Compact trie clustering for overlap detection in news

Richard Elling Moe, Dag Elgesem
Department of information science and media studies
University of Bergen

Abstract

We investigate document clustering through adaptation of Zamir and Etzioni's approach to online web document clustering. Specifically we generalize the Suffix Tree Clustering method to allow for a wider range of clustering techniques. We apply the modified technique to a corpus of news articles improving precision by 29% while running 8% faster than the original algorithm.

1 Background

The question of how much overlap and recirculation there is in news production is an important one for several reasons. It is important from the perspective of media pluralism: what is the range of diversity of news stories and what is just repetition? For media politics it is important to know where in the media system the news is produced and where it is mostly re-used [7, 16]. For the media industry it is important to be able to chart the overlaps in news production in order to monitor the profile of its competitors. For the general public it would give a better overview of the news stream to be able to see the degree of coverage of various topics and events. The development of automated methods for the detection of overlap and re-use of news content is therefore important. The long term goal of the work described in this paper is to model the flow of news in online newspapers over time and to visualize the concentration of coverage related to various topics.

Being able to cluster documents on the basis of having similar content would be instrumental to detecting reuse and overlap. In this paper we explore the application of a clustering technique to a news corpus.

Algorithms for document clustering have a rich history [9, 13, 14, 3, 17]. Zamir and Etzioni [18] demonstrate that documents can be clustered by applying the Suffix Tree method to short excerpts from them. This is an attractive feature in our context since such excerpts may be readily available for news articles in the form of front-page matter such as headlines, captions and ingresses. For this reason we chose to adapt Zamir and Etzioni's use of Suffix Tree Clustering and also because they report it to outperform a number of other algorithms, although in a different setting from ours. More recently, Eissen et al [5] presents a more nuanced picture. They point out that the technique has some weaknesses but maintains that these have little impact when applied to shorter texts and therefore pose no great threat in our context.

This paper was presented at the NIK-2013 conference; see http://www.nik.no/.

2 The Corpus

Since 2006 the Norwegian Newspaper Corpus [11] has downloaded the front pages of the 8 largest online newspapers and stored them in HTML format. Our corpus consisted of the 10 top stories in these newspapers, harvested daily from December 7 to 18, 2009.

We first wrote scripts that automatically pick the 10 top stories from the front pages and extracted the texts [10]. In the second step we manually coded 960 articles from the 12 day period . We developed a coding scheme that is based on categories that are used by Norwegian media scholars to classify news [1, 6]. The top categories in this topical classification are 'Domestic', 'International' and 'NorwegiansAbroad'. At the next level the categories are 'Economy', 'Crime', 'Social issues', 'Politics', 'Accidents', 'Culture and entertainment', 'Sport', 'Weather', and 'Science and technology'. In addition to an id, newspaper name, date and location on the front page, each article was coded according to the content categories mentioned, the topic, the event, the framing, the source of the article and the references it contained. We then wrote scripts that added this information from the manual coding as meta-tags to the extracted texts. Specifically, each article from the period received a tag, consisting of five categories, characterizing the content of the article. For example[1]:

$$Domestic - Crime - Traffic - Intoxication - Kristiansand$$
$$International - Economy - FinanceCrisis - Debt - Dubai$$

The tags represent a human judgement as to what the document is about. We think it is fair to assume that a high degree of overlap in tags will indicate overlap in content. Therefore we find it sensible to let these tags make up the ground truth for a precision/recall-style evaluation described in section 5.

The final preprocessing of the data was to reduce words to their ground form and to keep only certain kinds of words: nouns, verbs, adjectives and adverbs. This was achieved by marking up the text with syntactic information using the Oslo-Bergen tagger [12] and subsequently filter the document to leave only the desired words in the desired form.

3 Suffix Tree Clustering

A search engine retrieves a set of documents and presents them to the user as a list of snippets, i.e. a short excerpt taken from each document. However, the list can be overwhelmingly long so in order to further organize the result Zamir and Etzioni [18] proposed to cluster the documents wrt their content.

