Karlsruhe Institute of Technology

Institute of Theoretical Informatics
Algorithmics I

Seminar

Algorithmic Methods in the Humanities

Summer 2017

Algorithmics I
Institute of Theoretical Informatics
Department of Informatics
Karlsruhe Institute of Technology

Preface

Digital humanities is an area of research at the intersection of computing and the disciplines of the humanities, e.g., history, philosophy, linguistics, literature, art, archaeology, music, as well as cultural and social sciences. Multiple computer science fields have developed computational methods that are successfully used in the humanities sciences. The goal of the seminar *Algorithmic Methods in the Humanities* was to get a glimpse on the algorithms that lie at the core of these computational methods. In particular, the seminar considered the topics of Visualization in the Humanities, Analysis of Textual Variation, Topic Classification and Analysis of Debates. The seminar is part of the Master's program on Informatics at KIT. It is an interdisciplinary seminar supervised by Institute of Theoretical Informatics, Institute for Data Processing and Electronics and Institute of Philosophy. The aim of the seminar was to involve the students in the scientific research. They learned how to conduct literature research, identify, classify existing approaches and critically evaluate the literature. They practiced in devising a presentation in the context of a scientific topic in a way suitable for the audience. Finally they produced an extended written summary of the methods that they had identified; and tried to do it in a structured way, complying with the standards of scientific survey papers. This booklet is the collection of these summaries. The seminar took place in the summer semester 2017 and had 6 participants. The booklet contains the summaries of 4 participants.

1 Contents

2 Automatic Sentiment Summarization

Elisaweta Masserova

Abstract

Nowadays, online reviews are pervasive. Most internet users take a look at online reviews before they make a purchase decision. With the growth of the internet and, especially, e-commerce, the number of online reviews has grown immensively in the past years. On the one hand, this is a good thing – a high number of available subjective opinions allow a customer to make a confident and educated purchase decision. On the other hand, the number of reviews for some popular products or services can amount to hundreds or thousands, and such a large amount of data is difficult to process for a human. There can be multiple solutions for this problem, e.g., some services let users decide if a comment is useful or not, and show top comments first. In this work, we discuss the approach of automatically summarizing opinions, in particular the system introduced by S. Blair-Goldensohn et al. [1]. Our goal is to discuss the building blocks of the system itself and introduce available alternatives and possible applications of this system.

2.1 Introduction

A "summary" can be defined as a shortened version of one or more texts that contains a significant portion of the information from the initial text. In this work, we refer to a sentiment summarizer as being a system, that takes a set of reviews or comments as input and produces a summary. As the name suggests, such systems focus on extracting and aggregating the sentiments in the given data. A sample output of such a system is given in Figure 1. While multiple approaches to construct automatic sentiment summarizers exist, we focus on a system proposed by S. Blair-Goldensohn et al. [1] and discuss possible alternatives for parts of this system in process[1]. This system takes a set of reviews for some local service, e.g., a barber shop, as input, breaks them down into a set of text fragments, analyzes these fragments and produces a summary of the reviews as output. The key details and properties of this particular system are:

Utilization of an *aspect-based summarization model.* An aspect-based summarization system creates a summary separately for the important aspects of the service, see example summary for a greek restaurant in Figure 1.

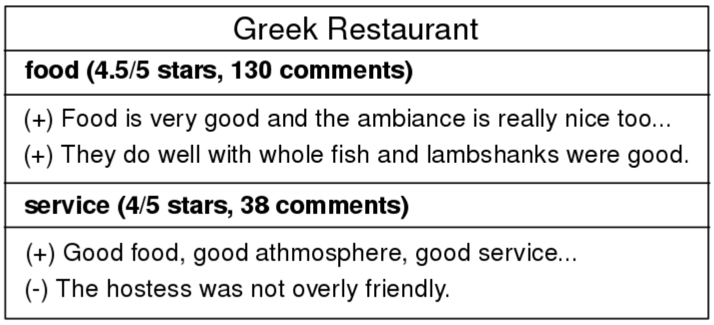
Exploitation of the *a priori knowledge* of the domain. The authors take advantage of the known information about the domain. For example, we might want to utilize the fact that some services, e.g., hotels and restaurants, are queried more often than others.

Consideration of *user provided labels.* In this system, the authors use not only the texts of the reviews, but also numerical labelling, e.g., ratings via zero to five stars, if available.

2.2 Preliminaries

In this report, the system takes advantage of some basic machine learning techniques. Machine learning is a study of algorithms that make predictions on data by inferring a general mapping

[1] Each time we mention "authors" or "system" without further specification, the system of S. Blair-Goldensohn et al. and its authors are meant



Greek Restaurant

food (4.5/5 stars, 130 comments)

(+) Food is very good and the ambiance is really nice too...
(+) They do well with whole fish and lambshanks were good.

service (4/5 stars, 38 comments)

(+) Good food, good athmosphere, good service...
(-) The hostess was not overly friendly.

