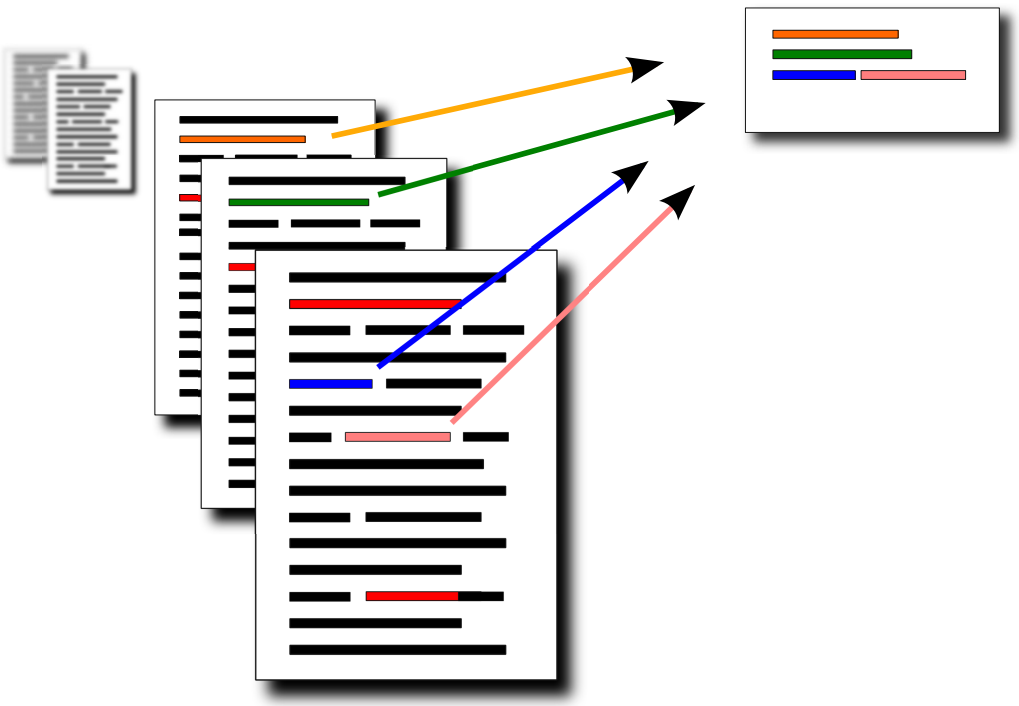
CHALMERS | UNIVERSITY OF GOTHENBURG



Multi-Document Summarization and Semantic Relatedness

OLOF MOGREN

Department of Computer Science and Engineering
CHALMERS UNIVERSITY OF TECHNOLOGY
UNIVERSITY OF GOTHENBURG
Gothenburg, Sweden 2015

THESIS FOR THE DEGREE OF LICENTIATE OF ENGINEERING

Multi-Document Summarization and Semantic Relatedness

OLOF MOGREN

Department of Computer Science and Engineering
CHALMERS UNIVERSITY OF TECHNOLOGY
UNIVERSITY OF GOTHENBURG

Gothenburg, Sweden 2015

Multi-Document Summarization
and Semantic Relatedness
OLOF MOGREN

Thesis for the degree of Licentiate of Engineering
ISSN 1652-876X
Technical Report No. 140L
Department of Computer Science and Engineering

Department of Computer Science and Engineering
Chalmers University of Technology and University of Gothenburg
SE-412 96 Gothenburg
Sweden
Telephone: +46 (0)31-772 1000

Cover:
Extractive Multi-Document Summarization; choosing the sentences that are most representative and informative in a set of input documents.

Chalmers Reproservice
Gothenburg, Sweden 2015

To Sofie, Tove, and Stina.

Abstract

Automatic summarization is the process of presenting the contents of written documents in a short, comprehensive fashion. Many approaches have been proposed for this problem, some of which extract content from the input documents (extractive methods), and others that generate the language in the summary based on some representation of the document contents (abstractive methods).

This thesis is concerned with extractive summarization in the multi-document setting, and we define the problem as choosing the most informative sentences from the input documents, while minimizing the redundancy in the summary. This definition calls for a way of measuring the similarity between sentences that captures as much as possible of the meaning. We present novel ways of measuring the similarity between sentences, based on neural word embeddings and sentiment analysis. We also show that combining multiple sentence similarity scores, by multiplicative aggregation, helps in the process of creating better extractive summaries.

We also discuss the use of information extraction for improving the quality of automatic summarization by providing ways of assessing the salience of information elements, as well as helping with the fluency of the output and providing the temporal dimension.

Furthermore, we present graph-based algorithms for clustering words by co-occurrence, and for summarizing short online user-reviews by computing bicliques. The biclique algorithm provides a fast, simple algorithm for summarization in many e-commerce settings.

Acknowledgements

I would like to thank both my supervisor Peter Damaschke and my co-supervisor Devdatt Dubhashi, for interesting discussions and inspiring ideas.

A deep thank you also goes to my office-mate and co-author Mikael Kågebäck; this thesis would not be much without your contributions and discussions. Thank you Fredrik Johansson, for inspiring chats and a creative atmosphere. Also, thank you Nina Tahmasebi for teaching me lots of important things, Jonatan Bengtsson for nice work on sentence similarity measures, and Richard Johansson for great advice and interesting discussions.

Thank you mum and dad for always encouraging me and believing in me, and my brother and my sisters for supporting, inspiring and challenging me from day one. I would never have made it here without you. Many others have been a great help to me during this time. Thank you all!

List of publications

This thesis is based on the following manuscripts.

Paper I M. Kågebäck et al. (2014). "Extractive Summarization using Continuous Vector Space Models". *Proceedings of The Second Workshop of Continuous Vector Space Models and their Compositionality*, pp. 31–39

Paper II O. Mogren, M. Kågebäck, and D. Dubhashi (2015). "Extractive Summarization by Aggregating Multiple Similarities". *Proceedings of Recent Advances in Natural Language Processing*, pp. 451–457

Paper III N. Tahmasebi et al. (2015). Visions and open challenges for a knowledge-based culturomics. *International Journal on Digital Libraries* **15**.2-4, 169–187

Paper IV P. Damaschke and O. Mogren (2014). Editing Simple Graphs. *Journal of Graph Algorithms and Applications, Special Issue of selected papers from WALCOM 2014* **18**.4, 557–576. DOI: 10.7155/jgaa.00337

Paper V A. S. Muhammad, P. Damaschke, and O. Mogren (2016). "Summarizing Online User Reviews Using Bicliques". *Proceedings of The 42nd International Conference on Current Trends in Theory and Practice of Computer Science, SOFSEM, LNCS*

Contribution summary

Paper I	I implemented the submodular optimization algorithm for sentence selection and created the setup for the experimental evaluation. I also wrote parts of the manuscript and made some illustrations.
Paper II	I am the main author of this work. I designed the study, performed the experiments, and wrote the manuscript.
Paper III	I wrote section 5, titled "Temporal Semantic Summarization", where I shared my views on possible research directions on generic multi-document summarization.
Paper IV	I contributed to the study and the analysis, and to the writing of the manuscript, including making the illustrations.
Paper V	I contributed to the study, did a substantial part of the experimental work, and contributed to the writing of the manuscript.

Contents

Part I
Extended summary

Chapter 1

Introduction

Automatic summarization is the process of presenting written documents in a condensed form. This is an active and exciting area of research, but also a highly relevant technology for making information easily available to people who struggle with information overload, or need to skim through massive amounts of text in the search of some particular piece of information.

In a very coarse-grained fashion, automatic summarization approaches can be divided into two different categories: abstractive and extractive. While abstractive summarization systems build some representation of the contents before generating the summary output, extractive systems select parts (typically sentences) of the input documents to present to the reader This thesis will focus on the latter category. In both, some form of optimization is normally performed, after stating an objective that gives a score of the output quality.

What is being optimized is at the core of this thesis. A natural way to state the objective is to say that we want the summary to have a high similarity to the input documents, and at the same time to have as little redundancy as possible. Many systems that have scored well in evaluations measure this similarity by counting word overlaps between sentences. In this thesis, we study ways to make the summaries more semantically aware by using ideas and techniques from deep learning, information extraction and sentiment analysis.

While the focus will be on multi-document summarization, many of the insights will be applicable to other problems in natural language processing (NLP), such as machine translation and semantic relatedness.

The thesis is divided into the following chapters: Chapter 2 presents some of the existing approaches to summarization that have been proposed as of today, and discusses some of the subtasks that have been considered. Chapter 3 provides a summary of the contributions in this thesis, while Chapter 4 concludes the work.

This work has mainly been done within the project "Data-driven secure business intelligence", grant IIS11-0089 from the Swedish Foundation for Strategic Research (SSF). Parts of the work has also been done within the projects "Algorithms for Inference" and "Towards a knowledge-based culturomics" supported by the Swedish Research Council (2012–2016; dnr 2012-5738).

Chapter 2

Background

The objective of automatic summarization is to capture the most important topics in a set of documents, and to present them briefly in a manner that is coherent and easy to consume. We will see how this can be broken down into subproblems in the following sections, discuss the relationship to semantic relatedness in Sections 2.1–2.2, then survey some existing methods for automatic document summarization in Sections 2.3–2.5, and conclude the background chapter in Section 2.6 with a discussion about existing techniques for summarization evaluation. All summarization approaches discussed here share the aim that the resulting summary has a high similarity with the input documents, and that it is short and non-redundant at the same time.

Many other NLP applications share parts of this objective, and could benefit from having more meaningful representations. One of them is machine translation, where the generated output should capture the semantics of the input, i.e. it should be semantically similar although expressed in a different language, and hence being lexicographically dissimilar. A different related application is textual entailment, the problem of deciding whether a sentence follows logically from another sentence.

In order to compute a summary that has a high semantic similarity to a set of documents D, one could envision an automatic summarizer that has a way of interpreting (and representing) the semantics of the content of D. This is however one of the more difficult problems within artificial intelligence, and so far, we attack it by doing the simplifications of measuring similarity between language units. In the following sections, we survey existing methods of measuring similarity in language, and existing approaches to summarization. In Chapter 3 we will present solutions to some of the shortcomings of existing methods, discuss how a sentence similarity score can capture as much semantics as possible, and how to best use the similarity scores to extract good sentences for a summary.

2.1 Semantic Relatedness

Semantic relatedness between two language units is a measure of how related their meanings are. In NLP settings, this is normally defined on the level of words or phrases.

Example: The phrase *"riding a bike"* is more semantically related to *"driving a car"* than to *"playing a game"*. This is easy to see for a human using just intuition (and some knowledge about the language involved), but a more involved problem for a computer program.

The SemEval 2014 task 1 for semantic relatedness was defined as predicting a relatedness score for pairs of sentences in English, the score ranging from 1 (completely unrelated) to 5 (highly related). For this, the SICK dataset was presented. Different approaches have been proposed for this task, from ontology-based to distributional representations. (Zhao, Zhu, and Lan 2014) presented a hybrid approach to the semantic similarity task, while (Tai, Socher, and Manning 2015) proposed a variant of the LSTM recurrent neural network architecture, where the LSTM modules are arranged according to a parse tree.

Semantic similarity is stricter than semantic relatedness, as it only includes the "is a" relationship (while relatedness includes any semantic relations). We will not go more into the distinction here, as it is not a focus of this thesis, and much literature use the two terms interchangeably.

Example: A *bicycle* is highly semantically related to a *wheel*, but they are not similar, neither semantically nor lexicographically. Conversely, *"driving someone crazy"* is lexicographically similar to *"driving a car"* but there is no apparent semantic similarity neither any semantic relatedness.

2.2 Measuring Similarity Between Language Units

Falling short of being able to understand and represent the semantic content in a text, one typically let computers use more shallow ways of relating different texts to each other. On character level, the edit-distance (Levenshtein 1966) gives a measure on how similar two strings of text are. On word-level, word overlap scores are another class of similarity measures that have been successfully used in summarization systems. Several different variants of such scores have been proposed in the summarization literature, with differences in preprocessing (such as stop-words removal, spelling corrections, and other filtering), weighting schemes, and normalization.

All similarity measures in this category are surface matching techniques, there is no semantical information used.

In 2011, H. Lin and Bilmes used a vector representation for each sentence, defined as its bag-of-terms representation, weighted by TFIDF (where terms are unigrams and bigrams). The similarity of each pair of sentences were then scored by the cosine similarity:

$$M_{\mathbf{s}_i,\mathbf{s}_j} = \frac{\sum_{w \in \mathbf{s}_i} tf_{w,i} \cdot tf_{w,j} \cdot idf_w^2}{\sqrt{\sum_{w \in \mathbf{s}_i} tf_{w,\mathbf{s}_i} idf_w^2} \sqrt{\sum_{w \in \mathbf{s}_j} tf_{w,\mathbf{s}_j} idf_w^2}}$$

A similar representation was also used by Radev et al. (2004).

Mihalcea and Tarau used a similar approach (2004), without the TFIDF weighting, and with a different normalization:

$$M_{\mathbf{s}_i,\mathbf{s}_j} = |\mathbf{s}_i \cap \mathbf{s}_j| / (\log |\mathbf{s}_i| + \log |\mathbf{s}_j|)$$

In their master's thesis, Bengtsson and Skeppstedt survey the landscape of similarity measures, including the ones mentioned in this section, and evaluate them together with TextRank, an existing summarization system (see Section 2.5).

2.3 Automatic Document Summarization

Automatic document summarization is the problem of presenting the most important content of a set of input documents. Methods for solving this can be classified as either abstractive or extractive. The former represents systems that process the input and then generate the language in the resulting summary, and the latter represents systems that choose parts (typically whole sentences) from the input documents and extract those, in their current form, to constitute the summary.

In this thesis we will focus on extractive systems, which date back to 1958, when Luhn presented an algorithm that looked at words' frequencies of occurring in a document. Luhn identified words as important for the document if they appeared not too commonly and not too rarely, and then scored sentences with a formula depending on the number of important words in them.

The idea of determining the importance of words has lived on, but threshold values have been replaced with other approaches, such as TFIDF weighting[1], and regression models (Hong and Nenkova 2014). Both of these make away with the hard constraints of threshold values, and instead provide a continuous measure of importance for different terms.

[1]TFIDF, Term frequency times inverse document frequency, gives high weight to words that are specific to this document

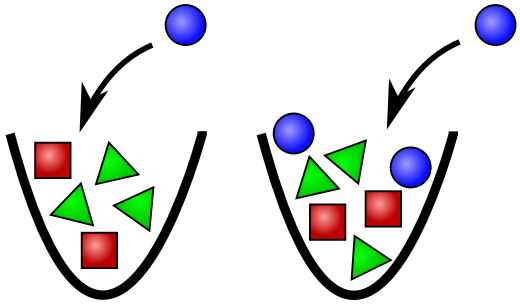


Figure 2.4.1: *Illustration of a monotonically non-decreasing, submodular set function $\mathcal{F}$. The left bowl contains a subset (S) of the objects in the right bowl (T). Let $\mathcal{F}$ be the number of different shapes in a bowl. The value of $\mathcal{F}$ changes at least as much when adding an object to S (left), compared to adding it to T (right). Adding a ball to a bowl with only triangles and squares increases the number of shapes, but adding it to a bowl with all three shapes does not.*

2.4 Submodular Optimization

One of the most successful extractive summarization systems was presented in (H. Lin and Bilmes 2011). Here, a summary is seen as a set of sentences, and the quality of a summary is scored with a function measuring its coverage (with regards to the input documents) and diversity (internally). The coverage and the diversity is defined based on a similarity measure between sentences. They showed that their set scoring function is submodular, which allows for a greedy approximation algorithm. The resulting system is fast and obtained good results in evaluations, that are still on-par with state-of-the-art.

A set function $\mathcal{F}$ is a function that has subsets of some universe U as its domain. Here we will consider only real-valued set functions. A (monotonically non-decreasing) submodular set function is a real-valued set function that obeys the principle of diminishing returns. Intuitively, this means that adding an element v to a small set S increases the function value more than adding the same element to a bigger set.

Formally, if $S \subseteq T \subseteq U \backslash v$, then

$$\mathcal{F}(S + v) - \mathcal{F}(S) \geq \mathcal{F}(T + v) - \mathcal{F}(T).$$

Maximizing monotone non-decreasing submodular set functions can be done with a simple greedy algorithm, which yields a $(1 - \frac{1}{e}) \approx 0.63$ approximation.

The objective functions of many approaches for extractive summarization have been shown to be submodular.

2.5 Other Approaches to Summarization

In TextRank (Mihalcea and Tarau 2004) a document is represented as a graph where each sentence is denoted by a vertex and pairwise similarities between sentences are represented by edges with a weight corresponding to a word-overlap similarity between the sentences. The PageRank ranking algorithm is used on this graph to estimate the importance of different sentences. Classy 04 (Conroy et al. 2004) was the best performing system in the official DUC 2004 evaluations. After some linguistic preprocessing, they used a Hidden Markov Model with a mutual information score based on "signature tokens", tokens that are more likely to occur in the input document than in the rest of the corpus. ISCI (Gillick, Favre, and Hakkani-Tur 2008) optimized coverage of key bigrams, using integer linear programming. (Kulesza and Taskar 2012) presented the use of Determinantal Point Processes for summarization, a probabilistic formulation that allows for a balance between diversity and coverage. Occams V (Davis, Conroy, and Schlesinger 2012) is a system using latent semantic analysis and a sentence selection algorithm based on budgeted maximum coverage and knapsack. (Hong and Nenkova 2014) presented RegSum, a supervised logistic regression model for predicting word importance based on engineered features. Using these weights they presented an extractive summarizer that performed well and gave insight in word categories that are important for summary extraction. (Bonzanini, Martinez-Alvarez, and Roelleke 2013) presented an iterative procedure that removed redundant sentences from the input documents, until a sufficiently short summary was obtained. The system performed well in the evaluations on short online user reviews.

2.6 Evaluating Automatic Summarization Systems

Assessing the quality of summaries is an active area of research, and it is not entirely unlike the process of producing actual summaries, as the existing approaches have focused on scoring a summary for its similarity to a gold-standard written by human experts.

The NIST Document Understanding Conferences (DUC), was a conference with a competition for the summarization task. In 2008, it was superseded by the Text Understanding Conferences (TAC), and continued as a track in the new conference. The dataset from DUC 2004 has lived on as the standard benchmark dataset for generic multi-document summarization. It consists of 50 document clusters, each comprising around ten news articles (between 149 and 590 sentences each) and accompanied with four gold-standard summaries created by manual experts.

The Opinosis dataset (Ganesan, Zhai, and Han 2010) contains short online user-reviews for 51 different topics. Each topic contains between 50 and 575 sentences of reviews made by different authors about a certain characteristic of a hotel, car, or a product. As online user-reviews typically are not authored by professional writers, the quality of the text is lower, and it contains more opinionated sentences.

In DUC, a series of different approaches for evaluation have been used through the years in different combinations.

Initially, the evaluation of the systems participating in the competition were evaluated manually. This is of course a tedious task and a solution that doesn't scale well.

Later, ROUGE (C.-Y. Lin 2004) was used: a tool that produces a recall-based measure based on n-gram overlaps between the evaluated summary and a provided set of gold-standard summaries. Just as BLEU scores, ROUGE has received criticism for some drawbacks, such as failure to capture coherence and synonymity. Still, it is the prevalent score used today in the literature. In many publications, three different ROUGE scores are presented, ROUGE-1, ROUGE-2, and ROUGE-SU4, corresponding to matches in unigrams, bigrams and skip-bigrams (with up to four words in between), respectively.

Pyramid scores (Nenkova and Passonneau 2005) aim at capturing more of the semantics in the similarity between summaries. It, too relies on hand-crafted gold-standard summaries, but in contrast to ROUGE, it also needs manual work in the evaluation process, finding so-called *Summarization Content Units* in both generated and gold-standard summaries, which are matched to each other.

Many other systems for evaluating summaries have been proposed. Basic Elements (with implementations in ROUGE-BE and BEwT-E) is one of them, a system that extracts syntactic units of variable length, and provides transformations to determine equivalence of the extracted elements.

The methods mentioned above score the content of the summaries. To evaluate summaries for readability, coherency, and fluency, manual evaluations are usually made.

Chapter 3

Multi-Document Summarization and Semantic Relatedness

The following sections will outline the main contributions of this thesis.

Chapter 2 surveyed previous work on extractive summarization, and described that most summarization techniques rely on measuring similarity between language units, such as sentences. In this chapter, we will build on these ideas, and propose ways to measure sentence similarity that capture more semantic information, using ideas from word embeddings and sentiment analysis.

In Section 3.1, we show how summaries can be computed by using word embeddings to compute the sentence similarity score, in Section 3.2 we build on this and show that aggregating several ways of measuring the sentence similarity at the same time helps to create better extractive summaries. These results are illustrated by incorporating the ideas into the existing submodular optimization approach to summarization (see Section 2.4). Section 3.3 presents some possible ideas for future research, such as using information extraction to improve the performance of summarization systems. The two final sections in this chapter takes a slightly different view, approaching NLP problems with ideas from graph theory. Section 3.4 discusses the complexity of finding clusters by graph modification. Section 3.5 presents a novel approach to summarizing online user-reviews based on detecting bicliques in the bipartite word-document graph.

3.1 Extractive Summarization using Continuous Vector Space Representations

In the first paper, we present a novel way of utilizing word embeddings for summarization.

Word embeddings are representations for words in a vector space. Simple examples of this are context vectors, where you represent a word w by a vector v that has a dimensionality the same as the size of the vocabulary. All components of v are zero, except at indices representing words that have been observed in the context of w. This kind of model is called *distributional*, and it naturally encodes distributional similarities. The resulting representations are however very high-dimensional and sparse.

In contrast, the word embeddings that we consider in this paper are continuous vector space representations of much lower dimension (hence, they are also denser). This class of word embeddings are also known as *distributed representations*, and they typically use (or at least are inspired by) artificial neural networks for training. They are also distributional representations, as they encode similar words close in the vector space. Collobert and Weston (2008) trained the representations using a neural network in a multitask training scenario. In the Word2Vec Skip-Gram model (Mikolov, Chen, et al. 2013), the vectors are obtained by training a model to predict the context words. The GloVe model is trained using global word co-occurrence statistics (Pennington, Socher, and Manning 2014). Vectors from the models mentioned above have been shown to encode many semantic aspects of words. This shows as relations between the corresponding vectors; e.g. the difference between the vectors for a country and its capital is almost constant over different countries:

$$v_{Stockholm} - v_{Sweden} \approx v_{Berlin} - v_{Germany}$$

We use the word embeddings to capture more semantics in the sentence similarity score. To be able to easily compare two sentences, we first create a sentence representation of fixed dimension for each sentence. We evaluate two different ways of doing this: simple vector addition (Mikolov, Sutskever, et al. 2013) and recursive auto-encoders (RAE) (Socher et al. 2011).

An RAE is an auto-encoder: a feed forward neural network that is trained to reconstruct the input at its output. In an auto-encoder, there is typically a lower-dimensional hidden layer, to obtain an amount of compression. A recursive auto-encoder uses the syntactic parse-tree of a sentence, to provide the layout of the auto-encoder network. In this way, intermediate representations are trained recursively for pairs of input representations. The dimensionality is the same after each pairwise input.

In the evaluation, two different word representations are used: those of (Mikolov, Chen, et al. 2013), and (Collobert and Weston 2008). Extending the representations to more complex structures than words (such as sentences and documents) is an active area of research.

The experimental evaluation suggests that using continuous vector space models affects the resulting summaries favourably, but surprisingly, the simple approach of summing the vectors to sentence representations outperforms the seemingly more sophisticated approach with RAEs.

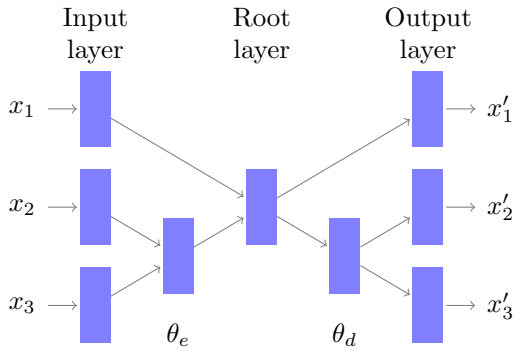


Figure 3.1.1: *An unfolding recursive auto-encoder. In this example, a three words phrase $[x_1, x_2, x_3]$ is the input. The weight matrix θ_e is used to encode the compressed representations, while θ_d is used to decode the representations and reconstruct the sentence.*

3.2 Extractive Summarization by Aggregating Multiple Similarities

This paper builds on the approach with word representations (Section 3.1). We also present a sentence similarity score based on sentiment analysis, and show that we can create better summaries by combining the word embeddings with the sentiment score and traditional word-overlap measures, by multiplying the scores together. Using multiplication corresponds to a conjunctive aggregation, where all involved similarity scores need to be high for the resulting score to be high, but only one of them needs to be low for the resulting score to be low. Products of experts have been shown to work well in many other applications and is a standard way of combining kernels.

It has been shown (Hong and Nenkova 2014) that when human authors create summaries for news paper articles, they use negative emotion words at a higher rate. This motivated our experiments with sentiment words. We define a new sentence similarity measure, comparing the sentiment charge of two sentences. Much as expected, this sentence similarity measure is in itself weak, but when used in aggregation with other similarity measures, it can provide an improvement.

We call the resulting system MULTSUM, an automatic summarizer that uses submodular optimization with multiplicative aggregation of sentence similarity scores, taking several aspects into account when selecting sentences.

$$M_{\mathbf{s}_i,\mathbf{s}_j} = \prod M^l_{\mathbf{s}_i,\mathbf{s}_j},$$

where M^l denotes the different similarity measures used.

The resulting summaries are evaluated on DUC 2004; the de-facto standard benchmark dataset for generic multi-document summarization, and obtain state–of–the–art results. Specifically, the scores where MULTSUM excels, are ROUGE-2 and ROUGE-SU4, with matches in bigrams and skip-bigrams (with up to four words in between), respectively, suggesting that the resulting summaries have high fluency.

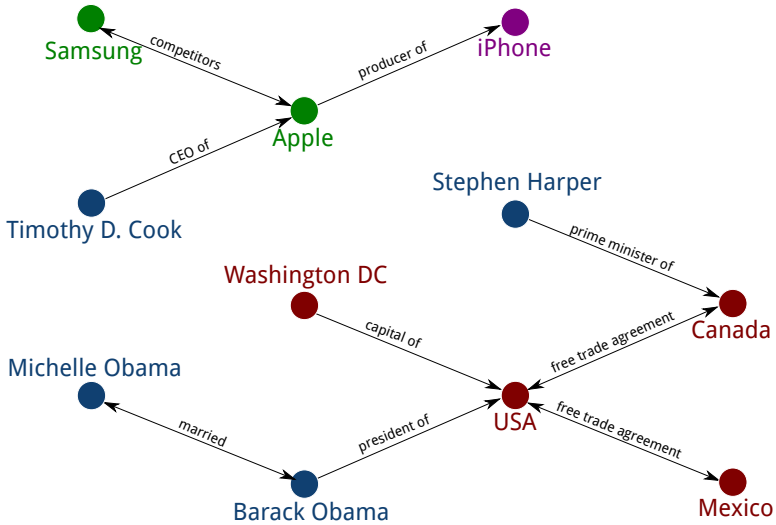


Figure 3.3.1: *Example of a graph of entities and relations. Entities are of different types, and are represented by nodes in the graph. The edges corresponds to (different kinds of) relations between entities.*

3.3 Visions and Open Challenges for a Knowledge–Based Culturomics

In this white paper we discuss automatic summarization in the setting of historical texts, and present ideas for making automatic summaries more aware of what is important in the text. Information extraction (IE) is the process of extracting entities, relations, and events from the text. The extracted information would be used as a first step in a summarization system to improve the information content in the summaries. Previous work has shown that incorporating IE can improve the results of multi-document summarization (Ji et al. 2013).

Entities, relations and events extracted from the documents are used to score sentences for extraction; a higher score is given to sentences with salient information units. Furthermore, temporal information in the input documents help create summaries that are fluent and keep the narrative. As a continuation of this, one can envision an abstractive summarization system that incorporates extracted information.

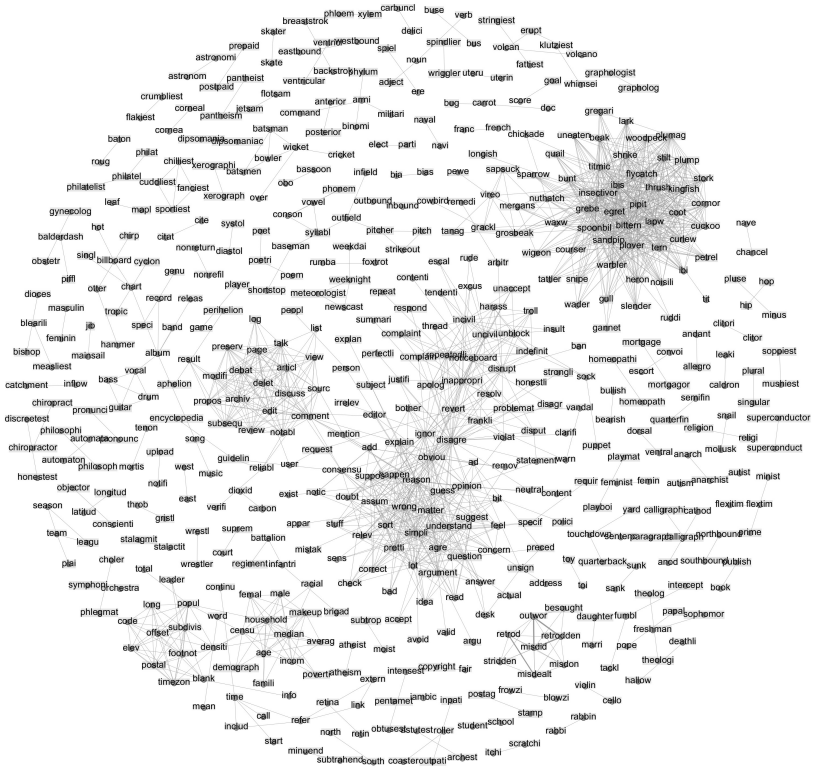


Figure 3.3.2: *A subset of the word co-occurrence graph generated using data from Wikipedia, using some edge weight threshold t. The visualization is created with Gephi. One can clearly see dense word-clusters forming with only few edges connecting clusters with each other.*

3.4 Editing Simple Graphs

Graph editing is the process of modifying an input graph by operations such as inserting, removing, or reversing edges until some prescribed property of the graph is met. This paper discusses the complexity of editing an input graph G, into a graph with a predefined critical-clique graph. A clique graph is a representation of the cliques of a graph, and the relationship between them. A critical-clique is an equivalence class defined by vertices that share the same neighborhood. Consequently, a critical-clique graph has a vertex for each critical clique, and edges between two vertices if the corresponding cliques in the original graph are connected.

The work in this paper is motivated by the wish to find clusters in word-co-occurrence graphs (see Figure 3.3.2 for a visualization of a word co-occurrence graph from Wikipedia data). Applications for this can be topic modelling or measures of similarity between words. The distributional hypothesis (Harris 1954) states that semantically similar words occur within similar contexts. Hence, the similarity found in this way could be a coarse variant of the kind of similarity found in word embeddings, and hence usable for the kind of semantic similarity tasks discussed for summarization. Using such clusters however, has been out of the scope of this thesis and is left for future work. The word "simple" in the title refers to the fact that we discuss some simple cases of graphs as the target (critical-clique) graph, corresponding to finding "simple" structures of clusters in the input graph.

We show that several variants of this problem is in SUBEPT, the complexity class of fixed parameter tractable problems solvable in subexponential time in the parameter ($2^{o(k)}$). In this case, the parameter k represents the number of edits. Of course, the time complexity also depends on the input size n, but only polynomially.

3.5 Summarizing Online User Reviews by Computing Bicliques

In this paper, we approach the problem of extractive summarization for short (one sentence) online user-reviews by considering the bipartite graph of documents and words (for a document vertex v, there is an edge between v and every word included in the corresponding document). The result is a simple combinatorial algorithm for producing an extractive summary based on bicliques in the document-word graph. A bipartite graph is a graph where the vertices can be divided into two sets, T and W, and edges must have exactly one endpoint in each of these sets. A bipartite clique, or *biclique* is a subset of the vertices in T and W that induces a complete bipartite subgraph (where every vertex in the first set is connected to every vertex in the second set). We extract bicliques where the number of sentences is two, then score them by the number of words that the sentences share. Thereafter sentences are scored by the number of bicliques they are part of, and finally the most representative sentences are presented as a summary.

