, or N following words in a sentence. Predicting a word is done by applying the softmax function to the output layer. Just as with the feedforward neural language model, the softmax computation can be made drastically more efficient by using a hierarchical softmax. Since the fully connected layer used by Bengio et al. was removed, the network can be trained much more efficiently.

Mikolov et al. [24] also propose an alternative solution to using the hierarchical softmax: Negative Sampling. The simple idea behind negative sampling is to not classify the target word among all words in the vocabulary, but rather classify it among a small sample of negative words. For example, this means that the task of predicting *boat* to appear in the context of *fishing* no longer requires choosing *boat* from among all words in the vocabulary, but instead in a smaller sample of words such as *boat, mother, duck, the, run, green*. With negative sampling, the number of parameters required to be accessed when computing the softmax, and during back propagation, is reduced drastically. Also, Mikolov et al. [24] showed that using negative sampling increased the quality of learned word embeddings as measured by their ability to find word analogies such as: *What word is to Sweden, what Berlin is to Germany?*

Another trick introduced by Mikolov et al. [24] to improve the quality of their word embeddings is to subsample common words. With the rationale that some common words like *the, in* and *with* will appear in the context of almost all words, they might not provide much information about the meaning of the words in their context. At the same time, they can also appear with frequencies magnitudes larger than less common words. To counter the problem of frequently training on these less informative examples, subsampling assigns all words a probability of being ignored during training based on their frequency in the text collection. Just as with negative sampling, subsampling was shown to speed up training and improve the quality of the resulting word embeddings.

Levy and Goldberg [17] have shown that Skipgram with negative sampling (SGNS) implicitly decomposes a word-context matrix, where the cells are the *pointwise mutual information* (PMI) (Equation 2.11) of a given word and context, shifted by a global constant. In other words, SGNS learn the same word embeddings that matrix decomposition of a PMI-matrix achieves.

2.2.4 Word Embedding Topology

The most common way of defining similarity between two words w_i and w_j in a word embedding is through the cosine similarity of their respective word vectors: v_i and v_j. Cosine similarity between two words in a word embedding is defined as

$$sim_{cos}(w_i, w_j) = \frac{v_i \cdot v_j}{\|v_i\|\|v_j\|}, \tag{2.4}$$

where $\cdot$ denotes the dot product and $\|v\|$ the Euclidean norm of vector v. In theory, cosine similarity can take values in the range $[-1, 1]$, where a value of -1 indicates that two word vectors have opposite directions, 0 that the vectors are orthogonal and 1 that they have the same direction. Cosine distance can be defined as

$$dist_{cos}(w_i, w_j) = 1 - sim_{cos}(w_i, w_j), \tag{2.5}$$

taking values in the range $[0, 2]$, where 0 means there is no distance and 2 is the maximum distance. Cosine distance is not a proper distance metric, since it does not satisfy the triangle inequality

$$dist(x, z) \leqslant dist(x, y) + dist(y, z). \tag{2.6}$$

In practice, this means that the cosine distance between two words can be larger than their combined distance to a common neighbour. This makes intuitive sense when considering a triplet of words such as *window*, *glass* and *mug*. Glass relates to both window and mug, but they do not relate to each other.

In a word embedding where words have been arranged based on relatively narrow contexts, words with similar semantics should be close to each other according to the distributional hypothesis. This can be verified empirically by inspecting the closest neighbours of words, which are often clearly semantically related to the root word. It has also been shown qualitatively using various benchmarks such as synonym detection and word intrusion [35].

Though cosine similarity has been used to measure similarities in multiple different word embeddings, the values of similarity are not comparable across different embeddings. Gyllensten and Sahlgren [8] have illustrated that the average cosine similarities between both the nearest neighbours in a word embedding and randomly sampled words vary significantly between different word embeddings.

Many questions remain about the actual topology of word embeddings though. Karlgren et al. [16] suggest that word embeddings have a filamentary structure, and that local structures have substantially lower dimensionality compared to the embedding itself. Gyllensten and Sahlgren [8] introduce a novel approach to inspect the local structures in a word embedding. By constructing a *relative neighbourhood graph* (RNG) (Section 2.4) over a word's closest neighbours they show that different senses of a polysemous word are typically related to the word in different ways, not necessarily being similar to the other senses. They also argue that every word in a word embedding has a semantic horizon, beyond which it is not meaningful to compare it to other words.

2.2.5 Collocation Detection

Collocation detection is the task of identifying phrases such as *Bank of America*, *moon lander* and *dark roast*; phrases where the collocation of multiple words carry a specific meaning. When training word embeddings, it would be desirable to treat such collocations as single word units. Mikolov et al. [24] employ a simple approach, in which they determine that a bi-gram should be treated as a collocation if two words frequently occur together, in relation to how often they occur on their own. For each bi-gram (w_i, w_j) in the data set, a score is assigned:

$$\text{score}(w_i, w_j) = \frac{\text{count}(w_i, w_j) - \delta}{\text{count}(w_i) \cdot \text{count}(w_j)}, \tag{2.7}$$

where $\text{count}(w_i, w_j)$ is the number of occurrences of the bi-gram (w_i, w_j), and $\text{count}(w_i)$ the number of occurrences of an individual uni-gram. δ is a discounting factor that will make sure that bi-grams of very infrequent words do not receive too high scores. After scoring all bi-grams, they decide upon a threshold above which to treat bi-grams as collocations. By combining these bi-grams into single word units and repeating the scoring process, collocations consisting of more than two words can be detected.

2.3 Graph clustering

In this thesis the input used to cluster words is a similarity matrix, which can be interpreted as an un-directed, weighted graph. Finding clusters in un-directed, weighted graphs is a well-researched problem. This section outlines the graph clustering algorithms used in this thesis, and introduces some alternatives.

2.3.1 Hierarchical Clustering

Hierarchical graph clustering is a type of clustering where elements are grouped in a hierarchy. At the top-level, all elements belong to the same cluster, and at the other end all elements are assigned to individual clusters. Hierarchical is typically either agglomerative (bottom-up), starting with all elements in individual clusters, or divisive (top-down), starting with all elements in the same cluster.

A simple agglomerative hierarchical approach is *single-linkage clustering*. It starts of with all nodes in separate clusters, and iteratively merges the two clusters that are most similar, or equivalently, least dissimilar. In the case of edges that measure dissimilarity, this means merging the two clusters that are connected with the weakest edge (least dissimilar). The same clusters can also be achieved in a divisive fashion, by constructing the *minimum spanning tree* (MST) of the dissimilarity graph, and iteratively removing the heaviest link of the MST. The MST can be constructed in $O(m \log n)$ in a graph with m edges and n nodes using Kruskals algorithm. Performing single-linkage clustering on the MST can then be performed in $O(n \log n)$. The worst case time complexity thus happens for fully connected graphs, which have $m = n2$, in which case the time complexity is $O(n^2 \log n)$, since it is dominated by constructing the MST. A drawback of single-linkage clustering is that a node with a single heavy link will usually be isolated, instead of being connected to the component which it has a link to.

2.3.2 The Girvan-Newman Algorithm

In an attempt to improve on the behaviour of the common hierarchical clustering algorithms like single-linkage clustering, Girvan and Newman [11] introduce the Girvan-Newman (GN) algorithm, which detects communities in a weighted graph in a divisive fashion based on edge centrality. In each step of their hierarchical process they remove the edge in the graph that is most central, defined as the edge through which the most number of shortest paths pass through. The steps of the algorithm are as follows:

1. Calculate the shortest paths between all nodes of the graph
2. Count the number of such shortest paths that pass through each edge (this is the edge's betweenness centrality)
3. Remove the edge with the highest centrality.
4. Repeat from step 1 until no edges remain.

The algorithm can be made more efficient by only recalculating edge centrality for edges that were connected to the edge removed in step 3, but this has no effect on the worst case time complexity which is $O(m^2 n)$ in a graph with m edges and n nodes.

2.3.3 Alternative Graph Clustering Algorithms

Zahn [40] proposes a graph clustering method where edges whose weight are inconsistent with the weights of edges in their surroundings are removed. Grygorash et al. [12] build

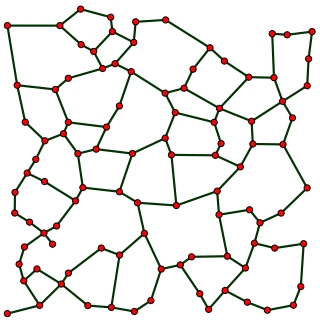


Figure 2.1: The RNG for 100 points on the unit square where distances were measured using Euclidean distance. Source: [10]

on the algorithm of Zahn, and define an algorithm for clustering an MST by iteratively removing edges in a way that minimizes the standard deviation of weights within the resulting components.

2.4 Relative Neighbourhood Graph

The *relative neighbourhood graph* (RNG) [36] is a special graph that spans elements based on their distances to each other. All elements are treated as nodes in the RNG, and edges are added between elements if there are no elements between them. If the RNG for the elements in the set S is constructed according to the binary distance metric $dist$, then an edge will be added between $a, b \in S$ if and only if $dist(a, b) \leqslant dist(a, c)$ and $dist(a, b) \leqslant dist(b, c) \forall c \in S$. Any binary distance metric between elements can be used to compute the RNG. For an arbitrary distance metric, the time complexity of constructing the RNG is cubic in terms of the size of the set, $O(|S|^3)$, assuming the distance metric can be computed in constant time, $O(1)$.

Figure 2.1 illustrates the RNG for 100 points in the unit square constructed according to the Euclidean distance metric.

2.5 Set Similarity Measures

One of the delimitations of this thesis is to only cluster single word topics, not bags-of-words. However, by using a set similarity metric, the framework introduced to cluster words in this thesis could be extended to also cluster bags-of-words.

To cluster bags-of-words, i.e. sets of words $\{w_1, \ldots, w_N\}$, each node in the graph being clustered should represent a bag-of-words. Word embeddings enable similarity to be measured between two words using cosine similarity, as defined in Equation 2.4. In agglomerative hierarchical clustering, similarities between clusters are either defined using

$$\text{single-linkage}(W_i, W_j) = \min_{w_i \in W_i, w_j \in W_j} sim(w_i, w_j), \tag{2.8}$$

or

$$\text{average-linkage}(W_i, W_j) = \frac{1}{|W_i||W_j|} \sum_{w_i \in W_i, w_j \in W_j} sim(w_i, w_j), \tag{2.9}$$

where W_i and W_j are two sets, and $sim(w_i, w_j)$ is a similarity metric between their elements. Single-linkage considers two sets to be precisely as similar as their two most similar elements, this is the same approach as is used in the single-linkage clustering approach presented earlier. Average-linkage defines the similarity of two sets as the average similarity of their elements.

2.6 Topic Modeling

Topic modeling is used to find themes in unlabeled document collections, enabling the documents to be categorized and analysed. This section briefly describes the two popular topic modeling techniques *Latent Dirichlet Allocation* (LDA) and *Probabilistic Latent Semantic Analysis* (pLSA). It also outlines research on improving the interpretability of topics produced by topic models.

2.6.1 pLSA and LDA

The two popular topic models *Latent Dirichlet Allocation* (LDA) [4] and *Probabilistic Latent Semantic Analysis* (pLSA) [13] analyse the distribution of words across multiple texts, and attempt to explain the distribution using a set of latent variables, interpreted as topics. Each topic is associated with a probability distribution over words, and they are often interpreted by looking at the words in the distribution that are given the largest probability. For example, a topic that gives large probability to the words *father, mother, brother, sister, kin* and *relative* etc. could be interpreted as being about family. Through the use of sampling methods such as *Markov-Chain Monte Carlo* (MCMC) these probability distributions are made to fit the statistics of a document collection.

2.6.2 Topic Interpretability

In their seminal paper, Chang et al. [7] study how humans perceive the coherence of the most probable words in a topic found by a topic model. Their findings showed that many topic models produced topics that were hard to interpret. Since then, making topics more interpretable has been the focus of much research.

One approach has been to study the correlation of intrinsic measures of topic coherence to human perception of coherence. A variety of metrics have been identified that correlated positively with human perception. Commonly, these metrics compute some confirmation metric between pairs of top words within a topic and aggregate these pairwise measures to estimate coherence. Given a pairwise confirmation measure sim, topic coherence can be defined as

$$coherence_{sim} = \frac{2}{N(N-1)} \sum_{i=1}^{N-1} \sum_{j=i+1}^{N} sim(w_i, w_j), \tag{2.10}$$

where $\{w_1 \dots w_N\}$ are the top words of a topic.

Newman et al. [28] measure pairwise confirmation using *Pointwise Mutual Information* (PMI). Defining pairwise confirmation as,

$$sim_{PMI}(w_i, w_j) = log \frac{P(w_i, w_j)}{P(w_i)P(w_j)}, \tag{2.11}$$

with $P(w_i)$ and $P(w_j)$ being the probability of observing the words w_i and w_j respectively and $P(w_i, w_j)$ the probability of observing them together. These probabilities are estimated based on a large document collection as

$$P(w) = \frac{D(w)}{N}, \tag{2.12}$$

$$P(w_i, w_j) = \frac{D(w_i, w_j) + \epsilon}{N}, \tag{2.13}$$

where N is the number of documents in the collection, $D(w)$ is the number of documents containing the word w, and $D(w_i, w_j)$ is the number of documents containing both the word

w_i and w_j. ϵ is a small factor added to shift extra probability mass to unseen co-occurrences in order to avoid assigning them zero probability.

Mimno et al. [26] introduce a similar measure, but instead of using PMI, they measure confirmation of two words as the conditioned probability of the less common word given the more common word,

$$sim_{UMass}(w_i, w_j) = log\frac{P(w_i, w_j)}{P(w_j)}. \tag{2.14}$$

Röder et al. [32] showed that both of these metrics correlate better with human perception of coherence when word probabilities were estimated based on a large external corpus, compared to when estimated using the documents used for topic modeling. They also show that many other topic coherence measures correlate well with human perception. One such measure suggested by Aletras and Stevenson [1] uses word embeddings to measure pairwise confirmation as the cosine similarity of the two words word vectors, as defined in Equation 2.4. By optimizing one of these topic coherence measures, topic models can be guided to produce topics that are perceived as more coherent by humans.

Another approach to improve topic coherence is to include external knowledge in the topic modeling process. Bartmanghelich et al. [2] make use of the von Mises-Fisher distribution to incorporate word embeddings into their topic model, thus taking semantic information in the word embeddings into account when creating topics. Yao et al. [39] also make use of the von Mises-Fisher distribution, but instead of using word embeddings, they use so-called knowledge graph embeddings. Knowledge graph embeddings embed the content of knowledge graphs into low dimensional space, capturing information about relationships among entities in the graph.