The backbone of Suffix Tree Clustering is the data structure known as a *compact trie*. A trie is a tree for storing sequences of tokens. Each arc is labelled with a token such that a path from the root represents the sequence of labels along the path. This simple structure effectively represents sequences and their subsequences as paths whereas the branching captures shared initial sequences. Note that a path does not necessarily represent a *stored* sequence. A stored sequence will have an end-of-sequence mark attached to its final node. (A star in figure 1.) The trie structure can be refined for the purpose of saving space. The idea is that sections of a path containing no end-of-sequence marks and no branching can be collapsed into a single arc. The *compact* trie thus allows arcs to be labelled with sequences of tokens. Figure 1 shows the compact trie for the sequences *aa*, *ab*, *abab*, *abc*, *babb*, *bc*, *cbba*, and *cbbc*.

[1] Tags are translated from Norwegian.

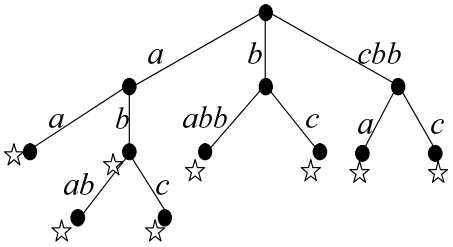

Figure 1: A compact trie

The suffix tree employed by Zamir and Etzioni is a compact trie storing all the suffixes of a set of given *phrases*[2], i.e. the snippets. That is, the arcs are labelled with sequences of *words*. Furthermore, the end-of-sequence mark is now the set of documents that the phrase occurs in. (In practice, the set of document ID's.) See figure 2.

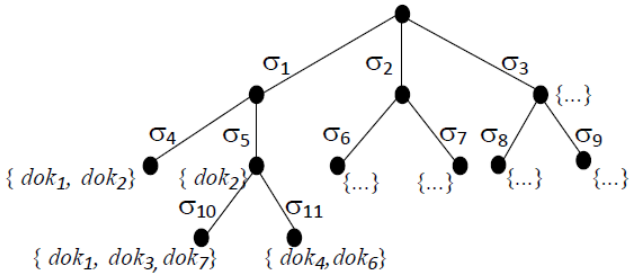

Figure 2: Suffix Tree

The suffix tree forms the basis for constructing clusters of documents. Each node in the tree corresponds to a *base-cluster*. The base-cluster is basically the set of documents associated with the subtree rooted in the node. The base-cluster also has a *label* which is composed of the labels along the path from the root to the node in question. This is illustrated in figure 3.

The base-clusters will be further processed to form the final clusters. That is, they are merged on grounds of being similar. Specifically, two base-clusters $\sigma : S$ and $\sigma' : S'$ are similar if and only if $\frac{|S \cap S'|}{|S|} > 0.5$ and $\frac{|S \cap S'|}{|S'|} > 0.5$

Note that there are alternative notions of similarity to be found, see for instance [2], but for the present discussion we stick to Zamir and Etzioni's original similarity-measure.

Now consider the *similarity-graph* where the base-clusters are nodes and there is an edge between nodes if and only if they are similar. The final clusters correspond to the connected components of the similarity-graph. That is, a cluster is the union of the document-sets found in the base-clusters of a connected component. Initially, the cluster is not given a designated label of its own. However, in other circumstances such labels could be useful [8]. In section 6 we will also find use for a cluster-label. So we add one, somewhat arbitrarily, by collecting the words from the base-cluster labels and sort them by their frequencies therein.

[2]Some would refer to this as a *generalized* suffix tree and reserve the term 'suffix tree' for such structures holding the suffixes of a single phrase.