The method scores well when compared to other methods on short online user-reviews, and due to a clever way to exploit statistical properties of word-frequencies (they follow power-laws), a very good running time can be obtained in practice.

Chapter 4

Conclusions

This thesis has discussed extractive multi-document summarization. Most extractive summarization systems rely on measuring the similarity of language units (typically sentences).

We have defined novel ways of measuring the sentence similarity score by leveraging word embeddings and sentiment analysis. When aggregating several similarity scores at the same time, using element-wise multiplication, our system obtains state–of–the–art results on standard benchmark datasets.

We have also discussed using information extraction to add temporal information and improve summarization systems in the setting of historical texts.

Finally, we have proposed graph-based approaches for clustering of words and for extracting sentences for summaries of online user-reviews. We show that both are fast (in the first case, we have a theoretical motivation, in the second case the motivation is backed by statistical properties of words).

So far, we have only scratched the surface of learning how to construct NLP systems that really work on the semantic level. Plenty remains to be done, such as exploring how to best create distributional representations for sentences. Recent work has been studying different ways to compose word embeddings to create sentence representations (Blacoe and Lapata 2012; Hermann and Blunsom 2013). Perhaps an even more promising direction is the use of deep learning and recurrent neural networks (Graves 2013). Such models have been successfully used for machine translation (Sutskever, Vinyals, and Le 2014) and could be useful both for extractive and abstractive summarization in an end–to–end deep learning approach.

References

Bengtsson, J. and C. Skeppstedt (2012). "Automatic extractive single document summarization". MA thesis. Chalmers University of Technology and University of Gothenburg. URL: http://publications.lib.chalmers.se/records/fulltext/174136/174136.pdf.

Blacoe, W. and M. Lapata (2012). "A Comparison of Vector-based Representations for Semantic Composition". *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*. EMNLP-CoNLL '12. Jeju Island, Korea: Association for Computational Linguistics, pp. 546–556. URL: http://dl.acm.org/citation.cfm?id=2390948.2391011.

Bonzanini, M., M. Martinez-Alvarez, and T. Roelleke (2013). "Extractive summarisation via sentence removal: condensing relevant sentences into a short summary". *SIGIR*, pp. 893–896.

Collobert, R. and J. Weston (2008). "A unified architecture for natural language processing: Deep neural networks with multitask learning". *Proceedings of ICML*, pp. 160–167.

Conroy, J. M. et al. (2004). "Left-brain/right-brain multi-document summarization". *Proceedings of DUC 2004*.

Damaschke, P. and O. Mogren (2014). Editing Simple Graphs. *Journal of Graph Algorithms and Applications, Special Issue of selected papers from WALCOM 2014* **18**.4, 557–576. DOI: 10.7155/jgaa.00337.

Davis, S. T., J. M. Conroy, and J. D. Schlesinger (2012). "OCCAMS–An Optimal Combinatorial Covering Algorithm for Multi-document Summarization". *Data Mining Workshops (ICDMW)*. IEEE, pp. 454–463.

Ganesan, K., C. Zhai, and J. Han (2010). "Opinosis: a graph-based approach to abstractive summarization of highly redundant opinions". *Proceedings of the 23rd International Conference on Computational Linguistics*. ACL, pp. 340–348.

Gillick, D., B. Favre, and D. Hakkani-Tur (2008). "The icsi summarization system at tac 2008". *Proceedings of TAC*.

Graves, A. (2013). Generating sequences with recurrent neural networks. *arXiv preprint arXiv:1308.0850*.

Harris, Z. S. (1954). Distributional structure. *Word*.

Hermann, K. M. and P. Blunsom (2013). "The Role of Syntax in Vector Space Models of Compositional Semantics." *ACL*, pp. 894–904.

Hong, K. and A. Nenkova (2014). "Improving the estimation of word importance for news multi-document summarization". *Proceedings of EACL*.

Ji, H. et al. (2013). "Open-Domain multi-document summarization via information extraction: Challenges and prospects". *Multi-source, Multilingual Information Extraction and Summarization*. Springer, pp. 177–201.

Kågebäck, M. et al. (2014). "Extractive Summarization using Continuous Vector Space Models". *Proceedings of The Second Workshop of Continuous Vector Space Models and their Compositionality*, pp. 31–39.

Kulesza, A. and B. Taskar (2012). Determinantal point processes for machine learning. *arXiv:1207.6083*.

Levenshtein, V. I. (1966). "Binary codes capable of correcting deletions, insertions, and reversals". *Soviet physics doklady*. Vol. 10. 8, pp. 707–710.

Lin, C.-Y. (2004). "Rouge: A package for automatic evaluation of summaries". *Text Summarization Branches Out: Proc. of the ACL-04 Workshop*, pp. 74–81.

Lin, H. and J. Bilmes (2011). "A Class of Submodular Functions for Document Summarization." *ACL*.

Luhn, H. P. (1958). The automatic creation of literature abstracts. *IBM Journal* **2**.2, 159–165.

Mihalcea, R. and P. Tarau (2004). "TextRank: Bringing order into texts". *Proceedings of EMNLP*. Vol. 4.

Mikolov, T., K. Chen, et al. (2013). Efficient Estimation of Word Representations in Vector Space. *arXiv:1301.3781*.

Mikolov, T., I. Sutskever, et al. (2013). "Distributed representations of words and phrases and their compositionality". *Advances in Neural Information Processing Systems*, pp. 3111–3119.

Mogren, O., M. Kågebäck, and D. Dubhashi (2015). "Extractive Summarization by Aggregating Multiple Similarities". *Proceedings of Recent Advances in Natural Language Processing*, pp. 451–457.

Muhammad, A. S., P. Damaschke, and O. Mogren (2016). "Summarizing Online User Reviews Using Bicliques". *Proceedings of The 42nd International Conference on Current Trends in Theory and Practice of Computer Science, SOFSEM, LNCS*.

Nenkova, A. and R. J. Passonneau (2005). "Evaluating Content Selection in Summarization: The Pyramid Method." *HLT-NAACL*, pp. 145–152. URL: `http://dblp.uni-trier.de/db/conf/naacl/naacl2004.html#NenkovaP04`.

Pennington, J., R. Socher, and C. D. Manning (2014). Glove: Global vectors for word representation. *Proceedings of the Empiricial Methods in Natural Language Processing (EMNLP 2014)* **12**, 1532–1543.

Radev, D. R. et al. (2004). Centroid-based summarization of multiple documents. *Information Processing & Management* **40**.6, 919–938.

Socher, R. et al. (2011). "Dynamic Pooling and Unfolding Recursive Autoencoders for Paraphrase Detection." *Advances in Neural Information Processing Systems*. Vol. 24, pp. 801–809.

Sutskever, I., O. Vinyals, and Q. V. Le (2014). "Sequence to sequence learning with neural networks". *Advances in neural information processing systems*, pp. 3104–3112.

Tahmasebi, N. et al. (2015). Visions and open challenges for a knowledge-based culturomics. *International Journal on Digital Libraries* **15**.2-4, 169–187.

Tai, K. S., R. Socher, and C. D. Manning (2015). Improved semantic representations from tree-structured long short-term memory networks. *arXiv preprint arXiv:1503.00075*.

Zhao, J., T. T. Zhu, and M. Lan (2014). Ecnu: One stone two birds: Ensemble of heterogenous measures for semantic relatedness and textual entailment. *SemEval 2014*, 271.

Part II
Publications

Paper I

Extractive Summarization using Continuous Vector Space Models

M. Kågebäck et al.

Reprinted from Proceedings of The Second Workshop of Continuous Vector Space Models and their Compositionality, **2014**

Extractive Summarization using Continuous Vector Space Models

Mikael Kågebäck, Olof Mogren, Nina Tahmasebi, Devdatt Dubhashi
Computer Science & Engineering
Chalmers University of Technology
SE-412 96, Göteborg
{kageback, mogren, ninat, dubhashi}@domain

Abstract

Automatic summarization can help users extract the most important pieces of information from the vast amount of text digitized into electronic form everyday. Central to automatic summarization is the notion of similarity between sentences in text. In this paper we propose the use of continuous vector representations for semantically aware representations of sentences as a basis for measuring similarity. We evaluate different compositions for sentence representation on a standard dataset using the ROUGE evaluation measures. Our experiments show that the evaluated methods improve the performance of a state-of-the-art summarization framework and strongly indicate the benefits of continuous word vector representations for automatic summarization.

1 Introduction

The goal of summarization is to capture the important information contained in large volumes of text, and present it in a brief, representative, and consistent summary. A well written summary can significantly reduce the amount of work needed to digest large amounts of text on a given topic. The creation of summaries is currently a task best handled by humans. However, with the explosion of available textual data, it is no longer financially possible, or feasible, to produce all types of summaries by hand. This is especially true if the subject matter has a narrow base of interest, either due to the number of potential readers or the duration during which it is of general interest. A summary describing the events of World War II might for instance be justified to create manually, while a summary of all reviews and comments regarding a certain version of Windows might not. In such cases, automatic summarization is a way forward.

In this paper we introduce a novel application of continuous vector representations to the problem of multi-document summarization. We evaluate different compositions for producing sentence representations based on two different word embeddings on a standard dataset using the ROUGE evaluation measures. Our experiments show that the evaluated methods improve the performance of a state-of-the-art summarization framework which strongly indicate the benefits of continuous word vector representations for this tasks.

2 Summarization

There are two major types of automatic summarization techniques, extractive and abstractive. *Extractive summarization* systems create summaries using representative sentences chosen from the input while *abstractive summarization* creates new sentences and is generally considered a more difficult problem.

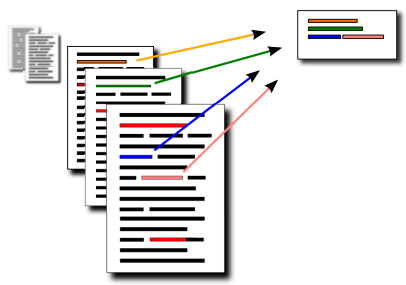


Figure 1: Illustration of Extractive Multi-Document Summarization.

For this paper we consider extractive multi-document summarization, that is, sentences are chosen for inclusion in a summary from a set of documents D. Typically, extractive summarization techniques can be divided into two components, the summarization framework and the similarity measures used to compare sentences. Next

we present the algorithm used for the framework and in Sec. 2.2 we discuss a typical sentence similarity measure, later to be used as a baseline.

2.1 Submodular Optimization

Lin and Bilmes (2011) formulated the problem of extractive summarization as an optimization problem using monotone nondecreasing submodular set functions. A submodular function F on the set of sentences V satisfies the following property: for any $A \subseteq B \subseteq V \backslash \{v\}$, $F(A + \{v\}) - F(A) \geq F(B + \{v\}) - F(B)$ where $v \in V$. This is called the diminishing returns property and captures the intuition that adding a sentence to a small set of sentences (i.e., summary) makes a greater contribution than adding a sentence to a larger set. The aim is then to find a summary that maximizes diversity of the sentences and the coverage of the input text. This objective function can be formulated as follows:

$$\mathcal{F}(S) = \mathcal{L}(S) + \lambda \mathcal{R}(S)$$

where S is the summary, $\mathcal{L}(S)$ is the coverage of the input text, $\mathcal{R}(S)$ is a diversity reward function. The λ is a trade-off coefficient that allows us to define the importance of coverage versus diversity of the summary. In general, this kind of optimization problem is NP-hard, however, if the objective function is submodular there is a fast scalable algorithm that returns an approximation with a guarantee. In the work of Lin and Bilmes (2011) a simple submodular function is chosen:

$$\mathcal{L}(S) = \sum_{i \in V} \min\{\sum_{j \in S} \mathrm{Sim}(i,j), \alpha \sum_{j \in V} \mathrm{Sim}(i,j)\} \tag{1}$$

The first argument measures similarity between sentence i and the summary S, while the second argument measures similarity between sentence i and the rest of the input V. $\mathrm{Sim}(i,j)$ is the similarity between sentence i and sentence j and $0 \leq \alpha \leq 1$ is a threshold coefficient. The diversity reward function $\mathcal{R}(S)$ can be found in (Lin and Bilmes, 2011).

2.2 Traditional Similarity Measure

Central to most extractive summarization systems is the use of sentence similarity measures ($\mathrm{Sim}(i,j)$ in Eq. 1). Lin and Bilmes measure similarity between sentences by representing each sentence using *tf-idf* (Salton and McGill, 1986) vectors and measuring the cosine angle between vectors. Each sentence is represented by a word vector $\mathbf{w} = (w_1, \ldots, w_N)$ where N is the size of the vocabulary. Weights w_{ki} correspond to the *tf-idf* value of word k in the sentence i. The weights $\mathrm{Sim}(i,j)$ used in the $\mathcal{L}$ function in Eq. 1 are found using the following similarity measure.

$$\mathrm{Sim}(i,j) = \frac{\sum_{w \in i} \mathrm{tf}_{w,i} \times \mathrm{tf}_{w,j} \times \mathrm{idf}_w^2}{\sqrt{\sum_{w \in i} \mathrm{tf}_{w,i}^2 \times \mathrm{idf}_w^2} \sqrt{\sum_{w \in j} \mathrm{tf}_{w,j}^2 \times \mathrm{idf}_w^2}} \tag{2}$$

where $\mathrm{tf}_{w,i}$ and $\mathrm{tf}_{w,j}$ are the number of occurrences of w in sentence i and j, and idf_w is the inverse document frequency (*idf*) of w.

In order to have a high similarity between sentences using the above measure, two sentences must have an overlap of highly scored *tf-idf* words. The overlap must be exact to count towards the similarity, e.g, the terms *The US President* and *Barack Obama* in different sentences will not add towards the similarity of the sentences. To capture deeper similarity, in this paper we will investigate the use of continuous vector representations for measuring similarity between sentences. In the next sections we will describe the basics needed for creating continuous vector representations and methods used to create sentence representations that can be used to measure sentence similarity.

3 Background on Deep Learning

Deep learning (Hinton et al., 2006) is a modern interpretation of artificial neural networks (ANN), with an emphasis on deep network architectures. Deep learning can be used for challenging problems like image and speech recognition (Krizhevsky et al., 2012; Graves et al., 2013), as well as language modeling (Mikolov et al., 2010), and in all cases, able to achieve state-of-the-art results.

Inspired by the brain, ANNs use a neuron-like construction as their primary computational unit. The behavior of a neuron is entirely controlled by its input weights. Hence, the weights are where the information learned by the neuron is stored. More precisely the output of a neuron is computed as the weighted sum of its inputs, and squeezed into the interval $[0,1]$ using a sigmoid function:

$$y_i = g(\theta_i^T \mathbf{x}) \tag{3}$$

$$g(z) = \frac{1}{1 + e^{-z}} \tag{4}$$

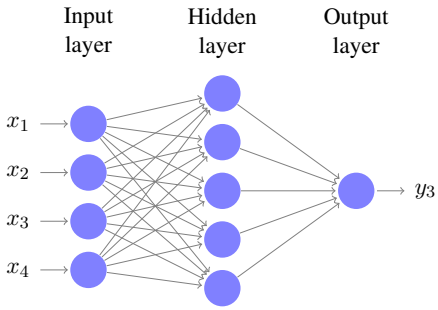


Figure 2: FFNN with four input neurons, one hidden layer, and 1 output neuron. This type of architecture is appropriate for binary classification of some data $\mathbf{x} \in \mathbb{R}^4$, however depending on the complexity of the input, the number and size of the hidden layers should be scaled accordingly.

where θ_i are the weights associated with neuron i and $\mathbf{x}$ is the input. Here the sigmoid function (g) is chosen to be the logistic function, but it may also be modeled using other sigmoid shaped functions, e.g. the hyperbolic tangent function.

The neurons can be organized in many different ways. In some architectures, loops are permitted. These are referred to as *recurrent neural networks*. However, all networks considered here are non-cyclic topologies. In the rest of this section we discuss a few general architectures in more detail, which will later be employed in the evaluated models.

3.1 Feed Forward Neural Network

A feed forward neural network (FFNN) (Haykin, 2009) is a type of ANN where the neurons are structured in layers, and only connections to subsequent layers are allowed, see Figure 2. The algorithm is similar to logistic regression using non-linear terms. However, it does not rely on the user to choose the non-linear terms needed to fit the data, making it more adaptable to changing datasets. The first layer in a FFNN is called the input layer, the last layer is called the output layer, and the interim layers are called hidden layers. The hidden layers are optional but necessary to fit complex patterns.

Training is achieved by minimizing the network error (E). How E is defined differs between different network architectures, but is in general a differentiable function of the produced output and

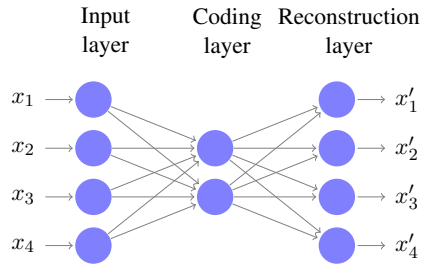


Figure 3: The figure shows an auto-encoder that compresses four dimensional data into a two dimensional code. This is achieved by using a bottleneck layer, referred to as a coding layer.

the expected output. In order to minimize this function the gradient $\frac{\partial E}{\partial \Theta}$ first needs to be calculated, where Θ is a matrix of all parameters, or weights, in the network. This is achieved using backpropagation (Rumelhart et al., 1986). Secondly, these gradients are used to minimize E using e.g. gradient descent. The result of this processes is a set of weights that enables the network to do the desired input-output mapping, as defined by the training data.

3.2 Auto-Encoder

An auto-encoder (AE) (Hinton and Salakhutdinov, 2006), see Figure 3, is a type of FFNN with a topology designed for dimensionality reduction. The input and the output layers in an AE are identical, and there is at least one hidden bottleneck layer that is referred to as the coding layer. The network is trained to reconstruct the input data, and if it succeeds this implies that all information in the data is necessarily contained in the compressed representation of the coding layer.

A shallow AE, i.e. an AE with no extra hidden layers, will produce a similar code as principal component analysis. However, if more layers are added, before and after the coding layer, non-linear manifolds can be found. This enables the network to compress complex data, with minimal loss of information.

3.3 Recursive Neural Network

A recursive neural network (RvNN), see Figure 4, first presented by Socher et al. (2010), is a type of feed forward neural network that can process data through an arbitrary binary tree structure, e.g. a

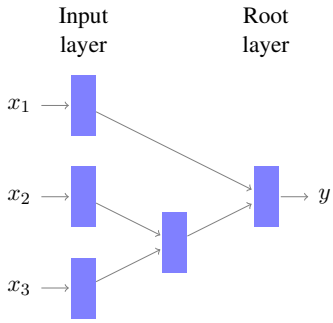


Figure 4: The recursive neural network architecture makes it possible to handle variable length input data. By using the same dimensionality for all layers, arbitrary binary tree structures can be recursively processed.

binary parse tree produced by linguistic parsing of a sentence. This is achieved by enforcing weight constraints across all nodes and restricting the output of each node to have the same dimensionality as its children.

The input data is placed in the leaf nodes of the tree, and the structure of this tree is used to guide the recursion up to the root node. A compressed representation is calculated recursively at each non-terminal node in the tree, using the same weight matrix at each node. More precisely, the following formulas can be used:

$$z_p = \theta_p^T [x_l; x_r] \tag{5a}$$

$$y_p = g(z_p) \tag{5b}$$

where y_p is the computed parent state of neuron p, and z_p the induced field for the same neuron. $[x_l; x_r]$ is the concatenation of the state belonging to the right and left sibling nodes. This process results in a fixed length representation for hierarchical data of arbitrary length. Training of the model is done using backpropagation through structure, introduced by Goller and Kuchler (1996).

4 Word Embeddings

Continuous distributed vector representation of words, also referred to as word embeddings, was first introduced by Bengio et al. (2003). A word embedding is a continuous vector representation that captures semantic and syntactic information about a word. These representations can be used to unveil dimensions of similarity between words, e.g. singular or plural.

4.1 Collobert & Weston

Collobert and Weston (2008) introduce an efficient method for computing word embeddings, in this work referred to as *CW* vectors. This is achieved firstly, by scoring a valid n-gram ($\mathbf{x}$) and a corrupted n-gram ($\bar{\mathbf{x}}$) (where the center word has been randomly chosen), and secondly, by training the network to distinguish between these two n-grams. This is done by minimizing the hinge loss

$$\max(0, 1 - s(\mathbf{x}) + s(\bar{\mathbf{x}})) \tag{6}$$

where s is the scoring function, i.e. the output of a FFNN that maps between the word embeddings of an n-gram to a real valued score. Both the parameters of the scoring function and the word embeddings are learned in parallel using backpropagation.

4.2 Continuous Skip-gram

A second method for computing word embeddings is the Continuous Skip-gram model, see Figure 5, introduced by Mikolov et al. (2013a). This model is used in the implementation of their word embeddings tool *Word2Vec*. The model is trained to predict the context surrounding a given word. This is accomplished by maximizing the objective function

$$\frac{1}{T} \sum_{t=1}^{T} \sum_{-c \leq j \leq c, j \neq 0} \log p(w_{t+j} | w_t) \tag{7}$$

where T is the number of words in the training set, and c is the length of the training context. The probability $p(w_{t+j}|w_t)$ is approximated using the hierarchical softmax introduced by Bengio et al. (2002).

5 Phrase Embeddings

Word embeddings have proven useful in many natural language processing (NLP) tasks. For summarization, however, sentences need to be compared. In this section we present two different methods for deriving phrase embeddings, which in Section 5.3 will be used to compute sentence to sentence similarities.

5.1 Vector addition

The simplest way to represent a sentence is to consider it as the sum of all words without regarding word orders. This was considered by Mikolov et al. (2013b) for representing short

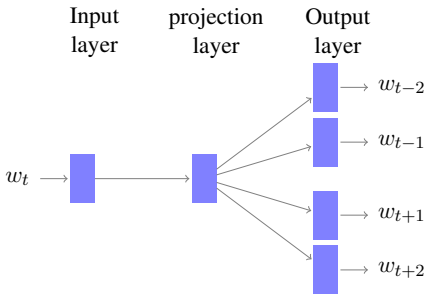


Figure 5: The continuous Skip-gram model. Using the input word (w_t) the model tries to predict which words will be in its context ($w_{t\pm c}$).

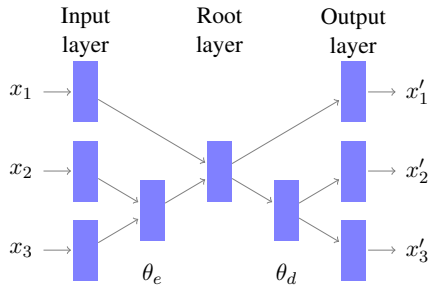


Figure 6: The structure of an unfolding RAE, on a three word phrase ($[x_1, x_2, x_3]$). The weight matrix θ_e is used to encode the compressed representations, while θ_d is used to decode the representations and reconstruct the sentence.

phrases. The model is expressed by the following equation:

$$\mathbf{x}_p = \sum_{\mathbf{x}_w \in \{\text{sentence}\}} \mathbf{x}_w \tag{8}$$

where x_p is a phrase embedding, and x_w is a word embedding. We use this method for computing phrase embeddings as a baseline in our experiments.

5.2 Unfolding Recursive Auto-encoder

The second model is more sophisticated, taking into account also the order of the words and the grammar used. An unfolding recursive auto-encoder (RAE) is used to derive the phrase embedding on the basis of a binary parse tree. The unfolding RAE was introduced by Socher et al. (2011) and uses two RvNNs, one for encoding the compressed representations, and one for decoding them to recover the original sentence, see Figure 6. The network is subsequently trained by minimizing the reconstruction error.

Forward propagation in the network is done by recursively applying Eq. 5a and 5b for each triplet in the tree in two phases. First, starting at the center node (root of the tree) and recursively pulling the data from the input. Second, again starting at the center node, recursively pushing the data towards the output. Backpropagation is done in a similar manner using backpropagation through structure (Goller and Kuchler, 1996).

5.3 Measuring Similarity

Phrase embeddings provide semantically aware representations for sentences. For summarization, we need to measure the similarity between two representations and will make use of the following two vector similarity measures. The first similarity measure is the cosine similarity, transformed to the interval of $[0, 1]$

$$\text{Sim}(i,j) = \left(\frac{\mathbf{x}_i^T \mathbf{x}_j}{\|\mathbf{x}_j\| \|\mathbf{x}_j\|} + 1 \right) /2 \tag{9}$$

where $\mathbf{x}$ denotes a phrase embedding The second similarity is based on the complement of the Euclidean distance and computed as:

$$\text{Sim}(i,j) = 1 - \frac{1}{\max\limits_{k,n} \sqrt{\| \mathbf{x}_k - \mathbf{x}_n \|^2}} \sqrt{\| \mathbf{x}_j - \mathbf{x}_i \|^2} \tag{10}$$

6 Experiments

In order to evaluate phrase embeddings for summarization we conduct several experiments and compare different phrase embeddings with *tf-idf* based vectors.

6.1 Experimental Settings

Seven different configuration were evaluated. The first configuration provides us with a baseline and is denoted *Original* for the Lin-Bilmes method described in Sec. 2.1. The remaining configurations comprise selected combinations of word embeddings, phrase embeddings, and similarity measures.

The first group of configurations are based on vector addition using both Word2Vec and CW vectors. These vectors are subsequently compared using both cosine similarity and Euclidean distance.

The second group of configurations are built upon recursive auto-encoders using CW vectors and are also compared using cosine similarity as well as Euclidean distance.

The methods are named according to: VectorType_EmbeddingMethod$_{SimilarityMethod}$, e.g. W2V_Add$_{Cos}$ for Word2Vec vectors combined using vector addition and compared using cosine similarity.

To get an upper bound for each ROUGE score, we use an exhaustive search on summaries. We evaluated each possible pair of sentences and maximized w.r.t the ROUGE score.

6.2 Dataset and Evaluation

The Opinosis dataset (Ganesan et al., 2010) consists of short user reviews in 51 different topics. Each of these topics contains between 50 and 575 sentences and are a collection of user reviews made by different authors about a certain characteristic of a hotel, car or a product (e.g. *"Location of Holiday Inn, London"* and *"Fonts, Amazon Kindle"*). The dataset is well suited for multi-document summarization (each sentence is considered its own document), and includes between 4 and 5 gold-standard summaries (not sentences chosen from the documents) created by human authors for each topic.

Each summary is evaluated with ROUGE, that works by counting word overlaps between generated summaries and gold standard summaries. Our results include R-1, R-2, and R-SU4, which counts matches in unigrams, bigrams, and skip-bigrams respectively. The skip-bigrams allow four words in between (Lin, 2004).

The measures reported are recall (R), precision (P), and F-score (F), computed for each topic individually and averaged. Recall measures what fraction of a human created gold standard summaries that is captured, and precision measures what fraction of the generated summary that is in the gold standard. F-score is a standard way to combine recall and precision, computed as $F = 2\frac{P*R}{P+R}$.

6.3 Implementation

All results were obtained by running an implementation of Lin-Bilmes submodular optimization summarizer, as described in Sec. 2.1. Also, we have chosen to fix the length of the summaries to two sentences because the length of the gold-standard summaries are typically around two sentences. The CW vectors used were trained by Turian et al. (2010)[1], and the Word2Vec vectors by Mikolov et al. (2013b)[2]. The unfolding RAE used is based on the implementation by Socher et al. (2011)[3], and the parse trees for guiding the recursion was generated using the Stanford Parser (Klein and Manning, 2003)[4].

6.4 Results

The results from the ROUGE evaluation are compiled in Table 1. We find for all measures (recall, precision, and F-score), that the phrase embeddings outperform the original Lin-Bilmes. For recall, we find that CW_Add$_{Cos}$ achieves the highest result, while for precision and F-score the CW_Add$_{Euc}$ perform best. These results are consistent for all versions of ROUGE scores reported (1, 2 and SU4), providing a strong indication for phrase embeddings in the context of automatic summarization.

Unfolding RAE on CW vectors and vector addition on W2V vectors gave comparable results w.r.t. each other, generally performing better than original Linn-Bilmes but not performing as well as vector addition of CW vectors.

The results denoted OPT in Table 1 describe the upper bound score, where each row represents optimal recall and F-score respectively. The best results are achieved for R-1 with a maximum recall of 57.86%. This is a consequence of hand created gold standard summaries used in the evaluation, that is, we cannot achieve full recall or F-score when the sentences in the gold standard summaries are not taken from the underlying documents and thus, they can never be fully matched using extractive summarization. R-2 and SU4 have lower maximum recall and F-score, with 22.9% and 29.5% respectively.

6.5 Discussion

The results of this paper show great potential for employing word and phrase embeddings in summarization. We believe that by using embeddings we move towards more semantically aware summarization systems. In the future, we anticipate improvements for the field of automatic summarization (as well as for similar applications), as the quality of the word vectors improves and we find

[1] http://metaoptimize.com/projects/wordreprs/

[2] https://code.google.com/p/word2vec/

[3] http://nlp.stanford.edu/ socherr/codeRAEVectorsNIPS2011.zip

[4] http://nlp.stanford.edu/software/lex-parser.shtml

Table 1: ROUGE scores for summaries using different similarity measures. OPT constitutes the optimal ROUGE scores on this dataset.

ROUGE-1			
	R	**P**	**F**
OPT_R	**57.86**	21.96	30.28
OPT_F	45.93	48.84	**46.57**
CW_RAE_{Cos}	27.37	19.89	22.00
CW_RAE_{Euc}	29.25	19.77	22.62
CW_Add_{Cos}	**34.72**	11.75	17.16
CW_Add_{Euc}	29.12	**22.75**	**24.88**
$W2V\_Add_{Cos}$	30.86	16.81	20.93
$W2V\_Add_{Euc}$	28.71	16.67	20.75
Original	25.82	19.58	20.57
ROUGE-2			
	R	**P**	**F**
OPT_R	**22.96**	12.31	15.33
OPT_F	20.42	19.94	**19.49**
CW_RAE_{Cos}	4.68	3.18	3.58
CW_RAE_{Euc}	4.82	3.24	3.67
CW_Add_{Cos}	**5.89**	1.81	2.71
CW_Add_{Euc}	5.12	**3.60**	**4.10**
$W2V\_Add_{Cos}$	5.71	3.08	3.82
$W2V\_Add_{Euc}$	3.86	1.95	2.54
Original	3.92	2.50	2.87
ROUGE-SU4			
	R	**P**	**F**
OPT_R	**29.50**	13.53	17.70
OPT_F	23.17	26.50	**23.70**
CW_RAE_{Cos}	9.61	6.23	6.95
CW_RAE_{Euc}	9.95	6.17	7.04
CW_Add_{Cos}	**12.38**	3.27	5.03
CW_Add_{Euc}	10.54	**7.59**	**8.35**
$W2V\_Add_{Cos}$	11.94	5.52	7.12
$W2V\_Add_{Euc}$	9.78	4.69	6.15
Original	9.15	6.74	6.73

enhanced ways of composing and comparing the vectors.

It is interesting to compare the results from different composition techniques on the CW vectors, where vector addition surprisingly outperforms the considerably more sophisticated unfolding RAE. However, since the unfolding RAE uses syntactic information in the text, this may be a result of using a dataset consisting of low quality text.

In the interest of comparing word embeddings, results using vector addition and cosine similarity were computed based on both CW and Word2Vec vectors. Supported by the achieved results CW vectors seems better suited for sentence similarities in this setting.

An issue we encountered with using precomputed word embeddings was their limited vocabulary, in particular missing uncommon (or common incorrect) spellings. This problem is particularly pronounced on the evaluated Opinosis dataset, since the text is of low quality. Future work is to train word embeddings on a dataset used for summarization to better capture the specific semantics and vocabulary.

The optimal R-1 scores are higher than R-2 and SU4 (see Table 1) most likely because the score ignores word order and considers each sentence as a set of words. We come closest to the optimal score for R-1, where we achieve 60% of maximal recall and 49% of F-score. Future work is to investigate why we achieve a much lower recall and F-score for the other ROUGE scores.