All the mentioned methods try to guide the topic modeling process by incorporating external information, hoping that it will better model the real world. An orthogonal approach is to make the topic modeling process interactive, thus giving a human the power to influence how the topics form. Hu et al. [14] introduce a framework that they call *interactive topic modeling* (ITM). The basic idea is to run the inference of a topic model for a couple of iterations, then let a human inspect the topics. The human is able to say that some words in a topic are incoherent, or that words in different topics should belong to the same topic. These two steps are repeated until the human user is satisfied with the topics.

3 Method

The word clustering approach presented in this thesis is a multi-step process, with several replaceable components, that can be seen as the algorithm's hyperparameters. This chapter starts out by outlining the algorithm, and then it describes each of the parameters in detail. Next, an unsupervised strategy for evaluating and determining promising hyperparameter compositions is introduced. Finally, a supervised evaluation scheme is described.

3.1 Algorithm Outline

To explore different design choices when constructing the word clustering algorithm, an algorithm outline with replaceable components has been constructed. Algorithm 1 outlines the algorithm, and the role of each component.

Algorithm 1 has five parameters: a word embedding W_e, a distance threshold p, a type of graph G_{type}, an edge weighting scheme w_{scheme} and a graph clustering algorithm $G_{cluster}$. The function of each component is explained in the following sections.

The type of graph clustering applied to the graph, controlled by the $G_{cluster}$ parameter, is very central to the algorithm. In this thesis two choices are considered: hierarchical clustering using divisive single-linkage (section 2.3.1) which is referred to simply as hierarchical clustering, and the Girvan-Newman (GN) algorithm (section 2.3.2).

Algorithm 1: Word clustering outline

Input: A set of words: S
Parameters: W_e, p, G_{type}, w_{scheme}, $G_{cluster}$
Output: Clusters of the words in S: C
$A \leftarrow$ Compute adjacency matrix for S using cosine distances in the word embedding W_e
$G \leftarrow$ Filter out distances outside of the p-th percentile in A, producing a sparse graph
$G \leftarrow$ Convert the sparse graph into a graph of type G_{type}
$G \leftarrow$ Update the weights in G following the weighting scheme w_{scheme}
$C \leftarrow$ Cluster the nodes of G using clustering algorithm $G_{cluster}$

3.1.1 Computing the Adjacency Matrix

The hierarchical clustering algorithm and the GN-algorithm expect low weights to indicate a high degree of similarity, thus, weights should indicate distances. For this reasons, cosine distance is used in favor of cosine similarity.

The adjacency matrix is simply a square matrix A such that

$$A_{i,j} = dist_{cos}(S_i, S_j), 1 \leqslant i, j \leqslant N, \tag{3.1}$$

where S is a set of N words, $dist_{cos}$ is the cosine distance metric from Equation 2.5 computed using the word embedding W_e .

Two word embeddings W_e are evaluated: Random Index and Skipgram embeddings. The Random Index embeddings have 2,000 dimensions and use a context window consisting of the two preceding and the two succeeding words for each word. They were trained on a large proprietary data set of news and social media texts provided by Gavagai, and contain collocations of up to three words.

A Skipgram architecture with 300 dimensions and context size $N = 5$ was trained with hierarchical softmax, using the Gensim[1] implementation. The model was trained for ten epochs on the complete English Wikimedia dump from January 20th, 2019[2]. Before training the Skipgram model, the Gensim Phrases[3] module was used to detect common collocations, based on the criterion used by Mikolov et al. [24] defined in Equation 2.7. Two passes of the collocation extraction algorithm were made over the Wikimedia dumps, in theory allowing collocations of up to four words to be detected.

Unfortunately, the vocabularies of both embeddings did not completely match, meaning that some words in the Random Index embeddings did not exist in the Skipgram embeddings and vice versa. As a result, the evaluation strategies introduced in Sections 3.2 and 3.3 had to be adjusted to not consider any words not part of both embeddings vocabularies. This is further explained in the respective sections.

3.1.2 Filtering out Irrelevant Similarities

Following the advise of Gyllensten and Sahlgren [8], similarities below a certain threshold are treated as irrelevant. Since the adjacency matrix A represents a fully connected graph with cosine distances as edge weights, this corresponds to removing edges with cosine distance above a certain threshold. The approach of Viegas et al. [38] is used to filter out weights, only keeping edges whose cosine distances are in the p-th percentile. Six values of p have been evaluated: $[2, 5, 10, 20, 50, 100]$.

The result of filtering out irrelevant similarities in the adjacency matrix A is a sparsely connected graph G (except for when $p = 100$, which maintains the fully connected graph).

3.1.3 Converting the Sparse Graph to Other Graph Types

At this point, the sparse graph G obtained by filtering out edges in the adjacency matrix A has no special properties except that all of its edges appear in the p-th percentile of cosine distances.

The divisive hierarchical clustering algorithm expects a Minimum Spanning Tree (MST) as input, which can be computed from the sparse graph. The GN-algorithm accepts any undirected graph, so it can be fed either the sparse graph G directly, its MST, or the Relative Neighbourhood Graph (RNG) (section 2.4) of G.

Clustering in the RNG of the sparse graph is an attractive idea because it is a simple representation that maintains a lot of topological information, yet can drastically reduce the number of edges compared to the sparse graph.

[1] https://radimrehurek.com/gensim/models/word2vec.html
[2] https://dumps.wikimedia.org/enwiki
[3] https://radimrehurek.com/gensim/models/phrases.html

This step of the algorithm converts the sparse graph into one of two graph types G_{type}: MST or RNG.

3.1.4 Apply Weighting Scheme

The GN-algorithm is applicable to both weighted and unweighted graphs. If it holds true that cosine similarities in a word embedding are not comparable across different regions, or even different words, then it could make sense to ignore weights once irrelevant similarities have been filtered out. This step of the algorithm applies one of two weighting schemes w_{scheme} to the weights of the graph produced by the previous step. Either it keeps all weights as they are, or it discards them, making the graph unweighted.

In fact, when the GN-algorithm is applied to the MST, it makes no difference if the graph is weighted or not, since the shortest path through an MST is independent of edge weights. Thus, a weighted MST is never used together with the GN-algorithm.

3.1.5 Graph Clustering

In this step, the graph output by the previous step of the algorithm is clustered. Both of the evaluated clustering algorithms, hierarchical and GN-clustering, produce a hierarchical clustering in a divisive fashion, starting out with all elements in the same cluster and eventually having one cluster per clustered element. Only the hierarchical process is of interest in this application, not presenting the final hierarchy. The clusters output by the word clustering algorithm are the clusters at some intermediate step of the divisive graph clustering process. To terminate at a good intermediate result, each cluster is evaluated against a stopping criterion before it is divided. The stopping criterion is

$$|C| \leqslant n, \tag{3.2}$$

where C is a cluster and n a constant. n represents a cluster sized deemed small enough that it can be evaluated bu a user without too much cognitive strain. Any cluster that does not meet the stopping criterion will be further divided, while clusters that meet the criterion will be kept intact. When all remaining clusters meet the stopping criterion the graph clustering algorithm is terminated. Four values of n have been evaluated: $[2, 4, 8, 20]$.

3.1.6 Application in the Gavagai Explorer

The clustering algorithm introduced in the previous sections takes a set of words and outputs a clustering of these words. In this section the algorithms application in the Gavagai Explorer is explained.

The Gavagai Explorer normally presents a list of topics ranked by their document frequency. Each topic is a bag-of-words, and its document frequency is calculated as the number of documents that contain any of the words in the bag-of-words. In this thesis, only the initial state of an exploration where topics consist of a single word, is considered. However, to more seamlessly fit with the framework of the Gavagai Explorer, the clustering can still be viewed as clustering bags-of-words of size one.

By clustering the bags-of-words, a second level of hierarchy is introduced: groupings. Words are still parts of topics, but topics can now be part of a group: a set of bags-of-words. The purpose of not merging all bags-of-words in a group into a single bag-of-words is to make it more obvious to a user what has been suggested by the clustering algorithm, and what words were already part of the same topic.

After grouping topics, the initial list of topics is re-arranged. Topics belonging to the same group are displayed together, ranked by the combined document frequency of the group, which is calculated as the number of documents that contain a word present in one of the topics in the group. Within the group, topics are ranked by their individual document frequency. Topics not belonging to a group are still ranked by their document frequency.

3.2 Unsupervised Evaluation

To gauge how different choices of hyperparameters affect the characteristics of the word clustering algorithm, an unsupervised evaluation was performed. The purpose of the unsupervised evaluation was to find a set of hyperparameters that reliably produce desirable cluster characteristics across multiple data sets analysed by users of the Gavagai Explorer. Three criteria for a good clustering were determined in the introduction.

First, a good clustering algorithm should produce topic clusters that are similar to those independently produced by a user. This criterion requires a silver standard in the form of a previously analysed project, and has thus not been analysed in the unsupervised evaluation.

Second, clustering the initial list of topics should make it easier to detect themes composed of low frequency terms.

Lastly, a user should effortlessly be able to determine if a cluster of topics is interesting enough to keep, or if it should be split up. On top of this, a user should not be overwhelmed by clusters they find inexplicable. Thus, clusters should be coherent, as measured by topic coherence (Equation 2.10).

Ideally, all three criteria should be met regardless of what data set is analysed, since users of the Gavagai Explorer analyse data from a wide range of domains. Because the first criterion requires a labeled gold standard to be evaluated, the possibility of evaluating it for a wide range of data sets is limited. However, the last two criteria can be measured in an unsupervised manner. This opens up the possibility of evaluating them across a wide range of unlabeled data sets, which are easier to obtain.

To estimate how well these criteria are met by different hyperparameter compositions, they have been evaluated on 110 data sets previously analysed by users of the Gavagai Explorer. Details on how the unsupervised evaluation has been performed are given in the next section, followed by a description of the metrics used to evaluate the two unsupervised criteria.

3.2.1 Unsupervised Evaluation Setup

Table 3.1 outlines the hyperparameter compositions that have been analysed in the unsupervised evaluation. The values in the cells along one row can be combined freely to create valid parameter compositions. All combinations of legal parameter choices have been evaluated, yielding a total of 192 evaluated compositions.

All hyperparameter compositions were evaluated on 110 data sets previously analysed by users of the Gavagai Explorer. Each data set consists of a varying number of texts of varying size and content. All texts were tokenized using the Lucene tokenizer[4], and stop words were removed. Collocations were identified by looking up bi-grams and tri-grams in the vocabulary of the Random Index embeddings. After combining collocations into single word tokens,

[4]https://lucene.apache.org/core/6_5_0/core/org/apache/lucene/analysis/Tokenizer.html

Table 3.1: Experiment configurations.

Clustering Algorithm, $G_{cluster}$	Graph Type, G_{type}	Weighting Scheme, w_{scheme}	Word Embedding, W_e	Max Cluster Size, n	Percentile, p
Hierarchical	MST	Weighted	Random Index, Skipgram	2, 4, 8, 20	2, 5, 10, 20, 50, 100
GN	MST	Unweighted	Random Index, Skipgram	2, 4, 8, 20	2, 5, 10, 20, 50, 100
GN	RNG	Weighted, Unweighted	Random Index, Skipgram	2, 4, 8, 20	2, 5, 10, 20, 50, 100

tokens that did not appear in both the Random Index vocabulary and the Skipgram vocabulary were removed. Lastly, the 100 terms with highest document frequency were selected for each data set. These 100 terms constitute the initial list of topics that would be suggested by the Gavagai Explorer when analyzing the data set, with the exception that some tokens not appearing in the Skipgram embeddings were removed.

The unsupervised evaluation of hyperparameters was done by clustering the 100 initial topics for each data set, using every hyperparameter composition. Then, the clusters were analysed quantitatively using the metrics introduced in Section 3.2.2.

3.2.2 Unsupervised Evaluation Metrics

Two criteria have been analysed in the unsupervised evaluation. This section introduces the metrics used to analyse them.

3.2.2.1 Criterion 1: Groups Should be Coherent

The clustering methods should suggest groups of topics that a user find coherent. Since topic coherence has been shown to correlate well with human judgement in multiple studies on topic models, it is used to measure the coherence of all words from the same group. Topic coherence (Equation 2.10) is measured using cosine similarity (Equation 2.4) in word embeddings as pairwise confirmation measure, as suggested by Alestras and Stevenson [1]. In all experiments, topic coherence was calculated using the same word embedding as that used for the parameter W_e. For each data set, the average topic coherence of non-singular clusters was measured.

The easiest strategy to maximize topic coherence is to minimize the size of groups. Since this is not desirable behaviour, two metrics concerning the size of groups have been analysed. First, the number of words not put in any group was counted. Second, the average size of clusters was measured, ignoring clusters consisting of a single word.

3.2.2.2 Criterion 2: Themes Comprised of Low Frequency Topics Should be Easier to Detect

The Gavagai Explorer presents suggested topics to a user in a list ranked by the topic's document frequency. One of the goals of clustering topics is to discover themes formed by a diverse set of key words. A theme with even a single high frequency term could potentially be discovered by expanding that initial term. However, if all terms in a theme have a document frequency that puts them far down the list of topics it is likely that the theme will not be discovered.

A simple metric was introduced to evaluate if grouping helps alleviate this problem. The metric is simply the count of groups that appear in the list of top 30 topics, that consist of no topics initially in the top 30. Initially, the list of top 30 topics simply contains the 30 topics with highest document frequency. After grouping, the top 30 still contains 30 topics, but the topics are ordered first based on the combined document frequency of their group, and second on their individual document frequencies. This metric gives a clear answer to the question if completely novel themes are now visible among the top 30 topics.

3.2.3 Evaluation Strategy

To understand how to select a good set of hyperparameters based on the metrics detailed above, first, an analysis of how the metrics correlate with each other was performed. For each pair of metrics, the Pearson correlation was calculated. This was done to understand what trade-offs to expect when optimizing for one metric.

After analyzing correlations among the metrics, an analysis was performed on how different hyperparameters and their compositions affected the metrics. The spread of average

cluster size, average coherence and number of single word clusters for all data sets were analysed using boxplots. The number of new clusters in the top 30 for a data set takes a discrete value, and in the experiments the range was very limited. For this reason, the quartiles of most experiments were very similar, so averages were compared instead.

Since half of the experiments measured coherence using Random Index vectors, and half using Skipgram vectors, coherence is not comparable across those experiments. Thus, much of the analysis had to be performed by considering these two sets of experiments separately.