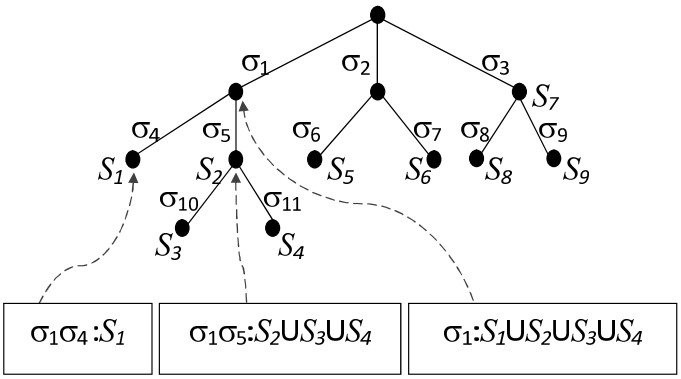

Figure 3: Baseclusters and labels

Clearly, the construction of final clusters requires every base-cluster to be checked for similarity with every other base-cluster. This process can be too computationally costly for practical purposes but Zamir and Etzioni circumvents the problem by restricting the merging process to just a selection of the base-clusters. For this purpose they introduce a *score* and form the final clusters from from only the 500 highest scoring base-clusters. We refer to this limit as λ, i.e. in [18] we have λ=500.

The score of a base-cluster $\sigma : B$ is defined to be the number $|B| \times f(\sigma)$ where the function f returns the *effective length* of σ. The effective length of a phrase is the number of words it contains that are neither too frequent, too rare nor appear in a given *stop-list* of irrelevant words. Specifically, 'too frequent' means appearing in more than 40% of the (total collection of) documents whereas a word is too rare if it appears in less that 3 documents. We refer to these thresholds as θ_{min} and θ_{max} respectively, i.e. in [18] we have θ_{min}=3 and θ_{max}=0.4. Furthermore, f penalizes single-word phrases, is linear for phrases that are two to six words long and becomes constant for longer phrases ([18, 8]).

Zamir and Etzioni applies this technique for clustering the results of a web search. The search engine will present a couple of lines from each retrieved document and these serve as the snippets from which the suffix-tree is built.

4 Variations on the scheme: Compact Trie Clustering

Zamir and Etzioni achieved good results but the technique was applied in a different context than ours. We will generalize their approach to make room for some adaptations. The idea is to keep the compact tries, but consider alternative ways of filling them. There are many conceivable ways of extracting snippets from the documents and we needn't confine ourselves to their suffixes. Furthermore, there is a number of parameters around that can be tweaked.

Why suffixes?

The use of suffixes will capture aspects of co-occurrence and order of words.

However, there seems to be a stronger focus towards the end of the phrase in the sense that a word will appear more times in the suffix-tree than the words that precede it. For example, the suffixes of the phrase 'one two three' are 'one two three', 'two three' and

'three' so the suffix-tree contains three occurrences of the word 'three' whereas 'two' occurs twice and 'one' only once.

A possible advantage of suffix-trees is the existence of a fast algorithm for building them. Ukkonen's algorithm [15] exploits the particular properties of suffixes to efficiently update a suffix-tree. This is referred to as the *on-line property* since it allows the suffix-tree to be built incrementally while meeting the user's expectations wrt response-time.

We disregard Ukkonen's algorithm for now because we will compare the use of suffixes with the use of alternative representations, which might not have such specialized algorithms. The comparison is better served by treating all representations with the same algorithm. Besides, the bottleneck lies elsewhere. Even a naive algorithm constructs the trie in a relatively short time compared to the massive job of merging base-clusters into final clusters. As mentioned, Zamir and Etzioni recognizes this too and tackles the problem by placing the limit λ on the number of base-clusters that are to be merged.