Our results suggest that the phrase embeddings capture the kind of information that is needed for the summarization task. The embeddings are the underpinnings of the decisions on which sentences that are representative of the whole input text, and which sentences that would be redundant when combined in a summary. However, the fact that we at most achieve 60% of maximal recall suggests that the phrase embeddings are not complete w.r.t summarization and might benefit from being combined with other similarity measures that can capture complementary information, for example using multiple kernel learning.

7 Related Work

To the best of our knowledge, continuous vector space models have not previously been used in summarization tasks. Therefore, we split this section in two, handling summarization and continuous vector space models separately.

7.1 Continuous Vector Space Models

Continuous distributed vector representation of words was first introduced by Bengio et al. (2003). They employ a FFNN, using a window of words

as input, and train the model to predict the next word. This is computed using a big softmax layer that calculate the probabilities for each word in the vocabulary. This type of exhaustive estimation is necessary in some NLP applications, but makes the model heavy to train.

If the sole purpose of the model is to derive word embeddings this can be exploited by using a much lighter output layer. This was suggested by Collobert and Weston (2008), which swapped the heavy softmax against a hinge loss function. The model works by scoring a set of consecutive words, distorting one of the words, scoring the distorted set, and finally training the network to give the correct set a higher score.

Taking the lighter concept even further, Mikolov et al. (2013a) introduced a model called Continuous Skip-gram. This model is trained to predict the context surrounding a given word using a shallow neural network. The model is less aware of the order of words, than the previously mentioned models, but can be trained efficiently on considerably larger datasets.

An early attempt at merging word representations into representations for phrases and sentences is introduced by Socher et al. (2010). The authors present a recursive neural network architecture (RvNN) that is able to jointly learn parsing and phrase/sentence representation. Though not able to achieve state-of-the-art results, the method provides an interesting path forward. The model uses one neural network to derive all merged representations, applied recursively in a binary parse tree. This makes the model fast and easy to train but requires labeled data for training.

The problem of needing labeled training data is remedied by Socher et al. (2011), where the RvNN model is adapted to be used as an auto-encoder and employed for paraphrase detection. The approach uses a precomputed binary parse tree to guide the recursion. The unsupervised nature of this setup makes it possible to train on large amounts of data, e.g., the complete English Wikipedia.

7.2 Summarization Techniques

Radev et al. (2004) pioneered the use of cluster centroids in their work with the idea to group, in the same cluster, those sentences which are highly similar to each other, thus generating a number of clusters. To measure the similarity between a pair of sentences, the authors use the cosine similarity measure where sentences are represented as weighted vectors of *tf-idf* terms. Once sentences are clustered, sentence selection is performed by selecting a subset of sentences from each cluster.

In TextRank (2004), a document is represented as a graph where each sentence is denoted by a vertex and pairwise similarities between sentences are represented by edges with a weight corresponding to the similarity between the sentences. The Google PageRank ranking algorithm is used to estimate the importance of different sentences and the most important sentences are chosen for inclusion in the summary.

Bonzanini, Martinez, Roelleke (2013) presented an algorithm that starts with the set of all sentences in the summary and then iteratively chooses sentences that are unimportant and removes them. The sentence removal algorithm obtained good results on the Opinosis dataset, in particular w.r.t F-scores.

We have chosen to compare our work with that of Lin and Bilmes (2011), described in Sec. 2.1. Future work is to make an exhaustive comparison using a larger set similarity measures and summarization frameworks.

8 Conclusions

We investigated the effects of using phrase embeddings for summarization, and showed that these can significantly improve the performance of the state-of-the-art summarization method introduced by Lin and Bilmes in (2011). Two implementations of word vectors and two different approaches for composition where evaluated. All investigated combinations improved the original Lin-Bilmes approach (using *tf-idf* representations of sentences) for at least two ROUGE scores, and top results where found using vector addition on CW vectors.

In order to further investigate the applicability of continuous vector representations for summarization, in future work we plan to try other summarization methods. In particular we will use a method based on multiple kernel learning where phrase embeddings can be combined with other similarity measures. Furthermore, we aim to use a novel method for sentence representation similar to the RAE using multiplicative connections controlled by the local context in the sentence.

References

Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Jauvin. 2003. A neural probabilistic language model. *Journal of Machine Learning Research*, 3:1137–1155.

Yoshua Bengio. 2002. New distributed probabilistic language models. Technical Report 1215, Département d'informatique et recherche opérationnelle, Université de Montréal.

Marco Bonzanini, Miguel Martinez-Alvarez, and Thomas Roelleke. 2013. Extractive summarisation via sentence removal: Condensing relevant sentences into a short summary. In *Proceedings of the 36th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '13, pages 893–896. ACM.

Ronan Collobert and Jason Weston. 2008. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of the 25th international conference on Machine learning*, pages 160–167. ACM.

Kavita Ganesan, ChengXiang Zhai, and Jiawei Han. 2010. Opinosis: a graph-based approach to abstractive summarization of highly redundant opinions. In *Proceedings of the 23rd International Conference on Computational Linguistics*, pages 340–348. ACL.

Christoph Goller and Andreas Kuchler. 1996. Learning task-dependent distributed representations by backpropagation through structure. In *IEEE International Conference on Neural Networks*, volume 1, pages 347–352. IEEE.

Alex Graves, Abdel-rahman Mohamed, and Geoffrey Hinton. 2013. Speech recognition with deep recurrent neural networks. *arXiv preprint arXiv:1303.5778*.

S.S. Haykin. 2009. *Neural Networks and Learning Machines*. Number v. 10 in Neural networks and learning machines. Prentice Hall.

Geoffrey E Hinton and Ruslan R Salakhutdinov. 2006. Reducing the dimensionality of data with neural networks. *Science*, 313(5786):504–507.

Geoffrey E Hinton, Simon Osindero, and Yee-Whye Teh. 2006. A fast learning algorithm for deep belief nets. *Neural computation*, 18(7):1527–1554.

Dan Klein and Christopher D Manning. 2003. Fast exact inference with a factored model for natural language parsing. *Advances in neural information processing systems*, pages 3–10.

Alex Krizhevsky, Ilya Sutskever, and Geoff Hinton. 2012. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems 25*, pages 1106–1114.

Hui Lin and Jeff Bilmes. 2011. A class of submodular functions for document summarization. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 510–520. ACL.

Chin-Yew Lin. 2004. Rouge: A package for automatic evaluation of summaries. In *Text Summarization Branches Out: Proceedings of the ACL-04 Workshop*, pages 74–81.

Rada Mihalcea and Paul Tarau. 2004. TextRank: Bringing order into texts. In *Proceedings of EMNLP*, volume 4. Barcelona, Spain.

Tomas Mikolov, Martin Karafiát, Lukas Burget, Jan Cernockỳ, and Sanjeev Khudanpur. 2010. Recurrent neural network based language model. In *INTERSPEECH*, pages 1045–1048.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013a. Efficient estimation of word representations in vector space. *ArXiv preprint arXiv:1301.3781*.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013b. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems*, pages 3111–3119.

Dragomir R Radev, Hongyan Jing, Małgorzata Styś, and Daniel Tam. 2004. Centroid-based summarization of multiple documents. *Information Processing & Management*, 40(6):919–938.

David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. 1986. Learning representations by back-propagating errors. *Nature*, 323(6088):533–536.

Gerard Salton and Michael J. McGill. 1986. *Introduction to Modern Information Retrieval*. McGraw-Hill, Inc., New York, NY, USA.

Richard Socher, Christopher D Manning, and Andrew Y Ng. 2010. Learning continuous phrase representations and syntactic parsing with recursive neural networks. In *Proceedings of the NIPS-2010 Deep Learning and Unsupervised Feature Learning Workshop*.

Richard Socher, Eric H. Huang, Jeffrey Pennington, Andrew Y. Ng, and Christopher D. Manning. 2011. Dynamic Pooling and Unfolding Recursive Autoencoders for Paraphrase Detection. In *Advances in Neural Information Processing Systems 24*.

Joseph Turian, Lev Ratinov, and Yoshua Bengio. 2010. Word representations: a simple and general method for semi-supervised learning. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 384–394. ACL.

Paper II

Extractive Summarization by Aggregating Multiple Similarities

O. Mogren, M. Kågebäck, and D. Dubhashi

Reprinted from Proceedings of Recent Advances in Natural Language Processing, **2015**

Extractive Summarization by Aggregating Multiple Similarities

Olof Mogren, Mikael Kågebäck, Devdatt Dubhashi
Department of Computer Science and Engineering
Chalmers University of Technology,
412 96 Göteborg, Sweden
mogren@chalmers.se

Abstract

News reports, social media streams, blogs, digitized archives and books are part of a plethora of reading sources that people face every day. This raises the question of how to best generate automatic summaries. Many existing methods for extracting summaries rely on comparing the similarity of two sentences in some way. We present new ways of measuring this similarity, based on sentiment analysis and continuous vector space representations, and show that combining these together with similarity measures from existing methods, helps to create better summaries. The finding is demonstrated with MULTSUM, a novel summarization method that uses ideas from kernel methods to combine sentence similarity measures. Submodular optimization is then used to produce summaries that take several different similarity measures into account. Our method improves over the state-of-the-art on standard benchmark datasets; it is also fast and scale to large document collections, and the results are statistically significant.

1 Introduction

Extractive summarization, the process of selecting a subset of sentences from a set of documents, is an important component of modern information retrieval systems (Baeza-Yates et al., 1999). A good summarization system needs to balance two complementary aspects: finding a summary that captures all the important topics of the documents (*coverage*), yet does not contain too many similar sentences (*non-redundancy*). It follows that it is essential to have a good way of measuring the similarity of sentences, in no way a trivial task. Consequently, several measures for sentence similarity have been explored for extractive summarization.

In this work, two sets of novel similarity measures capturing deeper semantic features are presented, and evaluated in combination with existing methods of measuring sentence similarity. The new methods are based on sentiment analysis, and continuous vector space representations of phrases, respectively.

We show that summary quality is improved by combining multiple similarities at the same time using kernel techniques. This is demonstrated using MULTSUM, an ensemble-approach to generic extractive multi-document summarization based on the existing, state-of-the-art method of Lin and Bilmes (2011). Our method obtains state-of-the-art results that are statistically significant on the de-facto standard benchmark dataset DUC 04. The experimental evaluation also confirm that the method generalizes well to other datasets.

2 MULTSUM

MULTSUM, our approach for extractive summarization, finds representative summaries taking multiple sentence similarity measures into account. As Lin and Bilmes (2011), we formulate the problem as the optimization of monotone non-decreasing submodular set functions. This results in a fast, greedy optimization step that provides a $(1 - \frac{1}{e})$ factor approximation. In the original version, the optimization objective is a function scoring a candidate summary by coverage and diversity, expressed using cosine similarity between sentences represented as bag-of-terms vectors. We extend this method by using several sentence similarity measures M^l (as described in Section 3) at the same time, combined by multiplying them together element-wise:

$$M_{\mathbf{s}_i,\mathbf{s}_j} = \prod M^l_{\mathbf{s}_i,\mathbf{s}_j}.$$

In the literature of kernel methods, this is the standard way of combining kernels as a conjunction (Duvenaud, 2014; Schölkopf et al., 2004, Ch 1).

3 Sentence Similarity Measures

Many existing systems rely on measuring the similarity of sentences to balance the coverage with the amount of redundancy of the summary. This is also true for MULTSUM which is based on the existing submodular optimization method. Similarity measures that capture general aspects lets the summarization system pick sentences that are representative and diverse in general. Similarity measures capturing more specific aspects allow the summarization system to take these aspects into account.

We list some existing measures in Table 3 (that mainly relies on counting word overlaps) and in Sections 3.1 and 3.2, we present sentence similarity measures that capture more specific aspects of the text. MULTSUM is designed to work with all measures mentioned below; this will be evaluated in Section 4. Interested readers are referred to a survey of existing similarity measures from the litterature in (Bengtsson and Skeppstedt, 2012). All these similarity measures require sentence splitting, tokenization, part-of-speech tagging and stemming of words. The Filtered Word, and TextRank comparers are set similarity measures where each sentence is represented by the set of all their terms. The KeyWord comparer and LinTFIDF represent each sentence as a word vector and uses the vectors for measuring similarity.

DepGraph first computes the dependency parse trees of the two sentences using Maltparser (Nivre, 2003). The length of their longest common path is then used to derive the similarity score.

The similarity measure used in TextRank (Mihalcea and Tarau, 2004) will be referred to as TRComparer. The measure used in submodular optimization (Lin and Bilmes, 2011) will be referred to as LinTFIDF. All measures used in this work are normalized, $M_{\mathbf{s}_i,\mathbf{s}_j} \in [0,1]$.

Name	Formula
Filtered	$M_{\mathbf{s}_i,\mathbf{s}_j} = \lvert(\mathbf{s}_i \cap \mathbf{s}_j)\rvert / \sqrt{\lvert(\mathbf{s}_i)\rvert + \lvert(\mathbf{s}_j)\rvert}$
TRCmp.	$M_{\mathbf{s}_i,\mathbf{s}_j} = \lvert\mathbf{s}_i \cap \mathbf{s}_j\rvert / (log\lvert\mathbf{s}_i\rvert + log\lvert\mathbf{s}_j\rvert)$
LinTFIDF	$M_{\mathbf{s}_i,\mathbf{s}_j} = \frac{\sum_{w \in \mathbf{s}_i} tf_{w,i} \cdot tf_{w,j} \cdot idf_w^2}{\sqrt{\sum_{w \in \mathbf{s}_i} tf_{w,\mathbf{s}_i} idf_w^2} \sqrt{\sum_{w \in \mathbf{s}_j} tf_{w,\mathbf{s}_j} idf_w^2}}$
KeyWord	$M_{\mathbf{s}_i,\mathbf{s}_j} = \frac{\sum_{w \in \{\{\mathbf{s}_i \cap \mathbf{s}_j\} \cap K\}} tf_w \cdot idf_w}{\lvert\mathbf{s}_i\rvert + \lvert\mathbf{s}_j\rvert}$
DepGraph	*See text description.*

Table 1: Similarity measures from previous works.

3.1 Sentiment Similarity

Sentiment analysis has previously been used for document summarization, with the aim of capturing an average sentiment of the input corpus (Lerman et al., 2009), or to score emotionally charged sentences (Nishikawa et al., 2010). Other research has shown that negative emotion words appear at a relative higher rate in summaries written by humans (Hong and Nenkova, 2014). We propose a different way of making summaries sentiment aware by comparing the level of sentiment in sentences. This allows for summaries that are both representative and diverse in sentiment.

Two lists, of positive and of negative sentiment words respectively, were manually created[1] and used. Firstly, each sentence $\mathbf{s}_i$ is given two sentiment scores, $positive(\mathbf{s}_i)$ and $negative(\mathbf{s}_i)$, defined as the fraction of words in $\mathbf{s}_i$ that is found in the positive and the negative list, respectively. The similarity score for positive sentiment are computed as follows:

$$M_{\mathbf{s}_i,\mathbf{s}_j} = 1 - |positive(\mathbf{s}_i) - positive(\mathbf{s}_j)|$$

The similarity score for negative sentiment are computed as follows:

$$M_{\mathbf{s}_i,\mathbf{s}_j} = 1 - |negative(\mathbf{s}_i) - negative(\mathbf{s}_j)|$$

3.2 Continuous Vector Space Representations

Continuous vector space representations of words has a long history. Recently, the use of deep learning methods has given rise to a new class of continuous vector space models. Bengio et al. (2006) presented vector space representations for words that capture semantic and syntactic properties. These vectors can be employed not only to find similar words, but also to relate words using multiple dimensions of similarity. This means that words sharing some sense can be related using

[1] To download the sentiment word lists used, please see `http://www.mogren.one/`

translations in vector space, e.g. $v_{king} - v_{man} + v_{woman} \approx v_{queen}$.

Early work on extractive summarization using vector space models was presented in (Kågebäck et al., 2014). In this work we use a similar approach, with two different methods of deriving word embeddings. The first model (CW) was introduced by Collobert and Weston (2008). The second ($W2V$) is the skip-gram model by Mikolov et al. (2013).

The Collobert and Weston vectors were trained on the RCV1 corpus, containing one year of Reuters news wire; the skip-gram vectors were trained on 300 billion words from Google News.

The word embeddings are subsequently used as building blocks for sentence level phrase embeddings by summing the word vectors of each sentence. Finally, the sentence similarity is defined as the cosine similarity between the sentence vectors.

With MULTSUM, these similarity measures can be combined with the traditional sentence similarity measures.

4 Experiments

Our version of the submodular optimization code follows the description by Lin and Bilmes (2011), with the exception that we use multiplicative combinations of the sentence similarity scores described in Section 3. The source code of our system can be downloaded from `http://www.mogren.one/`. Where nothing else is stated, MULTSUM was evaluated with a multiplicative combination of TRComparer and FilteredWordComparer.

4.1 Datasets

In the evaluation, three different datasets were used. DUC 02 and DUC 04 are from the Document Understanding Conferences, both with the settings of task 2 (short multi-document summarization), and each consisting of around 50 document sets. Each document set is comprised of around ten news articles (between 111 and 660 sentences) and accompanied with four gold-standard summaries created by manual summarizers. The summaries are at most 665 characters long. DUC 04 is the de-facto standard benchmark dataset for generic multi-document summarization.

Experiments were also carried out on Opinosis (Ganesan et al., 2010), a collection of short user reviews in 51 different topics. Each topic consists of between 50 and 575 one-sentence user reviews by different authors about a certain characteristic of a hotel, a car, or a product. The dataset includes 4 to 5 gold-standard summaries created by human authors for each topic. The the gold-standard summaries is around 2 sentences.

4.2 Baseline Methods

Our baseline methods are Submodular optimization (Lin and Bilmes, 2011), DPP (Kulesza and Taskar, 2012), and ICSI (Gillick et al., 2008). The baseline scores are calculated on precomputed summary outputs (Hong et al., 2014).

4.3 Evaluation Method

Following standard procedure, we use ROUGE (version 1.5.5) for evaluation (Lin, 2004). ROUGE counts n-gram overlaps between generated summaries and the gold standard. We have concentrated on recall as this is the measure with highest correlation to human judgement (Lin and Hovy, 2003), on ROUGE-1, ROUGE-2, and ROUGE-SU4, representing matches in unigrams, bigrams, and skip-bigrams, respectively.

The Opinosis experiments were aligned with those of Bonzanini et al. (2013) and Ganesan et al. (2010)[2]. Summary length was 2 sentences. In the DUC experiments, summary length is 100 words[3].

5 Results

Our experimental results show significant improvements by aggregating several sentence similarity measures, and our results for ROUGE-2 and ROUGE-SU4 recall beats state–of–the–art.

5.1 Integrating Different Similarity Measures

Table 2 shows ROUGE recall on DUC 04. MULTSUM[4] obtains ROUGE scores beating state-of-the-art systems, in particular on ROUGE-2 and ROUGE-SU4, suggesting that MULTSUM produce summaries with excellent fluency. We also note, that using combined similarities, we beat original submodular optimization.

Figure 5.1 shows, for each $n \in [1..9]$, the highest ROUGE-1 recall score obtained by MULTSUM, determined by exhaustive search

[2]ROUGE options on Opinosis: -a - m -s -x -n 2 -2 4 -u.

[3]ROUGE options on DUC: -a -n 2 -m -l 100 -x -c 95 -r 1000 -f A -p 0.5 -t 0 -2 4 -u.

[4]Here, MULTSUM is using TRComparer and FilteredWordComparer in multiplicative conjunction.

	ROUGE-1	ROUGE-2	ROUGE-SU4
MULTSUM	39.35	**9.94**	**14.01**
ICSISumm	38.41	9.77	13.62
DPP	**39.83**	9.62	13.86
SUBMOD	39.18	9.35	13.75

Table 2: ROUGE recall scores on DUC 04. Our system MULTSUM obtains the best result yet for ROUGE-2 and ROUGE-SU4. DPP has a higher ROUGE-1 score, but the difference is not statistically significant (Hong et al., 2014).

1	**2**	**3**	**4**
1.0	0.00038	0.00016	0.00016

Table 3: p-values from the Mann-Whitney U-test for combinations of similarity measures of size $n \in [1..4]$, compared to using just one similarity measure. Using 2, 3, or 4 similarity measures at the same time with MULTSUM, gives a statistically significant improvement of the ROUGE-1 scores. Dataset: DUC 04.

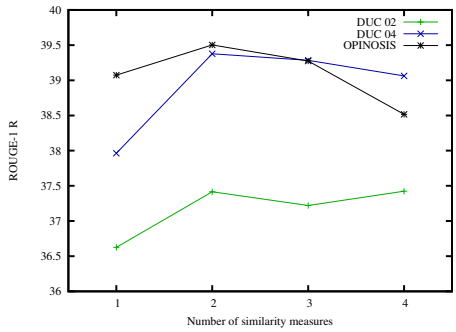


Figure 1: MULTSUM ROUGE-1 recall performance for each top-performing combination of up to four similarity measures. On all datasets, using combinations of two, three, and four similarity measures is better than using only one.

among all possible combinations of size n. The performance increases from using only one sentence similarity measure, reaching a high, stable level when $n \in [2..4]$. The behaviour is consistent over three datasets: DUC 02, DUC 04 and OPINOSIS. Based on ROUGE-1 recall, on DUC 02, a combination of four similarity measures provided the best results, while on DUC 04 and Opinosis, a combination of two similarity scores provided a slightly better score.

Table 3 shows p-values obtained using the Mann-Whitney U-test (Mann et al., 1947) on the ROUGE-1 scores when using a combination of n similarities with MULTSUM, compared to using only one measure. The Mann-Whitney U-test compares two ranked lists A and B, and decides whether they are from the same population. Here, A is the list of scores from using only one measure, and B is the top-10 ranked combinations of n combined similarity measures, $n \in [1..4]$). One can see that for each $n \in [1..4]$, using n sentence similarity measures at the same time, is significantly better than using only one.

On DUC 02, the best combination of similarity measures is using CW, LinTFIDF, NegativeSentiment, and TRComparer. Each point in Figure 5.1 represents a combination of some of these four similarity measures. Let n be the number of measures in such a combination. When $n = 1$, the "combinations" are just single similarity measures. When $n = 2$, there are 6 different ways to choose, and when $n = 3$, there are four. A line goes from each measure point through all combinations the measure is part of. One can clearly see the benefits of each of the combination steps, as n increases.

5.2 Evaluation with Single Similarity Measures

In order to understand the effect of different similarity measures, MULTSUM was first evaluated using only one similarity measure at a time. Table 4 shows the ROUGE recall scores of these experiments, using the similarity measures presented in Section 3, on DUC 04.

We note that MULTSUM provides summaries of high quality already with one similarity measure (e.g. with TRComparer), with a ROUGE-1 recall of 37.95 Using only sentiment analysis as the single similarity measure does not capture enough information to produce state-of-the-art summaries.

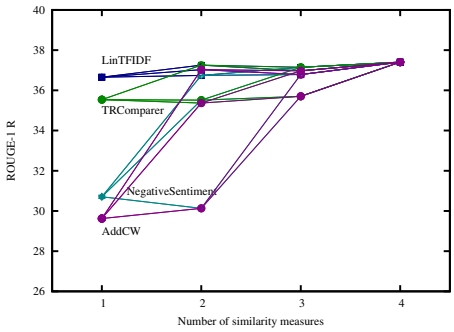


Figure 2: ROUGE-1 recall for the top-performing four-combination on DUC 2002 (CW, LinTFIDF, NegativeSentiment, and TRComparer), and all possible subsets of these four similarity measures. (When the number of similarity measures is one, only a single measure is used).

	ROUGE-1	ROUGE-2	ROUGE-SU4
$TRComparer$	**37.95**	**8.94**	**13.19**
$Filtered$	37.51	8.26	12.73
$LinTFIDF$	35.74	6.50	11.51
$KeyWord$	35.40	7.13	11.80
$DepGraph$	32.81	5.43	10.12
$NegativeSent.$	32.65	6.35	10.29
$PositiveSent.$	31.19	4.87	9.27
$W2V$	32.12	4.94	9.92
CW	31.59	4.74	9.51

Table 4: ROUGE recall of MULTSUM using different similarity measures, one at a time. Dataset: DUC 04. The traditional word-overlap measures are the best scoring when used on their own; the proposed measures with more semantical comparisons provide the best improvements when used in conjunctions.

6 Discussion

Empirical evaluation of the method proposed in this paper shows that using several sentence similarity measures at the same time produces significantly better summaries.

When using one single similarity at a time, using sentiment similarity and vector space models does not give the best summaries. However, we found that when combining several similarity measures, our proposed sentiment and continuous vector space measures often rank among the top ones, together with the TRComparer.

MULTSUM, our novel summarization method, based on submodular optimization, multiplies several sentence similarity measures, to be able to make summaries that are good with regards to several aspects at the same time. Our experimental results show significant improvements when using multiplicative combinations of several sentence similarity measures. In particular, the results of MULTSUM surpasses that of the original submodular optimization method.

In our experiments we found that using between two and four similarity measures lead to significant improvements compared to using a single measure. This verifies the validity of commonly used measures like TextRank and LinTFIDF as well as new directions like phrase embeddings and sentiment analysis.

There are several ideas worth pursuing that could further improve our methods. We will explore methods of incorporating more semantic information in our sentence similarity measures. This could come from systems for Information Extraction (Ji et al., 2013), or incorporating external sources such as WordNet, Freebase and DBpedia (Nenkova and McKeown, 2012).

7 Related Work

Ever since (Luhn, 1958), the field of automatic document summarization has attracted a lot of attention, and has been the focus of a steady flow of research. Luhn was concerned with the importance of words and their representativeness for the input text, an idea that's still central to many current approaches. The development of new techniques for document summarization has since taken many different paths. Some approaches concentrate on what words should appear in summaries, some focus on sentences in part or in whole, and some consider more abstract concepts.

In the 1990's we witnessed the dawn of the data explosion known as the world wide web, and research on multi document summarization took off. Some ten years later, the Document Understanding Conferences (DUC) started providing researchers with datasets and spurred interest with a venue for competition.

Luhn's idea of a frequency threshold measure for selecting topic words in a document has lived on. It was later superseded by $tf \times idf$, which measures the specificity of a word to a document,

The two bombers who carried out Friday's attack, which led the Israeli Cabinet to suspend deliberations on the land-for-security accord signed with the Palestinians last month, were identified as members of Islamic Holy War from West Bank villages under Israeli security control. The radical group Islamic Jihad claimed responsibility Saturday for the market bombing and vowed more attacks to try to block the new peace accord. Israel radio said the 18-member Cabinet debate on the Wye River accord would resume only after Yasser Arafat's Palestinian Authority fulfilled all of its commitments under the agreement, including arresting Islamic militants.

Table 5: Example output from MULTSUM. Input document: d30010t from DUC 04. Similarity Measures: W2V, TRComparer, and FilteredWordComparer.

something that has been used extensively in document summarization efforts. RegSum (Hong and Nenkova, 2014) trained a classifier on what kinds of words that human experts include in summaries. (Lin and Bilmes, 2011) represented sentences as a $tf \times idf$ weighted bag-of-words vector, defined a sentence graph with weights according to cosine similarity, and used submodular optimization to decide on sentences for a summary that is both representative and diverse.

Several other methods use similar sentence-based formulations but with different sentence similarities and summarization objectives (Radev et al., 2004; Mihalcea and Tarau, 2004).

(Bonzanini et al., 2013) introduced an iterative sentence removal procedure that proved good in summarizing short online user reviews. CLASSY04 (Conroy et al., 2004) was the best system in the official DUC 04 evaluation. After some linguistic preprocessing, it uses a Hidden Markov Model for sentence selection where the decision on inclusion of a sentence depends on its number of signature tokens. The following systems have also showed state–of–the–art results on the same data set. ICSI (Gillick et al., 2008) posed the summarization problem as a global integer linear program (ILP) maximizing the summary's coverage of key n-grams. OCCAMS_V (Davis et al., 2012) uses latent semantic analysis to determine the importance of words before the sentence selection. (Kulesza and Taskar, 2012) presents the use of Determinantal point processes (DPPs) for summarization, a probabilistic formulation that allows for a balance between diversity and coverage. An extensive description and comparison of these state–of–the–art systems can be found in (Hong et al., 2014), along with a repository of summary outputs on DUC 04.

Besides the aforementioned work, interested readers are referred to an extensive survey (Nenkova and McKeown, 2012). In particular, they discuss different approaches to sentence representation, scoring and summary selection and their effects on the performance of a summarization system.

8 Conclusions

We have demonstrated that extractive summarization benefits from using several sentence similarity measures at the same time. The proposed system, MULTSUM works by using standard kernel techniques to combine the similarities. Our experimental evaluation shows that the summaries produced by MULTSUM outperforms state-of-the-art systems on standard benchmark datasets. In particular, it beats the original submodular optimization approach on all three variants of ROUGE scores. It attains state-of-the-art results on both ROUGE-2 and ROUGE-SU4, showing that the resulting summaries have high fluency. The results are statistically significant and consistent over all three tested datasets: DUC 02, DUC 04, and Opinosis.

We have also seen that sentence similarity measures based on sentiment analysis and continuous vector space representations can improve the results of multi-document summarization. In our experiments, these sentence similarity measures used separately are not enough to create a good summary, but when combining them with traditional sentence similarity measures, we improve on previous methods.

Acknowledgments

This work has been done within "Data-driven secure business intelligence", grant IIS11-0089 from the Swedish Foundation for Strategic Research (SSF). We would like to thank Gabriele Capannini for discussion and help and Jonatan Bengtsson for his work on sentence similarity measures.

References

Ricardo Baeza-Yates, Berthier Ribeiro-Neto, et al. 1999. *Modern information retrieval*, volume 463. ACM press New York.

Yoshua Bengio, Holger Schwenk, Jean-Sébastien Senécal, Fréderic Morin, and Jean-Luc Gauvain. 2006. Neural probabilistic language models. In *Innovations in Machine Learning*. Springer.

Jonatan Bengtsson and Christoffer Skeppstedt. 2012. Automatic extractive single document summarization. Master's thesis, Chalmers University of Technology and University of Gothenburg.

Marco Bonzanini, Miguel Martinez-Alvarez, and Thomas Roelleke. 2013. Extractive summarisation via sentence removal: condensing relevant sentences into a short summary. In *SIGIR*, pages 893–896.

Ronan Collobert and Jason Weston. 2008. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of ICML*, pages 160–167.

John M Conroy, Judith D Schlesinger, Jade Goldstein, and Dianne P O'leary. 2004. Left-brain/right-brain multi-document summarization. In *DUC 2004*.

Sashka T Davis, John M Conroy, and Judith D Schlesinger. 2012. Occams–an optimal combinatorial covering algorithm for multi-document summarization. In *ICDMW*. IEEE.

David Duvenaud. 2014. *Automatic model construction with Gaussian processes*. Ph.D. thesis, University of Cambridge.

Kavita Ganesan, ChengXiang Zhai, and Jiawei Han. 2010. Opinosis: a graph-based approach to abstractive summarization of highly redundant opinions. In *Proceedings of COLING*, pages 340–348. ACL.

Dan Gillick, Benoit Favre, and Dilek Hakkani-Tur. 2008. The icsi summarization system at tac 2008. In *Proceedings of TAC*.

Kai Hong and Ani Nenkova. 2014. Improving the estimation of word importance for news multi-document summarization. In *Proceedings of EACL*.

Kai Hong, John M Conroy, Benoit Favre, Alex Kulesza, Hui Lin, and Ani Nenkova. 2014. A repository of state of the art and competitive baseline summaries for generic news summarization. *LREC*.

Heng Ji, Benoit Favre, Wen-Pin Lin, Dan Gillick, Dilek Hakkani-Tur, and Ralph Grishman. 2013. Open-domain multi-document summarization via information extraction: Challenges and prospects. In *Multi-source, Multilingual Information Extraction and Summarization*, pages 177–201. Springer.

Mikael Kågebäck, Olof Mogren, Nina Tahmasebi, and Devdatt Dubhashi. 2014. Extractive summarization using continuous vector space models. *Proceedings of (CVSC)@ EACL*, pages 31–39.

Alex Kulesza and Ben Taskar. 2012. Determinantal point processes for machine learning. *arXiv:1207.6083*.

Kevin Lerman, Sasha Blair-Goldensohn, and Ryan McDonald. 2009. Sentiment summarization: Evaluating and learning user preferences. In *Proceedings of EACL*, pages 514–522. ACL.

Hui Lin and Jeff Bilmes. 2011. A class of submodular functions for document summarization. In *ACL*.