First, both clustering algorithms $G_{cluster}$, combined with both word embeddings W_e, were analysed separately. The hierarchical clustering algorithm was only evaluated in conjunction with three other variable parameters: the word embedding W_e, maximum cluster size n and percentile p. For both word embeddings, a set of promising values n and p were selected. The GN-algorithm was evaluated on two different graph types G_{type} and two weighting schemes W_{scheme}. Before analyzing the impact of the n and p parameters, a promising choice of G_{type} and W_{scheme} was determined. Then, for both word embeddings, a set of promising values n and p were selected.

After selecting just a few promising hyperparamater compositions for each clustering algorithm $G_{cluster}$ and word embedding W_e, these compositions were compared to each other. One parameter composition per word embedding were selected as being the most promising, based on the balance they struck between the unsupervised metrics. These two composition were evaluated more thoroughly by more closely inspecting how they scored on the four unsupervised metrics across the 110 data sets. This was done by analysing the quartiles, as well as the 5th and 95th percentiles of all their scores.

3.3 Supervised Evaluation Scheme

The unsupervised evaluation enabled two out of three success criteria to be analysed across 110 data sets to detect promising hyperparameter compositions. This section describes the supervised evaluation strategy used to determine if clusters produced by the algorithm overlap with what users of the Gavagai Explorer have created manually.

3.3.1 Alignment with Silver Standard

Though there is no gold standard for a proper text analysis in the Gavagai Explorer, there are a lot of data sets that have already been analysed. A previously completed exploration of a data set might not have perfect coverage of all potentially interesting topics, but it indicates that the topics in it are interesting and meaningful to a user. It can be thought of as a silver standard, which can be used to evaluate a clustering algorithm in the same way it would be done with a gold standard.

Three such previously analysed data sets were converted to silver standards by sampling their top 60 topics. All words part of these topics were to be clustered, with the topics as targets. Since not all words recognized by the Gavagai Explorer appear in the Skipgram embeddings, some words needed to be removed from the silver standard.

All hyperparameter compositions from the unsupervised evaluation have been evaluated by clustering these three data sets and computing precision and recall. The two most promising hyperparameter compositions from the unsupervised evaluation were compared to the results of the other compositions.

The three data sets that were used for the supervised evaluation are all data sets analysed by one of Gavagai's analysts for the purpose of demonstrating their system to customers. The first is a data set of 130 reviews for a San Fransisco hotel from Tripadvisor. The second is a set of 34,563 Airbnb reviews from Trustpilot. The third consists of 29,072 reviews of airlines published on Airline Equity.

Though all three data sets consist of the words from the top 60 topics, not all such topics have been intentionally created. Some of the topics were intentionally created by the analyst

through merging or expanding terms, but the top 60 also contains many single word topics suggested because of their document frequency. The number of single word clusters vary for all three data sets, as does the average and maximum topic sizes. Table 3.2 outlines these differences.

Table 3.2: Statistics of the silver standard data sets.

Data Set	Words Clustered	Single Word Clusters	Average Cluster Size	Maximum Cluster Size
Hotel	125	31	4.09	9
Airbnb	78	46	3.20	6
Airlines	193	39	10.27	26

4 Results

This chapter presents the results for the unsupervised and supervised evaluations.

4.1 Unsupervised Evaluation

192 hyperparameter compositions from Table 3.1 have been evaluated on 110 data sets previously analysed by users of the Gavagai Explorer. The resulting clusters and ranked list of topics have been evaluated using the metrics from section 3.2.2. This section presents how the different hyperparameters impacted these metrics, as well as how the metrics correlate with each other.

The whiskers in all boxplots in this section are limited to show values within a factor of 1.5 of the inner quartile range (IQR). The error bars of all barplots denote the 95% confidence intervals, estimated using 1,000 bootstrap iterations.

4.1.1 Metric Correlation

Figure 4.1 shows the boxplots for the metrics *Average Cluster Size*, *Single Word Clusters* and *Average Coherence*, as well as the barplot for *New Clusters in Top 30*, from all unsupervised experiments. The number of single word clusters and the average size of cluster are very similarly distributed regardless of the choice of word embedding. On the other hand, average coherence is not at all comparable for the two different word embeddings. On average there is about one new cluster in the top 30 per four experiments for both word embeddings, though the average with Skipgram embeddings is slightly higher than that with Random Index embeddings.

Table 4.1 and 4.2 present the pearson correlation of the metrics *Average Cluster Size*, *Single Word Clusters*, *Average Coherence* and *New Clusters in Top 30*. Table 4.1 only takes into account measurements from experiments where Random Index vectors were used for the word embedding parameter W_e. Table 4.2 instead presents results from the experiments that used Skipgram embeddings. Correlations between metrics are very similar for both types of word embeddings. The correlation between new clusters in the top 30, number of single word clusters and average coherence vary by at most 0.01 between the two embeddings. Single word clusters and average cluster size are also very similarly correlated for the two embeddings, with values of -0.50 and -0.46 respectively. Correlation between average coherence and av-

erage cluster sizes varies the most, with values of -0.59 for Random Indexing and -0.67 for Skipgram. The largest correlation in both embeddings is between average coherence and the number of single word clusters, 0.84 for Random Index and 0.87 for Skipgram embeddings. Correlation between the number of new clusters in top 30 and all other metrics is relatively weak for both word embedding types. Its largest correlation coefficient is 0.40 with the number of single word clusters, for both embedding types.

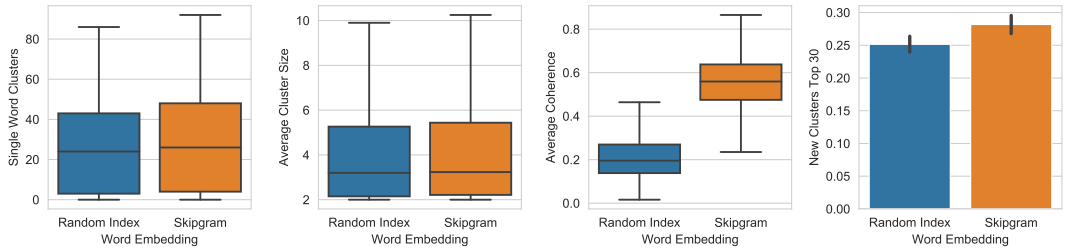


Figure 4.1: Boxplot of number of single word clusters, average cluster size and average coherence, and barplot of average number of new clusters in the top 30, for all unsupervised experiments.

Table 4.1: Pearson correlation of metrics in unsupervised experiments when using Random Index embeddings.

	Average Cluster Size	Single Word Clusters	Average Coherence	New Clusters Top 30
Average Cluster Size	1.00	-0.50	-0.59	-0.25
Single Word Clusters	-0.50	1.00	0.88	0.40
Average Coherence	-0.59	0.88	1.00	0.36
New Clusters Top 30	-0.25	0.40	0.36	1.00

Table 4.2: Pearson correlation of metrics in unsupervised experiments when using Skipgram embeddings.

	Average Cluster Size	Single Word Clusters	Average Coherence	New Clusters Top 30
Average Cluster Size	1.00	-0.46	-0.67	-0.26
Single Word Clusters	-0.46	1.00	0.87	0.40
Average Coherence	-0.67	0.87	1.00	0.37
New Clusters Top 30	-0.26	0.40	0.37	1.00

4.1.2 Hyperparameter Evaluation

This section presents a series of measurements from the unsupervised evaluation. First, the hierarchical clustering algorithm and the GN-algorithm are presented individually. Then, the results for the most promising parameter compositions are presented, followed by more detailed results for the hyperparameter composition determined to be the most promising.

4.1.2.1 Hierarchical Clustering

Figure 4.2 and 4.3 show experimental results for different values of max cluster size, n, and percentile, p, for the hierarchical clustering algorithm and Random Index word embeddings. Figure 4.4 and 4.5 show equivalent graphs but with Skipgram word embeddings.

Figure 4.2a, and 4.4a both show the same relationship between the number of single word clusters and the parameters max cluster size n and percentile p. With a larger value of n, the number of single word clusters decrease substantially. Increasing p from 2 to 5 also results in a notable decrease in the number of single word clusters, regardless of the choice of n. A similar, but slightly smaller effect is seen when increasing p from 5 to 10. However, further increasing p gives a much smaller effect.

Figure 4.2b and 4.4b show the opposite relation. For all values of $n > 2$, an increase of p from 2 to 5 results in a slight increase in average cluster size, but further increments have little effect. Given the nature of the average size metric, all clusters with more than one word are exactly of size two when $n = 2$.

Average coherence behaves almost identically to the number of single word clusters when changing the values of n and p, as seen when comparing Figure 4.2a with 4.2c and Figure 4.4a with 4.4c.

Both Figure 4.3 and 4.5 both show the same relation between n, p and the number of new clusters in top 30. A lower value of n gives a larger number of new clusters i in the top 30. The bottom of the 95% confidence interval for any experiment with $n \in [2, 4]$ does not overlap more than barely with the top of the 95% confidence interval for larger choices of n. Also, a lower value of p increases the metric, though not as significantly as n. For all experiments with $n = 2$, the average number of new clusters in top 30 is larger than 1 for both word embeddings. With $n = 4$ and $p = 2$, the Random Index embeddings yield about 0.75 new clusters per experiment, and Skipgram a little bit more, with 1 within its 95% confidence interval. Further increasing p with $n = 4$ drops the number of new clusters in the top 30 close to 0.5 for both embeddings.

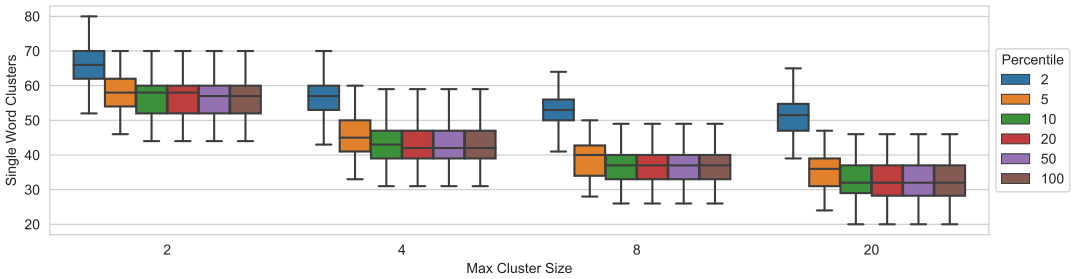
Three compositions thought to yield a good trade-of between the four metrics were selected for further analysis: $(n = 2, p = 5)$, $(n = 4, p = 2)$ and $(n = 4, p = 5)$, all combined with both word embeddings. This choice is further discussed and motivated in section 5.1.1.2

4.1.2.2 GN-Algorithm

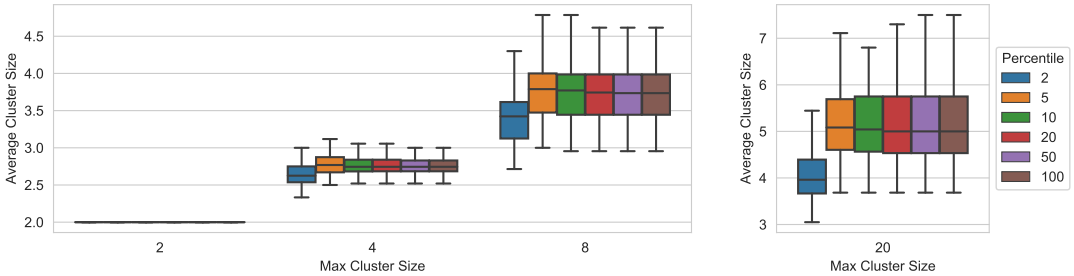
Evaluating G_{type} and W_{scheme} Figure 4.6 shows experimental results for the GN-algorithm in combination with both word embeddings W_e, both graph types G_{type} and both weighting schemes W_{scheme}. Max cluster size n and percentile p are not fixed, so all evaluated values contribute to the plot. The results are similar for both Random Index and Skipgram embeddings. For both word embeddings, the number of single word clusters are very similar for both the unweighted MST, and the two RNGs. The median of average cluster size is very similar for all three compositions in both Figure 4.6a and 4.6b, but the third quartile and the spread reaches slightly higher for the MST in both experiments. Average coherence is also slightly larger for the MST in both experiments. The only metric where the two word embeddings show slightly different results is the number of new clusters in the top 30. There, the MST gets slightly larger results for Random Index embeddings, but both the weighted and unweighted RNG slightly outperforms it with Skipgram embeddings.

Given the otherwise very small differences between the three choices of graph type and weight scheme, the choice of what composition to analyse further came down to the complexity of the graph types. The MST is far more widely known than the RNG, and has a lower time complexity. Thus, it was deemed that in the absence of results clearly favoring one graph type above the other, the simpler MST should be used.

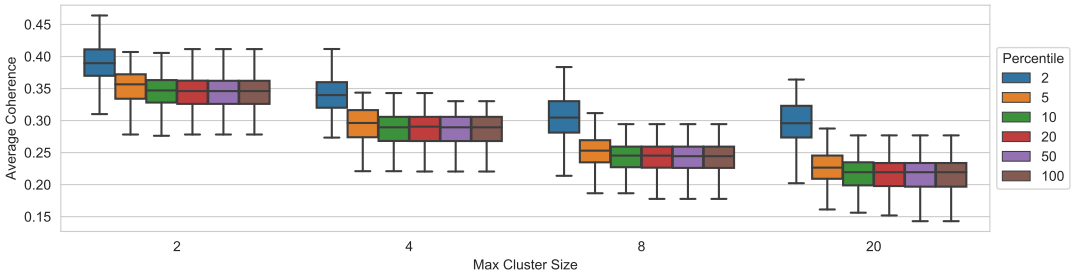
Evaluating n and p The experimental results from analyzing different values of n and p with the GN algorithm and the unweighted MST (Figure 4.7, 4.8, 4.9, 4.10) are similar to those from the experiments with the hierarchical clustering algorithm presented in section 4.1.2.1. Many of the relationships identified through those experiments also hold for the GN-algorithm, with some slight differences.



(a) Number of Single Word Clusters



(b) Average Cluster Size



(c) Average Coherence

Figure 4.2: Boxplots of a) number of single word clusters, b) average cluster size and c) average coherence, for the hierarchical clustering algorithm using Random Index word embeddings.