Snippets

Just what should go into the trie? We must consider how to represent the document appropriately. Will the first few sentences do or will it require larger chunks of text? There is even the option of using the whole article. In the present context we can make use of the front page matter, i.e. headline, ingress and caption, should there be a photo. So, for each news article its snippet could be the collection of such phrases, typically sentences, found on the front page. Given a snippet containing multiple phrases, each of them will be inserted into the trie separately, i.e. the suffix-tree would hold all the suffixes of each phrase in the snippet.

n-grams

We shall abandon the confinement to suffixes and in stead consider filling the compact trie with n-grams of the snippets, for some n. The 2-grams of 'one two three four' are 'one two', 'two three' and 'three four'. Like suffixes, n-grams will capture aspects of co-occurrence and order of words. In the example, the words 'one' and 'four' appear once while 'two' and 'three' appears twice. Generally, the words on the rims of the phrase will have some fewer occurrences in the trie, but the heavy bias toward the end is gone.

Apart from some tiny and uninteresting special cases, the number of words contained in the n-grams of a phrase is strictly fewer than the number of words in the corresponding suffixes. Thus, the use of n-grams may save some space but the question remains whether this reduction causes a loss of 'information' held by the trie and thereby harm the quality of the resulting clusters.

5 Research focus

We are faced with a wide range of options for setting up the clustering process. Roughly speaking, they vary along three dimensions:

1. *Snippet-extraction:* There are many conceivable selections of text that could make up the snippets. Say the first m% or every nth line or even the entire document.

2. *Snippet-expansion:* Given a snippet, how is it expanded into phrases that goes into the trie? Suffixes, n-grams for some (range of) n or something else? There is also a question of *granularity*, i.e. should the snippet be expanded as a whole or is it composed of multiple *units*, say the sentences, to be expanded one by one?

3. *Limits and thresholds:* There is a number of parameters built into the algorithm, including the frequency-thresholds θ_{min} and θ_{max} and the limit λ on the number of base-clusters to be fed into the merging phase.

In order to keep the scope of our investigation manageable we narrow down the focus. Specifically we will only compare snippets composed of headline, ingress and caption with snippets comprising the entire article. In either case the units for expansion are the sentences. Furthermore, regarding snippet-expansion, we aim only to compare suffixes with n-grams for a specific choice of n. Recall that the use of n-grams would reduce the amount of data to process but there is also the concern that the information held by the data then becomes impoverished. In an attempt to strike a balance, we choose n so as to maximize the number of words inserted into the trie. This is achieved by expanding a sentence of length k into its $\lceil k/2 \rceil$-grams.

Initially we keep Zamir and Etzioni's values for limits and thresholds but will also demonstrate a potential that lies in tweaking them.

Finally, even if we deem the on-line property less critical, we shall not completely abandon matters of efficiency. The difference in space-consumption between suffix-trees and n-gram tries is a natural consideration which could translate into differences of efficiency. If so, strategies for exploiting the difference could be to process the data quicker, to process as much data as possible in the same amount of time or something in between like trying to achieve the same effectiveness by spending a little more time on a little more data. The extra data in question could arise from extracting more text into snippets. Alternatively by feeding more base-clusters into the merging phase.

In section 6, efficiency is reported in terms of running times. These are of course highly dependent on the machinery that was used to obtain them. (In this case, a slightly outdated lap-top computer with standard hardware running standard software.) Such numbers are by no means absolute and are presented here solely for the purpose of comparison. Then, the main concern is that all experiments are conducted under equal conditions.

Accuracy, precision and recall

Zamir and Etzioni [18] measures the performance of their algorithm through precision. They apply the algorithm to the result of a web search, clustering the snippets returned from a query. The clusters are then assigned a score and sorted accordingly. Starting at the top, they run through the clusters and extract documents one by one until 10% of the total set of documents has been collected. This collection is then compared to the top 10% of the initial, non-clustered, output from the search engine. They report a precision of about 40% which is much better than the initial output and also outperforms a range of other clustering algorithms.

Although Zamir and Etzioni's use of precision is not directly transferrable to our context, the manually tagged portion of our corpus can serve as ground truth for precision/recall-style investigations. A *ground truth cluster* consists of all documents having identical tags, and only those documents. Thus, ground truth clusters are identifiable by tags. Precision would then measure the proportion of ground truth clusters among the clusters generated by the algorithm. Recall measures the extent to which the algorithm will recreate ground truth. Formally, if C is the set of clusters generated by the algorithm and G is the set of ground truth clusters then precision is calculated by $\frac{|C \cap G|}{|C|}$ while recall is $\frac{|C \cap G|}{|G|}$.