Chin-Yew Lin and Eduard Hovy. 2003. Automatic evaluation of summaries using n-gram co-occurrence statistics. In *Proceedings of NAACL/HLT*, pages 71–78.

Chin-Yew Lin. 2004. Rouge: A package for automatic evaluation of summaries. In *Text Summarization Branches Out: Proc. of the ACL-04 Workshop*, pages 74–81.

Hans Peter Luhn. 1958. The automatic creation of literature abstracts. *IBM Journal*, 2(2):159–165.

Henry B Mann, Donald R Whitney, et al. 1947. On a test of whether one of two random variables is stochastically larger than the other. *The annals of mathematical statistics*, 18(1):50–60.

Rada Mihalcea and Paul Tarau. 2004. Textrank: Bringing order into texts. In *Proceedings of EMNLP*, volume 4.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *arXiv:1301.3781*.

Ani Nenkova and Kathleen McKeown. 2012. A survey of text summarization techniques. In Charu C. Aggarwal and ChengXiang Zhai, editors, *Mining Text Data*, pages 43–76. Springer.

Hitoshi Nishikawa, Takaaki Hasegawa, Yoshihiro Matsuo, and Genichiro Kikui. 2010. Optimizing informativeness and readability for sentiment summarization. In *Proceedings of ACL*, pages 325–330. ACL.

Joakim Nivre. 2003. An efficient algorithm for projective dependency parsing. In *Proceedings of IWPT*. Citeseer.

Dragomir R. Radev, Timothy Allison, Sasha Blair-Goldensohn, John Blitzer, Arda Çelebi, Stanko Dimitrov, Elliott Drábek, Ali Hakim, Wai Lam, Danyu Liu, Jahna Otterbacher, Hong Qi, Horacio Saggion, Simone Teufel, Michael Topper, Adam Winkel, and Zhu Zhang. 2004. Mead - a platform for multidocument multilingual text summarization. In *LREC*.

Bernhard. Schölkopf, Koji. Tsuda, and Jean-Philippe. Vert. 2004. *Kernel methods in computational biology*. MIT Press, Cambridge, Mass.

Paper III

Visions and open challenges for a knowledge-based culturomics

N. Tahmasebi et al.

Reprinted from International Journal on Digital Libraries, **2015**

Int J Digit Libr
DOI 10.1007/s00799-015-0139-1

Visions and open challenges for a knowledge-based culturomics

Nina Tahmasebi · Lars Borin · Gabriele Capannini · Devdatt Dubhashi · Peter Exner · Markus Forsberg · Gerhard Gossen · Fredrik D. Johansson · Richard Johansson · Mikael Kågebäck · Olof Mogren · Pierre Nugues · Thomas Risse

Received: 30 December 2013 / Revised: 9 January 2015 / Accepted: 14 January 2015

Abstract The concept of *culturomics* was born out of the availability of massive amounts of textual data and the interest to make sense of cultural and language phenomena over time. Thus far however, culturomics has only made use of, and shown the great potential of, statistical methods. In this paper, we present a vision for a *knowledge-based culturomics* that complements traditional culturomics. We discuss the possibilities and challenges of combining knowledge-based methods with statistical methods and address major challenges that arise due to the nature of the data; diversity of sources, changes in language over time as well as temporal dynamics of information in general. We address all layers needed for knowledge-based culturomics, from natural language processing and relations to summaries and opinions.

Keywords Culturomics · Statistical analysis · Knowledge-based analysis · Temporal text analysis · Digital humanities · eScience · eInfrastructure · Natural language processing

The majority of this work was done while Nina Tahmasebi was employed at Chalmers University of Technology.

N. Tahmasebi ( ) · L. Borin · M. Forsberg · R. Johansson
Språkbanken, University of Gothenburg, Gothenburg, Sweden
e-mail: nina.tahmasebi@gu.se

G. Capannini · D. Dubhashi · F. D. Johansson · M. Kågebäck · O. Mogren
Chalmers University of Technology, Gothenburg, Sweden

P. Exner · P. Nugues
Lund University, Lund, Sweden

G. Gossen · T. Risse
L3S Research Center, Hannover, Germany

1 Introduction

The need to understand and study our culture drives us to read books, explore Web sites and query search engines or encyclopedias for information and answers. Today, with the huge increase of historical documents made available we have a unique opportunity to learn about the past, from the past itself. Using the collections of projects like Project Gutenberg [67] or Google books [25], we can directly access the historical source rather than read modern interpretations. Access is offered online and often minimal effort is necessary for searching and browsing. The increasing availability of digital documents, spread over a long timespan, opens up the possibilities to move beyond the study of individual resources. To study our history, culture and language, we can now computationally analyze the entire set of available documents to reveal information that was previously impossible to access.

The aim of the emerging field *Culturomics*, introduced by Aiden and Michel [3] and Michel et al. [53], is to study human behaviors and cultural trends by analyzing massive amounts of textual data that nowadays are available in digital format. By taking a purely statistical view, Michel et al. [53] unveiled information in Google books that would not have been found without the analysis of this large text corpus. Already by studying word frequencies interesting linguistic and cultural phenomena can be found.

One example of a linguistic phenomenon is the size of the English lexicon. The numbers of unique common words for the years 1900, 1950 and 2000 were compared and the authors found that by year 2000, the size of the lexicon had increased significantly (but see [46]).[1] This means that by the year 2000, more unique words are used in written text

[1] The authors define a common word as one with a frequency greater than one per billion.

Published online: 18 February 2015

than ever before. As an example of a cultural phenomenon the authors studied fame, approximated by the frequency of a person's name. The 50 most famous people born each year between 1800 and 1950 were studied based on several criteria, among others the age of peak celebrity and the half-life of decline. The authors found that, over time, people become famous earlier in their lives and are more famous than ever but are being forgotten faster.

The examples given above showcase the potential of culturomics and could not have been found by studying individual resources or by exclusively using manual analysis. However, despite the large potential of a purely statistical view, culturomics would gain from complementing with deeper, knowledge-based approaches as well as integrate pre-existing knowledge. In this paper, we introduce a knowledge-based culturomics that complements the purely statistical method of classic culturomics with NLP and recent information extraction techniques. We will present our vision and current progress toward a knowledge-based culturomics and discuss open challenges.

1.1 Knowledge-based culturomics

In a broad perspective, culturomics is the study of cultural and linguistic phenomena from large amounts of textual data distributed over a long timespan. Classical culturomics can be used to answer research questions using individual terms—understood as text word types—their frequencies and co-occurrence behaviors.

By "knowledge", we here understand *a priori knowledge relevant to the processing of the material* with these aims in mind. Most fundamentally this is linguistic knowledge about the language(s) present in the material, since the information that we wish to access is conveyed in language, and also general world knowledge and pertinent specific domain knowledge. Using such a priori knowledge allows us to provide additional insights using advanced linking and aggregation of information and relies on a combination of techniques from information extraction and automatic aggregation.

To return to the examples from above, using knowledge-based methods researchers can answer, in addition to the average age of fame, also the typical sentiment toward famous people over time. Have we become more or less positive toward our celebrities? Is there a difference between the start, middle or end of their fame period? What increases a celebrity's fame the most; actions that cause positive or negative reactions? Using *implicit* social graphs, i.e., social graphs extracted using relation extraction and semantic role labeling, we can answer questions about typical social behavior over time. How often do people get married in different parts of the world? Does getting married increase the chances of changing location? Are there differences between different parts of the world? Are people more or less likely to be famous by being married/a child to a famous person?

To answer these type of questions we need a three-layered pipeline: (1) extraction of first-order information; (2) aggregation of information; and (3) establishing connection with the primary resources.

First layer The first layer of processing (not considering the digitization process) is natural language processing (NLP). In this layer, we consider the extracted information to be first-order informational items, among which we include information from lemmatization, morphological, syntactic and lexical semantic analysis, term extraction, named entity recognition and event extraction.

This layer is crucial in our conception of knowledge-based culturomics. Linguistic processing allows for abstraction over text words which in many cases can lower the data requirements considerably, crucial in the case of languages where the amount of available material is smaller, and in the case of morphologically complex languages, and of course doubly crucial when both factors are present.

As an illustrative example of this, in the top diagram in Fig. 1, we replicate one of the examples from Michel et al. [53, p. 180], reproduced here using the *Google Books Ngram Viewer*. This graph shows the appearance of the three surnames *Trotsky*, *Zinovyev*, and *Kamenev* in the Russian portion of the Google Books material, illustrating the rise and fall of these three portal figures of the Bolshevik revolution in the wake of specific historical events.

Now, all examples in [53] are produced on the basis of a single text word form, in this case the base (citation) form of the three surnames. However, Russian nouns (including proper nouns) are inflected in six grammatical case forms (in two numbers), signalling their syntactic role in the sentence. The citation form, the nominative (singular), is mainly used for grammatical subjects, i.e., agents of actions. In the bottom diagram in Fig. 1, we have added the accusative/genitive form of the three surnames,[2] i.e., the form used for direct objects (patients/goals of actions) and possessors. It is obvious from the graph that there are almost as many instances of this form as there are of the nominative form in the material. This shows a clear indication that morphological analysis and lemmatization—using a priori linguistic knowledge—can help us to get more "culturomic leverage" out of our material.[3]

[2] With male surnames, the accusative form and the genitive form are always identical (although otherwise generally distinguished in Russian morphology).

[3] Mann et al. [51] describes an improved version of the Google Books Ngram Viewer where searches can be made for inflected forms of words. However, rather than using a dedicated morphological processor for each language, this project relies on Wiktionary data for collecting inflectional paradigms. Following a general lexicographical tradition, names are generally not listed in Wiktionary. However, names have

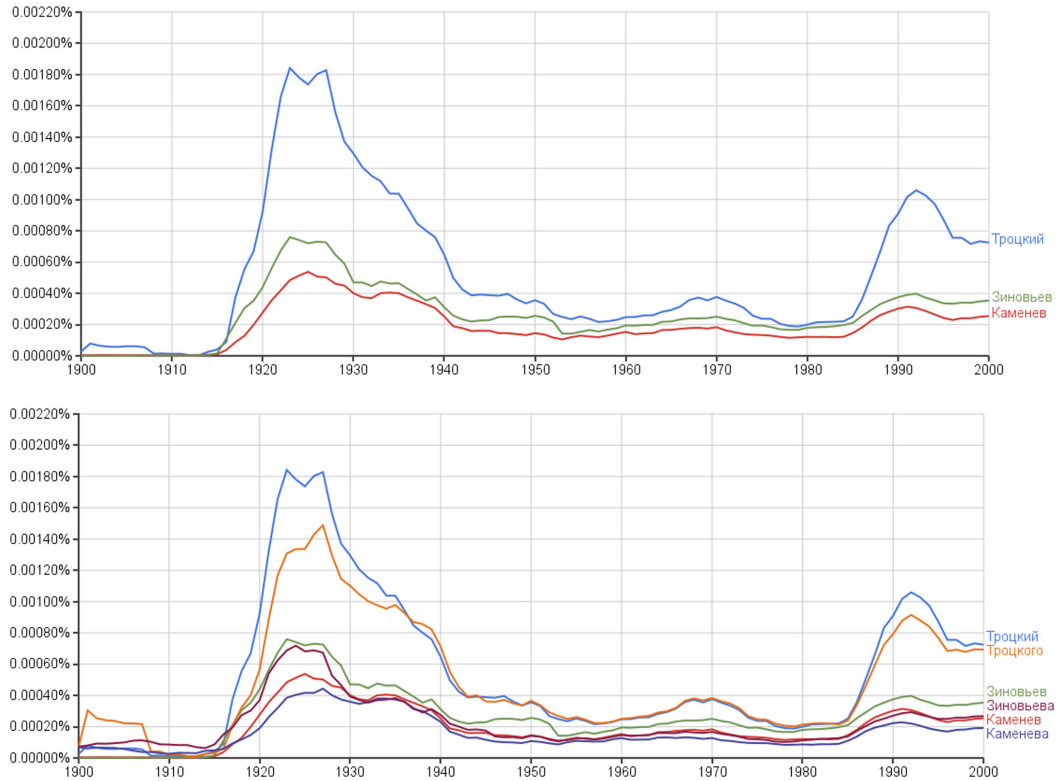


Fig. 1 The Russian surnames *Trotsky*, *Zinovyev*, and *Kamenev* in the Russian portion of the Google Books material, in the nominative case (*top*) and the nominative plus the accusative/genitive case (*bottom*)

A central aspect of knowledge-based culturomics as we construe it is consequently to apply as much linguistic processing to the material as is possible, using the most mature solutions for each language. This is arguably the step which can add most value in relation to the needed effort, and one where mature systems exist for many languages (although the picture tends to look less rosy when it comes to historical language stages or text types such as social media; see below).

In this paper, the technologies on the first layer are presented primarily as background and the focus lies on highlighting the specific challenges for culturomics.

Second layer On the second layer of processing, first-order informational items are aggregated to reveal "latent" information. We consider such information to be second-order information. Entities, events and their semantic roles can be combined to create implicit social graphs where people are related to each other (e.g., *parent-of*, *married-to*, *communicates-with*, *collaborates-with*) or more general entity graphs where, e.g., people can be related to events or locations (e.g., *born-in*).

Topics and opinions play an important role at this layer as they can reveal cultural phenomena. Opinions are extracted, either as general levels of sentiment or as sentiments held by an opinion holder toward a given entity, event or location. Topics can be extracted on general levels to allow exploration or with respect to a person or event. Summaries can be created to help exploration and ease understanding. These summaries can be on the basis of individual documents or documents related to entities, events, topics or opinions. Also language changes are detected on this layer.

Some of the mentioned technologies have been researched in detail during the past decades. However, large chal-

Footnote 3 continued
distinct inflectional paradigms in Russian and many other languages, and consequently, it is still not possible to retrieve all forms of Russian surnames automatically in the Ngram Viewer.

lenges lie ahead when placing these in the context of culturomics. One important challenge lies in including the temporal dimension and monitoring for change and dynamics. Tracking of information and temporal linking over large timespans must, among other things, handle language change. A second challenge lies in handling of historical texts without pre-existing reliable, digital knowledge resources like Wikipedia or WordNet, that sufficiently cover each period of time. As an example, opinion mining relies to a large extent on existing sentiment dictionaries. Such dictionaries exist only for modern content and cannot be generalized to cover the entire timespan without major efforts, and thus, opinion mining for culturomics requires new or adapted methods, as well as the development of new knowledge-rich resources, from scratch and/or by cross-linking from modern resources.

Third layer The aggregated information has the power to reveal patterns and phenomena that could not be detected by studying individual resources. However, the aggregation often brings us to a *distant reading scenario* [59] where we no longer have a direct link to the primary resources. Without access to the individual resources, it is difficult to verify, evaluate or perform detailed analysis of the information. Therefore, the final layer of knowledge-based culturomics is to re-connect extracted information with the primary resources. A large challenge here is to determine for which type of information this linking is at all possible. As an example, the size of the English vocabulary over time cannot be found in any one document in the collection. This information is reflected only in the aggregation and therefore there is no document that directly mentions or reflects this fact. However, relations, events, opinions and word senses might have primary representation. Therefore, this task is concerned with finding the set of most *descriptive* primary resources for a given fact or relation. Where the primary resources are known, the task is to choose the most descriptive resources among all known resources.

1.2 Structure of the paper

The rest of this paper is structured as follows: Sect. 2 covers natural language processing and first-order information such as entities and relations. This section also discusses entity resolution on the basis of extracted information. Sections 3–5 cover the second layer; Sect. 3 covers language change and Sect. 4 covers temporal opinion mining. Temporal semantic summarization is covered in Sect. 5. The third level—finding descriptive resources—is covered in Sect. 6. Related work and discussion are covered in each section individually. In Sect. 7 we present a general discussion on knowledge-based culturomics and finally we conclude the paper and present future work in Sect. 8.

2 First-order information, the NLP layer

The NLP layer is a linguistically motivated information layer added to the texts, which forms a basis for further analyses and allows us to extract first-order information items. Using NLP we can, e.g., identify that *mice* is the plural of the lemma *mouse*, make distinctions between words such as the noun *fly* and the verb *fly*, or determine the syntactic role of the named entity *Michelle Obama*. In this layer, a challenge is entity resolution as well as determining relation between entities.

2.1 Entity resolution

A key component in understanding and analyzing text is to identify and reason about entities mentioned in the text. In general text, the task is made difficult by the fact that entities are not always referred to using their full name, but with a pronoun, a noun phrase or by different names. Furthermore, several entities may have the same name. Merging mentioning of the same entity as well as differentiating between entities with the same lexical reference is an important challenge in culturomics as the entities come from a wide variety of sources and have been created over a long timespan introducing a larger variety.

As an example we can consider measuring the fame of a person over time, an example from Michel et al. [53]. We want to compare the artists *Michael Jackson* and *Madonna*. First, it is important to recognize that *the King of Pop* refers to *Michael Jackson*. However, mentioning of *Michael Jackson* in the 1880s as well as in domains like science, sports and religion is unlikely to refer to the same person and should be disregarded. For *Madonna*, entity resolution is even more important as the highest frequency for *Madonna* in Google books is around year 1910 in reference to the religious figure rather than the artist.

Central to entity resolution is the process of linking pronouns or phrases to entities, commonly called coreference, or entity resolution and is needed when aligning information about entities across documents as well as across time. While humans perform this task seemingly without effort, automating it has proved to be a greater challenge. Nowadays, because gathering massive amounts of text data is done with relative ease, the need for automatic analysis tools, such as extraction and resolution of entities, is increasing.

At its core, the problem of entity resolution is one of linking or grouping different manifestations of an underlying object, e.g., {*Michael Jackson, Michael Joseph Jackson, MJ, the King of Pop*} $\longrightarrow$ *Michael Jackson*. One of the earlier instances is that of record linkage [60] in which multiple database records, of the same object, are merged together. More recently, methods approaching this problem specifically for text have emerged. Focusing first on simple rule-based methods [42,69], the approaches became increasingly

more sophisticated, adopting the tools of machine learning [52] to unsupervised statistical [61] or clustering methods [10,71] and other statistical methods targeting temporal aspects [8,77].

In culturomics, we consider the entity resolution problem (ER) to be that of finding a mapping between references in text to an (unknown) set of underlying entities. Classically, ER is considered within a single document [52]. Leveraging an entire corpus instead involves reconciling entities across documents, increasing the complexity, but improving the analysis. It allows for use of additional features based on document metadata and richer statistical methods for more accurate resolution [9,66].

Rule-based methods are typically deterministic and easy to interpret but require a large set of rules and strong knowledge of the domain. Supervised methods, on the other hand, require labeled data, used for training a model, which may be hard to obtain. Statistical methods build on a set of assumptions on the data, such as an underlying distribution. Often features from rule-based methods are incorporated into statistical methods [30] to enforce linguistic constraints.

Recently, with the advent of global, freely accessible knowledge bases such as DBpedia or YAGO, a new approach to entity resolution has emerged, incorporating world knowledge to aid the process [70]. However, for many historical datasets (or fiction for that matter), such knowledge bases do not contain sufficient information about the entities involved. Another recent trend is to focus on very large scale problems, with 10s or 100s of millions of documents and entities. In such a setting, cheap identification of likely ambiguous identifiers is a helpful tool to avoid unnecessary computations. This problem has been approached by Hermansson et al. [33], using graph kernels and co-occurrence information, to classify identifiers as ambiguous or not.

As a special case of ER we consider temporal resolution, also called named entity evolution recognition (NEER), as the task of linking different names used for the same entity over time, e.g., cities and people and also different underlying concepts, like the *Great War* and *World War I*. For temporal resolution, statistical methods have exploited the *hand over* between different names of the same entity. This method shows great advantage for temporal entity resolution but would benefit from ER as a first step. Methods for temporal resolution of underlying concepts, rather than entities, are not well explored and need to be tackled more in depth for proper culturomics analysis. See further Sect. 3.

2.2 Relation extraction and semantic role labeling

Relation extraction is the task of extracting specific semantic relations between words or phrases, and the entities they refer to. While relation extraction considers mostly binary relations, semantic role labeling (SRL) is an extension to general predicate–argument structures.

Applying semantic role labeling to large corpora enables culturomics to extend the analysis from isolated words or n-grams to predicate–argument structures. Such structures may reveal finer cultural concepts as they exhibit relations between the concepts. The outcome of this analysis feeds extensively into the remaining technologies (e.g., temporal semantic summarization, Sect. 5). Below is a set of frequent triples consisting of a subject, a predicate and an object extracted from Wikipedia using the Athena system [20]:

Males have income
Schools include school
Students attend schools
Couple have children
Teams win championships
Album sell copies

The quality of the found relations affects the remaining methods in knowledge-based culturomics. Therefore, an important challenge of relation extraction is to keep a high quality while being able to scale to large datasets. In particular, for datasets that are diverse and vary over time.

Supervised methods have been used for relation extraction. They usually exhibit high performance in terms of precision/recall. However, they rely on a hand-annotated corpus for training. Since hand labeling is laborious and time consuming, these corpora are often small, making supervised extraction unscalable to Web size relations or large historical corpora. In contrast, distant supervision (DS) presents a novel alternative. It relies on existing relation facts extracted from a knowledge base to detect and label relations in a sentence. DS draws from both supervised and unsupervised methods in having relatively high performance and domain independence, providing canonical relational labels, without losing scalability in terms of documents and relations. DS was first introduced for information extraction (IE) in the biological domain by Craven et al. [17]. Since then, DS has been successfully applied to relation extraction, see [7,13,57,72,91].

In DS, training data are created using heuristic methods by matching entities in text to entities in relations from a knowledge base. For each matching set of entities, a relational tuple is formed by labeling with the corresponding relation from the knowledge base. By extracting features from the matching entities and sentence, a classifier can be trained. Because of the nature of the data, diversity of sources and changes in language, distant supervision seems to be the most appropriate way to go for culturomics.

2.3 Discussions

Since the NLP pipeline provides the first-order information layer for the subsequent analyses, it is crucial that special attention is spent on improving the quality of this information layer—a small error in a previous analysis step tends to multiply in the following ones. An endeavor such as culturomics must hence be informed by the latest developments in NLP research concerning the first-order analysis, in particular, the ones aimed at the target language at hand, instead of just unreflectingly using available off-the-shelf tools.

There are also many linguistic a priori distinctions built into the NLP pipeline that may influence later analyses substantially. These distinctions are concerned with linguistic identity; when are constructs in a language considered the same, and when are they different? This regards spelling variations as well as inflected forms of the same name (see Sect. 1 for an example). Also sense information is important; in addition to spelling variations, how many senses does a word like *color* have,[4] and how can these senses be distinguished computationally? The answers to these kinds of questions will have a significant effect on the subsequent analyses. As an example, Tahmasebi et al. [79] showed that by correcting OCR errors, the number of automatically derived word senses was increased by 24 % over a 201-year timespan and by 61 % in the period 1785–1815 where the amount of OCR errors was the largest.

Relation extraction and semantic role labeling may provide better insights on cultural patterns and their variation over time. FrameNet, an electronic dictionary based on frame semantics [22], the theory behind semantic role labeling, explicitly aims at building a repository of shared mental concepts. However, these semantic analyses are still less accurate than tagging or syntactic parsing as they come at the end of the NLP processing pipeline. In addition, they require predicate–argument dictionaries and annotated corpora that are, at writing time, only available for few languages: English, Chinese, German, Spanish, or Japanese and have poor coverage of historical variations.

The context of culturomics presents several new challenges to existing methods for entity resolution. First, the corpora involved have long timespans which make methods based on linguistic rules hard to employ, without allowing for evolving rules. While this problem has been approached in part [85], it is yet to be applied to historical data. Second, the large scale of many historical corpora makes supervised methods hard to use because of the amounts of annotated data needed for accurate classification. Also, it is not clear how the time dimension affects the results of existing methods. Unsupervised methods seem well suited, aiming to discover the underlying rules of the data, rather than state them. To the best of our knowledge, however, no such model exists, targeted specifically to historical data.

[4] The Princeton Wordnet posits 14 senses for the word *color*, of which seven are nouns, six are verbs, and one adjective.

3 Language change and variation

Change and variation are inevitable features of our language. With new inventions, changes in culture or major events, our language changes. Mostly, we are aware of all contemporary changes and can easily adapt our language use. However, over time, linguistic expressions and constructions fall out of use and are no longer a part of our collective memory. For most everyday tasks, this does not cause problems. However, in culturomics, when looking to the past, trying to make sense of cultural and language phenomena over time, recovering past language change is highly important.

There are different types of language change that we consider. The classification depends on how each type affects finding (i.e., information retrieval) and understanding of a given document.

The first type of change is *spelling variation* where words are spelled differently over time. To find the true frequency of a word, all different spellings must be found and the frequencies merged. For example, *infynyt*, *infinit*, *infinyte*, *infynit*, *infineit* are all historical spelling variants used at different times for the word now spelled *infinite*. To follow this controversial concept over time, frequencies and contexts from all spellings must be taken into account.

The second class of language change is *term to term evolution* or more generally *word to word evolution*. Different words are used to refer to the same concepts, people, places, etc. over time. To find the mentioning of such a concept, all different temporal references must be found. Because references do not need any lexical or phonetic overlap, this class is separate from spelling variation. Examples include *The Great War* and *World War I* that refer to the same war (i.e., underlying concept) or *St. Petersburg*, *Petrograd* and *Leningrad* that all refer to the same city (i.e., same entity).

The third class of change is *word sense evolution* or *semantic change*. Words change their meanings over time by adding, changing or removing senses. This means that even if a given word exists across the entire timespan, there is no guarantee that the word was always used to mean the same thing. As an example, assume that we are looking for *awesome leaders* over time. They are likely to appear today as well as several centuries ago; however, their interpretation over time has changed.

For normal users, not being aware of changes can limit their possibilities to find relevant information as well as interpret that information. As an example, not knowing of the name *Petrograd* or *Leningrad* will limit the amount of information that can be found on the history of the city. In the

context of knowledge-based culturomics, language change causes an additional set of problems. As an example, alignment of topics over time requires consolidating words and entities that represent the same concepts over time.

The classes of language change mentioned above fall into two general categories. Spelling, word to word as well as named entity evolution all fall into one category where the same concept or entity is represented using several different words over time. In contrast, word sense evolution falls into a different category where the word is stable over time but the senses change.

While the latter category is important for understanding of text, the former category of change is most relevant for knowledge-based culturomics and has a high impact on other technologies mentioned in this paper.

So far, spelling variation is the class that has received most attention by researchers and is the class that is best understood from a computational point of view. The work has focused on developing automatic/semi-automatic methods using rules as well as machine learning for creating dictionaries and mappings of outdated spelling. These resources are then used for search in historical archives to avoid missing out on important information [1,2,19,26,32].

Named entity evolution has been targeted using statistical methods as well as rule-based methods. Berberich et al. [8] proposes reformulating a query into terms prevalent in the past and measure the degree of relatedness between two terms when used at different times by comparing the contexts as captured by co-occurrence statistics. Kaluarachchi et al. [37] proposes to discover semantically identical concepts (=named entities) used at different time periods using association rule mining to associate distinct entities to events. Two entities are considered semantically related if their associated event is the same and the event occurs multiple times in a document archive. Tahmasebi et al. [77] makes use of special properties, namely *change periods* and *hand overs*, to find named entity evolution. If an entity is of general interest, then the name change will be of general interest as well during the period of change and the two names will be first-order co-occurrences in the text. E.g., *Sture Bergwall is better known to most people in Sweden as Thomas Quick*, ….[5]

For other classes of language change, these properties do not necessarily hold. Words used to refer to the same concepts, i.e., word to word evolution (see Fig. 2), are not likely to be first-order co-occurrences in a text and therefore more difficult to detect. E.g., in the past the word *fine* has been used to refer to the same concept as the modern word *foolish*. However, because there was no hand over as with the *Sture Bergwall* example above, the same methodology cannot be used to connect *fine* and *foolish*. Instead, to find general

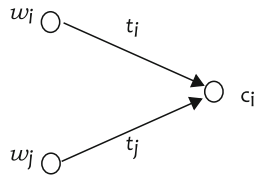


Fig. 2 Words w_i and w_j are considered temporal synonyms or word to word evolutions because they represent the same concept c_i at different points in time

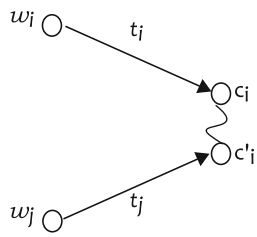


Fig. 3 When the gap between time points t_i and t_j is large, also the words constituting the concept are likely to change and hence we must be able to connect two concepts over time to find word to word evolution

word to word evolution, word senses must be used. A word is first mapped to a concept representing one of its word senses at one point in time. If one or several other words point to the same concept largely at the same period in time, then the words can be considered synonyms. If the time periods do not overlap, or only overlap in a shorter period, the words can be considered as *temporal synonyms* or word to word evolutions.

A key aspect to this methodology is the approximation of word senses. Because we cannot fully rely on existing contemporary dictionaries or other lexical resources, we must find these word senses automatically. Tahmasebi et al. [78] used word sense discrimination to approximate word senses. The curvature clustering algorithm was used and the output verified to correspond to word senses also for English text from the nineteenth century. Though the quality of the found word senses was high, comparably few word senses could be found. In other works, context-based approaches have been used [75] as well as methods making use of probabilistic topic models [40,88]. With the exception of Mitra et al. [58], Tahmasebi [78] and Wijaya and Yeniterzi [88], senses have not been followed over time. This is a key challenge, in particular for culturomics with large timespans, as very few senses remain exactly the same over longer periods of time, see Fig. 3. As words are exchanged in the vocabulary, also words that constitute the senses are exchanged. Therefore, to find word to word evolution, both word senses and the tracking of word senses must be handled automatically.

[5] http://www.gq.com/news-politics/newsmakers/201308/thomas-quick-serial-killer-august-2013.

3.1 Discussion

In culturomics, language change is a great obstacle for most other tasks like aligning and tracking opinions and topics or creating summaries.

Language change is interesting on its own; are there periods in which we change our language more than others? Is language change correlated to major events or cultural changes, e.g., introduction and wide spread use of Web log brought with it a whole set of new terms like *blog*, *blogging*, *blogger*. Are there other such events? Which type of change is more likely to be stable and which type of change is ephemeral and how have these classes changed over time? Did the introduction of user-generated content increase the amount of ephemeral changes?

The technical challenges lie in the lack of datasets and methods for automatic testing and evaluation over long timespans. Typically, tasks like word sense discrimination or named entity evolution have been developed and evaluated on short, modern timespans. While certain tasks are simpler to evaluate (e.g., named entity evolution) tasks like word sense discrimination or evolution are significantly harder, in particular when long timespans are taken into account and manual evaluation becomes infeasible.

4 Temporal opinion mining

Opinions and sentiments are an important part of our culture and therefore a central tool for analyzing cultural phenomena over time. The goal of opinion mining is to detect and classify expressions of approval or disapproval about a topic or entity from a text. The detected opinions can be used on their own to find documents expressing a given point of view, or aggregated to find opinions toward a given topic or event. Aggregation takes place both on the granularity of the topic and on the opinions contributed by different authors.

Research questions like: *are people generally more happy* can be answered by analyzing expressed opinions rather than counting the frequency of the word *happy*.

Currently, aggregation of opinions is typically only done for a single topic on a rather homogeneous dataset in a relatively short time period. However, we can gain more insights if we look at opinions over a longer period of time. In particular, recent years have seen much interest in applying opinion extraction techniques to texts from social media such as Twitter, which allows for a fast, realtime tracking of political sentiment developments [63]. Analyzing the temporal dynamics of these opinions, e.g., toward election candidates and their positions, can be useful for understanding the trends of political attitudes over time [18]. Furthermore, we can also extend this tracking to other cultural and social issues.

When we deal with historical collections, we will first have to analyze which documents contain opinions (e.g., personal letters and editorials) and how those opinions are expressed. Furthermore, the timescales in historical collections are larger and the number of documents is smaller. Whereas we can find opinion events on social media that last only some days or weeks and still contain opinions from thousands of users, the opinion events available in historical collections last for years or even decades while only having a small number of sources. This requires a more careful aggregation strategy that should also take into account external knowledge such as relations between the authors' contributing opinions.