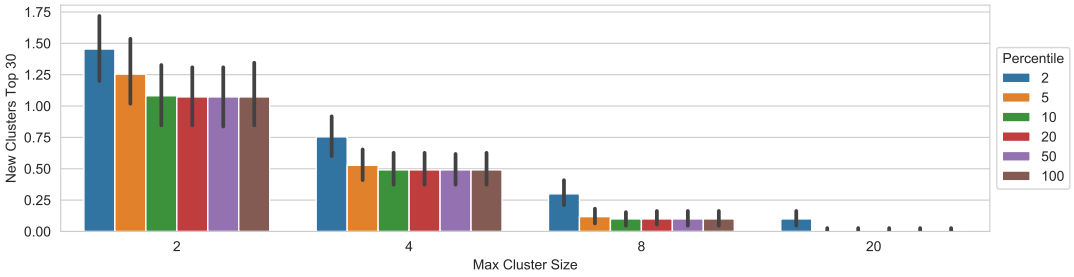
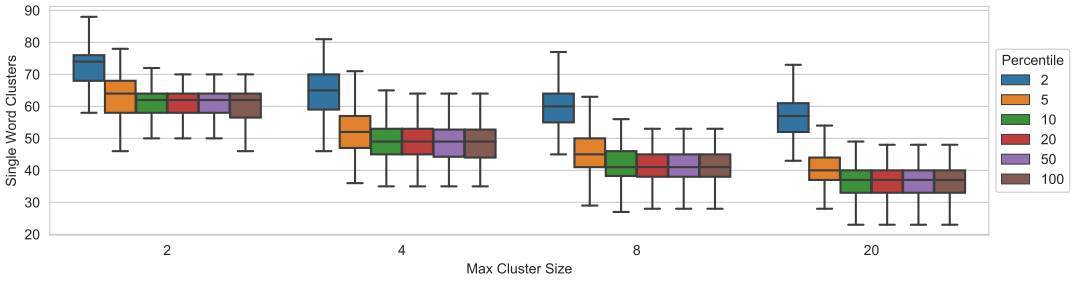
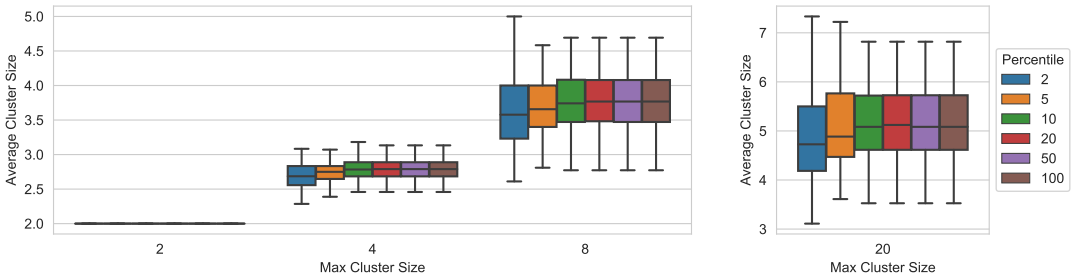


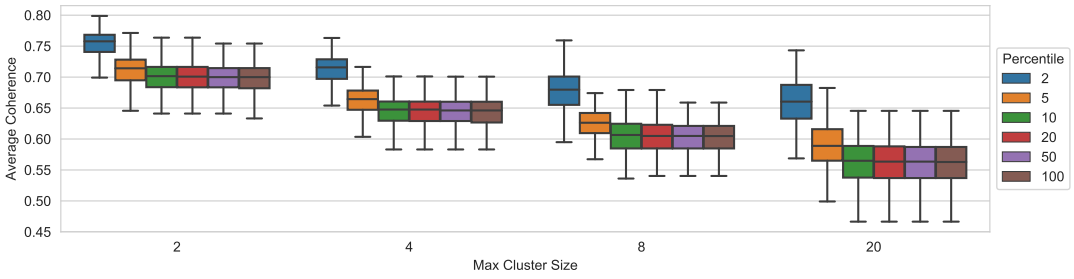
Figure 4.3: Average number of new clusters in top 30 for the hierarchical clustering algorithm using Random Index word embeddings.



(a) Number of Single Word Clusters



(b) Average Cluster Size



(c) Average Coherence

Figure 4.4: Boxplots of a) number of single word clusters, b) average cluster size and c) average coherence, for the hierarchical clustering algorithm using Skipgram word embeddings.

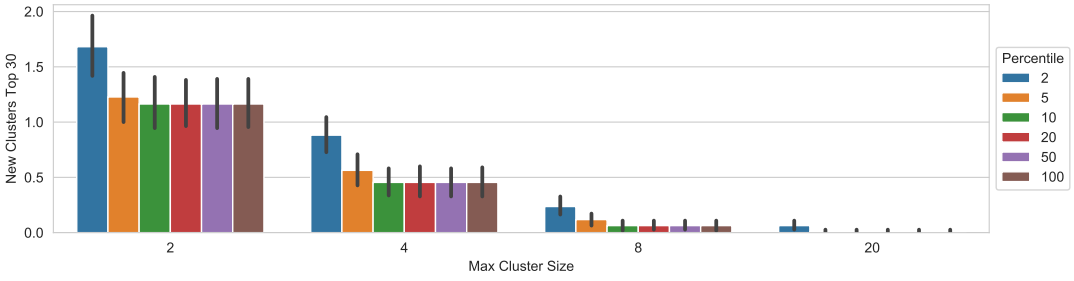
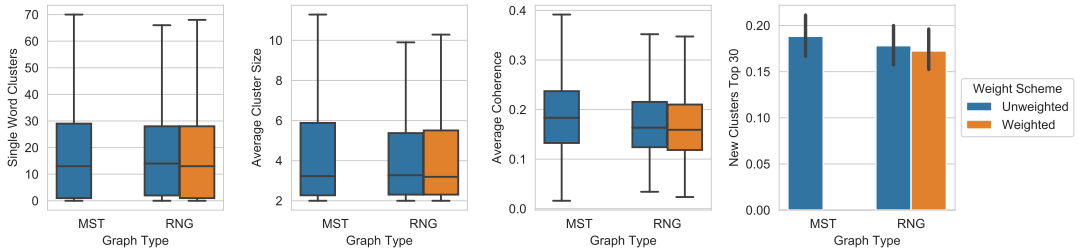
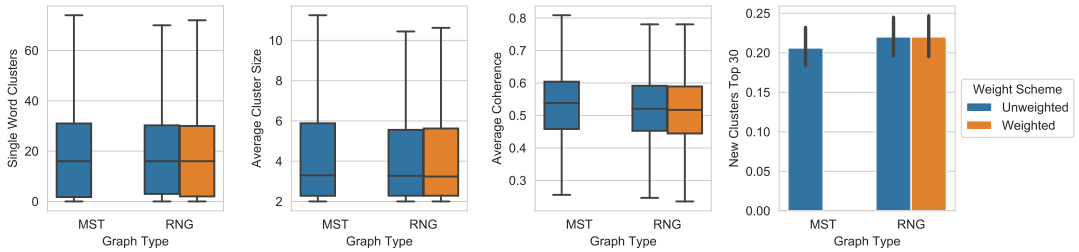


Figure 4.5: Average number of new clusters in top 30 for the hierarchical clustering algorithm using Skipgram word embeddings.



(a) Random Index



(b) Skipgram

Figure 4.6: Boxplot of number of single word clusters, average cluster size and average coherence, and barplot of average number of new clusters in the top 30 for different graph types in combination with the GN-algorithm using a) Random Index and b) Skipgram word embeddings.

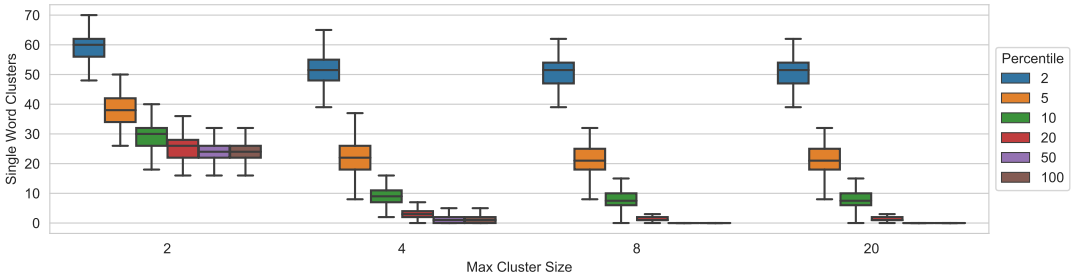
As seen in Figure 4.7a and 4.9a, increasing n still leads to a significant reduction in the number of single word clusters. However, the effect of p is much more pronounced for the GN algorithm, leading to a substantial reduction in the number of single word clusters with each increase of p, except for the increase from 50 to 100. As an indication of how much more pronounced the difference in increasing p is, consider the following example. For $n \in [2, 4]$, the upper spread of the experiments with $p = 5$ largely overlap with the third quartile of the corresponding experiment with $p = 2$ when using hierarchical clustering. However, when using the GN-algorithm, the top of the spread with $p = 5$ barely overlaps with the bottom of the spread of $p = 2$ in the corresponding experiments.

The same increased sensitivity to changes in p is seen for the average cluster size as well (Figure 4.7b, 4.9b), where increasing p results in larger cluster sizes for all values $n > 2$, but especially for $n > 4$.

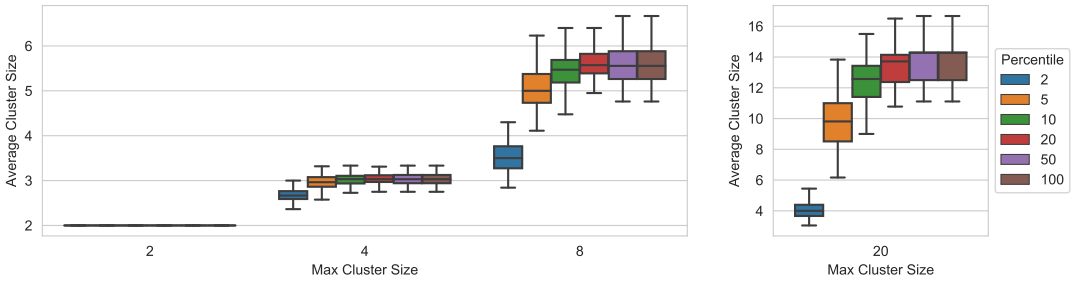
Just as was the case for hierarchical clustering, the average coherence closely follows the number of single word clusters. When increasing either n or p, the average coherence decreases. Yet, the decrease in coherence with larger values of p is not as pronounced as it is for the number of single word clusters. For average coherence, there is no single increase of p that leads to a spread that does not overlap with the spread of the lower value of p, for the same value of n.

The only large difference between the experiments with Random Index and Skipgram embeddings is the behaviour of the number of new clusters in the top 30 with increasing values of n and p. Figure 4.8 shows that the average number of new clusters drastically decreases for any values of $n > 2$ or $p > 2$ when combined with Random Index embeddings. The average starts out around 1.2, but drops to 0.4 for $(n = 2, p = 5)$ and 0.7 for $(n = 4, p = 2)$. On the other hand, Skipgram embeddings (Figure 4.10) shows a behaviour much more similar to that seen for the hierarchical clustering algorithm, where an increase in p only slightly reduced the number of new clusters, and the largest effect coming from increasing n.

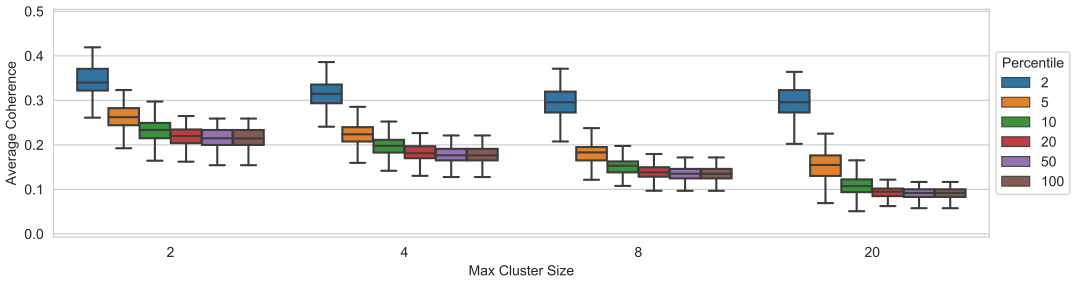
Given the similarities in behaviour with the hierarchical clustering, the same three compositions were selected for further analysis: $(n = 2, p = 5)$, $(n = 4, p = 2)$ and $(n = 4, p = 5)$.



(a) Number of Single Word Clusters



(b) Average Cluster Size



(c) Average Coherence

Figure 4.7: Boxplots of a) number of single word clusters, b) average cluster size and c) average coherence, for the GN-algorithm, applied to an unweighted MST constructed from Random Index word embeddings.

Again, they were all combined with both word embeddings. A deeper discussion on the choice is presented in section 5.1.1.3.

4.1.2.3 Best Performing Hyperparameter Compositions

Comparing the top 12 candidates For both clustering algorithms $G_{cluster}$ and word embeddings W_e a small set of potent hyperparameters were chosen for further analysis. Table 4.3 lists all of the resulting hyperparameter compositions. The performance of all of these compositions are shown in Figure 4.11 and 4.12.

The compositions that got the very lowest number of single word clusters were the GN-algorithm in combination with $n = 4$ and $p = 5$, in combination with both word embeddings. Both of these compositions also achieve the lowest coherence in their respective word embedding, and the fewest new clusters in top 30.

When combined with the GN-algorithm, $(n = 2, p = 5)$ yields fewer single word clusters compared to $(n = 4, p = 2)$, but has lower coherence and fewer new clusters. In combination with the hierarchical clustering algorithm it yields slightly larger coherence compared to $(n = 4, p = 2)$ and also slightly more new clusters in the top 30.