Figure 1 Summarizer outputs for a greek restaurant.

function from sample inputs and, sometimes, outputs. In the following, we briefly introduce some typical machine learning tasks.

Classification is a problem of identifying to which category of a given set a new data point belongs, given a training set of data points with corresponding categories.

Regression is a problem similar to the classification task, however the output is not discrete, but continuous.

Clustering is a problem of dividing a set of inputs into groups, which are not known beforehand.

In the work of S. Blair-Goldensohn et al., the classification task is of particular interest. The authors classify sentences as being positive, negative or neutral. Also, sentences are classified as corresponding to a certain service aspect, such as price, quality, location.

In machine learning, tasks typically fall into one of the following three categories: *supervised*, *unsupervised* and *reinforcement* learning. Supervised learning is a task, where a number of sample inputs and corresponding outputs are given, and the goal is to find a general function that maps such inputs to corresponding outputs. In unsupervised learning, no outputs are given, and the goal is to find structure in the input. Reinforcement learning is a task where the machine learning system interacts with a dynamic environment to achieve some goal. In contrast to supervised learning, no input/ouput pairs are given, the learning process is reinforced through a reward and punishment system. In the work of S. Blair-Goldensohn et al., the authors take advantage of the supervised learning. The typical order of events of such system is presented in Figure 2.

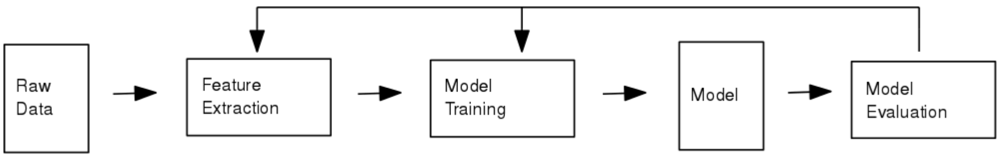


Figure 2 Typical order of events for a supervised learning approach.

Our task, given initial data, is to classify these data as corresponding to some category. Informally, we want to find some function/algorithm that maps given data to the expected output. This approximation function is called "model". At first, some rough approximation is assumed, then, in each iteration, parameters of the system are tweaked and changes in the results are evaluated.

The specific classifier suggested by S. Blair-Goldensohn et al. is a so-called maximum entropy classifier. Intuitively, a maximum entropy classifier is a classifier which prefers the uniform distribution if no additional data is available. Distribution constraints derived from the training data force the classifier to move from a uniform distribution to explain the data, however we still try to maintain the most uniform model possible. Such principle is natural for language processing, since it does not assume independence of the features. Take, for example, restaurant reviews. The word combination „chicken soup“ can occur frequently in such reviews and a maximum entropy classifier, in contrast to many other classifiers, does not assume that these two words are independent.

To evaluate machine learning algorithms, a number of standarad evaluation measures exist. The evaluation measures the authors used are *precision*, *recall* and *F1*.

Precision is defined as $\frac{|\{\text{relevant documents}\} \cap \{\text{retrieved documents}\}|}{|\{\text{retrieved documents}\}|}$. Intuitively it is the correct part of the system output, in other words, how *precise* the output is.

Recall is defined as $\frac{|\{\text{relevant documents}\} \cap \{\text{retrieved documents}\}|}{|\{\text{relevant documents}\}|}$. Intuitively it describes which part of the data we *wanted* to get we actually got.

F1 is defined as $2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}}$. It is the harmonic mean of precision and recall. The best value is one (perfect precision and recall) and worst is zero.

2.3 System overview

The system proposed by S. Blair-Goldensohn et al. consists of four main parts: *Text Extractor*, *Sentiment Classifier*, *Aspect Extractor*, and *Aggregator and Summarizer*. An overview of the system is shown in Figure 3. We briefly introduce the different parts of the system. Sentiment Classifier, Aspect Extractor and Aggregator and Summarizer are discussed in more detail in the next paragraph.

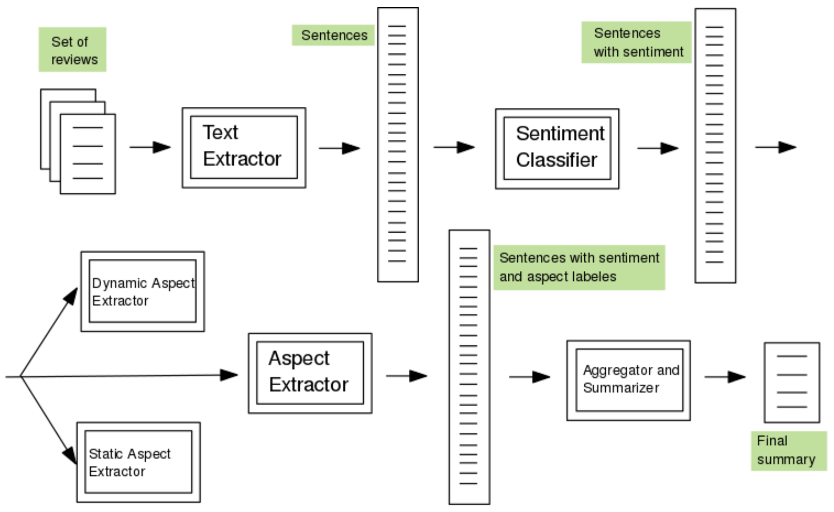


Figure 3 Overview of the automatic sentiment summarization system, as proposed by S. Blair-Goldensohn et al.