Finding opinion expressions in text is a hard problem. On the one hand, we have texts from social media where the messages are typically very short and often written in a very informal style, which makes NLP analysis harder than for traditional texts [28]. On the other hand, we have media such as newspapers that often contain opposing opinions from different parties in the same text, sometimes adding an author's viewpoint as well. The problem then is to correctly attribute each opinion to the correct speaker, similar to the problems of entity resolution discussed in Sect. 2.1. Common additional problems are domain-specific vocabulary and slang, as well as the use of irony and sarcasm, which make it harder to find the intended meaning.

4.1 Related work

Opinion extraction (or "opinion mining", "sentiment analysis", or "subjectivity analysis") is a wide and diverse field of research [50,65]. Automatic retrieval of opinionated pieces of text may be carried out on a number of different levels of granularity. On the coarsest level, *documents* are categorized as opinionated or factual [90]. This may for instance be used to distinguish editorials from news [93]. Classification of *sentences* is also widely studied; these classifiers have been based on linguistic cues [87] or bag-of-word representations [64]. While most work use supervised machine learning, there are also unsupervised approaches [41,48].

In contrast to the early work, recent years have seen a shift toward more detailed and fine-grained problem formulations where the task is not only to find the text expressing the opinion, but also analyzing it: *who* holds the opinion (the holder) and toward *what* is it directed (the target); is it positive or negative (polarity); what is its intensity [16,35,39,89]. The increasing complexity of representation leads us from retrieval and categorization deep into natural language processing territory; the methods employed here have been inspired by information extraction and semantic role labeling, combinatorial optimization and structured machine learning. For such tasks, deeper representations of linguistic structure have seen more use than in the coarse-

grained case [27,35,39,73,76]. An extensive survey of opinion mining is provided by Pang and Lee [65] and more current developments are summarized by Tsytsarau and Palpanas [82].

4.2 Opinion mining for culturomics

The source documents for culturomics are very diverse. They span a large time period, range of registers and are written by authors of widely different backgrounds for varying purposes. As a result, the language as well as the kind of opinions expressed vary. We will describe both aspects as well as proposed solutions in the following.

The language in the documents is different in the diachronic as well as synchronic axis. Diachronically, we have the problem of language change (see Sect. 3), where the connotations of terms change over time. For example, the word *gay* is used to express a positive sentiment (*cheerful*), but now expresses a neutral or in some contexts even a negative sentiment. Similarly, *euphemism treadmill* describes the phenomenon that terms intended as neutral substitutes for negative or "taboo" terms soon become negative themselves, a process that is often iterated as for example in the series *Negro*, *black*, and *African-American*. Synchronically, the language used in texts confers different opinions based on the speaker, topic and type of document. Continuing the example given above, *gay* is used as an approximately neutral synonym for *homosexual* in most Western newspapers, but as a pejorative in many youth cultures. As another example, *primitive* is typically used negatively, but in the context of art it is used neutral and descriptive.

Many opinion mining methods incorporate polarity dictionaries, i.e., lists of words with an a priori known opinion value. As we have however seen, these dictionaries will have a large error when applied to diverse collections. Therefore, it is necessary to automatically adapt the dictionaries to the processed texts.

Prior work has shown that it is possible to generate and extend polarity dictionaries in an unsupervised manner using grammatical [31] or co-occurrence relations [83] between words. By applying these methods on Web data, we can also infer the polarity for slang and common misspellings [84], which improves the quality of opinion mining on, e.g., social media data. The mentioned algorithms build word graphs, where the edges indicate that there is a correlation or anti-correlation of the corresponding node words, e.g., because the terms occur in a conjunction "good *and* cheap" or disjunction "good *but* expensive". Each node is assigned a polarity by propagating labels starting from a seed set of terms with known polarity, taking into account the edges between terms.

To improve the quality of the generated dictionaries, we can incorporate additional metadata such as the document type and the temporal context of the source documents into the graph creation context. For example, we can split nodes for terms that had a change in meaning and thus learn the polarity of the word in different time periods. In this way, we derive a polarity dictionary that contains information about the temporal and contextual validity of its entries.

Another way to improve the accuracy of opinion mining in heterogeneous collections is to use intra-document relations between contained entities and terms (see Sect. 2.2). Current methods only consider the local context of the opinion expression for the classification. By building a global opinion model for a document with the speakers and entities contained in it, we can classify the opinions in that document from this global perspective and correct for local misclassifications from incorrect models and dictionaries. This makes the opinion classifier more robust against heterogeneity in time and type of the input documents.

4.3 Opinion aggregation

The opinions expressed in a collection express the values of the members of the cultures that created the documents. By aggregating the opinions about specific topics we can gain insights into these values and thus understand and characterize the culture. E.g., what can be said about the general levels of satisfaction (as expressed in written documents) in different countries post-World War II? Were there different change rates for countries/regions? Was there a correlation with winning vs. loosing sides?

Opinion aggregation is a hard problem, as we not only have to find all relevant viewpoints, similar to what we need to do in relation extraction, but also need to judge how representative these opinions are. Here, we have the problem that documents are primarily created by people having strong opinions on a topic and are therefore biased, whereas other people may not find the topic relevant enough to express their opinions in written form [14]. Historically, common folk were not even offered the possibility to express themselves in published media, leaving their voices less heard today. Traditionally, the venue for publication has been used as an indicator of the relevance of an opinion, so that the opinion expressed in an article in the New York Times was deemed more important than one published in a local newspaper. However, the Internet and especially social media can help us gather opinions from a larger sample of people instead of only this obviously biased subset, though caution must be taken when gathering data to avoid introducing new bias by, e.g., oversampling from western, internet using teenagers.

Current research is on finding influential users and detecting the spread of information in online social networks either through the network structure (e.g., [5,15]) or through the linguistic influence [43]. These methods can give us a way to approximate the influence of an author's opinion toward the overall culture.

4.4 Opinion dynamics

When our collection spans a longer period of time, we can in addition to aggregating opinions also analyze the dynamics of the aggregate sentiment and see evidence for changes in the value system. On the one hand, we can analyze changes in the polarity, for example, in the confidence in political or social institutions. On the other hand, we can also observe changes in the overall importance to the culture, for example of religion, by looking at the changes in the total number of opinions expressed.

Opinion changes belong to two distinct categories: first are changes that occur in response to an extra-ordinary event. As an example, winning an important contest will prompt more positive opinions about an athlete. The second category of change contains slow but continuous shifts in the aggregated opinion. Continuing the previous example, a continuous series of losses would cause the athlete to slowly lose favor. It is possible for an entity or topic to experience both types of changes at different times. For example, the European sentiment toward the Euro currency had been growing more positive continuously until the financial crisis, which then caused flare-ups of anti-Euro sentiment in multiple countries.

Opinion changes of the first kind can be detected using the associated *opinion change event*. Existing approaches to opinion change detection therefore rely on existing event detection algorithms to detect such events, either indirectly by finding events and analyzing the opinions in their temporal context [81] or directly by detecting changes in the aggregated opinions [6,62].

Slow changes are harder to detect, as they are typically more implicit. For example, a change in the attitudes toward foreigners will only be partially observable through opinion expressions about "foreigners" per se, but rather through changes in aggregate of opinion expressions about individuals seen as members of that group. Therefore, it is necessary to aggregate opinions about groups of related entities and analyze the dynamics of the aggregated sentiment.

4.5 Discussion

Opinion mining can help to answer many culturomics research questions by providing insight into the opinions and values expressed in a document collection. The unique challenges of culturomics, such as the diversity of document sources and topics, as well as the longer time periods, have however not been tackled in previous work. As the techniques for opinion mining mature, these challenges will need to be addressed, especially as they also increase the robustness of opinion mining for more general applications.

A particular area where knowledge-based culturomics can help to drive further research is the detection of slow changes in cultural opinions and values. These manifest themselves through opinions expressed toward groups of related entities from different time periods that form the topic of the opinion change topic. Tools like automatic relation extraction (see Sect. 2.2) and Named Entity Evolution Recognition (see Sect. 3) can help us find the relevant entities, so that we can find all relevant opinions and analyze their dynamics. Furthermore, key resource finding (see Sect. 6) can help us corroborate the detected changes with relevant source documents.

5 Temporal semantic summarization

Automatic summarization is the process of producing a limited number of sentences that outline a set of documents and constitute a summary (Fig. 4). The summary should cover the most important topics and avoid redundancy. Automatic summarization is helpful for preventing information overload and can allow people to quickly digest a large set of documents by extracting the most useful information from them. In the context of culturomics, the goal is to make a digest of a chronological course of events, spanning some period of time, gathered from many different sources. Since the input will come from several different sources, this is called multi-document summarization, and summarizing one single document will be considered as a special case.

Because of the long timespans in culturomics, there is a need to go beyond traditional summarization and extend the summaries to temporal summaries, taking into account changes over time. In addition, a good summary needs awareness of important entities, relations and events to fully qualify into knowledge-based culturomics. Therefore, we envision *temporal semantic summarization*, a novel approach making use of open-domain information extraction to extract entities, relations and events from the text to create better and more complete summaries. In the event extraction phase, a temporal analysis will take place, making the summaries temporally aware.

5.1 Related work

Previous work can roughly be divided into three different categories: extractive methods, abstractive methods, and orthogonally, methods that use information extraction.

In *extractive summarization* every sentence in the input documents is considered as a candidate. The task is to choose the most representative sentences, according to some metrics, to include in the summary. A common approach is to represent each sentence as a weighted vector of TF*IDF terms, and compute the cosine similarity between all pairs. Some extractive approaches start by deriving representations of the topics, whereafter the sentences are scored based on *impor-*

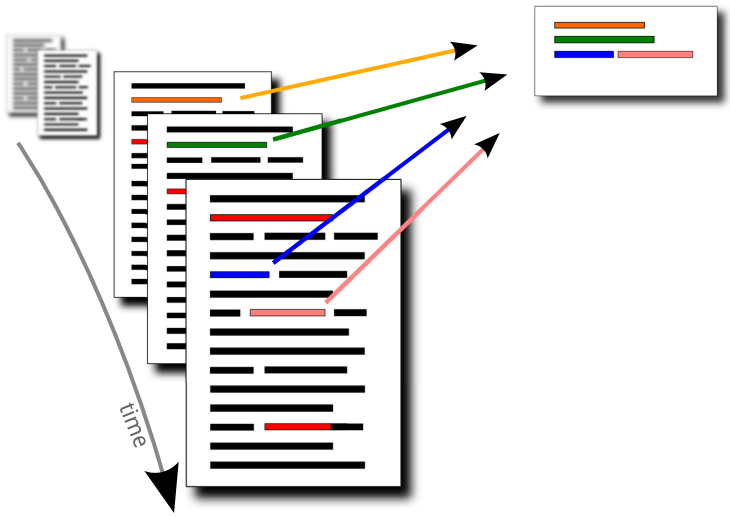


Fig. 4 Illustration of (Extractive) Automatic Summarization

tance. In other methods, the problem is seen as a trade-off between *diversity* and *similarity*. See [47,54,68] for more details.

In *abstractive summarization*, summaries are created by generating natural language sentences. All documents are parsed to build up knowledge in some representation and this knowledge is used to generate new sentences that summarize the documents. Ganesan et al. [23], Leskovec et al. [44], Rusu et al. [74] extracted subject, verb, object triplets from the documents to create a graph based on the semantic relations found. They use different techniques to extract important parts of this graph and generate the summary based on each part. While *abstractive summarization* has great potential, the inherent difficulties with changing language and large amount of noise in the collections used for culturomics leave abstractive methods for future work.

Recently, there has been some focus on using *information extraction to improve summarization*. Filatova and Hatzivassiloglou [21] presented a method that boosts event extraction coverage by relaxing the requirements for an event. Any pair of entities that appear in the same sentence and in combination with a connector word is considered an event. The connector word can be a verb or an action noun, as specified by WordNet. The sentences are scored using a greedy optimization algorithm and the results improve over an extractive baseline summarizer that did not account for redundancy.

Hachey [29] presented *General Relation Extraction*. To classify two entities as participating in a relation, they require the entities to be separated by no more than two words or by only one edge in a dependency parse tree. Connector words are derived from a model of relation types based on latent Dirichlet allocation, avoiding dependency on domain-specific resources like WordNet. The results were similar to those of Filatova and Hatzivassiloglou.

Ji et al. [34] used information extraction to perform relevance estimation and redundancy removal. This was done by combining the scores computed based on IE with scores based on coverage of bi-grams from the extractive summarizer used [24]. Combining these in the right way helped to create summaries that improved over the baselines.

All the above methods show an increased performance over an *extractive summarization* baseline. However, none perform any analysis on temporal information. Temporal summarization has been targeted in the past with a recent upswing [4,11,92]. Allan et al. [4] choose one sentence from each event within a news topic. Yan et al. [92] creates individual but correlated summaries on each date from a time-stamped collection of Web documents. Binh Tran [11] first ranks and chooses the top time points and then chooses the top sentences for each of the chosen time point. All works make use of news articles where typically one document describes one event and timestamps are exact and narrow. Automatic event detection or relation extraction was not considered. To achieve high-quality temporal summaries on historical texts with long timespans, temporal summarization techniques must be combined with information extraction.

5.2 Vision for temporal semantic summarization

For knowledge-based culturomics we envision temporal semantic summaries that build on the previous efforts [21, 29,34] in utilizing IE to create better extractive summaries. In the event extraction step, emphasis on extracting temporal information is needed, allowing for the final summary to be

coherent and chronologically structured. This is in particular important when documents are not properly timestamped, e.g., user-generated content from the Web or old books where timestamps are not fine-grained, alternatively events discussed can refer to historical events and do not correspond to time of publication.

By generating an entity–relation graph, extracted entities can be scored on importance, where importance is temporally variant. This allows to give higher scores to sentences that mention important entities given a time period. It is also possible to compare entity–relation graphs corresponding to different time periods to capture important changes and require these changes to be present in the summaries.

As an example, consider the life of *Marie Antoinette*. Born *Maria Antonia* in Austria as the daughter of the Holy Roman Emperor Francis I and Empress Maria Theresa. As a teenager she moved to France to marry Louis-Auguste and become the Dauphin of France and eventually the Queen of France. Modeling the entity–relation graph would show us significant changes over time as titles, place of residence and strongest connection (i.e., from parents to husband) change over time; also the general opinion toward her changed from being popular to being despised and in modern times again changed in a positive way. These general changes are important for a summary of her life and can be captured by a semantic temporal summarization approach.

Consider newspaper articles describing the world around us. They describe everything from people, events and conflicts to terrorist attacks, summits of world leaders, football stars scoring against some opposing team and companies changing leadership. Using information extraction (IE) to find important entities, relations and events in the input text gives many benefits in the summarization process. To mention a few:

1. Entity and relation extraction helps decide which sentences contain important information. Entity resolution and temporal resolution help to discriminate and consolidate mentioning of same and different entities. E.g., connecting *Maria Antonia* with *Marie Antoinette*.
2. Event extraction gives a natural way of filtering redundant mentions of an event (detecting, e.g., that a *bombing*, is the same as a *terrorist attack*).
3. Information extraction will ideally provide temporal information about events, helping us to order things in the output and disambiguate events.
4. Major changes in entity–relations or changes in events and their descriptions as found by comparing second-order information will provide a measure of importance and improve quality of the summaries.

The first step is extracting the necessary information. This includes building up an entity–relation graph of different kinds of entities (see below) and finding the events. The events can directly or indirectly (and to differing extent) relate to the entities that have been extracted. We will decide on entities in the entity–relation graph that are important, based on how they take part in events and based on the structure of the entity–relation graph.

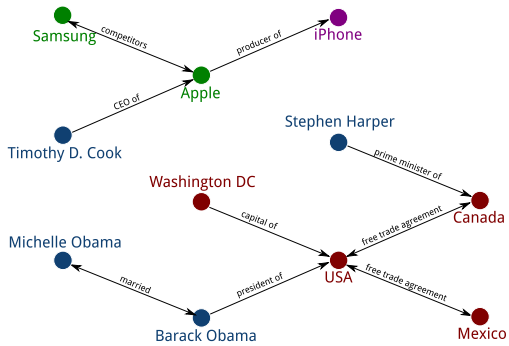


Fig. 5 Example of an entity–relation graph

Once the information is extracted, sentences can be scored based on what information they include. At this point in the process, the objective is similar to that of traditional extractive document summarization. The goal is to include sentences that are representative and relevant. One possibility is to use submodular optimization [47] to select the sentences based on the scores that they received in previous steps.

The process will make use of the following parts.

1. *Anaphora resolution* An anaphora is an expression that refers to some other expression, the most typical example being pronouns referring to some entity within the same sentence, or to other closely located sentences. Anaphora resolution is the process of resolving these expressions.
2. *Entity extraction* Entity extraction is the process of extracting entities from the text. Entities can be people, corporations and geo-political entities.
3. *Entity–relation graph extraction* A graph of relations will be extracted from the input documents (see Fig 5 and Sect. 2). This graph will contain relations between various kinds of entities, such as people, corporations and geo-political entities. Hence, a social graph could be seen as a subgraph of the entity–relation graph, containing only people and the relations between them. The system will also utilize properties from an entity–relation graph extracted from the entire corpus. From this, the aim is to compute a measure of global importance of different entities that are mentioned in the input text. This can be done by applying variants of the page rank algorithm to the network.

4. *Event extraction with temporal analysis* Events can happen once or repeatedly. They can also be hierarchical, with several smaller subevents spread over time, comprising one bigger event.

An entity in the entity–relation graph can be touched by a series of different events. This should be reflected in a summary, hence the temporal aspect of the event extraction is central.

5.3 Discussion

Temporal semantic summarization can help users deal with information overload, and follow developments within topics of interest over lengthier periods of time.

As the method of scoring information in the documents will rely on the success of extracting information, a great challenge lies in combining all these techniques in the best possible way. The aim is to develop a system that works on big amounts of input text, an aspect that can be seen both as a challenge and a source of strength. Algorithms need to be able to cope with the volume, variety and velocity of large amounts of data as well as be robust against noise (e.g., OCR errors or spelling variations in text).

An important aspect of a good summary in a culturomics context is *completeness* as well as *order*. A summary should contain at least the major events (e.g., marriage, mergers, battles) and the strongest relations (e.g., people, places and titles), all sorted in time. For temporal ordering, an important aspect becomes to sort information from documents written about an event during the event compared to information found in documents written after a significant time has passed. To sort the latter type of information, publication time cannot be used, instead time references must be extracted from the text itself, introducing an additional difficulty.

Finally, a proper summary should contain not only facts but also different viewpoints. Relying on the output of the opinion mining step, methods should be found to bring the opinions into the summary, also these in a temporal order. E.g., what were the different viewpoints of the Vietnam war during the war compared to the viewpoints after the war, and in particular, what are the differences.

An interesting addition to both summaries (and opinions) is the differences between automatically created summaries using information available during an event compared to information available post-event. What has been learnt afterwards and what has been forgotten? Which events or relations were deemed important in hindsight that were not considered important during the course of an event and vice versa?

6 Key resource finding

The primary focus within culturomics is the extraction of knowledge. Unfortunately, this is typically accomplished at the expense of losing the connection to the primary resources from where the knowledge arose. In some cases, this is unavoidable, e.g., corpus word frequencies can never be linked to any one primary resource but live in the combined space of all resources. Still, there are other instances where this may be remedied, e.g., relations, events, word senses, and summaries. In these cases *key resource finding*, in short KRF, which is the third processing layer in knowledge-based culturomics, can contribute credibility and traceability, with a secure basis in facts. In cases where all primary resources from where the knowledge arose are known, KRF is the task of choosing the most representative resources to avoid information overflow.

6.1 KRF and relations

By extracting entities from the text and inferring their relation, an entity–relation graph may be constructed, see Fig. 5 and Sect. 2.2. Given such a set of relations, the task of key resource finding is to find the minimal set of documents that best provide evidence of the correctness (or incorrectness) of the relations.

A simple example is finding a document describing some unspecified relation between two known entities, found for example using co-occurrence statistics or relation extraction. In the case of relation extraction, the relation is known and key resource finding is a straightforward search for documents containing the two entities exhibiting some relation. In the case where the two entities have many relations and where the user is only interested in one, the search is further constrained by specifying the relation. In a case where the relation is unknown (e.g., entities are often found in close proximity within text), key resource finding is the task of grouping documents according to different relations and choosing representative documents from each group.

More generally, a user may be interested in documents describing a complete entity–relation graph. In this case, the search turns in to the optimization problem of finding the minimum set of documents that cover the complete graph. This optimization may be further constrained by the number and length of documents that may be included. Also, the graph may be made more general and include, e.g., events.

6.2 KRF for entity disambiguation

If several entities share the same name, this will lead to ambiguity with regard to the identity of the referenced entity. Entity disambiguation aims to find and resolve this ambiguity. Using methodology described in Sect. 2.1 it is possible to automatically identify entities that are likely to be ambiguous.

Using a graph model like that presented by Hermansson et al. [33], the local graph structure surrounding the ambiguous

entity can be used and cliques can be identified. Then, key resource finding is the task of finding representative documents for each clique, which may be used both to resolve the ambiguity and to describe the disambiguated entities.

6.3 KRF for word senses

Clusters of words are used to approximate word senses, see Sect. 3 for a detailed discussion. These word senses can then be used to track meanings of words over time as well as find words that can be considered temporal synonyms. Concretely, a word sense for a given word is a bag-of-word representation of its context.[6] For example, the concept *music* can be described by words like *singing*, *guitar* and *Spotify*. However, automatically derived clusters lack labels on the word senses leaving it up to the user to interpret the meaning of the cluster from its words. This activity is both tedious and to some extent biased and would greatly benefit from being automated.

For labeling or specification of word senses, key resource finding is the task of finding the minimum set of documents (or sentences) that cover the largest amount of words among the describing words, preferably with linguistic relations between the words. Subsequently, documents or sentences can be ranked on informativeness. Presenting a few good sentences to the user, where the word sense is utilized. E.g., *Music is an art form whose medium is sound and silence.*[7]

An alternative method is to use the continuous space word representations presented in Mikolov et al. [55], infer categorical information regarding the words in the cluster, and use the most common category as label.

6.4 KRF for summarization

The objective of automatic summarization is to construct a set length summary that covers a maximal subset of the information conveyed in a set of documents. This can be done either by picking descriptive sentences (extractive summarization) or by constructing an abstract representation of the information contained in the text and subsequently generating a summary using natural language generation (abstractive summarization). More on summarization in Sect. 5.

Key resource finding constitutes the task of finding the smallest set of documents that cover the information in the derived summary to avoid redundancy and provide users with more details around the important elements covered in the summary. This can be accomplished using techniques similar to what is described in Sect. 6.1, with the summary as query.

[6] A bag of words is an unordered set containing all words represented in the original data.

[7] http://en.wikipedia.org/wiki/Music.

6.5 Related work

Key resource finding is an NLP task that falls under the category of information retrieval (IR). Traditional IR systems score the relevance of documents using keyword-based matching with respect to a given free text query. This approach has the advantage of being tractable for large-scale systems, but does not use higher level information in the text. Using semantic role labeling (SRL), it is possible to construct entity–relation graphs, that subsequently may be used to improve the precision of the search results.

Lin et al. [49] introduced an event-based information retrieval approach that finds relevant documents using entities and events extracted from the corpus. The query is created by the user in a structured form, but is limited to a single predicate. This approach could be generalized to the problem of finding a document describing a well-defined relation, as defined in Sect. 6.1.

A more general approach is employed by Kawahara et al. [38], where a predicate–argument graph is used to represent both the query and the documents. Document selection is done using binary operators (i.e., AND OR) on the predicate–argument structures as defined in the query. However, the method leaves SLR-based ranking as future work and does not provide a way to determine how many (or which) documents that are needed to cover the complete query, where the latter is necessary to answer the general question posed in Sect. 6.1.

6.6 Discussion

An important aspect of key resource finding is to step away from exact matching of words and sentences and allow for semantic similarity that captures, e.g., similarity between the words *guitar* and *music* because the guitar is an instrument used for creating music. For modern data, resources like WordNet [56] or DBpedia [12], their hierarchies or explicit relations can be used. For historical data where such resources may not be adequate, semantic similarity must be found using a combination of contemporary resources and unsupervised information extraction techniques. Because a large focus is given to historical resources in culturomics, finding semantic similarity measures and techniques for key resource finding is a challenge of great importance and future work is to investigate the utility of continuous word representations with temporal variance.

7 Discussion

As culturomics moves toward deeper analysis of large-scale corpora, the demand on processing capability moves beyond what any single computer can deliver regardless of its mem-

ory size or processor speed. This is a particularly important point as we move toward knowledge-based culturomics where many processes need more than one pass through the data and are computationally heavy. With the big data era, a new set of frameworks and tools have emerged that simplify the deployment of a distributed computing framework. To handle these challenges, focus must be given to large-scale distributed processing, for example, on frameworks like Hadoop [86].

Much of the discussion in this paper has regarded historical collections and the type of noise that must be handled with regard to, e.g., OCR errors and language change. However, it is important to keep in mind that similar problems arise when dealing with modern data, in particular from user-generated sources. A great variety in the form of slang, abbreviations and variations in spelling, word usage and grammar are present that can affect, e.g., NLP and semantic role labeling. To succeed with knowledge-based culturomics, attention must be paid to handle all these kinds of variation and provide methods that are robust against noise regardless of its origin.

One great advantage of knowledge-based culturomics is the possibility to iteratively apply different technologies and allow for automatic or semi-automatic improvements. As an example, while clustering word senses, we find that spelling and OCR errors often end up in the same clusters. Hence, using word sense clustering we can automatically determine which words to correct by, e.g., correcting words with low Levenshtein distance [45] to the word with the most frequent spelling. These corrections can be applied to the entire dataset and then clustering can take place again leading to higher quality clusters and possibly more or larger clusters. A concrete example is the following cluster derived from the Swedish Kubhist diachronic news corpus [80], for the year 1908: (*carbolinenm*, *kalk*, *cement*, *carbolineam*, *eldfast*, *carbolineum*). Three different spellings of *carbolineum* are present in the same cluster, two of them OCR errors which now can be corrected.

The strength of classical culturomics lies in its simplicity and low computational requirements. Because of the amount of data involved, many errors become statistically irrelevant and interesting patterns appear in the combination of (a) amount of data and (b) large timespans. However, there are many questions that cannot be answered using traditional culturomics; following concepts or entities over time that have changed their names or lexical representations, or distinguishing between different entities with the same name. Adding sentiment analysis, detecting changes in sentiment and the events that caused the changes is one example of moving beyond traditional culturomics.

Still, a deeper understanding of culturomics data requires human analysis. With knowledge-based culturomics the relevant resources can be found to help users focus their attention. Research questions like "what were the most relevant aspects of Marie Antoinette's life in terms of people, places, relations, what was the general sentiment of her and how and why did it change" can be answered. For each of these questions, documents can be provided for deeper analysis, going far beyond information retrieval with term matching techniques.

Current methods for NLP and statistical culturomics typically have the implicit assumption that the information expressed in documents is correct and can be taken at face value. For example, in opinion mining it is often assumed that the opinion expressed is the actual opinion of the speaker. First steps have been done to detected irony and sarcasm, where the implicit assumption obviously does not hold. However, this does not cover cases where authors misrepresent their opinions or factual knowledge for personal, political or social reason, e.g., to gain a financial advantage when talking about company stocks they own or to improve their standing in their peer group. Knowledge-based culturomics can be a first step to challenge the implicit assumption in two different ways: first, the richer model of the connections between entities, authors and documents makes it possible to aggregate first-order information in ways that are more representative than simple statistical summation. Furthermore, the findings of knowledge-based culturomics methods are linked back to source documents using key resource finding. This helps researchers validate the results instead of having to blindly trust the algorithms.

An interesting development in natural language processing, that we predict will have a strong impact on culturomics in the future, is the movement toward continuous vector space representation of words. These models embed words in a vector space that reveals semantic and syntactic similarities and can be used to lift the concrete text to a higher level of abstraction. This property was leveraged by Kågebäck et al. [36] to enhance the state-of-art extractive summarization algorithm introduced by Lin and Bilmes [47], using word embeddings to compare the information content in sentences. Further, these representations have been shown to exhibit interesting compositional properties, e.g., senses such as plurality and gender are captured as a linear translations in vector space, which can be exploited to draw conclusions not directly apparent from the text. An example of this is analogy testing, Mikolov et al. [55], where the question A relates to B as C relates to? Is answered using simple vector arithmetics $(v_B - v_A) + v_C \approx v_D$ where v_i denotes a vector representation of word i and v_D the vector representation of the sought answer. In culturomics word embeddings could be used for a variety of tasks, significantly extending the scope and impact of the project developed by Aiden and Michel [3] via the use of much richer representations in place of skip-grams. As a first example, we could directly compare words from different time periods by computing word embeddings

on time slices of history and projecting them into the same space, revealing trajectories for each word that plot the evolution of each corresponding word in a semantic space. Another example is to use the compositionality property, and compute how the vector space captures the popularity of a countries leader and subsequently track the evolution of a leaders corresponding word embedding for signs of dropping popularity, providing an early warning for civil unrest.

The aggregated data show that results of knowledge-based culturomics can be interesting for the general public and can help raise interest for historical resources. Imagine connecting modern newspaper articles with historical archives and for interesting current events being able to present the historical view. As an example, in an article on the topic *the Thailand Army declares martial law after unrest*, also historical aspects leading up to the event can be presented to users in condensed formats. Important people, places, events and different sentiments in summarized formats with links to descriptive original articles are all important to complement the current information and provide a more complete view.

8 Conclusion and future work

The notion of culturomics has turned out to be useful and interesting, not only from a research perspective but also for raising interest in historical data. However, without links to modern data it is less likely that culturomics will gain much interest from the general public. By moving toward knowledge-based culturomics, many more possibilities will be opened.

By offering support for integrating textual sources with modern and historical language, we can better bring information from historical resources (and the resources themselves) to users. This also allows researchers to better track information over long periods of time, without language change and variation getting in the way.

To achieve the goals set in this paper, particular focus must be given to natural language processing techniques and resources for entity, event and role extraction that constitute the basis of knowledge-based culturomics. Using this first-order information, knowledge-based culturomics opens up a large toolbox for researchers and the general public alike. Opinion mining and temporal semantic summaries allow for quick overviews and easy understanding of large amounts of textual resources over long periods of time. This analysis allows us to not only track information over time, but also clearly understand the change itself. As an example, we can find changes in opinion toward an entity over time as well as indications of the reason for the change, e.g., a specific event.

Finally, by linking extracted, second-order information to the primary resources, knowledge-based culturomics can offer credibility and traceability to this type of digital humanities research. Users can be offered a semantic interpretation of the important aspects of their queries as well as key resources that can serve as the starting point for further research.

While the benefits of knowledge-based culturomics are great, the challenges that lie ahead are equally large. Natural language processing is inherently hard as it is based on one of our most complex systems. However, tackling natural language at different periods in time, taking into consideration changes to the language, significantly increases the difficulty of the task. The increasing difficulty that comes with longer timespans applies to most other technologies that have thus far mostly been applied to data from short time periods. The strength of knowledge-based culturomics is the interconnected manner with which these challenges will be tackled, where the output of one technology can feed into another to improve the results.

Acknowledgments The authors would like to acknowledge the project "Towards a knowledge-based culturomics" supported by a framework Grant from the Swedish Research Council (2012–2016; dnr 2012-5738). We would also like to express our gratitude to the Centre for Language Technology in Gothenburg, Sweden (CLT, ⟨http://clt.gu.se⟩) for partial support. This work is also in parts funded by the European Commission under Alexandria (ERC 339233).