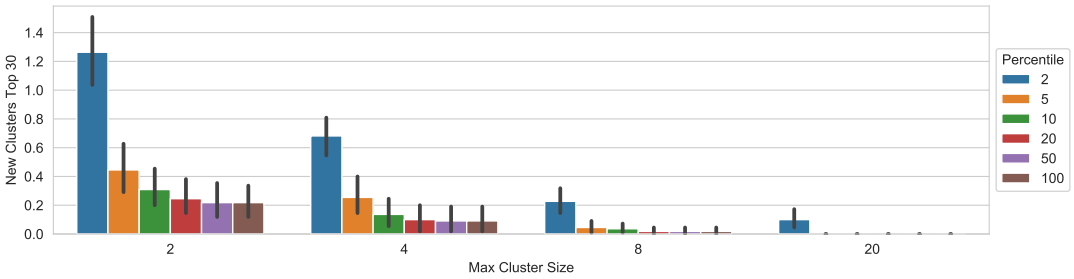
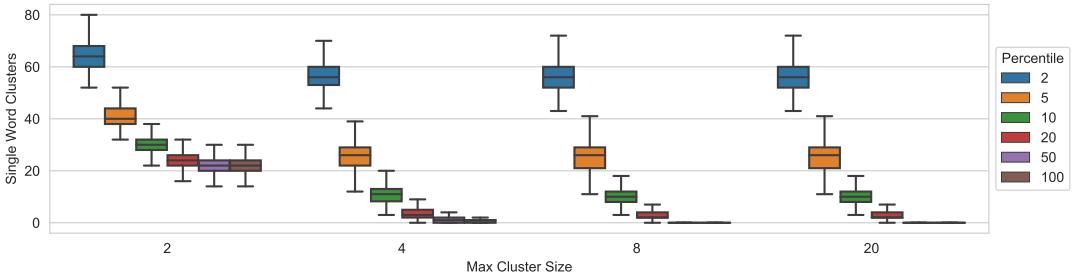
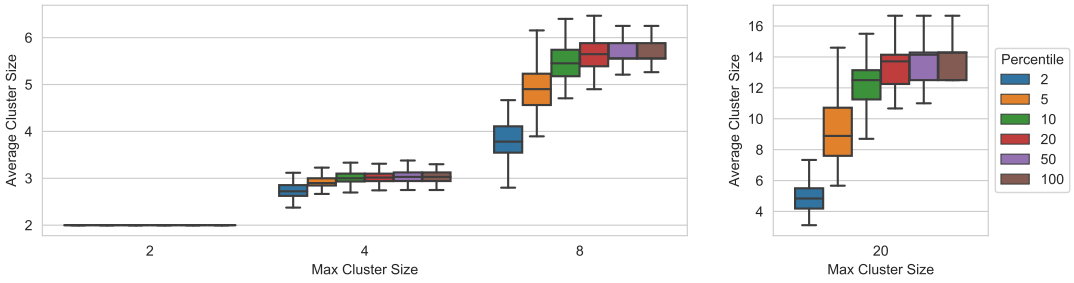


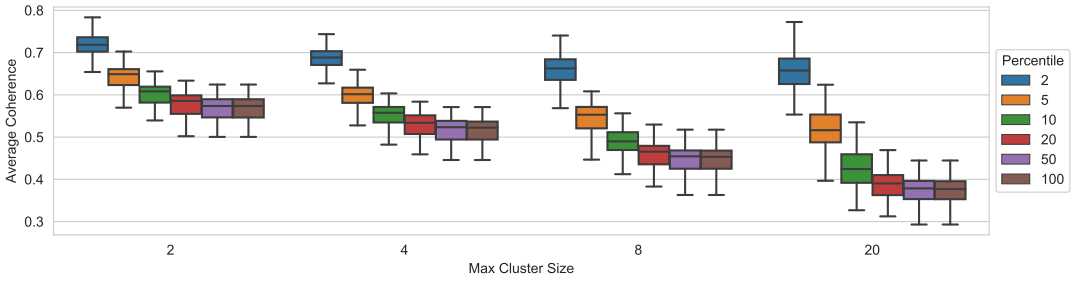
Figure 4.8: Average number of new clusters in top 30 for the GN-algorithm applied to an unweighted MST constructed from Skipgram word embeddings.



(a) Number of Single Word Clusters



(b) Average Cluster Size



(c) Average Coherence

Figure 4.9: Boxplots of a) number of single word clusters, b) average cluster size and c) average coherence, for the GN-algorithm, applied to an unweighted MST constructed from Skipgram word embeddings.

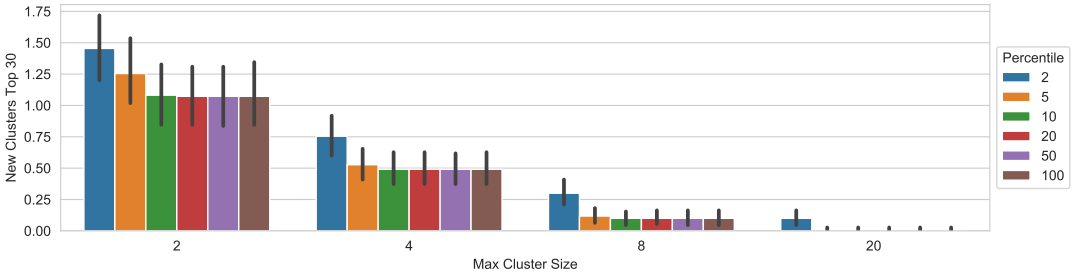


Figure 4.10: Average number of new clusters in top 30 for the GN-algorithm applied to an unweighted MST constructed from Skipgram word embeddings.

Setting $(n = 4, p = 2)$ with hierarchical clustering yields the highest median coherence of all compositions used with Skipgram. With Random Indexing, it yields slightly lower coherence than when setting $n = 2$. In comparison, setting $(n = 4, p = 5)$ yields slightly larger cluster sizes, and somewhat fewer single word clusters. However, coherence drops so far that the third quartile is lower than the first when comparing to $(n = 4, p = 2)$, for both word embeddings.

For both $(G_{cluster} = hierarchical)$ and $(G_{cluster} = GN)$ combined with $(n = 4, p = 2)$, Skipgram embeddings get slightly more new clusters in the top 30, and larger cluster sizes compared to when using Random Index embeddings. However, both compositions also get more single word clusters.

When paired with Random Index embeddings, $(G_{cluster} = GN, n = 4, p = 2)$ resulted in slightly fewer single word clusters, similar cluster sizes, but lower coherence and fewer new clusters in the top 30 compared to $(G_{cluster} = hierarchical, n = 4, p = 2)$. Similar results are seen for the Skipgram embedding as for the Random Index embeddings, but there the GN-algorithm performed very similarly to the hierarchical clustering in term of new words in the top 30.

$(G_{cluster} = hierarchical, n = 4, p = 2)$ was selected as the best performing parameter composition as it achieved high average coherence for both embeddings, while still allowing clusters larger than size 2 and yielding balanced results in terms of single word clusters and new clusters in the top 30. A more thorough motivation and discussion of this choice is presented in section 5.1.1.4.

Inspecting the best parameter composition Table 4.4 shows that with the composition $(G_{cluster} = hierarchical, n = 4, p = 2)$ combined with Random Index embeddings, 50% of all data sets got an average cluster size between 2.54 and 2.75, between 53 and 60 single word clusters, an average coherence between 0.32 and 0.36 and between 0 and 1 new clusters in the top 30. Inspecting the 5th and 95th percentile shows that 90% of all data sets got an average cluster size between 2.43 and 3.00, between 48.00 and 67.65 single word clusters, an average coherence between 0.25 and 0.39 and between 0 and 2 new clusters in the top 30. The 5th percentile of average coherence being 0.25 represents a drop of coherence compared to the median which is more than twice as large as the IQR. It also represent an average coherence of 74% of the median.

Table 4.5 shows that with the composition $(G_{cluster} = hierarchical, n = 4, p = 2)$ combined with Skipgram embeddings, 50% of all data sets got an average cluster size between 2.56 and 2.83, between 59 and 70 single word clusters, an average coherence between 0.70 and 0.73 and between 0 and 1 new clusters in the top 30. Inspecting the 5th and 95th percentile shows that 90% of all data sets got an average cluster size between 2.39 and 3.00, between 53.00 and 78.10 single word clusters, an average coherence between 0.65 and 0.75 and between 0 and 2 new clusters in the top 30. Just as with Random Index embeddings, the 5th percentile of

Table 4.3: Best performing hyperparameter compositions.

Clustering Algorithm, $G_{cluster}$	Graph Type, G_{type}	Weighting Scheme, w_{scheme}	Word Embedding, W_e	Max Cluster Size, n	Percentile, p
Hierarchical	MST	Weighted	RI	2	5
				4	2
				4	5
			Skipgram	2	5
				4	2
				4	5
GN	MST	Unweighted	RI	2	5
				4	2
				4	5
			Skipgram	2	5
				4	2
				4	5

Table 4.4: Select percentiles of the four unsupervised metrics for the best performing parameter composition ($G_{cluster} = hierarchical, n = 4, p = 2$), with Random Index word embeddings.

	5%	Q1 (25%)	Q2 (50%)	Q3 (75%)	95%
Average Cluster Size	2.43	2.54	2.62	2.75	3.00
Single Word Clusters	48.00	53.00	57.00	60.00	67.65
Average Coherence	0.25	0.32	0.34	0.36	0.39
New Clusters Top 30	0	0	1	1	2

Table 4.5: Select percentiles of the four unsupervised metrics for the best performing parameter composition ($G_{cluster} = hierarchical, n = 4, p = 2$), with Skipgram word embeddings.

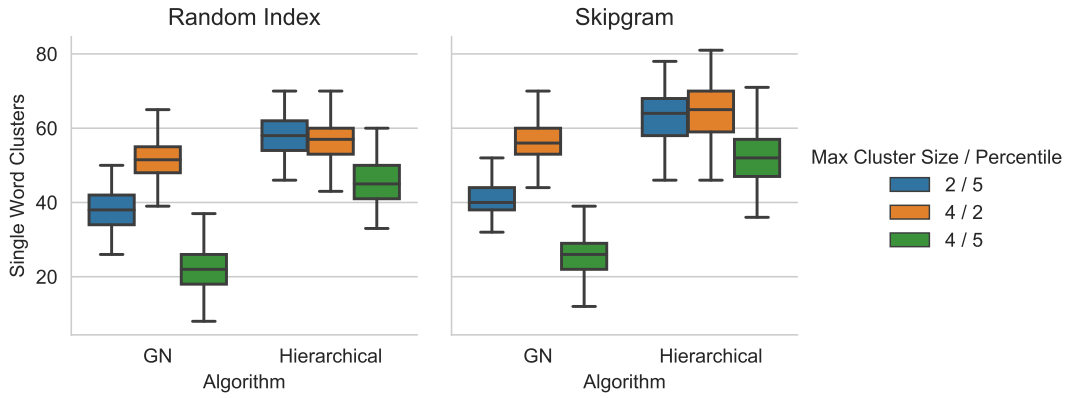
	5%	Q1 (25%)	Q2 (50%)	Q3 (75%)	95%
Average Cluster Size	2.39	2.56	2.69	2.83	3.00
Single Word Clusters	53.00	59.00	65.00	70.00	78.10
Average Coherence	0.65	0.70	0.72	0.73	0.75
New Clusters Top 30	0	0	1	1	2

average coherence is more than twice the IQR lower than the median. Direct comparison to the median shows that an average coherence of 0.65 is 90% of the median average coherence.

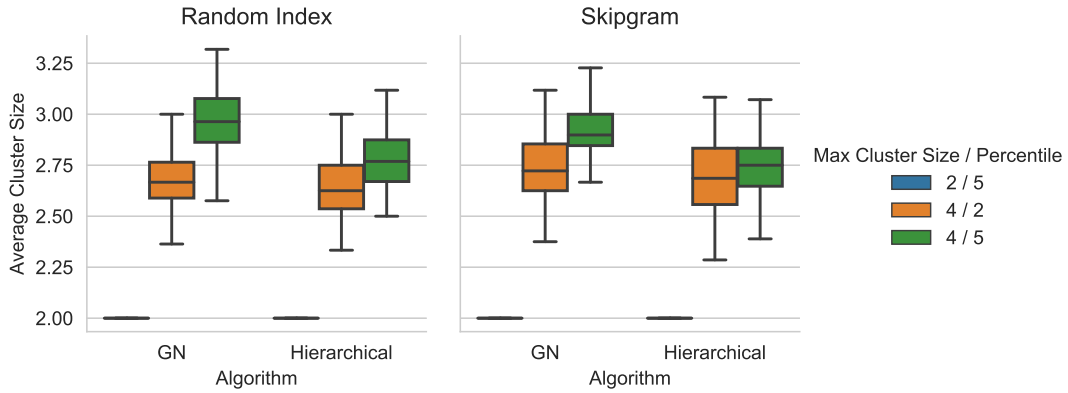
4.2 Supervised Evaluation

Figure 4.13 shows the precision and recall achieved by all hyperparameter compositions from Table 3.1 on all three data sets presented in section 3.3.1. The parameter composition selected for further analysis in section 4.1.2.3, ($G_{cluster} = hierarchical, n = 4, p = 2$) combined with both Random Index and Skipgram embeddings, have individual markers.

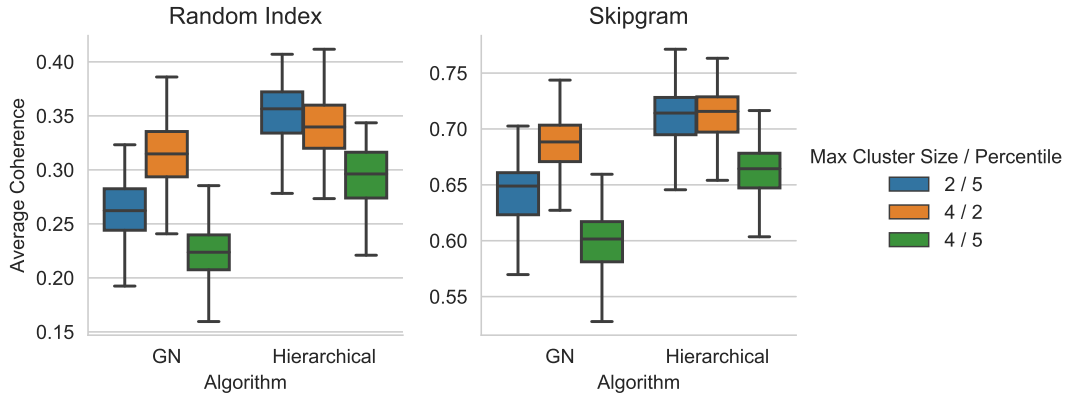
The maximum precision and recall achieved on the three data sets vary, with the highest precision scores happening for the hotel and airline reviews, and recall being highest for the Airbnb reviews. On all three data sets, the two selected parameter compositions perform very similarly in terms of both precision and recall. Also, for all three data sets they are among the top performers in terms of precision. For the two data sets with lower average cluster sizes, Hotel and Airbnb (see Table 3.2), there are few or no compositions that achieve higher precision than the selected compositions without paying for it in lower recall. However, on



(a) Number of Single Word Clusters



(b) Average Cluster Size



(c) Average Coherence

Figure 4.11: Boxplots of a) number of single word clusters, b) average cluster size and c) average coherence, for the best performing parameter compositions presented in Table 4.3.