Given a set of reviews, the *text extractor* breaks them into a set of text fragments. While different granularity levels are possible, the system of S. Blair-Goldensohn et al. focuses on sentence-level fragments.

Given a set of sentences, the *sentiment classifier* checks whether these are positive, negative or neutral. Sentences that are loaded with sentiment, are passed further to the Aspect extractor.

Given a number of sentiment loaded sentences, the *aspect extractor* checks whether these correspond to a certain important service aspect. To achieve better results, the extractor consists of two parts. The first one is a dynamic aspect extractor, which takes only the given set of reviews into account. The second one is a static aspect extractor, which utilizes our knowledge about the most common services, e.g., restaurants. The output of the Aspect extractor are sentences labelled with corresponding aspects.

Given a number of sentiment loaded sentences marked with corresponding aspects, the *Aggregator and Summarizer* averages the sentiment over the aspects and produces a final summary, which can be presented to a user.

We now discuss the Sentiment Classifier, Aspect Extractor and Aggregator and Summarizer as they are introduced by S. Blair-Goldensohn et al., and provide ideas for alternatives and possible applications.

2.4 Sentiment Classification

Knowing "What other people think" has always played an important role in the human decision-making process. With the growth of the World Wide Web, increased usage of social networks, blogs, and review platforms, customer reviews became an crucial factor in the customers' purchase decision process. During the last two decades, researchers started to focus on these reviews to automatically categorize them as being positive, negative, and neutral. The aim of the sentiment analysis is exactly this determination of polarity of the data.

The system of S. Blair-Goldensohn et al. utilizes a hybrid sentiment classification system. It employs a lexicon model along with machine learning techniques. In the next section we label words, relevant for the local service, with the corresponding sentiment. Next, we use the constructed sentiment lexicon to classify the sentences or other text fragments as corresponding to some sentiment using machine learning techniques. The advantages of this approach are domain independence (due to the constructed lexicon), and the option of system parameter optimization with machine learning tools.

2.4.1 Lexicon Construction

First, our goal is to create a sentiment lexicon for further use. Informally, we want to identify sets of positive, negative and neutral words of importance to the current service. We will do so by creating a small set of hand-labelled positive, negative and neutral words and expanding this initial data through synonym and antonym connections. The obvious advantage of this approach, in contrast to constructing an entire lexicon manually, is the reduced number of man-hours needed for labelling words. The algorithm is based on a label propagation algorithm, the pseudocode can be seen in Algorithm 1.

Similarly to a standard label propagation algorithm, we start with a number of initial nodes, that is, words with initial labelling (positive, negative, neutral). We also use a given matrix A, which contains information about the synonym and antonym relations between words. In each step, the matrix A is multiplied with the vector s, which represents current sentiment scores of the words. A matrix-vector multiplication can be seen as propagation of

```
**Input :** Scaling factor $\lambda$,
sets of words $W$, positive words $P$, negative words $N$, neutral words $M$:
$P \cup N \cup M \subseteq W$, $P \cap N = P \cap M = N \cap M = \emptyset$,
number of steps $k$,
**Output :** Vector $s$ containing sentiment scores for each word of the set $W$.
Define word relation matrix $A$: $a_{ij} = \begin{cases} 1+\lambda, & \text{if } i = j \\ +\lambda, & \text{if } w_i \in syn(w_j), w_j \notin M \\ -\lambda, & \text{if } w_i \in ant(w_j), w_j \notin M \\ 0, & \text{otherwise} \end{cases}$
Initialize score vector $s^m = (s_0^m, s_1^m, \ldots, s_{|W|}^m)$: $s_i^0 = \begin{cases} +1, & \text{if } w_i \in P \\ -1, & \text{if } w_i \in N \\ 0, & \text{if } w_i \in W \setminus (P \cup N) \end{cases}$
**for** $i = 1$ **to** $k$ **do**
    $s^i = \textbf{sign-correct}(As^i)$
**end**
Scale $s^k$: $s_i = \begin{cases} \log(|s_i^k|) \cdot \text{sign}(s_i^k), & \text{if } s_i^k > 1 \\ 0, & \text{otherwise} \end{cases}$
```

Algorithm 1 : Sentiment Classifier - Lexicon Construction

some information (in our case, sentiment scores) through the network (in our case, network connections are represented by the synonym and antonym relations of the words). In the last step, the scores are logarithmically scaled. This is done to mitigate the impact of the extremely high scoring words on the final results. An example of the algorithm execution can be seen in Figure 4.