References

1. Adesam, Y., Ahlberg, M., Bouma, G.: bokstaffua, bokstaffwa, bokstafwa, bokstaua, bokstawa... towards lexical link-up for a corpus of Old Swedish. In: Proceedings of the 11th Conference on Natural Language Processing (KONVENS), Vienna, pp. 365–369. ÖGAI (2012). http://www.oegai.at/konvens2012/proceedings/54_adesam12w/54_adesam12w.pdf
2. Ahlberg, M., Bouma, G.: A best-first anagram hashing filter for approximate string matching with generalized edit distance. In: Proceedings of COLING 2012, Mumbai, pp. 13–22. ACL (2012). http://gup.ub.gu.se/records/fulltext/172769/172769.pdf
3. Aiden, E., Michel, J.-B.: Uncharted: Big Data as a Lens on Human Culture. Riverhead Books, New York (2013)
4. Allan, J., Gupta, R., Khandelwal, V.: Temporal summaries of new topics. In: Proceedings of the 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR 2001, pp. 10–18 (2001). doi:10.1145/383952.383954
5. Bakshy, E., Hofman, J.M., Mason, W.A., Watts, D.J.: Everyone's an influencer: quantifying influence on twitter. In: Conference on Web Search and Data Mining, WSDM 2011, pp. 65–74 (2011). doi:10.1145/1935826.1935845
6. Balog, K., Mishne, G., de Rijke, M.: Why are they excited?: Identifying and explaining spikes in blog mood levels. In: Conference of the European Chapter of the Association for Computational

Linguistics: Posters & Demonstrations, EACL '06, pp. 207–210 (2006). http://dl.acm.org/citation.cfm?id=1608974.1609010
7. Bellare, K., McCallum, A.: Learning extractors from unlabeled text using relevant databases. In: Sixth International Workshop on Information Integration on the Web (2007)
8. Berberich, K., Bedathur, S.J., Sozio, M., Weikum, G.: Bridging the terminology gap in web archive search. In: Proceedings of the 12th International Workshop on the Web and Databases, WebDB 2009 (2009). http://webdb09.cse.buffalo.edu/papers/Paper20/webdb2009-final.pdf
9. Bhattacharya, I., Getoor, L.: A latent Dirichlet model for unsupervised entity resolution. In: Siam International Conference on Data Mining (2006)
10. Bhattacharya, I., Getoor, L.: Collective entity resolution in relational data. ACM Trans. Knowl. Discov. Data (TKDD) **1**(1) (2007). doi:10.1145/1217299.1217304
11. Binh Tran, G.: Structured summarization for news events. In: International Conference on World Wide Web Companion, WWW '13 Companion, pp. 343–348 (2013). http://dl.acm.org/citation.cfm?id=2487788.2487940
12. Bizer, C., Lehmann, J., Kobilarov, G., Auer, S., Becker, C., Cyganiak, R., Hellmann, S.: DBpedia—a crystallization point for the Web of Data. J. Semant. **7**(3), 154–165 (2009). doi:10.1016/j.websem.2009.07.002
13. Bunescu, R.C., Mooney, R.: Learning to extract relations from the web using minimal supervision. In: Annual Meeting of the Association for Computational Linguistics, ACL 2007, p. 576 (2007)
14. Calais Guerra, P.H., Veloso, A., Meira Jr, W., Almeida, V.: From bias to opinion: a transfer-learning approach to real-time sentiment analysis. In: Conference on Knowledge Discovery and Data Mining, KDD 2011, pp. 150–158 (2011). doi:10.1145/2020408.2020438
15. Cha, M., Haddadi, H., Benevenuto, F., Gummadi, K.: Measuring user influence in twitter: The million follower fallacy. In: International AAAI Conference on Weblogs and Social Media, ICWSM 2010 (2010). http://www.aaai.org/ocs/index.php/ICWSM/ICWSM10/paper/view/1538
16. Choi, Y., Breck, E., Cardie, C.: Joint extraction of entities and relations for opinion recognition. In: Conference on Empirical Methods in Natural Language Processing, EMNLP 2006, pp. 431–439 (2006)
17. Craven, M., Kumlien, J., et al.: Constructing biological knowledge bases by extracting information from text sources. In: Conference on Intelligent Systems for Molecular Biology, pp. 77–86 (1999)
18. Demartini, G., Siersdorfer, S., Chelaru, S., Nejdl, W.: Analyzing political trends in the blogosphere. In: Fifth International AAAI Conference on Weblogs and Social Media, ICWSM 2011 (2011)
19. Ernst-Gerlach, A., Fuhr, N.: Retrieval in text collections with historic spelling using linguistic and spelling variants. In: Joint International Conference on Digital Libraries, JCDL 2007, pp. 333–341 (2007). doi:10.1145/1255175.1255242
20. Exner, P., Nugues, P.: Constructing large proposition databases. In: International Conference on Language Resources and Evaluation, LREC 2012, p. 5 (2012)
21. Filatova, E., Hatzivassiloglou, V.: A formal model for information selection in multi-sentence text extraction. In: International Conference on Computational Linguistics, COLING 2004 (2004). doi:10.3115/1220355.1220412
22. Fillmore, C.J.: Frame semantics and the nature of language. Ann. N. Y. Acad. Sci. **280**, 20–32 (1976)
23. Ganesan, K., Zhai, C., Han, J.: Opinosis: a graph-based approach to abstractive summarization of highly redundant opinions. In: International Conference on Computational Linguistics, COLING 2010, pp. 340–348 (2010). http://dl.acm.org/citation.cfm?id=1873781.1873820
24. Gillick, D., Favre, B., Hakkani-tür, D., Bohnet, B., Liu, Y., Xie, S.: The ICSI/UTD summarization system at TAC 2009. In: Text Analysis Conference (2009)
25. Google Books. http://books.google.com/ (2013). Retrieved 26 June 2013
26. Gotscharek, A., Neumann, A., Reffle, U., Ringlstetter, C., Schulz, K.U.: Enabling information retrieval on historical document collections: the role of matching procedures and special lexica. In: Workshop on Analytics for Noisy Unstructured Text Data, AND 2009, pp. 69–76 (2009). doi:10.1145/1568296.1568309
27. Greene, S., Resnik, P.: More than words: syntactic packaging and implicit sentiment. In: Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics, Boulder, pp. 503–511. ACLs (2009). http://www.aclweb.org/anthology/N/N09/N09-1057
28. Günther, T.: Sentiment analysis of microblogs. Master's thesis, University of Gothenburg (2013)
29. Hachey, B.: Multi-document summarisation using generic relation extraction. In: Conference on Empirical Methods in Natural Language Processing, EMNLP 2009, pp. 420–429 (2009). http://dl.acm.org/citation.cfm?id=1699510.1699565
30. Haghighi, A., Klein, D.: Coreference resolution in a modular, entity-centered model. In: Human Language Technologies, HLT 2010, pp. 385–393 (2010). http://dl.acm.org/citation.cfm?id=1857999.1858060
31. Hatzivassiloglou, V., McKeown, K.R.: Predicting the semantic orientation of adjectives. In: Annual Meeting of the Association for Computational Linguistics and Conference of the European Chapter of the Association for Computational Linguistics, pp. 174–181 (1997)
32. Hauser, A., Heller, M., Leiss, E., Schulz, K.U., Wanzeck, C.: Information access to historical documents from the Early New High German Period. In: Digital Historical Corpora—Architecture, Annotation, and Retrieval, number 06491 in Dagstuhl Seminar Proceedings (2007). http://drops.dagstuhl.de/opus/volltexte/2007/1057
33. Hermansson, L., Kerola, T., Johansson, F., Jethava, V., Dubhashi, D.: Entity disambiguation in anonymized graphs using graph kernels. In: International Conference on Information and Knowledge Management, CIKM '13, pp. 1037–1046 (2013). doi:10.1145/2505515.2505565
34. Ji, H., Favre, B., Lin, W.-P., Gillick, D., Hakkani-Tur, D., Grishman, R.: Open-domain Multi-Document summarization via information extraction: Challenges and prospects. In: Saggion, H., Poibeau, T., Yangarber, R. (eds.) Multi-source Multilingual Information Extraction and Summarization. Lecture Notes in Computer Science. Springer (2011)
35. Johansson, R., Alessandro, M.: Relational features in fine-grained opinion analysis. Comput. Linguist. **39**(3), 473–509 (2013)
36. Kågebäck, M., Mogren, O., Tahmasebi, N., Dubhashi, D.: Extractive summarization using continuous vector space models. In: Proceedings of the 2nd Workshop on Continuous Vector Space Models and their Compositionality (CVSC), Gothenburg, Sweden, pp. 31–39. Association for Computational Linguistics (2014). http://www.aclweb.org/anthology/W14-1504
37. Kaluarachchi, A., Roychoudhury, D., Varde, A.S., Weikum, G.: SITAC: discovering semantically identical temporally altering concepts in text archives. In: International Conference on Extending Database Technology, EDBT/ICDT '11, pp. 566–569 (2011). doi:10.1145/1951365.1951442
38. Kawahara, D., Shinzato, K., Shibata, T., Kurohashi, S.: Precise information retrieval exploiting predicate–argument structures. In: Proceeding of the IJCNLP (2013)

39. Kim, S.-M., Hovy, E.: Extracting opinions, opinion holders, and topics expressed in online news media text. In: Workshop on Sentiment and Subjectivity in Text, pp. 1–8 (2006)
40. Lau, J.H., Cook, P., McCarthy, D., Newman, D., Baldwin, T.: Word sense induction for novel sense detection. In: Conference of the European Chapter of the Association for Computational Linguistics, EACL 2012, pp. 591–601 (2012). http://aclweb.org/anthology-new/E/E12/E12-1060.pdf
41. Lazaridou, A., Titov, I., Sporleder, C.: A Bayesian model for joint unsupervised induction of sentiment, aspect and discourse representations. In: Annual Meeting of the Association for Computational Linguistics, ACL 2013, pp. 1630–1639 (2013)
42. Lenhert, W., Cardie, C., Fisher, D., Riloff, E., Williams, R.: Description of the CIRCUS system as used for MUC-3. In: Message Understanding Conference. Morgan Kaufmann (1991). http://acl.ldc.upenn.edu/M/M91/M91-1033.pdf
43. Leskovec, J., Backstrom, L., Kleinberg, J.: Meme-tracking and the dynamics of the news cycle. In: SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 497–506 (2009). doi:10.1145/1557019.1557077
44. Leskovec, J., Grobelnik, M., Milic-Frayling, N.: Learning substructures of document semantic graphs for document summarization. In: Workshop on Link Analysis and Group Detection, LinkKDD 2004 (2004)
45. Levenshtein, V.I.: Binary codes capable of correcting deletions, insertions, and reversals. Sov. Phys. Dokl. **10**(8), 707–710 (1966)
46. Liberman, M.: String frequency distributions. In: Language Log posting, 3rd Feb (2013). http://languagelog.ldc.upenn.edu/nll/?p=4456
47. Lin, H., Bilmes, J.: A class of submodular functions for document summarization. In: Annual Meeting of the Association for Computational Linguistics: Human Language Technologies, ACL 2011, pp. 510–520 (2011)
48. Lin, C., He, Y.: Joint sentiment/topic model for sentiment analysis. In: Conference on Information and Knowledge Management, CIKM 2009, pp. 375–384 (2009)
49. Lin, C.-H., Yen, C.-W., Hong, J.-S., Cruz-Lara, S., et al.: Event-based textual document retrieval by using semantic role labeling and coreference resolution. In: IADIS International Conference WWW/Internet 2007 (2007)
50. Liu, B.: Sentiment analysis and opinion mining. In: Synthesis Lectures on Human Language Technologies. Morhan & Claypool Publishers (2012)
51. Mann, J., Zhang, D., Yang, L., Das, D., Petrov, S.: Enhanced search with wildcards and morphological inflections in the Google Books Ngram Viewer. In: Proceedings of ACL Demonstrations Track, Baltimore. ACL (2014) (to appear)
52. McCarthy, J.F., Lehnert, W.G.: Using decision trees for coreference resolution. In: International Joint Conference On Artificial Intelligence, pp. 1050–1055 (1995)
53. Michel, J.-B., Shen, Y.K., Aiden, A.P., Veres, A., Gray, M.K., Pickett, J.P., Hoiberg, D., Clancy, D., Norvig, P., Orwant, J., et al.: Quantitative analysis of culture using millions of digitized books. Science **331**(6014), 176–182 (2011)
54. Mihalcea, R., Tarau, P.: Textrank: Bringing order into texts. In: Conference on Empirical Methods in Natural Language Processing, EMNLP 2004 (2004)
55. Mikolov, T., Yih, W., Zweig, G.: Linguistic regularities in continuous space word representations. In: Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, pp. 746–751 (2013). http://www.aclweb.org/anthology/N13-1090
56. Miller, G.A.: WordNet: a lexical database for English. Commun. ACM **38**, 39–41 (1995)
57. Mintz, M., Bills, S., Snow, R., Jurafsky, D.: Distant supervision for relation extraction without labeled data. In: Joint Conference of the Annual Meeting of the ACL and the International Joint Conference on Natural Language Processing of the AFNLP, ACL 2009, pp. 1003–1011 (2009)
58. Mitra, S., Mitra, R., Riedl, M., Biemann, C., Mukherjee, A., Goyal, P.: That's sick dude!: Automatic identification of word sense change across different timescales. CoRR, abs/1405.4392 (2014). http://arxiv.org/abs/1405.4392
59. Moretti, F.: Graphs, Maps, Trees: Abstract Models for a Literary History. Verso (2005). ISBN 9781844670260
60. Newcombe, H.B., Kennedy, J.M., Axford, S.J., James, A.P.: Automatic linkage of vital records. Science **130**(3381), 954–959 (1959)
61. Ng, V.: Unsupervised models for coreference resolution. In: Conference on Empirical Methods in Natural Language Processing, EMNLP 2008, pp. 640–649 (2008)
62. Nguyen, T., Phung, D., Adams, B., Venkatesh, S.: Event extraction using behaviors of sentiment signals and burst structure in social media. Knowl. Inf. Syst. 1–26 (2012)
63. O'Connor, B., Balasubramanyan, R., Routledge, B.R., Smith, N.A.: From tweets to polls: linking text sentiment to public opinion time series. In: International AAAI Conference on Weblogs and Social Media, ICWSM 2010 (2010)
64. Pang, B., Lee, L., Vaithyanathan, S.: Thumbs up? Sentiment classification using machine learning techniques. In: Conference on Empirical Methods in Natural Language Processing, University of Pennsylvania, United States, pp. 79–86 (2002). doi:10.3115/1118693.1118704
65. Pang, B., Lee, L.: Opinion mining and sentiment analysis. Found. Trends Inf. Retr. **2**(1–2), 1–135 (2008)
66. Poon, H., Domingos, P.: Joint unsupervised coreference resolution with markov logic. In: Conference on Empirical Methods in Natural Language Processing, EMNLP 2008, pp. 650–659 (2008). http://www.aclweb.org/anthology/D08-1068
67. Project Gutenberg. http://www.gutenberg.org/. (2013). Retrieved 26 June 2013
68. Radev, D.R., Jing, H., Styś, M., Tam, D.: Centroid-based summarization of multiple documents. Inf. Process. Manag. **40**(6), 919–938 (2004)
69. Raghunathan, K., Lee, H., Rangarajan, S., Chambers, N., Surdeanu, M., Jurafsky, D., Manning, C.D.: A multi-pass sieve for coreference resolution. In: Conference on Empirical Methods in Natural Language Processing, EMNLP 2010, pp. 492–501 (2010)
70. Rahman, A., Ng, V.: Coreference resolution with world knowledge. In: Annual Meeting of the Association for Computational Linguistics: Human Language Technologies, HLT 2011, pp. 814–824 (2011). http://dl.acm.org/citation.cfm?id=2002472.2002575
71. Rastogi, V., Dalvi, N., Garofalakis, M.: Large-scale collective entity matching. VLDB Endow. **4**(4), 208–218 (2011). http://dl.acm.org/citation.cfm?id=1938545.1938546
72. Riedel, S., Yao, L., McCallum, A.: Modeling relations and their mentions without labeled text. In: Machine Learning and Knowledge Discovery in Databases, vol. 6323 of LNCS, pp. 148–163. Springer (2010)
73. Ruppenhofer, J., Somasundaran, S., Wiebe, J.: Finding the sources and targets of subjective expressions. In: International Conference on Language Resources and Evaluation, LREC 2008, pp. 2781–2788 (2008)
74. Rusu, D., Fortuna, B., Grobelnik, M., Mladenic, D.: Semantic graphs derived from triplets with application in document summarization. Informatica (Slovenia) **33**(3), 357–362 (2009)
75. Sagi, E., Kaufmann, S., Clark, B.: Semantic density analysis: comparing word meaning across time and phonetic space. In: Workshop on Geometrical Models of Natural Language Semantics, GEMS 2009, pp. 104–111 (2009). http://dl.acm.org/citation.cfm?id=1705415.1705429

76. Somasundaran, S., Namata, G., Wiebe, J., Getoor, L.: Supervised and unsupervised methods in employing discourse relations for improving opinion polarity classification. In: Conference on Empirical Methods in Natural Language Processing, EMNLP 2009, Singapore, pp. 170–179 (2009)
77. Tahmasebi, N., Gossen, G., Kanhabua, N., Holzmann, H., Risse, T.: NEER: an unsupervised method for Named Entity Evolution Recognition. In: International Conference on Computational Linguistics, COLING 2012, pp. 2553–2568 (2012). http://www.aclweb.org/anthology/C12-1156
78. Tahmasebi, N.: Models and algorithms for automatic detection of language evolution. Ph.D. thesis, Gottfried Wilhelm Leibniz Universität Hannover (2013)
79. Tahmasebi, N., Niklas, K., Zenz, G., Risse, T.: On the applicability of word sense discrimination on 201 years of modern english. Int. J. Digit. Libr. **13**(3–4), 135–153 (2013). doi:10.1007/s00799-013-0105-8. ISSN 1432-5012
80. The Kubhist Corpus. http://spraakbanken.gu.se/korp/?mode=kubhist. Språkbanken, Department of Swedish, University of Gothenburg
81. Thelwall, M., Buckley, K., Paltoglou, G.: Sentiment in twitter events. J. Am. Soc. Inf. Sci. Technol. **62**(2), 406–418 (2011). doi:10.1002/asi.21462
82. Tsytsarau, M., Palpanas, T.: Survey on mining subjective data on the web. Data Min. Knowl. Discov. **24**, 478–514 (2012). doi:10.1007/s10618-011-0238-6
83. Turney, P.D.: Thumbs up or thumbs down?: Semantic orientation applied to unsupervised classification of reviews. In: Annual Meeting of the Association for Computational Linguistics, ACL 2002, pp. 417–424 (2002)
84. Velikovich, L., Blair-Goldensohn, S., Hannan, K., McDonald, R.: The viability of web-derived polarity lexicons. In: Annual Meeting of the Association for Computational Linguistics: Human Language Technologies, ACL 2010, pp. 777–785 (2010)
85. Whang, S.E., Garcia-Molina, H.: Entity resolution with evolving rules. VLDB Endow. **3**(1–2), 1326–1337 (2010). http://dl.acm.org/citation.cfm?id=1920841.1921004
86. White, T.: Hadoop: The Definitive Guide. O'Reilly Media Inc (2012)
87. Wiebe, J., Bruce, R., O'Hara, T.: Development and use of a gold standard data set for subjectivity classifications. In: Annual Meeting of the Association for Computational Linguistics, ACL 1999, pp. 246–253 (1999)
88. Wijaya, D.T., Yeniterzi, R.: Understanding semantic change of words over centuries. In: Workshop on DETecting and Exploiting Cultural diversiTy on the social web, DETECT 2011, pp. 35–40 (2011). doi:10.1145/2064448.2064475
89. Wilson, T.A.: Fine-grained subjectivity and sentiment analysis: recognizing the intensity, polarity, and attitudes of private states. Ph.D. thesis, University of Pittsburgh, Pittsburgh, United States (2008)
90. Wu, Y., Oard, D.W.: Beyond topicality, finding opinionated documents. In: Annual Conference of the Association for Information Science and Technology, Vancouver (2000)
91. Wu, F., Weld, D.S.: Autonomously semantifying Wikipedia. In: Conference on Information and Knowledge Management, CIKM 2007, pp. 41–50 (2007)
92. Yan, R., Kong, L., Huang, C., Wan, X., Li, X., Zhang, Y.: Timeline generation through evolutionary trans-temporal summarization. In: Conference on Empirical Methods in Natural Language Processing, EMNLP 2011, pp. 433–443 (2011). http://dl.acm.org/citation.cfm?id=2145432.2145483
93. Yu, H., Hatzivassiloglou, V.: Towards answering opinion questions: separating facts from opinions and identifying the polarity of opinion sentences. In: Conference on Empirical Methods in Natural Language Processing, EMNLP 2003, pp. 129–136 (2003)

Paper IV

Editing Simple Graphs

P. Damaschke and O. Mogren

Reprinted from Journal of Graph Algorithms and Applications, Special Issue of selected papers from WALCOM 2014, **2014**

Editing Simple Graphs*

Peter Damaschke and Olof Mogren
Department of Computer Science and Engineering
Chalmers University, 41296 Göteborg, Sweden
[ptr,mogren]@chalmers.se

Abstract

We study the complexity of turning a given graph, by edge editing, into a target graph whose critical-clique graph is any fixed graph. The problem came up in practice, in an effort of mining huge word similarity graphs for well structured word clusters. It also adds to the rich field of graph modification problems. We show in a generic way that several variants of this problem are in SUBEPT. As a special case, we give a tight time bound for edge deletion to obtain a single clique and isolated vertices, and we round up this study with NP-completeness results for a number of target graphs.

1 Introduction

Graphs in this paper are undirected and have no loops or multiple edges. In an edge modification problem, an input graph must be modified by edge insertions or deletions or both, to get a target graph with some prescribed property. Edge editing means both insertions and deletions. Edge insertion is also known as fill-in. The computational problem, for a given target graph property and a given type of editsm is to use a minimum number k of edits. There is a rich literature on the complexity of such problems for a number of target graph properties, and also on their various applications. Here we cannot possibly survey them all, we only refer to a few representative papers on hardness results [1, 13]. Other edit operations related to graph minors are studied in [9], and the target graph is a single fixed graph. Ironically, results are missing on edge modification problems for some structurally very simple target graphs. Informally, "simple" here means that the graph becomes small after the identification of its twin vertices (see Section 2 for technical definitions). For any fixed graph H, our target graphs will be the graphs obtained from H by replacing vertices with bags of true twins.

Our motivation of this type of problem is the concise description of graphs with very few cliques (that may overlap) and some extra or missing edges. They appear, e.g., as subgraphs in co-occurence graphs of words, and they constitute meaningful word clusters there. Within a data mining project we examined a similarity matrix of some 26,000 words, where similarity is defined by co-occurence in English Wikipedia. By thresholding we obtain similarity graphs (Figure 1 shows a part of such a graph), and we consider subgraphs that have small diameter

*Original publication in Journal of Graph Algorithms and Applications 18 (2014),557–576

and only few cut edges to the rest of the graph. Words occurring in the same contexts form nearly cliques. These are often not disjoint, as words appear in several contexts. Furthermore, synonyms may not always co-occur (as different authors prefer different expressions), but they co-occur with other words. Relations like this give rise to various cluster structures. As opposed to partitioning entire graphs into overlapping clusters (as in [7]), we want to single out simple subgraphs of the aforementioned type. Experience in our project shows that some existing standard clustering methods generate poor word clusters which are either too small or dragged out and not internally dense. This suggested the idea of defining the clusters directly by the desired properties, and then to determine them by edge editing of candidate subgraphs. Next, instead of describing the clusters naively as edge lists we can list their vertices along with the few edited edges (to achieve cliques). Altogether this yields very natural word clusters, and by varying the threshold we also obtain different granularities. Applications of word clusters include sentence similarity measures for text summarization, search query result diversification, and word sense disambiguation. Thus, we believe that the problems are of importance, but they are also interesting as pure graph-algorithmic problems.

Overview of contributions:

For any fixed H, our edge modification problems are easily seen to be fixed-parameter tractable (FPT) with k as the parameter. As our main result we get in Section 3 that they even belong to the smaller class SUBEPT of problems solvable in subexponential time in the parameter. (Not very many natural SUBEPT problems are known so far, as discussed in [8].) Each of our edge modification problems has a $2^{\sqrt{k}\log k}$ time bound. The special case known as p-Cluster Editing, where H is the graph with p vertices and no edges, was recently treated in [8], using techniques like enumeration of small cuts. Our result is more general, and the quite different algorithm looks conceptually simpler, but at the price of a somewhat worse time for the special case. Therefore it remains interesting to tighten the time bounds for other specific graphs H as well.

Consequently, we then turn to the absolutely simplest graphs H: In Section 4 we study the (NP-complete) edge deletion problem towards a single clique plus isolated vertices. We give a refined FPT time bound where the target clique size c appears explicitly. Intuitively, $2k/c^2$ is an "edit density". Using an evident relationship to vertex covers we achieve, for small edit densities, essentially $O^*(1.2738^{k/c})$ time. For large enough k/c we invoke a naive algorithm instead, and the time can be bounded by $O(1.6355^{\sqrt{k \ln k}})$. The base 1.2738 is due to the best known Vertex Cover algorithm from [4]. Moreover, the bound is tight: We show that the base of k/c cannot beat the base in the best FPT algorithm for Vertex Cover.

Section 5 gives a similar FPT time bound for edge editing towards a single clique plus isolated vertices, a problem that has recently been proved to be NP-complete. In Section 6 we make some progress in proving NP-completeness results systematically, for many graphs H. The results indicate that almost all our modification problems, with rather few exceptions, might be NP-complete. But recall that, on the positive side, they are in SUBEPT.

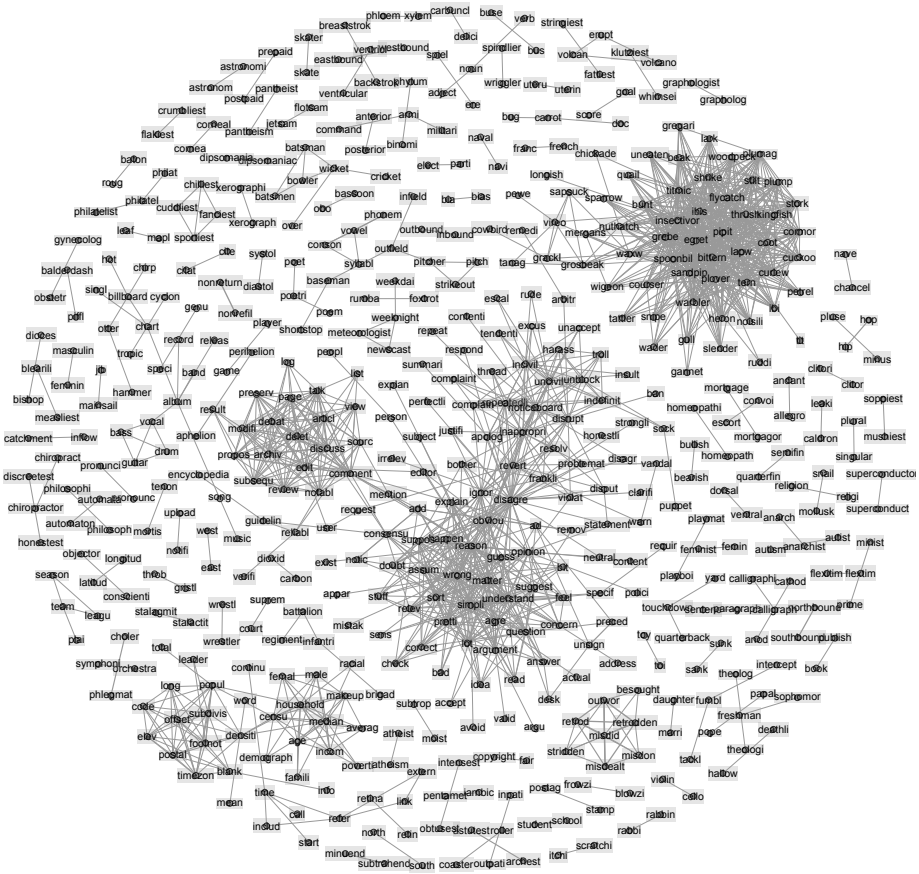


Figure 1: This Gephi visualization shows a small part of our word similarity graph, for some similarity threshold. Words have been stemmed prior to the calculations. One can clearly recognize the "almost cliques" structure, and in the middle we see an example of two overlapping cliques (the $H = P_3$ case). Also, the clusters make sense, in that they comprise related words. The data support our approach to define word clusters by edge-editing towards unions of very few cliques.

2 Preliminaries

The number of vertices and edges of a graph $G = (V, E)$ is denoted n and m, respectively. The complement graph $\bar{G}$ of G is obtained by replacing all edges with non-edges and vice versa. We also use standard notation for some specific graphs: K_n, C_n, P_n is the complete graph (clique), the chordless cycle, the chordless path, respectively, on n vertices, and $K_{n_1,n_2,\ldots,n_p}$ is the complete multipartite graph with p partite sets of n_i vertices ($i = 1, 2, \ldots, p$). The disjoint union $G + H$ of graphs G and H consists of a copy of G and a copy of H on disjoint vertex sets. $G - v$ denotes the graph G after removal of vertex v and all incident edges. Similarly, $G - X$ denotes the graph G after removal of a vertex set $X \subseteq V$ and all incident edges. The subgraph of G induced by $X \subseteq V$ is denoted $G[X]$.

A vertex cover of $G = (V, E)$ is a subset of V being incident to all edges. The Vertex Cover problem asks for a minimum vertex cover. The vertex covers in a graph are exactly the complements of independent sets, hence the complements of cliques in $\bar{G}$.

A graph class $\mathcal{G}$ is called hereditary if, for every graph $G \in \mathcal{G}$, all induced subgraphs of G are also members of $\mathcal{G}$. Any hereditary graph class $\mathcal{G}$ can be characterized by its forbidden induced subgraphs: F is a forbidden induced subgraph if $F \notin \mathcal{G}$, but $F - v \in \mathcal{G}$ for every vertex v.

The open neighborhood of a vertex v is the set $N(v)$ of all vertices adjacent to v, and the closed neighborhood is $N[v] := N(v) \cup \{v\}$. For a subset X of vertices, $N[X]$ is the union of all $N[v]$, $v \in X$. Vertices u and v are called true twins if uv is an edge and $N[u] = N[v]$. Vertices u and v are called false twins if uv is a non-edge and $N(u) = N(v)$. The true twin relation is an equivalence relation whose equivalence classes are known as the critical cliques of the graph. (The false twin relation is an equivalence relation as well.) In the critical-clique graph H of a graph G, every critical clique of G is represented by one vertex of H, and two vertices of H are adjacent if and only if some edge exists (and hence all possible edges exist) between the corresponding critical cliques of G. For brevity we refer to the critical cliques as *bags*, and we say that G is a "graph H of bags".

As stated earlier, an edge edit is an edge insertion or deletion. For every fixed graph H we define three edge modification problems as follows:

H-Bag Insertion / H-Bag Deletion / H-Bag Editing
Given: an input graph G and a parameter k.
Problem: Change G by at most k edge insertions / deletions / edits, such that the critical-clique graph of the resulting graph is H or an induced subgraph thereof.

We allow induced subgraphs of H in order to allow bags to be empty. Similarly we define the problems $H[0]$-Bag Deletion and $H[0]$-Bag Editing. The difference is that the target graph may additionally contain isolated vertices, that is, false twins with no edges attached. Thus, not all vertices are forced into the bags. More formally:

$H[0]$-Bag Deletion / $H[0]$-Bag Editing
Given: an input graph G and a parameter k.
Problem: Change G by at most k edge deletions / edits, such that the critical-clique graph of the resulting graph, after removal of all isolated vertices, is H or an induced subgraph thereof.

Problem $H[0]$-Bag Insertion is not mentioned above, as it easily reduces to H-Bag Insertion: As only insertions are permitted, the isolated vertices in an optimal solution are exactly the isolated vertices of G. Thus it remains to solve H-Bag Insertion on G without its isolated vertices.

We also consider problem variants where the bags have prescribed sizes. We sometimes refer to all the mentioned problems collectively as *bag modification problems*, for any fixed H. We say that editing an edge uv affects its end vertices u and v. A vertex is called unaffected if it is not affected by any edit.

Without loss of generality we can always assume that H has no true twins, because they could be merged, which leads to the same problems with a smaller graph in the role of H. For any fixed graph H understood from context, we use $\mathcal{H}$ to denote the class all graphs whose critical-clique graph is H or an induced subgraph thereof. Similarly, we use $\mathcal{H}[0]$ to denote the class of graphs consisting of all graphs from $\mathcal{H}$, but possibly with additional isolated vertices. All these classes are hereditary, and they are our classes of target graphs. (One could modify the problems by, e.g., allowing any target graphs with modular decompositions of some maximum size, but in this paper we also want a fixed structure given by H.)