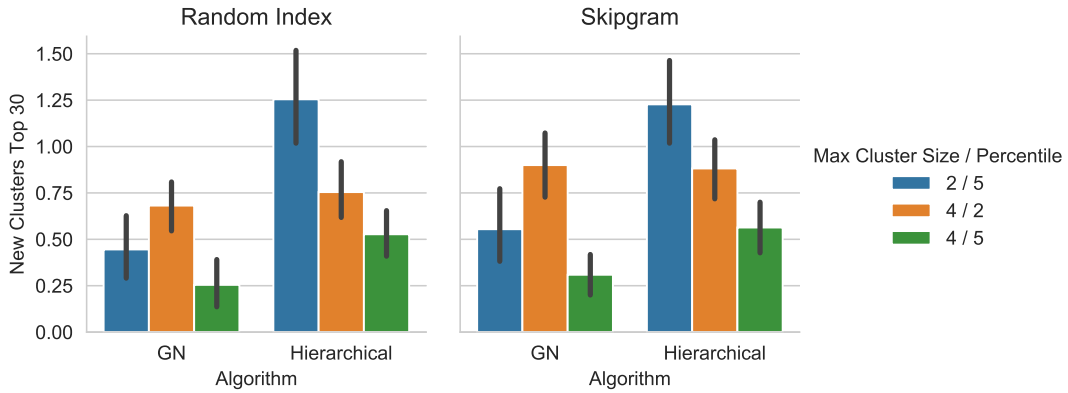


Figure 4.12: Average number of new clusters in top 30 for the best performing parameter compositions presented in Table 4.3.

the Airlines data set there are a few compositions that perform at similar precision but far higher recall.

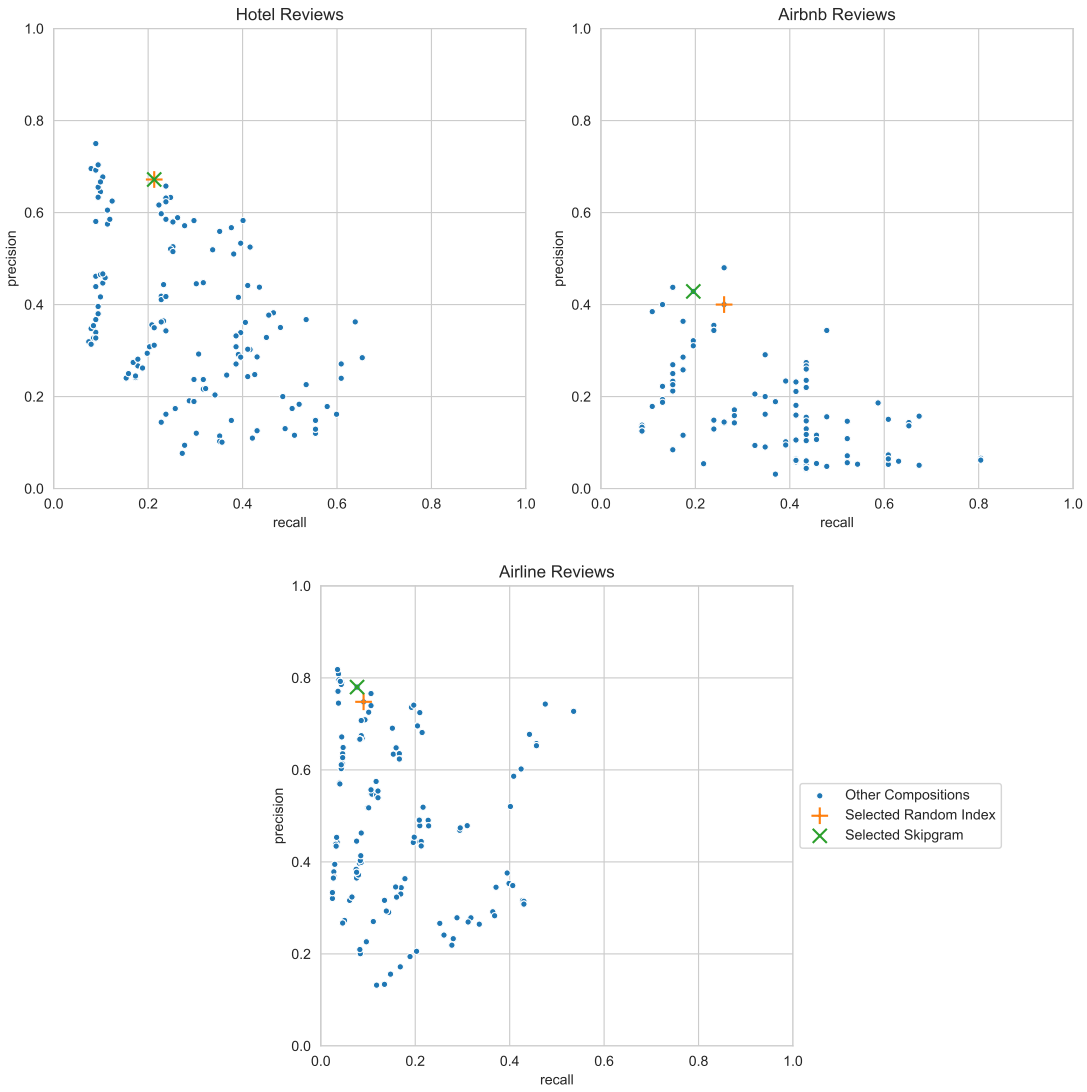


Figure 4.13: Precision and recall for all evaluated hyperparameter compositions on the three data sets. The compositions $(G_{cluster} = hierarchical, n = 4, p = 2)$ combined with Random Index and Skipgram embeddings are marked separately.

5 Discussion

This chapter is dedicated to discussing the results and methods of the thesis.

5.1 Results

In this section the thesis results presented in chapter 4 are discussed. Also, some example clusters produced by the word clustering algorithm are presented and discussed to give a more concrete idea of what the algorithm accomplishes.

5.1.1 Unsupervised Evaluation

Analysing the results of the unsupervised evaluation gives some intuition on how the investigated clustering algorithm behaves with different hyperparameter compositions. In this section the results are discussed, and motivations are provided for the different choices of parameters that were analysed more thoroughly.

5.1.1.1 Metric Correlations

Section 4.1.1 presents the distribution of the four unsupervised metrics for all experiments, as well as the Pearson correlation among the metrics. The fact that average coherence is not comparable for the two word embeddings was expected, as discussed in section 3.2.3. Perhaps more surprising is the fact that the choice of embedding seems to have a very small impact on the number of single word clusters and the average cluster size. It is also interesting to note how low the average number of new clusters in the top 30 is, with only about one new cluster per four experiments.

The clear negative correlation between average coherence and average cluster size was expected, as discussed in section 3.2.2.1. Though, even more notable was the large positive correlation between average coherence and the number of single word clusters.

5.1.1.2 Hierarchical Clustering

It is very interesting that the hierarchical clustering algorithm behaves so similarly for both Random Index and Skipgram embeddings, as seen in section 4.1.2.1.

Section 4.1.1 showed that there was a large negative correlation between average cluster size and number of single word clusters in both embeddings, and an even larger positive correlation between average coherence and the number of single word clusters. Thus, it is not surprising that changing n and p has a very similar effect on both the number of single word clusters and average coherence, and the opposite effect on average cluster size. Perhaps the fact that average size is exactly two for many experiments has some interesting results on its correlation coefficients with the other metrics.

Three combinations of n and p were selected for further analysis in section 4.1.2.3: $(n = 2, p = 5)$, $(n = 4, p = 2)$ and $(n = 4, p = 5)$. Since the hyperparameters n and p had very similar relationships with the unsupervised metrics for both embedding types, the same values were deemed to have the largest potential for both. Choosing low values of n and p is desirable to get high average coherence and more new clusters in the top 30. $n = 2$ has the desirable characteristic of yielding high average coherence and the largest number of new clusters in top 30. However, with $p = 2$, it also stands out as having far more single word clusters than any other composition, for both Random Index and Skipgram embeddings. Therefore, the composition $(n = 2, p = 5)$ was chosen to strike a balance between higher average coherence and not having to many single word clusters. Choosing $n = 4$ with $p = 2$ yields similar average coherence and a similar number of single word clusters as $(n = 2, p = 5)$, though it decreases the number of new clusters in the top 30. However, it comes with the benefit of increasing the average cluster size from 2 to almost 3. The last chosen composition, $(n = 4, p = 5)$ further sacrifices some coherence and new clusters in the top 30 to achieve fewer single word clusters. Any further sacrifices of coherence were deemed too large to consider.

5.1.1.3 GN-Algorithm

Evaluating G_{type} and W_{scheme} Section 4.1.2.2 shows that the choice of G_{type} and W_{scheme} has little impact on the unsupervised metrics. The largest difference is in average coherence, which is slightly larger with $G_{type} = MST$ compared to $G_{type} = RNG$.

The choice to select $G_{type} = MST$ and $W_{scheme} = unweighted$ was mainly motivated by the absence of clear advantages for the RNG, and the fact that the MST is both more well known, and can be computed with a lower asymptotic time complexity. Using a simpler and more well known representation has the advantage of making the algorithm more easily understandable. An improved time complexity also has the advantage of making the algorithm scale better if more words need to be clustered. However, since the algorithm is intended to be used on relatively small set of N words ($N \approx 100$), the cubic time complexity of computing the RNG is not particularly restrictive.

Evaluating n and p Section 4.1.2.2 showed that many of the relationships between n, p and the four unsupervised metrics were similar for the GN-algorithm and the hierarchical algorithm (section 4.1.2.1). The arguments for choosing values of n and p to use with the GN-algorithm were very similar to those used for the hierarchical clustering algorithm, and the same three compositions were chosen: $(n = 2, p = 5)$, $(n = 4, p = 2)$ and $(n = 4, p = 5)$.

Just like for hierarchical clustering, $(n = 2, p = 2)$ was deemed to yield too many single word clusters, which motivates the use of $(n = 2, p = 5)$ to strike a balance between coherence and the number of single word clusters. In the hierarchical experiments $(n = 4, p = 2)$ yielded similar results as $(n = 2, p = 5)$ in the number of single word clusters and average coherence, with fewer new clusters in the top 30, but larger cluster sizes. However, with the GN-algorithm it actually yields higher coherence, but more single word clusters. The average cluster size increases for both embeddings, but the number of new clusters in the top 30 goes up for Random Index, while dropping for Skipgram. The composition $(n = 4, p = 5)$ was chosen mainly because it yields fewer single word clusters compared to the other two candidates, without sacrificing too much average coherence.

5.1.1.4 Best Performing Hyperparameter Compositions

Comparing the top 12 candidates Section 4.1.2.3 presents a more through comparison of the 12 parameter compositions selected for further analysis in section 4.1.2.1 and 4.1.2.2. Comparing coherence values between Random Index and Skipgram embeddings is not possible, due to the difference in how they were calculated. However, all other metrics are comparable for all 12 compositions.

The GN-algorithm in combination with $n = 4$ and $p = 5$ performed poorly compared to all other selected compositions, and was thus ruled out for further analysis.

The fact that setting $n = 2$ prohibits any cluster from being larger than 2 makes it an unattractive choice. When combined with the GN-algorithm, $(n = 2, p = 5)$ yields fewer single word clusters compared to $(n = 4, p = 2)$ at the cost of lower coherence, fewer new clusters and the limited cluster size. Thus, it was ruled as an unattractive choice for the GN-algorithm. Results were better for $(n = 2, p = 5)$ when combined with hierarchical clustering, Still the advantages were not deemed significant enough to justify the hard limit of cluster sizes to two words. Thus, six compositions had been eliminated, with six more to go.

The large loss of coherence of $(n = 4, p = 5)$ compared to $(n = 4, p = 2)$ resulted in both compositions with $(n = 4, p = 5)$ being ruled out.

At this point, two compositions of $G_{cluster}$, n and p remained: $(G_{cluster} = GN, n = 4, p = 2)$ and $(G_{cluster} = hierarchical, n = 4, p = 2)$, both combined with both word embeddings. For both compositions, the experiments with Skipgram embeddings yielded slightly larger cluster sizes and more new clusters in the top 30. Unfortunately, the important coherence metric is not comparable across the two embeddings, making it very hard to rule one of them out. For this reason, one candidate composition was chosen for each word embedding.

For Random Index embeddings, it was deemed that the reduced coherence of the GN-algorithm compared to hierarchical clustering did not justify the slight reduction in single word clusters, so the composition $(G_{cluster} = hierarchical, n = 4, p = 2)$ was selected for further analysis. For Skipgram embeddings, higher coherence of the hierarchical clustering was also deemed as the most important decision point, so just as for the Random Index embeddings the composition $(G_{cluster} = hierarchical, n = 4, p = 2)$ was selected for further analysis.

Inspecting the best parameter composition The median, quartiles, and the 5th and 95th percentiles of the unsupervised metrics when using $(G_{cluster} = hierarchical, n = 4, p = 2)$ are presented in section 4.1.2.3. With both embeddings, the spread between the lower and upper quartiles is relatively small for all four metrics, with none of the quartiles representing an unacceptable outcome. However, it is noteworthy that the third quartile of number of single word clusters when using Skipgram embeddings is larger than the 95th percentile when using Random Index embeddings.

The 5th and 95th percentiles of average cluster size seem acceptable for both embeddings, meaning more than 90% of all data sets get acceptable cluster sizes. The 95th percentile of single word clusters being 67.65 with Random Index embeddings is pretty high, but given that 100 words are clustered it still means that more than 32 words were clustered, which is at least not a complete breakdown. The number of single word clusters reach as high as 78.10 in the 95th percentile when using Skipgram, which means that only 22 words end up being clustered. This is 10 words less than the 95th percentile when using Random Index embeddings. The 95th percentile of new clusters in top 30 is just two for both embeddings, so seeing more than two new clusters will be a very rare occurrence.

The low 5th percentile of average coherence is perhaps the least acceptable outcome for both word embeddings. The relative loss of coherence compared to the median is worse for Random Index compared to Skipgram. However, what ranges of coherence are actually acceptable depends on the general distribution of similarities in the word embedding, so it is still not possible to compare these numbers for the two word embeddings. Also, though there

is a correlation between human perception of coherence and topic coherence, it is not large enough to state how much a specific increase in coherence would impact human perception.

5.1.2 Supervised Evaluation

Section 4.2 shows the results from the supervised evaluation explained in section 3.3.1. The variations in maximum precision and recall between the three data sets probably comes down to two different factors: 1) what words are in the data set and 2) the number of clusters and their size in the silver standard. For example, most parameter compositions achieve lower recall on the Airlines data set compared to the other two data sets. This is not unexpected since it has a much larger average cluster size, and even a cluster of size 26, which none of the evaluated compositions could hope to get right given that the largest evaluated maximum cluster size n was 20. The two other data sets have lower average cluster sizes, making it possible even for compositions with lower values of n to successfully group a large portion of the target clusters. Also, with lower average cluster sizes, optimizing recall over precision becomes possible by creating large groups containing multiple target clusters.

It is interesting that the two parameter compositions selected for further analysis in section 4.1.2.3 are among the top performers in terms of precision for all three data set. Even more noteworthy is the result that for the Hotel and Airbnb data sets there exist few or no other parameter compositions that achieve higher precision without paying in recall, making both of the selected compositions attractive. The compositions that achieve a much better recall on the Airline data set without paying much in precision all have one thing in common. They have the maximum cluster size parameter n set to 20, which was deemed as leading to unacceptably low average coherence in the unsupervised experiments.