W = {Good, Smart, Dull, Tasty}
P = {Good}, $N = M = \emptyset$
Synonyms: Good and Smart,
Antonyms: Smart and Dull

$$A = \begin{array}{c} \\ Good \\ Smart \\ Dull \\ Tasty \end{array} \begin{array}{c} \begin{array}{cccc} G & S & D & T \end{array} \\ \begin{pmatrix} 1.2 & 0.2 & 0 & 0 \\ 0.2 & 1.2 & -0.2 & 0 \\ 0 & -0.2 & 1.2 & 0 \\ 0 & 0 & 0 & 1.2 \end{pmatrix} \end{array}, s^0 = \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \end{pmatrix}$$

$$s^0 = \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \end{pmatrix} \rightarrow s^1 = \begin{pmatrix} 1.2 \\ 0.2 \\ 0 \\ 0 \end{pmatrix} \rightarrow s^2 = \begin{pmatrix} 1.48 \\ 0.48 \\ -0.04 \\ 0 \end{pmatrix} \rightarrow s^3 = \begin{pmatrix} 1.872 \\ 0.88 \\ -0.144 \\ 0 \end{pmatrix}$$

Figure 4 Lexicon construction algorithm sample execution.

2.4.2 Classification

Next, the actual classification takes place. For this task, the authors take advantage of the previously constructed lexicon and use a maximum entropy classifier. The authors evaluate several systems, depending on the features of the data that were taken into account:

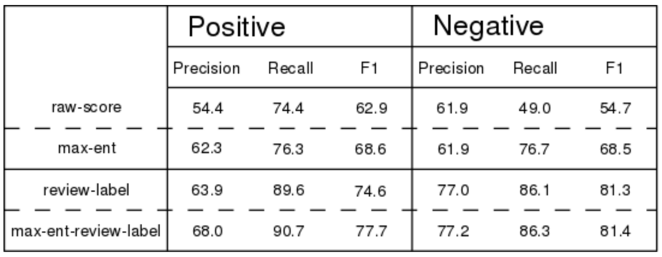
Review-Label system - this system assigns for each sentence in the review an overall score that the user provided for the service. This approach is obviously often inaccurate, since even the most positive reviews can contain negative sentences and vice versa.

Raw-Score system - this system assigns for each sentence in the review a so-called raw score. The raw score of the sentence is defined as $\sum_{i=1}^{n} s_i$, where s_i represents the sentiment score of the i-th word of the sentence. Informally, it is the aggregated sentiment of the words in the sentence. Afterwards, the system sorts sentences by their raw score.

Max-Ent system - this system trains a model based on the raw score and the purity features. The purity score of the sentence is defined as $\frac{\text{raw-score(x)}}{\sum_{i=1}^{n} |s_i|}$. The purity score correlates to the weighted fraction of words in the sentence that have the same sentiment as the entire sentence. For example, ”Outstanding food, bad service” and ”Average food, average service” can have the same raw score, however the first sentence achieves this score by having highly positive and highly negative words in it, and the second one achieves this score by having two average sentiments. The second sentence is thus more pure.

Max-Ent-Review-Label system - this system trains a model using the raw score, the purity features and the overall user-provided ranking for the review.

The authors compared the systems using precision, recall and F1 scores. Results can be seen in Figure 5. Upon these results, several observations can be made:



	Positive			Negative		
	Precision	Recall	F1	Precision	Recall	F1
raw-score	54.4	74.4	62.9	61.9	49.0	54.7
max-ent	62.3	76.3	68.6	61.9	76.7	68.5
review-label	63.9	89.6	74.6	77.0	86.1	81.3
max-ent-review-label	68.0	90.7	77.7	77.2	86.3	81.4

Figure 5 Sentiment Classification Precision, Recall and F1. Systems above the line do not use any user provided labelling information.

The system based on the raw score performs poorly when used alone, however the performance improves substantially when more features (purity, overall user-provided score) are added. The performance improvement when using user-provided overall rating is substantial - improvements of up to 10-15% were observed.

Based on these observations, the authors chose to go with the max-ent system, when no review scores are provided by the users, and the max-ent-review-label otherwise.

2.4.3 Alternative approaches

While the system proposed by the authors already reaches good results, we want to take a look at some alternatives for the sentiment classification. The sentiment classifcation approaches usually fall into one of the three categories: machine learning, lexicon based and hybrid.

Machine learning approaches include, besides the introduced maximum entropy classifier, Bayesian Networks, Naive Bayes Classification, Neural Networks and Support Vector Machines. In the work of T. Mullen and N. Collier [3], Support Vector Machines are used to conduct sentiment analysis for movie reviews. An interesting study has been done by Y. Yuan and Y. Zhou [5]. They used Recursive Neural Networks to analyse sentiment of a Twitter data set.