We assume that the reader is familiar with fixed-parameter tractability (FPT) and basic facts, in particular with the notion of a branching vector. Otherwise we refer to [6, 15] for general introductions. A problem with input size n and an input parameter k is in FPT if some algorithm can solve it in $f(k) \cdot p(n)$ for some computable function f and some polynomial p. The $O^*(f(k))$ notation suppresses the polynomial factor $p(n)$. The subexponential parameterized tractable problems where $f(k) = 2^{o(k)}$ form the subclass SUBEPT. In our time analysis we will encounter branching vectors of a special form.

Lemma 1 *The branching vector $(1, r, \ldots, r)$ with q entries r has a branching number bounded by $1 + \frac{\log_2 r}{r}$, if r is large enough compared to the fixed q.*

Proof. Denoting the branching number by $1 + x$, we get the characteristic polynomial $(1 + x)^{r+1} = (1 + x)^r + q$, thus $x(1 + x)^r = q$. Trying $x := \frac{\log_2 r}{r}$, the left-hand side becomes $\frac{\log_2 r}{r}(1 + \frac{\log_2 r}{r})^{\frac{r}{\log_2 r}\log_2 r}$. As r grows, $(1 + \frac{\log_2 r}{r})^{\frac{r}{\log_2 r}}$ tends to $e > 2$, thus, there is a threshold r_0 such that, for $r > r_0$, the left-hand side exceeds $\frac{\log_2 r}{r} 2^{\log_2 r} = \log_2 r > q$. Clearly, the latter inequality holds since q is fixed, and we can just make r_0 large enough. Next, as $x(1 + x)^r$ is monotone in x, the true x is smaller than $x := \frac{\log_2 r}{r}$, for all $r > r_0$. It follows that $1 + \frac{\log_2 r}{r}$ is an upper bound on the branching number. $\diamond$

3 Fixed-Parameter Tractability

Some of our bag modification problems (in different terminology) are known to be NP-complete, among them cases with very simple graphs H. Specifically, for $H = K_1$, problem $H[0]$-Bag Deletion can be stated as follows. Given a graph G, delete at most k edges so as to obtain a clique C and a set I of isolated vertices. Equivalently, delete a set I of vertices incident to at most k edges, and delete all these incident edges, so as to retain a clique. The problem is NP-complete due to an obvious reduction from Maximum Clique.

Next, for any fixed p, the p-Cluster Editing problem asks to turn a graph, by editing at most k edges, into a disjoint union of at most p cliques. p-Cluster Insertion and p-Cluster Deletion are similarly defined. Observe that these are the bag modification problems for $H = \bar{K}_p$. It is known that p-Cluster Insertion is polynomial for every p, and so is p-Cluster Deletion for $p = 2$, but it remains NP-complete for every $p \geq 3$, whereas p-Cluster Editing remains NP-complete even for every $p \geq 2$ [16].

These hardness results provoke the question of fixed-parameter tractability of bag modification problems. By a well-quasi ordering argument based on Dickson's lemma [5] one can show that $\mathcal{H}$ and $\mathcal{H}[0]$ have only finitely many induced subgraphs, and then the general result from [2] implies that the bag modification problems are in FPT, for every fixed graph H. Although the argument is neat, we omit the details, because we will prove a stronger statement: membership in SUBEPT.

The following observation is known for Cluster Editing (that is, $H = \bar{K}_p$) due to [10]; here we show it for general H.

Proposition 2 *Any bag modification problem has an optimal solution where any two true twins of the input graph belong to the same bag (or both are isolated) in the target graph.*

Proof. First we consider H-Bag Editing. For a vertex v, an input graph, and a solution, we define the edit degree of v to be the number of edits that affect v. Consider any solution. For any equivalence class T of true twins, let $v \in T$ be some vertex with minimum edit degree. Consider any $u \in T \setminus \{v\}$. If the solution puts u in a different bag than v, then we undo all edits that affect u, and instead proceed as follows: We edit every edge uw, $w \neq v$, if and only if vw is edited. We also move u to the bag of v and undo the deletion of edge uv (if it happened). – Clearly, this yields a valid solution and does not increase the number of edits between u and the vertices $w \neq v$. Since we do not incur an additional edit of uv either, the new solution is no worse. Doing these changes for all $u \in T \setminus \{v\}$, and also for all T, we get a solution where any true twins end up in the same bag. This proves the assertion for H-Bag Editing.

For $H[0]$-Bag Editing we treat the set of isolated vertices as yet another bag. Then the same arguments apply. What is, however, not covered in the previous reasoning is the case when v is isolated and u is in a bag of true twins. But then u and v are not adjacent, neither before nor after the move, hence again the number of edits does not increase.

Finally, for the Insertion and Deletion problems, again the same arguments go through in all cases. Just replace "edit" with "insert" or "delete". $\diamond$

We make another simple observation.

Lemma 3 *In any bag modification problem, for a fixed graph H with p vertices, the input graph has at most $2k + p$ critical cliques (isolated vertices not counted), or the instance has no solution.*

Proof. The unaffected vertices induce a subgraph that belongs to $\mathcal{H}$ or $\mathcal{H}[0]$, respectively, hence it has at most p bags. Any affected vertex is adjacent to either all or none of the vertices of any of these bags (since the latter ones are unaffected). In the worst case, k edits

affect $2k$ vertices, and each of them becomes a critical clique of its own. Together this yields the bound. $\diamond$

Lemma 3 implies again that all bag modification problems for fixed H are in FPT: Assign every critical clique in the input graph to some bag of the target graph (or make its vertices isolated, in the $H[0]$ case). These are at most $p+1$ options for every critical clique. For the isolated vertices it suffices to decide how many of them we put in each bag, which are $O(n^p)$ options in total. Hence the time for this naive branching algorithm is $O^*((p+1)^{2k+p})$. Instead of this poor bound we will now show:

Theorem 4 *Any bag modification problem with a fixed graph H can be solved in $O^*(2^{\sqrt{k}\log k})$ time, hence it belongs to SUBEPT.*

Proof. First we focus on H-Bag Editing. The other problem variants can then be treated in the same way, with minor modifications.

Let a, $0 < a < 1$, be some fixed number to be specified later. To avoid bulky notation, we omit rounding brackets and work with terms like k^a as if they were integers. Let p denote the number of vertices of our fixed graph H. One difficulty is that the sizes of the p bags are not known in advance. Our preprocessing phase takes care of that.

Preprocessing phase: Initially all bags are declared *open*. For every bag we create k^a *places* that we successively treat as follows. At every place we branch: Either we *close* the bag and leave it, or we decide on a critical clique of the input graph and put any of its vertices in the bag. (Clearly, the latter choice of a vertex from a critical clique is arbitrary. By Proposition 2 we can even immediately fill further places with the entire critical clique, but our analysis will not take advantage of that.) Due to Lemma 3 these are at most $2k+p+1$ branches, hence the total number of branches is $(2k+p+1)^{pk^a} = O(k)^{pk^a} = 2^{k^a \log k}$. Note that p is fixed, and constant factors are captured by the unspecified base of log in the exponent.

Every open bag has now k^a vertices (where k is the initially given parameter value). We will not add any further vertices to closed bags. Vertices that are not yet added to bags are called *undecided*. Finally we do all necessary edits of edges between different bags, to stick to the structure of the target graph given by H, and subtract the number of these edits from k, the number of remaining edits.

Main phase: In every branch obtained in the preprocessing phase we apply further branching rules that will further reduce the parameter k by edits. The branching rules are applied exhaustively in the order described below. In the following we first consider the special case that all bags are open. Later we show how to handle the presence of closed bags, too.

All bags are open: We describe the branching rules and the situations after their exhaustive application.

- If there exists an undecided vertex u and a bag B such that u is adjacent to some but not all vertices of B, then we branch as follows: either insert all missing edges between u and B, or delete all edges between u and B. (But for now, u is not yet added to any bag.) The branching vector is some $(i, k^a - i)$ with two positive entries, or a better vector if already more than k^a vertices got into B.

- Now every undecided vertex u is either completely adjacent or completely non-adjacent to each bag B. We say that u fits in B, if u is adjacent to exactly those bags that belong to $N[B]$. Remember that H has no true twins. It follows that every vertex u fits in at most one bag.

- If there exists an undecided vertex u that fits in no bag, we branch as follows: we decide on a bag for u, put u in this bag, and do the necessary edits. Since u does not fit anywhere, we need at least k^a edits, thus the branching vector, of length p, is $(k^a, \ldots, k^a)$ or better.

- After that, every undecided vertex u fits in exactly one bag $B(u)$. Suppose that two undecided vertices u and v have the wrong adjacency relation. That is, either uv is an edge but $B(u)$ and $B(v)$ are not adjacent, or uv is not an edge but $B(u)$ and $B(v)$ are adjacent or $B(u) = B(v)$. We branch as follows: either we edit uv or not. If we don't, then u and v cannot be both added to their designated bags. Then we also decide on u or v and put that vertex in one of the other $p-1$ bags, which again costs at least k^a edits. Thus, the worst-case branching vector is $(1, k^a, \ldots, k^a)$ with $2p-2$ entries k^a.

- Finally, all undecided vertices have their correct adjacency relations, hence the graph belongs to $\mathcal{H}$.

Some bags are closed: We cannot treat closed bags like the open bags, since closed bags can be small, hence the above branching rules would not guarantee at least k^a edits. Therefore we modify the above procedure. Remember that all edits between bags were already done in the preprocessing phase, and undecided vertices can be put in open bags only.

Let U be the set of vertices of H corresponding to the open bags. Note that $H[U]$ may have true twins. In that case we merge every critical clique of $H[U]$ into one superbag. Since every open bag entered the main phase with k^a vertices, trivially, each superbag is larger than k^a.

To place the undecided vertices we perform exactly the same branching rules as above, but only on $H[U]$ (where superbags have the role of bags). Since we have fewer branches, the branching vectors do not get worse. A new twist is needed only when we actually add a vertex u to a superbag S. In every such event we also decide on the bag within S that will host u. Since every S came from a critical clique of $H[U]$, these latter choices do not change any more the adjacency relations of u with other vertices in the open bags and with other undecided vertices. Therefore we can take these decisions independently for all u, and always choose some bag in S that causes the minimum number of edits of edges between u and the closed bags.

Complexity result: The worst branching vector we encounter (see above) is $(1, k^a, \ldots, k^a)$ with $2p-2$ entries k^a. From Lemma 1 we obtain the bound $(1+\frac{a \log_2 k}{k^a})^k = 2^{k^{1-a} \log k}$ for some suitable logarithm base. Since the main phase is applied to every branch resulting from the prepocessing phase, we must multiply the two bounds: $2^{k^a \log k} 2^{k^{1-a} \log k}$. Choosing $a = 1/2$ yields the product $2^{\sqrt{k} \log k}$. (Note that the base is, arbitrarily, 2 because of the unspecified logarithm base.)

For H-Bag Deletion and H-Bag Insertion we proceed similarly. The only difference is that only one type of edits is permitted, hence some of the branches are disabled, which cannot make the branching vectors worse. In $H[0]$-Bag Deletion and $H[0]$-Bag Editing we can treat the set of isolated vertices like another bag; some necessary adjustments are straightforward. $\diamond$

4 Clique Deletion

If H is the one-vertex graph, then the $H[0]$ edge modification problems have as target graphs a single clique plus isolated vertices. Instead of the $H[0]$-Bag terminology we speak in this case of Clique Insertion, Clique Deletion, and Clique Editing, which is more suggestive. Clique Insertion is a trivial problem: Since only edge insertions are permitted, all vertices except the isolated ones must be connected to a clique, thus there is no choice. In this section we study:

Clique Deletion
Given: an input graph G and a parameter k.
Problem: Change G by at most k edge deletions so as to obtain a clique C and a set I of isolated vertices. Equivalently: Delete a set I of vertices incident to at most k edges, and delete all these incident edges as well, so as to retain a clique.

This should not be confused with problems having a split graph as target graph, as split graphs can have additional edges between C and I.

Lemma 5 *A partitioning of the vertex set of a graph G into sets C and I is a valid solution to* Clique Deletion *if and only if I is a vertex cover of $\bar{G}$. Moreover, a minimum vertex cover I of $\bar{G}$ also yields a minimum number of edge deletions in G. Consequently,* Clique Deletion *is NP-complete.*

Proof. The first assertion is evident. For the second assertion, note that Clique Deletion requests a vertex cover I of $\bar{G}$ being incident to the minimum number of edges of G. Since C is a clique, and every edge of G is either in C or incident to I, we get the following chain of equivalent optimization problems: minimize the number of edges incident to I, maximize the number of edges in C, maximize $|C|$, minimize $|I|$. The final assertion follows from the NP-completeness of Vertex Cover. $\diamond$

Clique Deletion is also in SUBEPT by Theorem 4, but besides the generic time bound with unspecified constants in the exponent, we are now aiming at an FPT algorithm with a tight time bound, as a function of k and $c := |C|$. With m being the number of edges in the input graph, clearly, c must satisfy $m - k \leq \frac{1}{2}c(c-1)$, thus $c \geq \frac{1}{2} + \sqrt{\frac{1}{4} + 2(m-k)}$. In an algorithm for Clique Deletion we may guess the exact clique size c above this threshold and try all possible sizes c, which adds a factor smaller than n to the time bound. Therefore we may assume in the following that c is already prescribed.

Before we turn to an upper complexity bound, we first give an implicit lower bound, relative to Vertex Cover parameterized by the solution size.

Proposition 6 *Any* Clique Deletion *algorithm with a time bound* $O^*(b^{k/c})$*, where* $b > 1$ *is some constant base, yields a* Vertex Cover *algorithm with a time bound* $O^*(b^v)$*, where* v *is the vertex cover size.*

Proof. We join our input graph $G = (V, E)$ with a clique K, and define $c^* := |K|$. Joining means that all possible edges between K and V are created. Observe that an optimal solution for the joined graph consists of an optimal solution for G (a partitioning of V into some C and I), with K added to C. Thus, if k edges are deleted in G, then $k + (n - c)c^*$ edges are deleted in the joined graph, and the size of the solution clique is $c^* + c$. Furthermore, the size of the vertex cover I in $\bar{G}$ is $n - c$.

The above reasoning holds for every size c^*. If we choose c^* "large" compared to n, but still polynomial in n, then the number of deleted edges and the clique size are $(n - c)c^*$ and c^*, respectively, subject to lower-order terms. Their ratio is the vertex cover size $v := n - c$. Using the original notations k and c for the number of deleted edges and the clique size, respectively, it follows that any FPT algorithm for Clique Deletion that runs in time bounded by some function $O^*(f(k/c))$ could be used to solve also Vertex Cover in $\bar{G}$ within $O^*(f(n - c)) = O^*(f(v))$ time. $\diamond$

Therefore, the best we can hope for is a Clique Deletion algorithm with a time bound $O^*(b^{k/c})$, with some constant base $b > 1$ that cannot be better than in the state-of-the-art Vertex Cover algorithm. This bound is also tight in a sense, as we will see below.

The exponent k/c is not an arbitrary measure, rather, it has a natural interpretation: It can be rewritten as $c\frac{k}{c^2}$, where the second factor can be viewed as an "edit density"; note that $\frac{2k}{c(c-1)}$ is the ratio of deleted edges and remaining edges in the target graph. For technical reasons it will be convenient to define the edit density slightly differently as $d := 2k/c'^2$ where $c' := c - 1$. Furthermore, in applications we are mainly interested in instances that are already nearly cliques, thus we keep a special focus on the case $d < 1$ in the following.

We start our algorithm for Clique Deletion with a single reduction rule: Consider any vertex v of degree smaller than c'. Clearly, v cannot be in a clique C of size c, hence it must be in I, and we can remove it without changing the problem. It also follows that it is correct to iterate this process.

Reduction rule: Remove any vertex v of degree smaller than c', along with all incident edges, and subtract the degree of v from the parameter.

After exhaustive application of this rule there remains a graph where all vertex degrees are at least c'. In order words, we compute the c'-core. From now on we can suppose without loss of generality that, already in G, all vertices have degree at least c'.

Lemma 7 *If a graph where all vertex degrees are at least* c' *admits a solution to* Clique Deletion *with at most* k *edge deletions,* $|C| = c' + 1$*, and* $|I| = i$*, then we have* $i \leq 2k/c'$*, and in the case* $d := 2k/c'^2 < 1$ *this can be improved to* $i \leq \frac{2}{1+\sqrt{1-d}} \cdot k/c'$*.*

Proof. Let h be the number of edges in I. Since at most k edge deletions are permitted, we have $ic' - h \leq k$. Since $h \leq k$ (or we must delete too many edges already in I), it follows $i \leq 2k/c' = dc'$.

For $d < 1$, this further implies $i \leq c'$. Using $h < i^2/2$, the previous inequality $ic' - h \leq k$ yields $ic' - i^2/2 \leq k$, thus $i^2 - 2c'i + 2k \geq 0$ with the solution $i \leq c' - \sqrt{c'^2 - 2k}$. (Recall that $i \leq c'$, thus the other solution is already excluded.) By using simple algebra this can be rewritten as

$$i \leq c' - \sqrt{c'^2 - 2k} = \frac{c'^2 - (c'^2 - 2k)}{c' + \sqrt{c'^2 - 2k}} = \frac{2k}{c' + \sqrt{c'^2 - 2k}} = \frac{2}{1 + \sqrt{1 - 2k/c'^2}} \cdot k/c'.$$

Finally remember $2k/c'^2 = dc$. $\diamond$

Note that the factor in front of k/c' grows only from 1 to 2 when d grows from 0 to 1. To make this factor more comprehensible, we may also simplify it to a slightly worse upper bound: Since $\sqrt{1-d} > 1-d$, we have $i \leq \frac{2}{2-d} \cdot k/c'$. We also remark that Clique Deletion is trivial if $k < c'$, because, after reduction to the c'-core, either there remains a clique, or the instance has no solution.

Theorem 8 Clique Deletion *can be solved in* $O^*(1.2738^{\frac{2}{1+\sqrt{1-d}} \cdot k/c'})$ *time if* $d < 1$*, and in general in* $O^*(1.2738^{2k/c'})$ *time.*

Proof. First we apply our reduction rule, in polynomial time. Let G be the remaining graph. Due to Lemma 5 it suffices then to compute a vertex cover of minimum size in $\bar{G}$. As for the time bound, the base comes from the Vertex Cover algorithm in [4], and the exponent comes from the bounded size i in Lemma 7. For large edit densities we may still use the algorithm with the simpler bound from Lemma 7. $\diamond$

In the time bound from Theorem 8 we may replace c' with c, which does not make a difference asymptotically, and write $O^*(1.2738^{2k/c})$. The following result gives a time bound as a function of k only, as in the previous section, but with a specified base.

Corollary 9 Clique Deletion *can be solved in* $O^*(1.6355^{\sqrt{k \ln k}})$ *time.*

Proof. Depending on c, either we run the algorithm from Theorem 8 in $O^*(1.2738^{2k/c})$ time, or we check in a brute-force manner all subsets of c vertices for being cliques. The latter method runs in $O^*((2k+c)^c)$ time, since at most $2k + c$ non-isolated vertices exist due to Lemma 3. For any fixed k, compare the expressions $1.2738^{2k/c}$ and $(2k+c)^c$. They decrease and increase, respectively, as functions of c. Hence their minimum is maximized if they are equal. This happens at approximately $c = 0.492\sqrt{k/\ln k}$. Plugging in this c yields the asserted time bound. $\diamond$

One may wish to improve the naive $O^*(2k + c^c)$ bound, and hence the Corollary, by fast exclusion of most c-vertex subsets as candidates for the clique C. However, the maximum clique problem cannot be solved in $O(f(c) \cdot n^{o(c)})$ time for any function f, under the Exponential Time Hypothesis [3].

5 Clique Editing

Recall that Clique Editing is the problem of editing at most k edges so as to obtain a clique C, say of size c, and a set I of $n - c$ isolated vertices. In the following theorem, c is part of the input.

Theorem 10 Clique Editing *with prescribed size c of the target clique is W[1]-complete in parameter $n - c$, hence also NP-complete.*

Proof. We argue with the (non-parameterized) optimization version and show that minimizing the number of edited edges is equivalent to finding a set I of $n-c$ vertices being incident to the minimum number of edges. This claim is verified as follows. Note that the edges incident to I are exactly those to be deleted, and minimizing deletions means maximizing the number of remaining edges. Since c is presecribed, this also minimizes the number of edge insertions needed to make C a clique. Due to [11], finding at least s vertices that cover at most t edges, known as Minimum Partial Vertex Cover, is W[1]-complete in the parameter s. Thus our assertion follows by letting $s := n - c$. $\diamond$

Note that we cannot simply use Theorem 10 to conclude NP-completeness of Clique Editing when the size c is arbitrary. The catch is that the prescribed clique sizes c in the reduction graphs may be different from c in optimal solutions to Clique Editing on these graphs, and our problem might still be polynomial for the "right" c. Actually, NP-completeness has been proved in [12].

Another equivalent way to state the Clique Editing problem is: Given a graph G, find a subset C of vertices that induces a subgraph that maximizes the number of edges minus the number of non-edges. Denoting the number of edges by $m(G)$, the objective can be written as $m(G[C]) - m(\bar{G}[C])$. This formulation also gives rise to an extension to a weighted version: For a given real number $w > 0$, maximize $m(G[C]) - w \cdot m(\bar{G}[C])$. Now the effect of w becomes interesting. The problem is trivial for $w = 0$ (the whole vertex set is an optimal C), and NP-complete if w is part of the input (since a maximum clique is an optimal C if w is large enough). But what happens in between? What is the complexity for any constant $w > 0$? We must leave this question open.

Next we propose an FPT algorithm for Clique Editing when k is the parameter. It works if c is part of the input, and hence also for free c, as we can try all, at most n, values of c. Membership in SUBEPT follows from Theorem 4, but as earlier we are also interested in the dependency of the time bound on c. The following algorithm uses similar ideas as the earlier ones.

Theorem 11 Clique Editing *can be solved in $O^*(2^{\log c \cdot k/c})$ time.*

Proof. We have to decide for every vertex whether to put it in C or in I.

Consider the following reduction rule: Put every vertex v of degree at most $(c-1)/2$ in I, and delete the incident edges. The correctness is seen as follows. Assume that $v \in C$ in the final solution. Since v has degree at most $(c-1)/2$, at least $(c-1)/2$ edges between v and the rest of C have been inserted. If we had instead decided $v \in I$, we would have inserted

no edges incident to v, but deleted the at most $(c-1)/2$ incident edges, which is not more expensive.

After exhaustive application of the reduction rule, there remains a graph of minimum degree at least $c/2$. We can assume without loss of generality that already the input graph has minimum degree at least $c/2$. We begin with branching. A vertex is called undecided if it is not yet put in C or I. Initially we guess one vertex of C, which adds only a linear factor to the time bound. All other vertices are undecided in the beginning.

As long as there exists an undecided vertex v which is not adjacent to all of C, we branch on v. In the $I := I \cup \{v\}$ branch we delete the, at least $c/2$, incident edges. (Whenever some vertex degrees fall below $c/2$ because of the deletions, we first apply the reduction rule again.) In the $C := C \cup \{v\}$ branch we insert at least one edge that is missing in C. After exhaustive application, all undecided vertices are adjacent to all vertices in C. If the undecided vertices form a clique, we are done, as we can add the undecided vertices to C, and if we get $|C| > c$, some surplus vertices are moved to I without branching. Suppose that the other case holds: two non-adjacent undecided vertices u and v exist. Then we branch by setting $C := C \cup \{u, v\}$ or $I := I \cup \{u\}$ or $I := I \cup \{v\}$. In the first branch we must insert an edge, and otherwise delete at least $c/2$ edges.

Our rules have, obviously, the worst-case branching vectors $(1, c/2)$ and $(1, c/2, c/2)$, and Lemma 1 yields the time bound. $\diamond$

6 Some Hardness Results

All bag modification problems are trivially in NP. In this section we prove the NP-completeness of bag modification problems for many target graphs H. We give a general construction that "lifts" NP-completeness from a considered H to larger graphs H'. That is, we will in polynomial time reduce H-Bag Editing to H'-Bag Editing, for certain graphs H and H' specified later on.

The general situation is as follows. Let the graph G and parameter k be any instance of H-Bag Editing, and let H' be a graph that contains H as an induced subgraph.

We choose a particular subset S of vertices of H' such that $H'[S]$ is isomorphic to H. Note that H may have several occurrences as induced subgraph in H', but we fix some set S. Let S_0 be some set of vertices of $H' - S$ which are adjacent to no vertices of S. (But S_0 is not necessarily the set of all such vertices.) Similarly, let S_1 be some set of vertices of $H' - S$ which are adjacent to all vertices of S. From G we construct a graph G' as follows, in polynomial time. We take $S_0 \cup S_1$ from H' and replace every vertex of $S_0 \cup S_1$ with a bag of size c, for some number $c > 2k$. Two bags are joined by all possible edges (by no edges) if the corresponding vertices in H' are adjacent (not adjacent). Then we add G and insert all possible edges between S_1 and the vertices of G. Figure 2 is a simple sketch of the construction that shall help remember the role of S_0 and S_1.

If G with parameter k is a yes-instance of H-Bag Editing, then we can take the at most k edits that yield a solution, and mimic them in the subgraph G of G'. This immediately implies that G' with parameter k is a yes-instance of H'-Bag Editing: we can map the vertices of G onto S to obtain an edited graph whose critical-clique graph is an induced

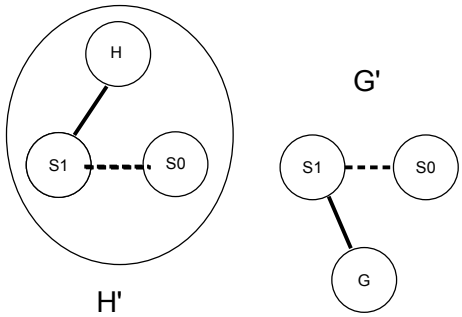


Figure 2: Thick lines mean that all possible edges between the induced subgraphs exist, dotted lines mean that some of the edges may exist. Inflation of the vertices to bags is not shown here.

subgraph of H'. (Assumption $c > 2k$ is not needed for this direction.)

However the converse does not hold in general: "If G' with parameter k is a yes-instance of H'-Bag Editing, then G' with parameter k is a yes-instance of H-Bag Editing." Our aim in the following is to show this converse under certain conditions on H and H'. The equivalence will then establish the desired reduction.

Specifically, suppose that the following technically looking condition is satisfied. Here, an embedding of a graph into another graph means that edges are mapped to edges, and non-edges are mapped to non-edges, i.e., the embedded graph is an induced subgraph of the host graph. Remember that H is isomorphic to $H'[S]$.

Embedding condition: Let J be any induced subgraph of H' isomorphic to $H'[S_0 \cup S_1]$. Accordingly, we embed J into any graph of $\mathcal{H}'$ and divide the vertex set of J in two sets U_0 and U_1. These are the sets of those vertices which come from S_0 and S_1, respectively. For every such embedding, let T be the set of vertices t such that $N[t]$ contains all vertices of U_1 and no vertex of U_0. Then the subgraph induced by T is always in $\mathcal{H}$.

Note that there may exist many possible embeddings of J into a host graph from $\mathcal{H}'$, and our condition must hold for each of them. We also remark that T may contain some vertices of U_1.

Now suppose that G' with parameter k is a yes-instance of H'-Bag Editing, that is, at most k edge edits in G' have produced a graph in $\mathcal{H}'$. Since k edits affect at most $2k$ vertices, but $c > 2k$, we conclude that every bag in the edited graph corresponding to a vertex of $S_0 \cup S_1$ still has at least one unaffected vertex. Accordingly, let U be some set of unaffected vertices, containing one such vertex from each of the bags. The subgraph induced by U in the edited graph is then isomorphic to $H'[S_0 \cup S_1]$. Let U_0 and U_1 be the subset of those vertices of U corresponding to vertices of S_0 and S_1, respectively. Then we have $U = U_0 \cup U_1$. Furthermore, since U is unaffected, all vertices of G are still adjacent (non-adjacent) to all vertices of U_1 (U_0).

Recall that H and H' are assumed to satisfy our embedding condition. Take as T the vertex set of G. It follows that, after editing, the vertices of G form a graph in $\mathcal{H}$. Since at most k edits have been done in the whole graph, we get that G with parameter k is a yes-instance of H-Bag Editing. To summarize:

Lemma 12 *If the embedding condition holds for graphs H and H' and some vertex sets S, S_0, S_1 (as specified above), then H-Bag Editing is reducible to H'-Bag Editing in polynomial time.*

The embedding condition looks more complicated than it is. When it comes to specific graphs H and H', it is often easy to apply, as we will see in the examples below. We only have to find suitable sets S_0 and S_1. For convenience we refer to the vertices in S_0 and S_1 as 0-vertices and 1-vertices, respectively, and we call any graph in $\mathcal{H}$ a *graph H of bags*.

Theorem 13 *H'-Bag Editing is NP-complete for, at least, the following graphs H':*
complete multipartite graphs where some partite set has at least 3 vertices;
the complete multipartite graph with partite sets of exactly 2 vertices;
K_3-free graphs with maximum degree at least 3.

Proof. H-Bag Editing for $H = \bar{K}_p$ is p-Cluster Editing, which is known to be NP-complete for every $p \geq 2$ [16]. By virtue of Lemma 12 we reduce H-Bag Editing for $H = \bar{K}_p$, with a suitable $p \geq 2$, to H'-Bag Editing for the mentioned graphs H'.

In a complete multipartite graph H', let b denote the size of some largest partite set, and assume $b \geq 3$. We choose $p = b - 1 \geq 2$. We let S_1 be empty, and we let S_0 consist of a single vertex in a partite set of size b. The vertices of H' being non-adjacent to this 0-vertex are in the same partite set, hence they induce a graph $\bar{K}_{b-1} = \bar{K}_p$. No matter where we embed our 0-vertex in a graph H' of bags, the set T as defined in the embedding condition forms a graph $H = \bar{K}_{b-1}$ of bags. (For partite sets smaller than b, note that bags are allowed to be empty.) Hence the embedding condition is satisfied.

Next consider $H' = K_{2,2} = C_4$. We choose $p = 2$, and we let S_1 consist of two non-adjacent vertices, while S_0 is empty. Clearly, the common neighbors of the two 1-vertices induce the graph $\bar{K}_2 = H$. The only possible embedding of our two non-adjacent 1-vertices in a graph $H' = C_4$ of bags is to put them in two non-adjacent bags. Then the set T forms again a graph $\bar{K}_2 = H$ of bags, thus the embedding condition is satisfied.

To prove NP-completeness of H'-Bag Editing for $H' = K_{2,\ldots,2}$, let $H = K_{2,\ldots,2}$ but with two vertices less. Then literally the same arguments as in the previous paragraph show that the embedding condition is satisfied. Using this observation as induction step and the case $H' = K_{2,2}$ as the induction base, this proves the claim by induction on the number of partite sets.

Finally, consider any K_3-free graph H' of maximum degree $d \geq 3$. (That is, H' is K_3-free but not merely a disjoint union of paths and cycles.) Fix some vertex v of degree d, and some neighbor u of v. We choose $p = d - 1$, $S_1 = \{v\}$, and $S_0 = \{u\}$. The vertices adjacent to v and non-adjacent to u obviosuly induce a graph $\bar{K}_{d-1} = \bar{K}_p$. For any embedding of an adjacent pair of a 1-vertex and a 0-vertex into a graph H' of bags, the set T forms a graph $H = \bar{K}_p$ of bags. This holds because every vertex in H' has at most d neighbors, they are

pairwise non-adjacent, and one of the d bags neighbored to the 1-vertex is already occupied by the 0-vertex. Once more, the embedding condition is satisfied. $\diamond$

The same construction also lifts NP-completeness results from $H[0]$ to $H'[0]$, whenever we can choose $S_1 = \emptyset$ and a suitable S_0. Our construction also works for H'-Bag Deletion and H'-Bag Insertion; the only modification is that only one type of edge edits is permitted. However, note that we need an NP-complete case to start with. For H'-Bag Deletion we can use $\bar{K}_p$ with $p \geq 3$. As for H'-Bag Insertion, remember that $\bar{K}_p$-Bag Insertion is polynomial [16] for every p. Still we can start from P_3 instead and get, for instance, the following results:

Theorem 14 *H'-Bag Insertion is NP-complete for, at least, the graphs $H' = P_3$, and $H' = P_n$ and $H' = C_n$ for each $n \geq 6$.*

Proof. P_3-Bag Insertion in G means to delete in $\bar{G}$ a minimum number of edges so as to obtain a graph consisting of a complete bipartite graph (biclique) and isolated vertices. This in turn is equivalent to the problem of finding a biclique with a maximum number of edges. The latter problem is NP-complete (even in bipartite graphs and hence in general graphs) due to [14]. Finally we reduce P_3-Bag Insertion to P_n-Bag Insertion for each $n \geq 6$ by setting $S_1 = \emptyset$ and S_0 isomorphic to P_{n-4}. Similarly, we reduce P_3-Bag Insertion to C_n-Bag Insertion for each $n \geq 6$ by setting $S_1 = \emptyset$ and S_0 isomorphic to P_{n-5}. It is straightforward to verify the embedding condition. $\diamond$

These results are applications of only one reduction technique. We may get out more NP-complete cases, but the construction also fails for some other graphs. We also remark that P_3-Bag Deletion is polynomial: consider the complement graph and proceed similarly as in [16]. It would be interesting to achieve a complexity dichotomy for all target graphs.