In fact, it could be argued that achieving perfect precision and recall on the silver standards would not be a good indicator that a clustering would be useful to a user. A finished analysis of a data set might include some large topics, created by an analyst by repeatedly adding new terms and merging topics, based on an incremental understanding of what the topics is about. To achieve perfect recall on such a silver standard would require the clustering algorithm to immediately suggest that whole topic, which might feel overwhelming to the user, who did not get the possibility to gradually adjust their understanding of the topic.

Achieving perfect precision would perhaps not have as apparent drawbacks as maximizing recall, but it is perhaps not necessary. Perfect precision suggests that none of the suggested clusters contain any words not grouped by the analyst. It could very well be the case that the analyst missed some words, that would have made sense to put in the topics she created. It could also be the case that some clusters not deemed interesting by the analyst still would not be harmful to suggest, since they could simply be ignored. For these reasons, the supervised evaluation is perhaps not as interesting as the unsupervised evaluation, which was both more robust due to the larger number of data sets evaluated, bot also since the metrics measured there more clearly describe the type of clusters produced with different parameter compositions.

5.1.3 Examples of Top 30 Topics

In order to shed some light on what kind of word clusters are created by the word clustering algorithm, some example clusters are presented and discussed here.

First, two data sets were randomly sampled among the 110 data sets used in the unsupervised evaluation. Second, the hyper parameter composition deemed the best in section 4.1.2.3, i.e. $(G_{cluster} = Hierarchical, G_{type} = MST, W_{scheme} = Weighted, n = 4, p = 2)$, was used to cluster both data sets. Table 5.1 shows the original top 30 list for both data sets, as well as the top 30 lists after clustering with Random Index and Skipgram embeddings. Words in the same cluster are grouped together in the list, separated by white space. Bold

Table 5.1: The top 30 words before and after clustering, for two data sets randomly sampled among the 110 data sets used in the unsupervised evaluation. Words appear in order of document frequency, with clustered words separated by a wider margin. To avoid displaying partial clusters at the bottom of the top 30, the clustered top list have been extended past 30 words when necessary. Words in the new top 30 that do not appear in the original top 30 are bold. Clustering was performed with hierarchical clustering, max cluster size of 4 and filtering similarities outside of the top 2%. Results from using both Random Index and Skipgram word embeddings are shown.

Example 1			**Example 2**		
Original	Random Index	Skipgram	Original	Random Index	Skipgram
dissertation students degree one chair would courses classes professors time feedback many get years work never money chairs student process faculty assignments ncu experience even university program instructors feel due	students professors student **professor** dissertation **phd** **doctoral** courses classes **class** **changed** **changes** **change** **changing** many **two** **different** **multiple** chair chairs even **much** **little** get **getting** **got** feedback **information** **also** **however** **often**	dissertation degree **phd** **doctoral** professors faculty university **professor** courses classes **course** even due **much** **however** **changed** **changes** **change** **changing** feel **really** **felt** **want** get **getting** **go** **got** students student chair chairs	rent property tenants would maintenance like pets inspections rental owners repairs better managers done bond expensive renting able lease tenant cheaper pay less allow lower allowed nothing need make increases	rent renting **rents** maintenance inspections repairs **inspection** property tenants **renters** able need **ability** **needs** would pets **pet** like cheaper **faster** **easier** **quicker** tenant **landlords** **landlord** rental **rentals** better **good** lease **leases**	rent rental **rentals** **rents** tenants tenant **landlords** **landlord** property **estates** able allow allowed would maintenance like pets **pet** owners **owner** expensive cheaper inspections **inspection** make **get** **getting** **give** lease **leases**

words did not appear in the original top 30, so completely bold clusters are those counted as new clusters in the top 30.

Example 1 The first data set, Example 1, appears to be about university, with words such as *dissertation*, *students*, *degree* and *professors* among the top 30 words. The top 30 lists for both Random Index and Skipgram embeddings share many identical and similar clusters.

A cluster that is both identical for the two word embeddings, and new in the top 30, is {*changed*, *changes*, *change*, *changing*}. It seems like many of the texts in the data set mentioned

some kind of change, but different tenses of the word was used. By grouping four tenses together the theme was promoted to the top 30. Grouping words with the same root probably makes sense in most cases, since different tenses of the same word will commonly be used when discussing the same general theme in different tenses. It is nice that these relationships are picked up by both embeddings, though it would also be easy to detect them by using a dictionary.

Another cluster that appears for both word embeddings is {*chair, chairs*}, created by grouping the singular and plural form of *chair*. This grouping is also useful, though it could also easily have been detected using a dictionary.

The cluster {*courses, classes, class*} in the Random Index list, and {*courses, classes, course*} in the Skipgram list, differ in only one word: *class* was exchanged with *course*. Both clusters contain one singular/plural pair, but also a synonym pair: *courses* and *classes*. Synonyms could potentially be detected using a thesaurus such as WordNet[1].

Both word embeddings put *dissertation*, *phd* and *doctoral* in the same cluster, though Skipgram also included *degree*. These words are obviously related, but not as varied tenses or as synonyms. Thus, they are perhaps the first example that could not have been trivially constructed based on a handcrafted lexical resource.

One of the Random Index clusters is {*students, professors, student, professor*}, which contains two singular/plural pairs, whose roots *student* and *professor* are obviously related with each other. However, in the context of analysing a data set that mostly seems to be about university, it could be the case that a user would like to keep texts about students and texts about professors separate. Thus, this might be a case where the cluster will be split up by the user. The similar cluster {*professors, faculty, professor, university*} in the Skipgram embeddings would perhaps get *university* removed by the user, since it is less strongly related to the other words than they are to each other.

Both embeddings generate some clusters that makes little sense when analysing themes in the texts. For example, the cluster {*also, however, often*} in the Random Index embeddings seems to be created based on the fact that these words can have similar functions in a text, but they could all be used to discuss a widely varied set of themes. Similarly, the Skipgram embeddings generated the cluster {*even, due, much, however*}, which has the same problem.

Example 2 The second data set, Example 2, seems to be about housing, with words such as *rent*, *property* and *tenants* in the top 30.

Many of the same themes can be seen in the groups as those discussed for the first example. For example, the clusters {*pet, pets*} and {*lease, leases*}, consisting of a singular/plural pairs, are identified with both word embeddings.

The Skipgram cluster {*tenants, tenant, landlords, landlord*} groups two singular/plural pairs, whose roots are clearly related. However, in a data set about housing they might not be related enough to warrant a merged topic. Thus, they might end up being split up by the user. In the Random Index embeddings, *tenants* is instead grouped with *renters*, while {*tenant, landlords, landlord*} is a separate cluster. A user unhappy with the grouping of *tenant* and *landlord* would perhaps move *tenant* to the {*tenants, renters*} cluster.

The Random Index cluster {*cheaper, faster, easier, quicker*} contains four adjectives in comparative form, out of which *faster* and *quicker* are the only semantically similar words. Using Skipgram embeddings, *cheaper* was instead clustered with *expensive*. This is an interesting cluster, since the two words are antonyms, i.e. they have opposite meaning. Both words are used to talk about the same thing, price, but in the opposite way. Thus, a user might find it useful to split the two words into separate topics.

With Random Index embeddings, *able* is grouped with *need*, *ability* and *needs*, which could make sense when talking about housing, since tenants will have needs based on their abilities. With Skipgram another theme containing *able* is clustered: {*able, allow, allowed*}. A tenant is

[1] https://wordnet.princeton.edu/

able to do something which they have been allowed. Both uses of the word *able* are valid, and it is hard to tell which one a user would prefer. Unfortunately, seeing one of the clusters might bias a user towards that use. Given then cluster {*able, allow, allowed*}, it is a longer reach to group *able* with *need*, compared to if *able* was left un-clustered. This potential to introduce bias is something to be mindful of when automatically clustering terms.

General comments All four clustered top 30 lists include some clusters that make little sense. They also contain many useful clusters, such as clusters created by merging different word forms, synonyms or otherwise semantically similar words. The effect of clustering these words is twofold: it promotes some themes to higher positions in the top list, and it also makes it easier to get an overview of the whole list. In fact, I would argue that by arranging the top lists in clusters of related words, they become considerably more easy to analyse. Possibly to such an extent that a longer top list could be analysed in a similar amount of time.

Using Random Index or Skipgram embeddings makes a difference in many clusters. However, it is not clear that one of the embeddings performs better than the other.

5.2 Method

In this section, the method presented in chapter 3 is discussed and critiqued.

5.2.1 Word Clustering Approach

Though much research has been done on word clustering, no previous research was found that was directly applicable to the task of arranging a relatively small set of words in a user application. Thus, the chosen word clustering approach used in this thesis is not directly comparable with any previous methods. However, the method is mostly a composition of previously well-studied methods, making it possible to discuss both the chosen composition and the individual components.

5.2.1.1 Clustering Based on Word Embeddings

As mentioned before, most word clustering literature precedes the publication on Word2Vec by Mikolov et al. [23], which sparked the recent surge in research on word embeddings. Given how word embeddings have become ubiquitous for representing words in other machine learning research, including recent work on word clustering [38], the choice to cluster words based on word embeddings is well-motivated. Mutual information measures such as the ones used by Brown et al. [6] and Lin [19] are less scalable since they require storing co-occurrence statistics for all words in the vocabulary. Given a vocabulary of size $|V|$, a co-occurrence matrix will have size $|V| \times |V|$, as opposed to the $|V| \times D$ used by a word embedding with dimensionality D. When V is in the order of millions of words, and D is in the order of hundreds or thousands, using word embeddings is much more manageable.

5.2.1.2 Graph Clustering

Given vector representations of the words to be clustered, many clustering approaches are applicable. The approach taken in this thesis was to treat words as nodes in a graph, with edges representing the cosine distance between pairs of words, and apply graph clustering to that graph. It would also have been possible to cluster the words directly given their vector representation, using a clustering method such as K-means [20] clustering or Gaussian Mixture Modeling [37, e.g.]. These two methods have a fundamental drawback for this application though, in that they try to optimize a global criterion by fitting all words in a fixed number of clusters. Since the work of Karlgren et al. [16] suggests that only local structures

are meaningful in word embeddings, it makes little sense to take global structures into account when clustering words. The graph clustering approaches used in this thesis instead define a local criterion to satisfy in order to reach a final clustering. Any time a cluster is split, the words in one part will have no further influence on how the words in the other part are clustered.

A nice effect of the chosen approach is that it allows clusters to have varying degrees of density. A dense cluster of multiple words with relatively high cosine similarities can be further divided into two or more clusters, while still allowing less dense clusters to remain intact. This means the maximum size of clusters can be limited, without requiring less dense clusters to be split up. Presuming that cosine similarities between words in one region of a word embedding are not necessarily comparable to the cosine similarities in another region, this is a desirable property of the graph clustering algorithms. However, to avoid generating some very sparse clusters that a user might not find meaningful, an initial threshold of cosine similarity was applied to all edges in the graph. All edges with a cosine similarity below that threshold were removed, essentially saying that they do not indicate a meaningful relationship between the words. This threshold was determined based on the similarities among the words to be clustered. Based on the argument that cosine similarities are not comparable across different regions in a word embedding, setting a global threshold of similarity makes little sense, though it proved to have some positive effects in the unsupervised evaluation. A more sound approach would have been to find an individual threshold for each word, or perhaps for different word classes. Maybe such thresholds could be determined by inspecting the cosine similarities each word has to its closest neighbours in the entire word embedding vocabulary.

5.2.2 Choice of Word Embeddings

The similarity graph used to cluster words in this thesis was created by measuring cosine similarities in two different word embeddings: Random Index embeddings trained on a large private data set, and Skipgram embeddings with hierarchical softmax trained on Wikipedia. First, some critique of the way these embeddings were trained, then some possible alternative embeddings are presented.

5.2.2.1 Critique of the Chosen Word Embeddings

Random Index The Random Index embeddings were provided by Gavagai. The fact that they were trained on a large private data set unfortunately makes the results from clustering with them non-reproducible.

Skipgram with hierarchical softmax Based on the results of Mikolov et al. [24], it is likely that training the Skipgram embeddings using negative sampling instead of hierarchical softmax would have been both quicker and resulted in higher quality embeddings. Unfortunately, the long training times required to train the embedddings on the entirety of the Wikimedia dump is prohibitive, and retraining the embeddings was not plausible at the point when this conclusion was made.

5.2.2.2 Alternative Word Embeddings

Multiple popular word embeddings could have been evaluated, such as GloVe [30] or Fasttext[5]. However, given that they are trained on relatively small contexts, just as the evaluated Random Index and Skipgram embeddings were, the similarity relations represented would be similar in the sense that all four embeddings model similarities in terms of words appearing in similar, small, contexts. Since it was already hard to tell the results from Random Index and Skipgram embeddings apart, adding more word embeddings that learn similar word relationships might not have added much to the results of this thesis.

Something that would have perhaps been more interesting would be to use a word embedding with a drastically different definition of context. For example, using LSA word embeddings, where similarity means that words often occur in the same paragraph, could perhaps have allowed different kinds of clusters to be created.

5.2.3 Evaluation

5.2.3.1 Unsupervised Evaluation

By evaluating the effect of different hyperparameter compositions for the word clustering algorithm on multiple data sets previously analysed by users of the Gavagai Explorer, the expected results on new data sets could be analysed. The evaluation gives no indication of whether the clusters would have been useful to a user, but helped find a parameter composition that often strikes a balance in terms of the number of single word clusters, the size of clusters, the potential of promoting new clusters to the top 30, and having decently coherent clusters.

Using data sets previously analysed by users of the Gavagai Explorer ensures that the measurements were made on relevant data sets, though it does not mean that the data sets are representative of all possible data sets users might want to analyse in the future. A drawback of performing the analysis on customer data is that the data sets can not be made available for reproducibility.

Measuring coherence using the cosine similarities in the word embedding being used to cluster words made it impossible to compare coherence between the experiments with Random Index and Skipgram embeddings. This is unfortunate, and perhaps using a third embedding to measure coherence would have been a possible solution. However, such a solution could have introduced a bias toward a certain word embedding which it was more similar to.

5.2.3.2 Supervised Evaluation

The supervised evaluation helped answer the question if clusters produced by the word cluster algorithm overlap with those previously produce by users of the Gavagai Explorer. Unfortunately, previous clusters created by users are not a perfect representation of the clusters that the algorithm should produce. First, users could have missed some relevant clusters in their analysis, so it is not necessarily a bad thing that the algorithm suggests some clusters not found by the users. Second, a cluster could be useful to a user, regardless if it is deemed useful to the final analysis, since clustering words make the list of suggested topics more structured. For example, when analysing hotel reviews, a topic such as *hotel*, *hotels* might not be very interesting to analyse, since all texts are expected to be about hotels, but grouping the words still makes it easier to discard them both. Third, a user might iteratively create large topics, improving their understanding of the topic in the process. By immediately suggesting such a large topic, the user might be overwhelmed and not understand the reason for grouping those words in the same topic. For these reasons, optimizing silver standard performance was not a suitable method for finding a good hyperparameter composition.