For the lexicon based approach, many resources, such as SentiWordNet, WordNet-Affect, MPQA and SenticNet are available. In [4], C. Musto et al. compared different resources for the lexicon-based sentiment analysis. They concluded that from the resources they compared, MPQA and SentiWordNet are the best ones.

Unfortunately, we do not know how the system of S. Blair-Goldensohn et al. performs, when combined with some alternative approaches. It would be also interesting to know whether some performance improvements can be achieved when using different sentiment lexicons, such as MPQA or SentiWordNet.

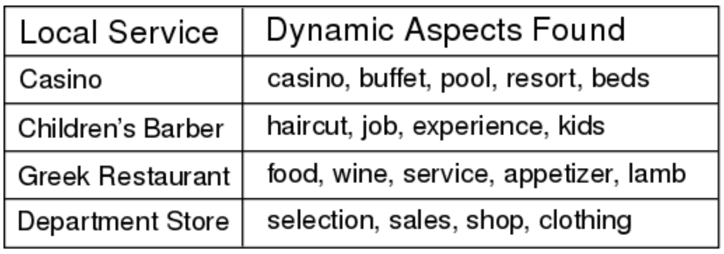
2.5 Aspect extraction

Hand in hand with the sentiment classification goes the aspect classification task. Here, given a set of sentences (in our case pre-labelled with the corresponding sentiment), the goal is to classify these sentences by correspondence to some aspect. Again, the authors employ a hybrid model. The first component is a dynamic aspect extractor, which uses the given set of reviews only. This approach gives us the flexibility of working with any service type. The second component is a static aspect extractor, which utilizes the fact that some services are queried for a lot more often than other ones. Searches for hotels and restaurants constitute a bulk of searches for local services. Thus, having highly accurate models for this service types will allow us to improve the performance of the whole system substantially.

2.5.1 Dynamic aspect extraction

We start with the dynamic aspect extraction. Given a set of reviews for the service, our goal is to identify aspects that are important for this particular service. Note that we do not want to generalize the results of the aspect extractor to all services - imagine a restaurant with a truly outstanding tomato soup. Many reviews for this restaurant will probably say something about this soup, and it certainly makes sense to identify ”tomato soup” as an important aspect for this service. However, it does not make sense to identify ”tomato soup” as an important aspect for the whole restaurant category.

The basic idea of S. Blair-Goldensohn et al. is that aspects are short strings that appear frequently in sentiment-loaded sentences. These strings are identified with the help of syntactic patterns, relative word frequencies and the previously computed sentiment lexicon. Afterwards, a number of filters are applied: strings consisting of the so-called stopwords, e.g., ”a”, ”of”, ”the”, are removed, as well as strings with low sentiment score. The entire algorithm in pseudocode can be seen in Algorithm 2. An example output of the dynamic aspect extraction can be seen in Figure 6.



Local Service	Dynamic Aspects Found
Casino	casino, buffet, pool, resort, beds
Children’s Barber	haircut, job, experience, kids
Greek Restaurant	food, wine, service, appetizer, lamb
Department Store	selection, sales, shop, clothing

Figure 6 Sample output of the dynamic aspect extractor.

```
Input : Sentences with given sentiment score
Output : List containing aspect candidates:
Step 1: Find short strings that occur often in:
  - Sentiment-loaded sentences
  - Certain syntactic patterns
Step 2: Remove candidates consisting of stopwords
Step 3: Remove candidates with low sentiment score
Step 4: Rank the resulting list by sentiment score
```

Algorithm 2 : Dynamic aspect extraction algorithm

2.5.2 Static aspect extraction

Dynamic aspect extractor results are fine-grained. This is often a good thing, however it also has some disadvantages:

Sometimes the results are so fine-grained that we get a ”laundry list” of aspects. For example, for a restaurant category we could identify each dish as a separate category. Thus, the problem of information overload would not be solved.

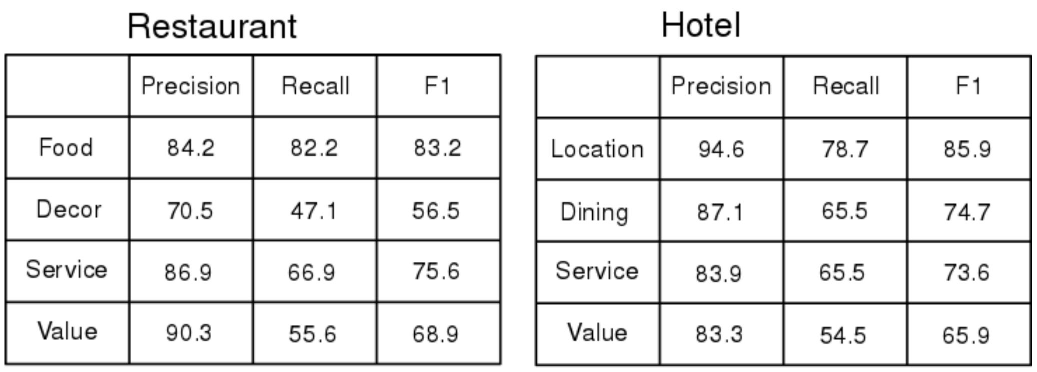
With such a fine-grained aspect list, loss of information may occur. For example, three reviews about restaurant food - one about the tomato soup, one about the salad and one about the dessert may be enough to rate the overall ”food” category, but they are probably not enough to rate each specific category separately. Thus, this information will be lost.