As described above, the algorithm sets the limit λ on the number of base-clusters that proceed to be merged into final clusters. This poses a serious threat to recall. Specifically, only 500 of the original 25378 base-clusters are retained. When the initial data for generating clusters is cut short by such a large amount we can not expect the algorithm to be able to recreate all the ground truth clusters. In fact, our data contains 667 ground truth clusters, while Zamir and Etzioni's original setup of the algorithm produces a total of only 159 clusters. For the case at hand, our news corpus, this has a devastating effect. A large proportion of the articles make up a ground truth cluster of its own, being the only texts on their topics. The original scoring favors bigger base-clusters, leaving many singleton clusters out and causing great harm to recall. We believe this particular choice of scoring is somewhat arbitrary and less suitable for our purposes. A more generally applicable scoring-scheme would be a separate dimension of research but for reasons of space we will not pursue it here. In stead we resort to the quick-fix of reversing the order induced by the original score. For these reasons, recall will receive less attention in the following.

We will primarily measure precision as described, but there are some points that deserve to be discussed. Precision/recall studies rely on a notion of *relevance*. In Zamir and Etzioni's case this is about a retrieved document being a sensible response to the query. In our setting, a cluster is considered relevant if it matches a ground truth cluster. The question is, should we require a *perfect* match?

Consider a cluster containing three articles, two of which are tagged

$$International - Politics - Climate - Obama - Copenhagen$$

and one with the tag

$$International - Politics - Climate - Draft - Copenhagen$$

These articles are all about the 2009 Copenhagen Climate Change Conference, but the cluster can not match a ground truth cluster perfectly because of the discrepancy of one word in the tags. Is it reasonable to deem this cluster irrelevant? This is largely a matter of the intended use, but also subject to human opinion. In precision/recall investigations, the notion of relevance typically rely on human judgement. Zamir and Etzioni [18] is no exception. In our context, relevance is decided by matching against ground truth clusters but there is a human factor here too in that the tags have been determined manually. We are reluctant to introduce more human judgement into the picture, fearing that we might open the door to mere whim. However we may obtain some flexibility by adding a degree of perfection to matching.

A cluster C *matches ground truth with discrepancy* $5-d$ if and only if d is the maximal number for which there is a ground-truth cluster G such that

- $G \subseteq C$
- d is the number of words common to all tags in $C \cup G$

Intuitively, discrepancy 0 is a perfect match while 5 means that there is no category that appears in all tags. Relaxing the requirement for perfection by some degree of discrepancy would allow more clusters that appear relevant to the human eye to be deemed relevant. This will be our main approach. Unfortunately, we can not claim that it is flawless.

First, it may still be stricter than desirable. Consider the cluster of four articles, three of which constitute the ground truth cluster

$$NorwegiansAbroad - Sport - Alpine - SuperG - Italy$$

while the last article is tagged

$$Domestic - Culture - Fashion - Blogging - Troms$$

This cluster should be dismissed for having discrepancy 5. After all, the discrepancy can't get worse. But the cluster nevertheless appears to be good, it's just slightly contaminated by one small piece of junk. The junk-article might even be relevant if it happens to be a report on blogging about the Super G racers in Italy.

Secondly, consider a cluster containing five articles, all tagged

$$NorwegiansAbroad - Sports - Swimming - Championships - Istanbul$$

This cluster would be deemed irrelevant because in the corpus there is a sixth article carrying the same tag. It might seem unreasonable to dismiss this cluster. Particularly since the algorithm is not likely to reproduce all ground truth clusters anyway, as discussed above.