Acknowledgments

This work has been partially supported by the following project grants: "Generalized and fast search strategies for parameterized problems", grant 2010-4661 from the Swedish Research Council (Vetenskapsrådet), and "Data-driven secure business intelligence", grant IIS11-0089 from the Swedish Foundation for Strategic Research (SSF). The authors would also like to thank Gabriele Capannini for sharing the data and for initial discussions of the applications, and Henning Fernau for intensive discussions of some open problems during a short visit at Chalmers.

References

[1] P. Burzyn, F. Bonomo, G. Durán: NP-completeness Results for Edge Modification Problems. *Discr. Appl. Math.* 154, 1824–1844 (2006)

[2] L. Cai: Fixed-Parameter Tractability of Graph Modification Problems for Hereditary Properties. *Info. Proc. Lett.* 58, 171–176 (1996)

[3] J. Chen, X. Huang, I.A. Kanj, G. Xia: Strong Computational Lower Bounds via Parameterized Complexity. *J. Comput. Syst. Sci.* 72, 1346–1367 (2006)

[4] J. Chen, I.A. Kanj, G. Xia: Improved Upper Bounds for Vertex Cover. *Theor. Comp. Sci.* 411, 3736–3756 (2010)

[5] L.E. Dickson: Finiteness of the Odd Perfect and Primitive Abundant Numbers with n Distinct Prime Factors. *Amer. J. Math.* 35, 413–422 (1913)

[6] R.G. Downey, M.R. Fellows: *Parameterized Complexity*. Springer, New York (1999)

[7] M.R. Fellows, J. Guo, C. Komusiewicz, R. Niedermeier, J. Uhlmann: Graph-Based Data Clustering with Overlaps. *Discr. Optim.* 8, 2–17 (2011)

[8] F.V. Fomin, S. Kratsch, M. Pilipczuk, M., Pilipczuk, Y. Villanger: Tight Bounds for Parameterized Complexity of Cluster Editing. In: Portier, N., Wilke, T. (eds.) *30th Symp. Theor. Aspects of Computer Science STACS 2013*. LIPIcs, vol. 20, pp. 32–43, Dagstuhl (2013)

[9] P.A. Golovach, D. Paulusma, I. Stewart: Graph Editing to a Fixed Target. In: Lecroq, T., Mouchard, L. (eds.) 24th Int. Workshop on Combin. Algor. IWOCA 2013. LNCS, vol. 8288, pp. 192–205, Springer, Heidelberg (2013)

[10] J. Guo: A More Effective Linear Kernelization for Cluster Editing. *Theor. Comp. Sci.* 410, 718–726 (2009)

[11] J. Guo, R. Niedermeier, S. Wernicke: Parameterized Complexity of Vertex Cover Variants. *Theory Comput. Syst.* 41, 501–520 (2007)

[12] I. Kováč, I. Selečéniová, M. Steinová: On the Clique Editing Problem. In: *39th Int. Symp. on Math. Found. of Computer Science MFCS 2014*. LNCS, to appear (2014)

[13] A. Natanzon, R. Shamir, R. Sharan: Complexity Classification of some Edge Modification Problems. *Discr. Appl. Math.* 113, 109–128 (2001)

[14] R. Peeters: The Maximum Edge Biclique Problem is NP-complete. *Discr. Appl. Math.* 131, 651–654 (2003)

[15] R. Niedermeier: *Invitation to Fixed-Parameter Algorithms*. Oxford Lecture Series in Math. and its Appl., Oxford Univ. Press (2006)

[16] R. Shamir, R. Sharan, D. Tsur: Cluster Graph Modification Problems. *Discr. Appl. Math.* 144, 173–182 (2004)

Paper V

Summarizing Online User Reviews Using Bicliques

A. S. Muhammad, P. Damaschke, and O. Mogren

Reprinted from Proceedings of The 42nd International Conference on Current Trends in Theory and Practice of Computer Science, SOFSEM, LNCS, **2016**

Summarizing Online User Reviews Using Bicliques

Azam Sheikh Muhammad[1], Peter Damaschke[2], and Olof Mogren[2]

[1] College of Computing and Informatics, Saudi Electronic University (SEU), Riyadh 13316, Kingdom of Saudi Arabia, m.sheikh@seu.edu.sa
[2] Department of Computer Science and Engineering, Chalmers University, 41296 Göteborg, Sweden, [ptr,mogren]@chalmers.se

Abstract. With vast amounts of text being available in electronic format, such as news and social media, automatic multi-document summarization can help extract the most important information. We present and evaluate a novel method for automatic extractive multi-document summarization. The method is purely combinatorial, based on bicliques in the bipartite word-sentence occurrence graph. It is particularly suited for collections of very short, independently written texts (often single sentences) with many repeated phrases, such as customer reviews of products. The method can run in subquadratic time in the number of documents, which is relevant for the application to large collections of documents.

1 Introduction

Extractive summarization, i.e., selection of a representative subset of sentences from input documents, is an important component of modern information retrieval systems. Existing methods usually first derive some intermediate representation, then score the input sentences according to some formula and finally select sentences for the summary [16].

The field of automatic summarization was founded in 1958 by Luhn [14], who related the importance of words to their frequency of occurring in the text. This importance scoring was then used to extract sentences containing important words. With the advent of the World Wide Web, large amounts of public text became available and research on multi-document summarization took off. Luhn's idea of a frequency threshold measure for selecting topic words in a document has lived on. It was later superseded by $tf \times idf$, which measures the specificity of a word to a document and has been used extensively in document summarization efforts.

Radev et al. pioneered the use of cluster centroids in summarization in their work [18], with an algorithm called MEAD that generates a number of clusters of similar sentences. To measure the similarity between a pair of sentences, the authors use the cosine similarity measure where sentences are represented as a weighted vector of $tf \times idf$ terms. Once sentences are clustered, a subset from each cluster is selected. MEAD is among the state-of- the-art extractive

summarization techniques and frequently used as baseline method for comparing new algorithms.

Summarization is abstractive when new content is generated while summarizing the input text. Ganesan et al. [8], considered online user reviews, which are typically short (one or two sentences) and opinionated. They presented an abstractive approach that used parts of the input sentences to generate the output. For evaluations they provided the Opinosis dataset, consisting of user reviews on 51 different topics. Their system performed well evaluating generated summaries against those written by human experts. They also compare their results with the aforementioned MEAD [18] method.

Bonzanini et al. [4] introduced an iterative sentence removal procedure that proved good in summarizing the same short online user reviews from the Opinosis dataset. Usually, an extractive summarization method is focused on deciding which sentences are important in a document and considered for inclusion in the summary. The sentence removal algorithm by Bonzanini et al., [4] would instead iteratively choose sentences that are unimportant and remove them, starting with the set of all sentences in the input. The process continues until the required summary length is reached.

SumView [19] is also specialized on collections of short texts and uses feature extraction and a matrix factorization approach to decide on the most informative sentences. Besides the aforementioned work we refer to an extensive survey [16] discussing different approaches to sentence representation, scoring, and summary selection, and their effects on the performance of a summarization system.

Our contribution:
We propose a novel, purely combinatorial approach aimed at extractive summarization of collections of sentences. Since we will consider online user reviews as input, that typically are single sentences from independent authors, we speak of *sentences* rather than *documents* from now on. The idea is simply to find the key combinations of words appearing in several sentences. We work with a bipartite graph where the two vertex sets correspond to sentences and words, respectively, and edges exist between words and the sentences where they appear. Then, finding sets of sentences that share the same combination of words is equivalent to finding the bicliques in this graph. (Formal definitions follow in Section 2.) Finally we select bicliques for the summary according to further criteria. Leveraging recent advances in fast algorithms for determining the most similar sets, the approach is also computationally fast. Altogether this enables us to present a method for extractive summarization of short independent texts that should be attractive due to its conceptual simplicity and direct interpretability of the output. We show that it performs well on the benchmark Opinosis dataset [8]. It also outperforms state-of-the-art systems achieving the best precision, F_1 and F_2 measures. (See definitions and results in Section 6.2.)

As a delimitation, due to its very idea the method cannot be expected to extract good summaries of single complex texts, but this type of application is beyond the scope of this work.

2 Preliminaries

The following notation will be used throughout this paper. We consider any sentence as a bag of words, that is, as the set of distinct words, disregarding order and multiplicity of words. The ith sentence is denoted s_i. and $|s_i|$ is the number of distinct words, after removal of multiple occurrences and stopwords. The given set of sentences is T, and a summary is denoted $S \subset T$. Note that an extractive summary is merely a selection of the most informative sentences; no new text is generated. The length of a summary in terms of the number of sentences is denoted by ℓ.

Symbol $M_{s_i;s_j}$ stands for a sentence similarity measure. Several measures for sentence similarity have been explored for extractive summarization. The simplest ones are called the surface matching sentence similarity measures [15], because they do not require any sophisticated linguistic processing, and the similarity value is merely based on the number of words that occur in both of the sentences. In the present paper we are considering two of them:

- the *match* measure $M_{s_i;s_j} = |s_i \cap s_j|$,
- the *cosine* measure $M_{s_i;s_j} = |s_i \cap s_j| / \sqrt{|s_i| \times |s_j|}$.

Note that the cosine measure can be viewed as the match measure normalized by the sentence size.

Given a collection of sentences, we consider the *word-sentence occurrence graph*, that is, the bipartite graph $B = (T, W; E)$ with T and W being the set of sentences and words, respectively, where $sw \in E$ is an edge if and only if word w occurs in sentence s. Here the number of occurrences is disregarded, that is, edges are not weighted.

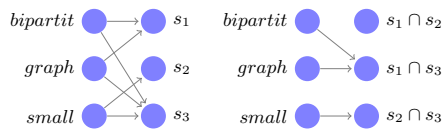
We use the standard notation $N(v)$ for the set of neighbored vertices of a vertex v in a graph. It naturally extends to vertex sets V, by defining $N(V) := \bigcup_{v \in V} N(v)$. A bipartite clique, *biclique* for short, is a pair (X, Y) of sets $X \subseteq T$ and $Y \subseteq W$ that induces a complete bipartite subgraph of B, that is, $N(X) \supseteq Y$ and $N(Y) \supseteq X$. A maximal biclique (not to confuse with a biclique of maximum size) is a biclique not contained in other bicliques. We call a subset Y of W of the form $Y = N(s_i) \cap N(s_j)$, with $s_i, s_j \in T$, a *2-intersection*. It corresponds to a biclique (X, Y) with $|X| = 2$.

3 Overall Idea

Customer reviews and similar text collections consist of sentences written independently by many authors. Intuitively, combinations of words appearing in several sentences should be important for a summary. Obviously they are bicliques in the word-sentence occurrence graph.

The problem of enumerating the maximal bicliques (and hence, implicitly, all bicliques) in a graph is well investigated, in particular, several output-sensitive algorithms with different running times are known [1, 6, 9, 11]. However, in general

Fig. 1. Biclique Summarization. Example sentences s_1: "This is a bipartite graph", s_2: "Look how small it is", s_3: "This bipartite graph is very small". After stemming and stopwords removal: s_1: "bipartit graph", s_2: "small", s_3: "bipartit graph small".



a graph has very many bicliques, and we need to select those which appear most relevant for a small extractive summary. We found that the large 2-intersections (see definitions above) are good candidates, by the following reasoning.

The word set Y of a biclique (X, Y) with $|X| = 2$ has been used by at least two independent authors, hence it has not only occurred by chance, in one sporadic statement. Moreover, Y is a maximal word set with this property. More repetitions of less specific word sets, that is, bicliques with $|X| > 2$ but smaller Y, do not seem to add much to a summary. The restriction to $|X| = 2$ also allows the use of standard pairwise similarity measures in the heuristic rules that afterwards select the sentences to be put in the summary.

An example of the concept is visualized in Figure 1. The bipartite graph is shown on the left while all possible 2-intersections are on the right. Since s_3 is present in both of the nonempty 2-intersections, it may be regarded as representative of the other two and thus selected for the summary.

4 Implementation Details

The given set of sentences T undergoes a preprocessing before passing it to the main algorithm. This involves stopwords removal and stemming. Words such as "am", "are", "by", "is", are called stopwords. They are common to many sentences and usually do not carry meaningful information when comparing texts in the context of summarization. Stemming reduces a word to its stem, base or root form. Stemming prevents mismatch between two words which apparently differ but are actually grammatical variations of the same word, such as singular and plural. In many cases stemming is achieved by chopping off the ends of words. Details about the stopwords list and stemming technique used in our implementation (which, however, do not affect the core ideas of the method) are described in Section 6.2.

In Algorithm 1 we give a pseudocode description of the method. As said in Section 3, in a biclique (X, Y), the Y part is the word set and X with the restriction $|X| = 2$ (because we need 2-intersections only), is the set containing i, j corresponding to a pair of sentences (s_i, s_j), $i \neq j$, in T. Our implementation using the subroutine $FindBiclique(s_i, s_j, sim)$ collects bicliques (X, Y) with $|Y| \geq 2$, that is, there are at least two words common in the corresponding

pair of sentences (s_i,s_j). For every such pair we also compute the value of the similarity measure specified by the parameter `sim` which is either equal to `match` or `cosine`; see Section 2. We find all such bicliques and this concludes the first part of the algorithm. There are two more major parts in our algorithm.

Algorithm 1 Biclique Summarization

Input: sim, top, ℓ, T
Output: S
Part-1: Find Bicliques:
$nBicliques \leftarrow 0$
repeat
  Choose a new pair (s_i, s_j), $i \neq j$, *in* T
  $[X, Y, simValue] \leftarrow FindBiclique(s_i, s_j, sim)$
  $Bicliques.add(X, Y, simValue)$
  $nBicliques \leftarrow nBicliques + 1$
until $NoMoreBicliques$
Part-2: Filter Important Bicliques:
$Bicliques \leftarrow SortBySimValue(Bicliques)$
$max \leftarrow \lceil \frac{nBicliques \times top}{100} \rceil$
$impBicliques \leftarrow Select(Bicliques[1...max])$
Part-3: Summary Selection:
$SentenceIDs \leftarrow UniqueSortedByFreq(impBicliques.X)$
$S \leftarrow GetSentences(T, SentenceIDs[1...\ell])$
return S

In the second part we filter out important bicliques, making use of the parameter `top` which defines percentage of the bicliques that should be kept. Typical values used are `10`, `30` and `50`.

First, the bicliques are sorted with respect to their decreasing similarity value and then we select best `top%` of the entire bicliques collection. In this way, the most informative bicliques are kept while the rest are discarded. For example when `top=10`, we select top `10` percent of the bicliques from the sorted list.

Finally, to select a summary, we need a ranking of the sentences appearing in the filtered important bicliques. In our implementation we simply count the occurrences of sentences in the filtered bicliques and take the ℓ most frequent sentences.

5 Processing Time

The time complexity is dominated by the time needed to determine the sentence pairs (2-intersections) with top similarity scores. This subproblem is known as *top-k set similarity joins* (more precisely: self-joins), where our k is the number of bicliques to keep. A basic implementation as displayed in Algorithm 1 loops through all pairs of sentences and therefore costs quadratic time in the number of sentences. However, the top-k set similarity joins problem is well studied for

different standard similarity measures, and fairly simple heuristic algorithms as proposed in [20, 2, 7] and related work have experimental running times far below the naive quadratic time bound. They can replace Part 1 and 2 of Algorithm 1. These heuristics rely on the very different word frequencies in texts, which essentially follow power laws [10]. Some further theoretical analysis of time bounds is given in [5].

For the sake of completeness we briefly outline these techniques. The first main idea is to process the words by increasing frequencies f and collect the pairs of sentences where any common words have been detected. A word appearing in f sentences trivially appears in fewer than $f^2/2$ of the 2-intersections. Since most words have low frequencies, the total number of such sentence pairs does not grow much in the beginning. The second main idea is called prefix filtering, which is actually a branch-and-bound heuristic. It bounds for every considered sentence pair the maximum possible value of the similarity measure, in terms of the number of (frequent) words not yet considered. This allows early exclusion of many pairs of sentences that cannot be among the top k pairs any more. Building on these principles, the "segment bounding algorithm" for the overlap measure [3] divides the set of words into segments according to their frequencies. Then the largest 2-intersections within the segments are computed by subset hashing, and finally the partial results are combined following some simple priority rules, until the top k pairs are established for a prescribed k. The authors of [3] report that their algorithm runs faster than those from [20] especially on large datasets. Finally, for the largest 2-intersections found, one can also filter those where other similarity measures are maximized.

Thanks to these techniques, our method can be implemented to run in subquadratic time. By way of contrast, this is apparently not possible for other summarization approaches like sentence removal [4] which is intrinsically more than quadratic.

6 Experimental Results

In this section, we present an empirical evaluation of the proposed method.

6.1 Dataset

Considering large amounts of highly redundant short text opinions expressed on the web, our experimental study focuses on assessing the performance of the proposed method on such a dataset which includes users stating their single line opinions about products or services, or commenting on some hot topics or issues on certain discussion forums and social media sites.

The Opinosis dataset [8] is particularly relevant to our purpose because it contains short user reviews in 51 different topics. Each of these topics consists of between 50 and 575 one-sentence user reviews made by different authors about a certain characteristic of a hotel, a car or a product (e.g. *"Location of Holiday*

Inn, London" and *"Fonts, Amazon Kindle"*). The dataset includes 4 to 5 gold-standard summaries created by human authors for each topic. The length of the gold-standard summaries is around 2 sentences.

6.2 Evaluation Method and Baseline Selection

Following standard procedure, we use ROUGE [12] for evaluation. ROUGE compares the system-generated summary with the corresponding gold-standard summaries for a topic and reports the assessment in terms of quantitative measures: recall, precision and F-measure. Recall is the number of words in the intersection of system-generated and the gold-standard summaries divided by the number of words in the gold-standard summary. Precision is the number of words in the intersection divided by the number of words in the system-generated summary.

F-measure, also called F_1 score, is a composite measure defined as the harmonic average of precision and recall. Sometimes, in order to emphasize the importance of recall over precision, another F-measure called F_2 score is also computed. On our benchmark dataset, Opinosis, F_2 scores are also reported in state-of-the art results.

The general definition od F-measure for positive real β is:

$$F_\beta = (1 + \beta^2) \cdot \frac{\text{precision} \cdot \text{recall}}{(\beta^2 \cdot \text{precision}) + \text{recall}}.$$

ROUGE works by counting n-gram overlaps between generated summaries and the gold standard. Our results show ROUGE-1, ROUGE-2 and ROUGE-SU4 scores, representing matches in unigrams, bigrams and skip-bigrams respectively. The skip-bigrams allow four words in between.

The experiments are aligned (in terms of ROUGE settings etc.,) with those of Bonzanini et al. [4], which provide state-of-the art results on extractive summarization on the Opinosis dataset. As mentioned in Section 1, they use a sentence removal (SR) algorithm. They used ROUGE to evaluate their methods, and MEAD [18], an extractive multi-document summarizer (see Section 1, too), as a baseline.

There are two different versions of the SR algorithm in [4], one based on similarity (SR_{SIM}) and the other one based on diversity (SR_{DIV}). We compare our method to both of them.

Summary length ℓ was fixed at 2 sentences. This matches the supplied gold-standard summaries and is also necessary to align our results to [4].

In addition to MEAD [18], they [4] use a brute-force method as a baseline which, for any given topic, enumerates all combinations of 2 sentences and chooses the pair that optimizes on the same scoring functions as used in their sentence removal algorithm.

Our implementation maximizes ROUGE scores: We consider all possible pairs of sentences within each topic, compute ROUGE-1, ROUGE-2 and ROUGE-SU4 scores (of recall, precision, F_1, and F_2) and choose the sentence pair with highest value. Choosing such pairs for all topics in the dataset gives us the maximum

scores, denoted with OPTIMAL, in our evaluations. These are the ideal scores attainable by an extractive summarization algorithm on this dataset.

We evaluate an implementation of the Biclique algorithm with sentence similarity measures *match* and *cosine*. For the *cosine* measure we let `top` vary between `10` and `30`. For *match* we evaluate with `top` fixed at `50`, which gave us the highest scores. Accordingly, the methods are abbreviated $Biclique_{Cosine1}$, $Biclique_{Cosine3}$, and $Biclique_{Match5}$, respectively.

The systems are evaluated with ROUGE version 1.5.5 using the following options: `-a - m -s -x -n 2 -2 4 -u`. For F_2, alongside the previous option, we also add: `-p 0.2`.

For preprocessing we make use of a Porter stemmer [17]. Stopwords were removed using the stopword list distributed as part of the ROUGE evaluation system [12].

6.3 Results

In Table 1 we show results for the experiments. R-1, R-2 and R-SU4 represent scores of ROUGE-1, ROUGE-2 and ROUGE-SU4 respectively. The best results (with the exception of the brute-force optimal scores) are shown in bold. The best scores among biclique alternatives are shown in italic. The brute-force optimal scores with respect to the evaluated measure in Table 1, marked in gray in the first row in each sub-table, are the maximum attainable scores. Baselines are shown at the bottom of each sub-table.

Recall:
With respect to recall, $Biclique_{Match5}$ has attained a ROUGE-1 value of 43.54 and ROUGE-SU4 of 16.36. This is better than the corresponding values (37.46 and 13.80) of the baseline method SR_{SIM} making an improvement of 16.23% and 18.55%, respectively. Similarly, with respect to ROUGE-2 value, $Biclique_{Match5}$ has attained 8.91 compared to the value 8.67 of the baseline method SR_{DIV}. Comparing ROUGE-1 and ROUGE-SU4, SR_{DIV} has attained better scores, and $MEAD$ has overall higher recall scores. However, it should be kept in mind that both of the latter are biased towards recall and do not perform well on precision compared to our method $Biclique_{Match5}$ in all three ROUGE settings.

Precision:
$Biclique_{Cosine1}$ is the best system for precision in all three settings of ROUGE, increasing the best scores among the baseline methods by over 85% for the ROUGE-1, over 90% for ROUGE-2, and over 223% for ROUGE-SU4.

F_1 :
The performance of $Biclique_{Cosine1}$ is consistent when we consider the measures F_1 and F_2, showing that the high precision is also combined with a high recall. For example, using $Biclique_{Cosine1}$, the best F_1-scores of the best performing baseline methods are improved by at least 34%, 35%, and 120% for ROUGE-1, ROUGE-2, and ROUGE-SU4, respectively.

Recall	R-1	R-2	R-SU4
$OPTIMAL$	57.86	22.96	29.73
$Biclique_{Cosine1}$	30.48	7.62	12.40
$Biclique_{Cosine3}$	31.94	8.13	13.40
$Biclique_{Match5}$	*43.54*	*8.91*	*16.36*
$MEAD$	**49.32**	**10.58**	**23.16**
SR_{DIV}	46.05	8.67	20.10
SR_{SIM}	37.46	9.29	13.80

Precision	R-1	R-2	R-SU4
$OPTIMAL$	57.35	22.07	36.07
$Biclique_{Cosine1}$	***36.85***	***9.88***	***17.58***
$Biclique_{Cosine3}$	33.29	9.07	15.03
$Biclique_{Match5}$	11.70	2.26	3.36
$MEAD$	9.16	1.84	1.02
SR_{DIV}	9.64	1.77	1.10
SR_{SIM}	19.87	5.18	5.44

F_1	R-1	R-2	R-SU4
$OPTIMAL$	46.57	19.49	23.76
$Biclique_{Cosine1}$	***32.67***	***8.41***	***13.93***
$Biclique_{Cosine3}$	31.85	8.35	13.46
$Biclique_{Match5}$	17.93	3.50	5.40
$MEAD$	15.15	3.08	1.89
SR_{DIV}	15.64	2.88	2.03
SR_{SIM}	24.38	6.23	6.31

F_2	R-1	R-2	R-SU4
$OPTIMAL$	48.41	20.45	24.72
$Biclique_{Cosine1}$	31.16	7.88	12.86
$Biclique_{Cosine3}$	***31.73***	***8.17***	***13.28***
$Biclique_{Match5}$	26.91	5.36	8.66
$MEAD$	26.27	5.43	4.34
SR_{DIV}	25.39	4.70	4.16
SR_{SIM}	29.92	7.54	8.08

Table 1. ROUGE scores for $Biclique_{Dice}$, $Biclique_{Cosine}$ - Biclique (with Match and Cosine sentence similarity, respectively) obtains the highest Precision, as well as F_1 and F_2 scores. $OPTIMAL$ scores (gray) contain the corresponding score for an optimal summary for each cell. SR_{SIM}, SR_{DIV} - Bonzanini et al (2013).

F_2 :
Here the best results are achieved by our method $Biclique_{Cosine3}$. We consistently improve on the best scoring method among the baselines by at least 6%, 8% and 64% for ROUGE-1, ROUGE-2, and ROUGE-SU4, respectively.

6.4 Discussion

Empirical evaluation of the method proposed in this paper suggests that we have a clear improvement over state-of-the-art extractive summarization results on the Opinosis dataset. Our method has shown substantial improvement compared to the existing results, especially for precision, F_1, and F_2 on all ROUGE settings. The only state-of-the-art result we cannot beat is the recall scores of the baseline methods MEAD and SR_{DIV}, which achieve the high recall at the expense of a sharp drop in precision (explained by the fact that these methods tend to choose larger sentences which results in high recall only).

Generally our method provides a balance between the two metrics, recall and precision, which is clear from the F_1-scores (Table 1). Still the biclique method has the flexibility of optimizing a certain metric, e.g., $Biclique_{Match5}$ is obtained using a parameter setting favoring the recall.

To conclude, supported by the best scores for all ROUGE settings for precision, F_1, and F_2 on the Opinosis dataset, our biclique method should be a good addition to the existing multi-document summarization systems, and it is particularly well suited to summarizing short independent texts like user reviews. With all its strengths, the method should also be appealing because of its simplicity and good time complexity. However, it is not expected to perform equally well on datasets of more complex texts which are not in the focus of our study.

7 Conclusions

We have proposed a novel method for extractive multi-document summarization, and showed with empirical results that it outperforms state–of–the–art summarization systems. Our method is based on the detection of bicliques in a bipartite graph of sentences and their words. To keep it simple and to highlight the strength of the main idea, we have evaluated only a basic version of the method which is already better than existing top-performing systems. The technique is also flexible as it can be easily adapted for a higher recall, a higher precision, or a balance between the two metrics. Considering the time efficiency, our proposed biclique algorithm offers, for standard similarity measures, the possibility of subquadratic running time, as opposed to the at least quadratic running time of the baseline sentence removal method.

We believe that more elaborate versions making use of deep similarity measures and combining with ideas from other methods, such as MEAD, can further enhance the performance. A natural extension of the preprocessing would be to cluster semantically related words (synonyms, etc.) and to replace the words from each cluster with one representative. As mentioned in Section 6.2 we use stopwords from ROUGE that also include negations. As the method does not rely on the meaning of words this is not an issue, still one could study the effect of different stopword lists.

Acknowledgments

This work has been supported by Grant IIS11-0089 from the Swedish Foundation for Strategic Research (SSF), for the project "Data-driven secure business intelligence". We thank our former master's students Emma Bogren and Johan Toft for drawing our attention to similarity joins, and the members of our Algorithms group and collaborators at the companies Recorded Future and Findwise for many discussions.

References

1. Alexe, G., Alexe, S., Crama, Y., Foldes, S., Hammer, P. L., Simeone, B.: Consensus Algorithms for the Generation of all Maximal Bicliques. Discr. Appl. Math. 145, 11–21 (2004)
2. Arasu, A., Ganti, V., Kaushik, R.: Efficient Exact Set-Similarity Joins. In: Dayal, U. et al. (Eds.) VLDB 2006. pp. 918–929, ACM (2006)
3. Bogren, E., Toft, J.: Finding Top-k Similar Document Pairs – Speeding up a Multi-Document Summarization Approach. Master's thesis, Dept. of Computer Science and Engin., Chalmers, Göteborg (2014)
4. Bonzanini, M., Martinez-Alvarez, M., Roelleke, T.: Extractive Summarisation via Sentence Removal: Condensing Relevant Sentences into a Short Summary. In: Jones, G.J.F. et al. (Eds.), SIGIR 2013. pp. 893–896, ACM (2013)
5. Damaschke, P.: Finding and Enumerating Large Intersections. Theor. Comp. Sci. 580, 75–82 (2015)
6. Dias, V.M.F., de Figueiredo, C.M.H., Szwarcfiter, J.L. 2007. On the Generation of Bicliques of a Graph. Discr. Appl. Math. 155, 1826–1832 (2007)
7. Elsayed, T., Lin, J., Oard, D.W.: Pairwise Document Similarity in Large Collections with MapReduce. In: ACL-08: HLT, Short Papers (Companion Volume), pp. 265–268, Assoc. Comp. Ling. (2008)
8. Ganesan, K., Zhai, C., Han, J.: Opinosis: a Graph Based Approach to Abstractive Summarization of Highly Redundant Opinions. In: Huang, C.R., Jurafsky, D. (Eds.) COLING 2010, pp. 340–348, Tsinghua Univ. Press (2010)
9. Gely, A., Nourine, L., Sadi, B.: Enumeration Aspects of Maximal Cliques and Bicliques. Discr. Appl. Math. 157, 1447– 1459 (2009)
10. Li, W.: Random Texts Exhibit Zipf's-law-like Word Frequency Distribution. IEEE Trans. Info. Th. 38, 1842–1845 (1992)
11. Li, J., Liu, G., Li, H., Wong, L. 2007. Maximal Biclique Subgraphs and Closed Pattern Pairs of the Adjacency Matrix: A one-to-one Correspondence and Mining Algorithms. IEEE Trans. Knowl. Data Eng. 19, 1625–1637 (2007)
12. Lin, C.Y.: ROUGE: A Package for Automatic Evaluation of Summaries. In: Moens, M.F., Szpakowicz (Eds.) ACL Workshop "Text Summarization Branches Out", pp. 74–81 (2004)
13. Lin, H., Bilmes, J.A.: A Class of Submodular Functions for Document Summarization. In: Lin, D., Matsumoto, Y., Mihalcea, R. (Eds.) ACL 2011, pp. 510–520, Assoc. Comp. Ling. (2011)
14. Luhn, H.P.: The Automatic Creation of Literature Abstracts. IBM Journal 2, 159–165 (1958)
15. Manning, C.D., Schütze, H.: Foundations of Statistical Natural Language Processing. MIT Press (1999)
16. Nenkova, A., McKeown, K.: 2012. A Survey of Text Summarization Techniques. In: Aggarwal, C.C., Zhai, C. (Eds.) Mining Text Data. pp. 43–76, Springer (2012)
17. Porter, M.F.: An Algorithm for Suffix Stripping. Program 14, 130–137 (1980)
18. Radev, D.R., Allison, T., Blair-Goldensohn, S., Blitzer, J., Celebi, A., Dimitrov, S., Drábek, E., Hakim, A., Lam, W., Liu, D., Otterbacher, J., Qi, H., Saggion, H., Teufel, S., Topper, M., Winkel, A., Zhang, Z.: 2004. MEAD - a platform for multidocument multilingual text summarization. In: LREC (2004)
19. Wang, D., Zhu, S., Li, T.: SumView: A Web-based Engine for Summarizing Product Reviews and Customer Opinions. Expert Systems with Appl. 40, 27–33 (2013)
20. Xiao, C., Wang, W., Lin, X. Haichuan Shang, H.: Top-k Set Similarity Joins. In: Ioannidis, Y.E., Lee, D.L., Ng, R.T. (Eds.) ICDE 2009, pp. 916–927, IEEE (2009)