That is why we want to identify some coarse-grained aspects of the service. Multiple solutions exist for this problem: aspect clusters can be defined, a machine learning approach with hand-labelled data set is also possible. It would take a lot of time to hand-label training data for all possible service types. Thus, we take advantage of the fact that some service types, hotels and restaurants in particular, constitute a bulk of local services searches. Thus, by creating high-quality models for these service types, we can expect the performance of the whole system to be improved significantly.

The authors again utilize a maximum entropy classifier, this time in its binary version. For a given sentence, it identifies whether or not it corresponds to some given aspect. The aspects the authors are testing against are food, decor, service, and value for restaurants, and rooms, location, dining, service, and value for hotels. It took the authors less than two person days to hand-label the training data and the result of the system improved significantly. It is important to note that different service types often share some aspect categories. Restaurants and hotels both share the service and value categories. The authors expected improvements of the combination of the training data of both types - and the experimental results proved they were right. The overall evaluation results for the static aspect extractor can be seen in Figure 7.

2.5.3 Dynamic and static extractor combination

The authors propose to combine dynamic and static extractor results by checking the weighted sum of phrases and sentences corresponding to some aspect, and excluding those sentences where the sum does not reach some predefined threshold. Static aspect extractor results have priority over the dynamic aspect extractor results. The pseudocode of the entire algorithm is presented in Algorithm 3.



Restaurant

	Precision	Recall	F1
Food	84.2	82.2	83.2
Decor	70.5	47.1	56.5
Service	86.9	66.9	75.6
Value	90.3	55.6	68.9

Hotel

	Precision	Recall	F1
Location	94.6	78.7	85.9
Dining	87.1	65.5	74.7
Service	83.9	65.5	73.6
Value	83.3	54.5	65.9

Figure 7 Sample output of the static aspect extractor.

2.5.4 Alternative aspect extractor implementations

The aspect extraction approaches can be roughly divided into four categories: *Frequency-based*, *Relation-based*, *Supervised Learning* and *Topic Modelling*.

The frequency-based approach searches for frequent nouns and noun phrases in the review and identifies them as aspects. The dynamic aspect extractor of the system of S. Blair-Goldensohn et al. is an example of a frequency-based aspect extractor.

The relation-based approach involves identifying relationships between aspect and opinion words to find aspects.

Supervised learning methods used for the aspect extraction include Hidden Markov Models (HMM) and Conditional Random Fields (CRF).

The topic modelling approach is an unsupervised learning method for discovering topics in the given document. In our context, discovered topics would correspond to aspects. The most frequently used models are pLSA (Probabilistic Latent Semantic Analysis) and LDA (Latent Dirichlet Allocation).

In the work of P. More and A. Ghotkar [2], different aspect extraction approaches were studied. They concluded that each method has its strengths and weaknesses, namely:

Frequency-based methods, while simple and intuitive, often generate a lot of non-aspects, miss some important aspects and need manual tuning of parameters for each dataset.

Relation-based approach helps finding low frequency aspects, however may also produce a lot of non-aspects.

Supervised learning methods highly depend on the accuracy of the training data.

Topic modelling methods do not need manually labelled data, however they usually require a large amount of data.

In our opinion, the hybrid approach of S. Blair-Goldensohn et al. is the way to go. This method utilizes the strength of both the frequency-based and the supervised learning methods and mitigates their weaknesses at the same time.

2.6 Aggregator and Summarizer

The last part of the system proposed by S. Blair-Goldensohn et al. is the Aggregator and Summarizer. The goal is, given the results of sentiment and aspect classifiers, to provide an aspect-level summary, where each important service aspect is represented not only by numerical sentiment score, but also a textual evidence. For the first task, the scores of the

```
Input : List of static aspects $S$ and dynamic aspects $D$;
List $A$ such that $A_{asp}$ for some aspect $asp$ equals the weighted sum of phrases and
sentences from all reviews which has been classifed under aspect $asp$;
Manually tuned thresholds $\lambda$, $\beta$, $max\_size$;
Set $G$, containing all opinion mentions about any aspect (general comments).
Output : List $C$ of ranked aspects to show in the summary.
for $asp$ in $D$ do
    if $asp$ in $S$ or $A_{asp} \leq \lambda$ then
        $D.remove(\{asp\})$;
    end
end
for $asp$ in $S$ do
    if $A_{asp} \leq \lambda$ then
        $S.remove(\{asp\})$;
    end
end
sort $asp \in S$ by descending $A_{asp}$;
sort $asp \in D$ by descending $A_{asp}$;
$C := \emptyset$;
for $s$ in $S$ do
    $C.append(\{s\})$;
end
for $d$ in $D$ do
    if $A_{asp} \geq \beta$ and $|C| \leq max\_size$ then
        $C.append(\{d\})$;
    end
end
$C.append(G)$
```