In critique of the method, it should be noted that since the Gavagai Explorer suggests new words to add to topics using Random Index vectors, the silver standards might be biased toward grouping words that are similar in the Random Index embeddings. However, as seen in Figure 4.13 it is not clear that the candidate parameter composition performed better with Random Index embeddings than it did with Skipgram embeddings.

5.2.3.3 Alternative Evaluation

Since the ultimate goal of clustering words in the Gavagai Explorer is to help users analyse their data more easily, it would have been interesting to measure user satisfaction through

a user study. However, user studies are expensive and time-consuming, and can potentially end up measuring the wrong thing. In the context of this thesis, a potential pitfall could have been users judging the presentation of the clustering algorithm instead of the usefulness of the clusters themselves. For this reason, performing a user study was deemed out of scope for this thesis.

5.2.4 Source Criticism

Theory presented throughout this thesis mostly come from peer-reviewed journals, with only a few pre-prints being cited. The only pre-print on which major parts of the theory has been based is the initial publication of Word2Vec [23], which has nonetheless been scrutinized by the scientific community, and cited more than 10,000 times according to Google Scholar.

5.3 The Work in a Wider Context

By introducing automated steps in the process of analysing text data, there is potential to bias the resulting analysis. Following the suggestions of the algorithm will be more convenient for a user compared to splitting up the suggestions and creating novel clusters. So, if the algorithm is biased towards grouping certain types of words, more users are likely to end up with those words in the same topics. The algorithms inclination towards grouping some words will originate from the word embeddings used to represent the words. Since these embeddings are trained on large collections of texts, it is not hard to imagine that they might pick up on human biases; biases about gender, race, sexuality, politics etc. If biased clusters are suggested by the algorithm, these biases might end up being reinforced if they are included in the final analysis.

Another problem of automated suggestions of clusters is that it might be hard to look beyond a suggestion, to see alternative clusters. For example, given the cluster {*jaguar, porsche*}, it could be harder to detect that {*jaguar, lion*} would also make sense as a cluster. If the algorithm is biased to one type of similarity, it might end up systematically suppressing some insights. Luckily, most data sets analysed in the Gavagai Explorer have a specific domain, in which most otherwise ambiguous words have a most likely meaning. In a data set of zoo reviews, a user will probably be predisposed to interpreting *jaguar* as an animal over a car, so in the case that {*jaguar, porsche*} is suggested, the user would still figure out that {*jaguar, lion*} makes more sense.

6 Conclusion

This thesis has presented an algorithm for arranging a limited set of words into clusters to be presented to a user of the text analysis tool Gavagai Explorer. Multiple hyperparameters were evaluated across 110 data sets, providing a good estimate of how many words are typically left un-clustered, how large an average cluster is, how coherent clusters are, and how many completely novel clusters typically appear in the top 30 list of suggested words. This gives a good idea about how well the algorithm generalises to new data sets. A specific hyperparameter composition was deemed to strike the best balance among the four metrics, but the thorough evaluation of other parameters also provides a foundation to understand other compositions if another balance is desired.

6.1 Research Questions

6.1.1 Do words grouped by the proposed word clustering algorithm overlap with groups previously created by users of the Gavagai Explorer?

Yes, a certain degree of overlap was found between topics created by a user of the Gavagai Explorer and the suggestions produced by all evaluated hyperparameter compositions of the word clustering algorithm. Out of three data sets evaluated, the two compositions chosen as having the highest potential in the unsupervised evaluation never scored below 40% precision, meaning that more than 40% of the word pairs put in the same cluster by the algorithm were also put in the same cluster by the user.

6.1.2 Does grouping keywords affect the rank of some less frequent keywords enough to make novel themes visible in the top 30 list of topics?

Yes, but out of 110 evaluated data sets this happened less than once per data set for the hyperparameter compositions deemed most useful in the unsupervised evaluation.

6.1.3 How reliable is the behaviour of the proposed word clustering algorithm across different data sets?

When evaluating the selected hyperparameter composition for the word clustering algorithm on 110 data sets, it was shown that for 90% of all data sets the average cluster size was between 2.4 and 3.0, regardless of what word embedding was used. With Random Index embeddings 90% of all data sets got between 48.0 and 67.7 single word clusters, and with Skipgram embeddings the same range was between 53.0 and 78.1. The average coherence of 90% of the data sets was between 0.25 and 0.39 with Random Index embeddings and between 0.65 and 0.75 with Skipgram embeddings. It is important to note that the coherence values are not comparable between the two word embeddings since they were also measured with cosine similarities from the respective embeddings. Regardless of the embedding, 90% of all data sets got between 0 and 2 new clusters in the top 30.

6.1.4 What hyperparameters of the proposed word clustering algorithm have the biggest impact on its behaviour?

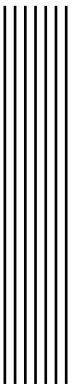
The hyperparameters with the largest impact on the behaviour of the word clustering algorithm were the maximum cluster size n and the similarity threshold p. An increase of both n and p consistently led to fewer single word clusters and larger cluster sizes at the cost of lower coherence and fewer new words in the top 30.

The choice of computing word similarities using Random Index or Skipgram embeddings had little impact on the behaviour of the algorithm. Combining the GN-algorithm with an unweighted MST, a weighted RNG as well as an unweighted RNG also yielded very similar results. The GN-algorithm produced fewer single word clusters and larger clusters, but lower average coherence compared to the single-linkage hierarchical algorithm. However, not by a particularly large margin.

6.2 Future Work

This work quantifies the coherence of clusters using topic coherence, which has been shown to correlate with human perception of coherence. However, a cluster being interpreted as coherent does not necessarily make it useful to a user of the Gavagai Explorer. It would be interesting to study how users interact with the clusters in a user study, where users can either accept or discard suggested clusters in the context of the Gavagai Explorer. This would give a better indication of whether clusters suggested by the algorithm are useful to users.

Since cosine similarities are not necessarily comparable across different regions of a word embedding, it would be interesting to investigate if the thresholds used to filter out irrelevant similarities could be set individually for each word, instead of globally. This could potentially be done by inspecting the distance between each word and its closest neighbours in the entire vocabulary of the word embedding, and either setting a threshold based on the distance to the k-th neighbour, or by means of the elbow method.



Bibliography

[1] Nikolaos Aletras and Mark Stevenson. "Evaluating topic coherence using distributional semantics". In: *Proceedings of the 10th International Conference on Computational Semantics (IWCS 2013)–Long Papers*. 2013, pp. 13–22.

[2] Kayhan Batmanghelich, Ardavan Saeedi, Karthik Narasimhan, and Sam Gershman. "Nonparametric spherical topic modeling with word embeddings". In: *arXiv preprint arXiv:1604.00126* (2016).

[3] Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Jauvin. "A neural probabilistic language model". In: *Journal of machine learning research* 3.Feb (2003), pp. 1137–1155.

[4] David M Blei, Andrew Y Ng, and Michael I Jordan. "Latent dirichlet allocation". In: *Journal of machine Learning research* 3.Jan (2003), pp. 993–1022.

[5] Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. "Enriching word vectors with subword information". In: *Transactions of the Association for Computational Linguistics* 5 (2017), pp. 135–146.

[6] Peter F Brown, Peter V Desouza, Robert L Mercer, Vincent J Della Pietra, and Jenifer C Lai. "Class-based n-gram models of natural language". In: *Computational linguistics* 18.4 (1992), pp. 467–479.

[7] Jonathan Chang, Sean Gerrish, Chong Wang, Jordan L Boyd-Graber, and David M Blei. "Reading tea leaves: How humans interpret topic models". In: *Advances in neural information processing systems*. 2009, pp. 288–296.

[8] Amaru Cuba Gyllensten and Magnus Sahlgren. "Navigating the Semantic Horizon using Relative Neighborhood Graphs". In: *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, 2015, pp. 2451–2460.

[9] Scott Deerwester, Susan T Dumais, George W Furnas, Thomas K Landauer, and Richard Harshman. "Indexing by latent semantic analysis". In: *Journal of the American society for information science* 41.6 (1990), pp. 391–407.

[10] David Eppstein. *The relative neighborhood graph of 100 random points in a square*. File: `Relative neighborhood graph.svg`. 2010. URL: `https://commons.wikimedia.org/wiki/File:Relative_neighborhood_graph.svg`.

[11] Michelle Girvan and Mark EJ Newman. "Community structure in social and biological networks". In: *Proceedings of the national academy of sciences* 99.12 (2002), pp. 7821–7826.

[12] Oleksandr Grygorash, Yan Zhou, and Zach Jorgensen. "Minimum spanning tree based clustering algorithms". In: *2006 18th IEEE International Conference on Tools with Artificial Intelligence (ICTAI'06)*. IEEE. 2006, pp. 73–81.

[13] Thomas Hofmann. "Probabilistic latent semantic analysis". In: *Proceedings of the Fifteenth conference on Uncertainty in artificial intelligence*. Morgan Kaufmann Publishers Inc. 1999, pp. 289–296.

[14] Yuening Hu, Jordan Boyd-Graber, Brianna Satinoff, and Alison Smith. "Interactive topic modeling". In: *Machine learning* 95.3 (2014), pp. 423–469.

[15] Pentii Kanerva, Jan Kristoferson, and Anders Holst. "Random indexing of text samples for latent semantic analysis". In: *Proceedings of the Annual Meeting of the Cognitive Science Society*. Vol. 22. 22. 2000.

[16] Jussi Karlgren, Anders Holst, and Magnus Sahlgren. "Filaments of meaning in word space". In: *European Conference on Information Retrieval*. Springer. 2008, pp. 531–538.

[17] Omer Levy and Yoav Goldberg. "Neural word embedding as implicit matrix factorization". In: *Advances in neural information processing systems*. 2014, pp. 2177–2185.

[18] Hang Li and Naoki Abe. "Word clustering and disambiguation based on co-occurrence data". In: *Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics-Volume 2*. Association for Computational Linguistics. 1998, pp. 749–755.

[19] Dekang Lin. "Automatic retrieval and clustering of similar words". In: *COLING 1998 Volume 2: The 17th International Conference on Computational Linguistics*. Vol. 2. 1998.

[20] James MacQueen et al. "Some methods for classification and analysis of multivariate observations". In: *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*. Vol. 1. 14. Oakland, CA, USA. 1967, pp. 281–297.

[21] Christopher Manning, Prabhakar Raghavan, and Hinrich Schütze. "Introduction to information retrieval". In: *Natural Language Engineering* 16.1 (2010), pp. 100–103.

[22] Elaine McColl, A Jacoby, Lois Thomas, J Soutter, C Bamford, Nick Steen, Rosemann Thomas, E Harvey, A Garratt, and J Bond. *Design and use of questionnaires: a review of best practice applicable to surveys of health service staff and patients*. Core Research, 2001.

[23] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. "Efficient estimation of word representations in vector space". In: *arXiv preprint arXiv:1301.3781* (2013).

[24] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. "Distributed representations of words and phrases and their compositionality". In: *Advances in neural information processing systems*. 2013, pp. 3111–3119.

[25] George A Miller. "WordNet: a lexical database for English". In: *Communications of the ACM* 38.11 (1995), pp. 39–41.

[26] David Mimno, Hanna M Wallach, Edmund Talley, Miriam Leenders, and Andrew McCallum. "Optimizing semantic coherence in topic models". In: *Proceedings of the conference on empirical methods in natural language processing*. Association for Computational Linguistics. 2011, pp. 262–272.

[27] Frederic Morin and Yoshua Bengio. "Hierarchical probabilistic neural network language model." In: *Aistats*. Vol. 5. Citeseer. 2005, pp. 246–252.

[28] David Newman, Jey Han Lau, Karl Grieser, and Timothy Baldwin. "Automatic evaluation of topic coherence". In: *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*. Association for Computational Linguistics. 2010, pp. 100–108.

[29] Alicia O'Cathain and Kate J Thomas. "" Any other comments?" Open questions on questionnaires–a bane or a bonus to research?" In: *BMC medical research methodology* 4.1 (2004), p. 25.

[30] Jeffrey Pennington, Richard Socher, and Christopher Manning. "Glove: Global vectors for word representation". In: *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*. 2014, pp. 1532–1543.

[31] William M Rand. "Objective criteria for the evaluation of clustering methods". In: *Journal of the American Statistical association* 66.336 (1971), pp. 846–850.

[32] Michael Röder, Andreas Both, and Alexander Hinneburg. "Exploring the space of topic coherence measures". In: *Proceedings of the eighth ACM international conference on Web search and data mining*. ACM. 2015, pp. 399–408.

[33] Herbert Rubenstein and John B Goodenough. "Contextual correlates of synonymy". In: *Communications of the ACM* 8.10 (1965), pp. 627–633.

[34] Magnus Sahlgren. "An introduction to random indexing". In: *Methods and Applications of Semantic Indexing Workshop at the 7th International Conference on Terminology and Knowledge Engineering, TKE 2005*. 2005.

[35] Magnus Sahlgren. "The Word-Space Model: Using distributional analysis to represent syntagmatic and paradigmatic relations between words in high-dimensional vector spaces". PhD thesis. 2006.

[36] Godfried T Toussaint. "The relative neighbourhood graph of a finite planar set". In: *Pattern recognition* 12.4 (1980), pp. 261–268.

[37] Hastie Trevor, Tibshirani Robert, and Friedman JH. *The elements of statistical learning: data mining, inference, and prediction*. 2009.

[38] Felipe Viegas, Sérgio Canuto, Christian Gomes, Washington Luiz, Thierson Rosa, Sabir Ribas, Leonardo Rocha, and Marcos André Gonçalves. "CluWords: Exploiting Semantic Word Clustering Representation for Enhanced Topic Modeling". In: *Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining*. ACM. 2019, pp. 753–761.

[39] Liang Yao, Yin Zhang, Baogang Wei, Zhe Jin, Rui Zhang, Yangyang Zhang, and Qinfei Chen. "Incorporating Knowledge Graph Embeddings into Topic Modeling." In: *AAAI*. 2017, pp. 3119–3126.

[40] Charles T Zahn. "Graph theoretical methods for detecting and describing gestalt clusters". In: *IEEE Trans. Comput.* 20.SLAC-PUB-0672-REV (1970), p. 68.