Finally, it is a general problem that relevance varies with human perspective. The algorithm may discriminate too much or too little relative to some users' preference. For instance, there is a large ground truth cluster

$$NorwegiansAbroad - Sport - Handball - Championships - China$$

The algorithm does not produce it as a whole, but rather identify sub-clusters covering narrower topics such as individual players or matches. We can hardly claim that either way of clustering these particular articles is objectively correct.

These observations may raise questions about the suitability of precision in our context. Perhaps measuring up against some ground truth doesn't do full justice to the performance of the algorithm. If so, we could leave ground truth out of the picture altogether. A cluster has *tag accuracy* $5 - d$ if and only if d is the number of words common to all tags in it. That is, tag accuracy 0 means that all documents agree completely on the tags while 5 indicates poor accuracy. Note that this approach too has the problem of dismissal for even slight contamination as mentioned above.

The numbers obtained using tag accuracy will appear more favorable than the ones obtained with precision. Partly because our data are rich on singleton clusters, which necessarily will have perfect tag accuracies. We will not maintain that either of the two measures is generally more suitable than the other. Both can anyhow be used to register changes in the performance of the algorithm. We will consider tag accuracy as a supplementary measure.

6 Experiments

Experiments have been carried out to explore a number of modifications to the algorithm. Each falls within one of the the three dimensions outlined in section 5.

Suffixes vs $\lceil k/2 \rceil$-grams

Table 1 compares the performances of the suffix-tree and $\lceil k/2 \rceil$-gram varieties of the algorithm under Zamir and Etzioni's original setup [18]. The suffix-algorithm obtain somewhat better results in terms of precision and accuracy but the $\lceil k/2 \rceil$-gram algorithm works faster. One way of cashing in on this advantage would be to process more data to achieve better results. The question is, which results would actually benefit? As a rule of

	Precision		Tag accuracy	
Discrepancy	Suffix	$\lceil k/2 \rceil$-gram	Suffix	$\lceil k/2 \rceil$-gram
0	0.438	0.408	0.726	0.718
≤ 1	0.444	0.416	0.742	0.729
≤ 2	0.454	0.434	0.753	0.747

	Suffix	$\lceil k/2 \rceil$-gram
# clusters	372	380
Time	1.148s	1.073s

Table 1: $\lambda = 500$, $\theta_{min} = 3$, $\theta_{max} = 0.4$

thumb, when precision goes up, recall goes down and vice versa. We have already noted that recall suffers as a result of discarding base-clusters. Indeed, increasing the mass by merging more base-clusters seems to boost recall. Specifically:

- Increasing λ from 500 to 540 causes the n-gram algorithm to increase recall to the same level as the suffix-algorithm achieved for λ=500 (0.244), while still being faster (1.108 seconds).
- Increasing further to λ=1000 improves the recall (to 0.402) spending approximately the same time as the suffix-algorithm needed for λ=500 (1.140 seconds).

Alternatively, one could increase the amount of data by using more text for snippets.

Snippet-extraction

Table 2 shows the performances when snippets contain the entire text from the article. Compare with corresponding numbers in table 1, where snippets consists of the headline,

	Precision		Tag accuracy	
Discrepancy	Suffix	$\lceil k/2 \rceil$-gram	Suffix	$\lceil k/2 \rceil$-gram
0	0.384	0.341	0.571	0.502
≤ 1	0.407	0.363	0.598	0.530
≤ 2	0.417	0.391	0.609	0.557

	Suffix	$\lceil k/2 \rceil$-gram
# clusters	396	402
Time	12.644s	8.229s

Table 2: $\lambda = 500$, $\theta_{min} = 3$, $\theta_{max} = 0.4$, Full text snippets

ingress and caption. We observe that the n-gram algorithm is significantly faster but the performance in terms of precision actually deteriorates with full snippets. Then one might expect that recall would be boosted but, it is not. In fact, it drops a bit too, from 0.232 to 0.205. This is not so remarkable. Even if an ingress is brief it is carefully formulated to bring out the essence of the article, whereas in the full text there will inevitably be common words and phrases that may cause unrelated articles to gravitate towards each other when clusters are formed.