Algorithm 3 : Dynamic and Static extractor - Result combination

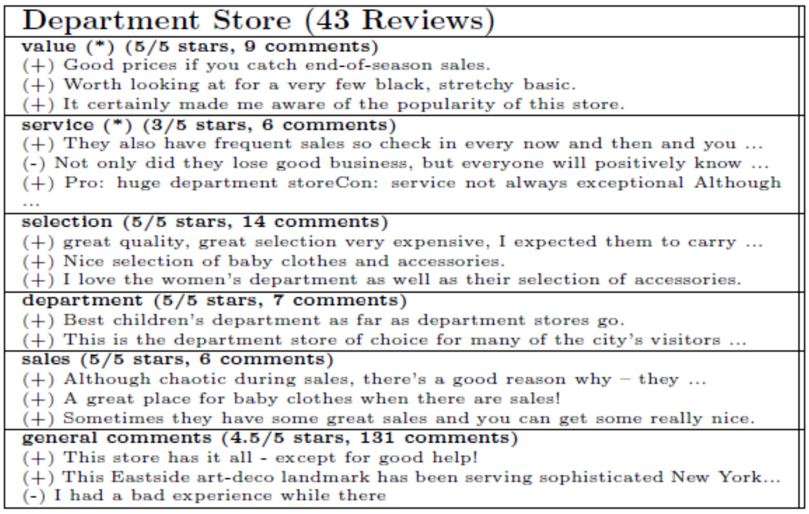
sentences of each identified aspect are aggregated and transformed into a ”star rating”. For the second task, a number of positive and negative sentences of some aspect are chosen such that the fraction of the positive sentences is approximately equal to the fraction $\frac{\text{overall aspect score}}{\text{maximum posssible score}}$.

2.7 System Evaluation

Example summaries of the system proposed by S. Blair-Goldensohn et al. are shown in Figure 1, Figure 8, Figure 9 and Figure 10. The input set used was the set of local service search results on Google Maps (http://maps.google.com). The number of reviews given for a service varies, however each example has a minimum of 16 input reviews.

The authors explain the ”Children’s barber shop” example in detail: the ”haircut” and ”general comments” aspects were identified by the dynamic extractor, ”service” and ”value” by the static extractor. This shows the importance of such a static extractor: without it, only two aspects would have been identified, and a lot of comments would either fall into the ”general comments” category, or be lost entirely (by falling into a category that does not have enough comments to get mentioned as a separate category in the summary).

Unfortunately, the authors do not provide a formal evaluation of the output summaries produced by the system. This is a difficult task since the usefulness and quality of such summary is highly subjective. However, the authors propose a possible approach to evaluate such summaries in future: to track users online and determine whether time savings are achieved when searching for a service.



Department Store (43 Reviews)

value (*) (5/5 stars, 9 comments)
(+) Good prices if you catch end-of-season sales.
(+) Worth looking at for a very few black, stretchy basic.
(+) It certainly made me aware of the popularity of this store.

service (*) (3/5 stars, 6 comments)
(+) They also have frequent sales so check in every now and then and you ...
(-) Not only did they lose good business, but everyone will positively know ...
(+) Pro: huge department storeCon: service not always exceptional Although ...

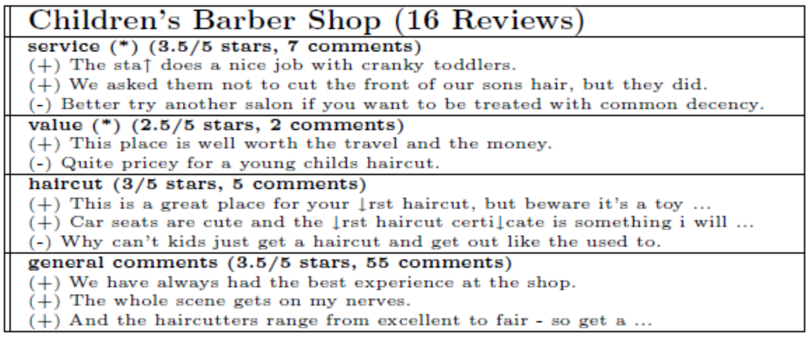
selection (5/5 stars, 14 comments)
(+) great quality, great selection very expensive, I expected them to carry ...
(+) Nice selection of baby clothes and accessories.
(+) I love the women's department as well as their selection of accessories.

department (5/5 stars, 7 comments)
(+) Best children's department as far as department stores go.
(+) This is the department store of choice for many of the city's visitors ...

sales (5/5 stars, 6 comments)
(+) Although chaotic during sales, there's a good reason why – they ...
(+) A great place for baby clothes when there are sales!
(+) Sometimes they have some great sales and you can get some really nice.

general comments (4.5/5 stars, 131 comments)
(+) This store has it all - except for good help!
(+) This Eastside art-deco landmark has been serving sophisticated New York...
(-) I had a bad experience while there

Figure 8 Summarizer outputs for a department store.