Zamir and Etzioni [18] conducts a similar experiment and their findings are consistent with ours. So is Eissen's remark that Suffix Tree Clustering is best suited for shorter

texts [5]. Since the focus has been very restricted, merely comparing small snippets with big snippets, there are no grand conclusions for us to make. If anything, these findings suggest that snippet-extraction is an intricate matter and perhaps a research question in its own right.

Tweaking and tinkering

Playing around with the different parameters and inner workings of the algorithm reveals a potential for improvement compared to the performance of Zamir and Etzioni's original setup shown in table 1. Let us consider the $\lceil k/2 \rceil$-gram algorithm and the parameters λ, θ_{min} and θ_{max}.

Table 3 shows an effect of tweaking the θ_{min} and θ_{max} parameters. Compared to the

Discrepancy	Precision	Tag accuracy
0	0.442	0.772
≤ 1	0.447	0.778
≤ 2	0.758	0.789

Table 3: Tweaked $\lceil k/2 \rceil$-gram: $\lambda = 500$, $\theta_{min} = 6$, $\theta_{max} = 0.5$

original setup, both precision and tag accuracy is improved. The run time (1.076 seconds) is better and even recall increases a bit, from 0.232 to 0.238.

Even if the use of a stop-list can help reduce the impact of some very common and irrelevant words we can not blacklist every common word there is. There will inevitably be clusters cemented by the co-occurrences of a single common word. Such clusters are often large and inaccurate but identifying them is no easy matter. We can surely use the cluster-label to find clusters formed on the presence of a single word, but a single word may well carry essential information about content. Hence, a one-word cluster is not necessarily a bad cluster. However, we might hope that the one-word clusters are more often based on common words than on essential ones. Then the net effect of removing them would be positive. Table 4 shows the performance when one-word clusters are excluded. Compared to table 3 we see a considerable gain in precision and tag accuracy.

Discrepancy	Precision	Tag accuracy
0	0.564	0.970
≤ 1	0.564	0.970
≤ 2	0.564	0.970

Table 4: $\lceil k/2 \rceil$-gram: $\lambda = 500$, $\theta_{min} = 6$, $\theta_{max} = 0.5$. No one-word clusters

Moreover, run-time performance does not seem to suffer: 1.065 seconds.

Finally, we have already remarked the possibility of boosting recall by increasing λ. Indeed, setting λ=5000 yields a recall of 0.757. The corresponding precision and tag accuracy is shown in table 5. The gain in recall is at the expense of run-time: 14.326 seconds. Clearly, the higher number of base-clusters floods the bottleneck of merging them.

7 Conclusion

We aimed to adapt Suffix-tree clustering for application to a corpus of news-articles. We have considered a number of modifications to the algorithm, with some success.

Discrepancy	Precision	Tag accuracy
0	0.507	0.883
≤ 1	0.523	0.903
≤ 2	0.535	0.916

Table 5: $\lceil k/2 \rceil$-gram: $\lambda = 5000$, $\theta_{min} = 6$, $\theta_{max} = 0.5$. No one-word clusters

For instance, comparing the performance of the original suffix-tree approach shown in table 1 with the modified $\lceil k/2 \rceil$-gram algorithm shown in table 4 we observe an increase in precision and tag accuracy of about 29% and 33%, respectively, while efficiency is approximately 8% better.

Because of the confinement to a particular data set we can not claim external validity for our results. Beyond the case at hand, our contribution is mainly to demonstrate a potential for improving the algorithm, but also to point out directions for further work. There are several issues that could be pursued.

First, we see that there are many factors at play. Our modification and tweaking is by no means an attempt to exhaust the possibilities. There are more limits and thresholds around than just λ, θ_{min} and θ_{max}. Furthermore, a richer use of n-grams is conceivable. For instance expansion to n-grams in the range $\lceil k/2 \rceil \pm \alpha$ where α is determined by a suitable function of k. Snippet-extraction certainly deserves more scrutiny. Our investigation has often taken an explorative approach. A more extensive and systematic investigation could aim to optimize a multiplicity of parameters found in the algorithm. This is a complex task but work in this direction is already in progress. Davøen [4] will apply genetic algorithms for multiple parameter optimization to a selection of parameters.

Secondly, the similarity measure is worth a closer look. It has thresholds that can be tweaked, but beyond that it would be interesting to explore more sophisticated measures exploiting cluster characteristics such as word frequencies and label overlaps.

Thirdly, unlike Zamir and Etzioni [18] we have not employed scoring for the final clusters but doing so would allow us to replicate their research design and carry out an investigation more comparable to theirs. Scoring would also be useful for trimming down a big collection of clusters in order to make it more manageable for a user. We believe scoring could make more sophisticated use of cluster characteristics, such as labels, size, word-frequencies etc, than the score used here for base-clusters.

Finally, we have seen that the potential for good recall values is severely hampered by the computational bottleneck of merging base-clusters. A research challenge lies in finding faster algorithms or alternatives ways of forming clusters from base-clusters.

Acknowledgements

Great contributions have been made by Gyri Losnegaard and Linn Lorgen in preparing the raw data for processing and classifying articles. The project is supported by Rådet for Anvendt Medieforskning.

References

[1] S. Allern. Nyhetsverdier: Om markedsorientering og journalistikk i ti norske aviser. IJ Forlaget 2001

[2] H. Chim, X. Deng. A New Suffix Tree Similarity Measure for Document Clustering. Proceedings of the International World Wide Web conference 2007.

[3] D. R. Cutting, D. R. Karger, J. O. Pedersen, J.W. Tukey. Scatter/Gather: a cluster-based approach to browsing large document collections. In Proceedings of the 15th International ACM SIGIR Conference on Research and Development in Information Retrieval, 1992

[4] S. M. Davøen. Compact Trie Clustering Parameter Optimization. Forthcoming 2013.

[5] S. M. zu Eissen, B. Stein, M. Potthast. The Suffix Tree Document Model Revisited. Proceedings of the 5th International Conferense on Knowledge Management, 2005.

[6] D. Elgesem, H. Moe, H. Sjøvaag, E. Stavelin. NRKs nyhetstilbud på Internett i 2009. Rapport til medietilsynet, Institutt for informasjons og medievitenskap UiB, 2010

[7] J. Erdal. Hvor kommer nyhetene fra? Nyhetsflyt mellom papiraviser, kringkasting og nettmedier. NOU2010:14, appendix 1. 2010

[8] Gulla, Borch, Ingvaldsen. Contextualized Clustering in Exploratory Web Search, 2007.

[9] D. R. Hill. A vector clustering technique. In Samuleson (ed) mechanised Information Storage, Retrieval and Dissemination. North-Holland, Amsterdam 1968

[10] G. Losnegaard. Automatic extraction of news text from online newspapers. Project report, Department of information science and media studies, UoB 2012.

[11] Norwegian Newspaper Corpus http://avis.uib.no

[12] Oslo-Bergen Tagger. http://tekstlab.uio.no/obt-ny/index.html

[13] E. Rasmussen. Clustering Algorithm. In W. B. Frakes and R. Baeza-Yates (eds). Information retrieval. Prentice-Hall 1992

[14] J. J. Rocchio. Document retrieval systems - optimization and evaluation. PhD Thesis, Harvard University 1966

[15] B. Smyth. Computing Patterns in Strings. Addison Wesley

[16] R. Waldahl, M. B. Andersen, H. Rønning. TV-nyhetenes verden. Universitetsforlaget 2009

[17] P. Willet. Recent trends in hierarchical document clustering: A critical review. Information Processing and Management, 1988

[18] O. Zamir, O. Etzioni. Web Document Clustering: A Feasibility Demonstration. SIGIR98 1998