Children's Barber Shop (16 Reviews)

service (*) (3.5/5 stars, 7 comments)
(+) The staf does a nice job with cranky toddlers.
(+) We asked them not to cut the front of our sons hair, but they did.
(-) Better try another salon if you want to be treated with common decency.

value (*) (2.5/5 stars, 2 comments)
(+) This place is well worth the travel and the money.
(-) Quite pricey for a young childs haircut.

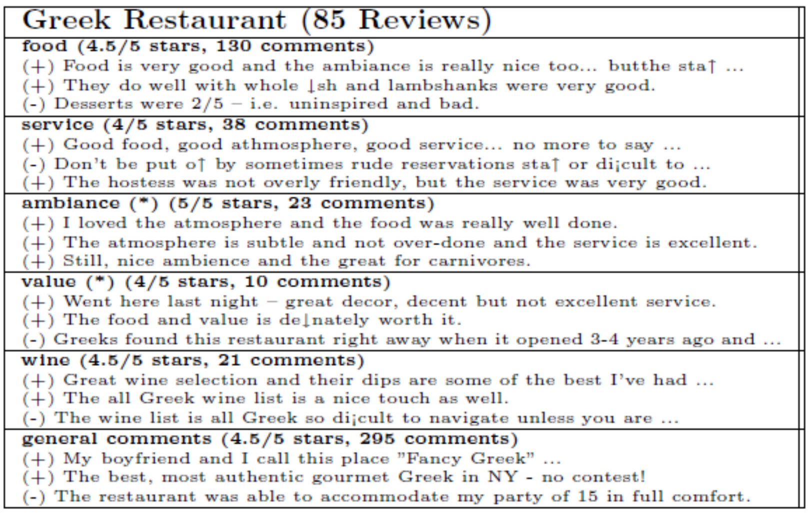
haircut (3/5 stars, 5 comments)
(+) This is a great place for your ⌊rst haircut, but beware it's a toy ...
(+) Car seats are cute and the ⌊rst haircut certi⌊cate is something i will ...
(-) Why can't kids just get a haircut and get out like the used to.

general comments (3.5/5 stars, 55 comments)
(+) We have always had the best experience at the shop.
(+) The whole scene gets on my nerves.
(+) And the haircutters range from excellent to fair - so get a ...

Figure 9 Summarizer outputs for a department store.

2.8 Further Application Ideas

The system of S. Blair-Goldensohn et al. is sufficiently general to produce quality summaries for all service types. In our opinion, with some adjustments to the initial lexicon of the sentiment classifier and an appropriate static aspect extractor, their system might be suitable for even more tasks. Some possible application examples include discussion analysis and media response analysis. For an analysis of a discussion, it would be nice to have a summary of the discussion with an overview of the important aspects and the sentiment of each involved party concerning these aspects. Media response analysis answers questions like ”How good is the public image of some company A?”, ”Did the image change after the last advertising


Greek Restaurant (85 Reviews)

food (4.5/5 stars, 130 comments)
(+) Food is very good and the ambiance is really nice too... butthe sta↑ ...
(+) They do well with whole ↓sh and lambshanks were very good.
(-) Desserts were 2/5 – i.e. uninspired and bad.

service (4/5 stars, 38 comments)
(+) Good food, good athmosphere, good service... no more to say ...
(-) Don't be put o↑ by sometimes rude reservations sta↑ or di¡cult to ...
(+) The hostess was not overly friendly, but the service was very good.

ambiance (*) (5/5 stars, 23 comments)
(+) I loved the atmosphere and the food was really well done.
(+) The atmosphere is subtle and not over-done and the service is excellent.
(+) Still, nice ambience and the great for carnivores.

value (*) (4/5 stars, 10 comments)
(+) Went here last night – great decor, decent but not excellent service.
(+) The food and value is de↓nately worth it.
(-) Greeks found this restaurant right away when it opened 3-4 years ago and ...

wine (4.5/5 stars, 21 comments)
(+) Great wine selection and their dips are some of the best I've had ...
(+) The all Greek wine list is a nice touch as well.
(-) The wine list is all Greek so di¡cult to navigate unless you are ...

general comments (4.5/5 stars, 295 comments)
(+) My boyfriend and I call this place "Fancy Greek" ...
(+) The best, most authentic gourmet Greek in NY - no contest!
(-) The restaurant was able to accommodate my party of 15 in full comfort.

Figure 10 Summarizer outputs for a greek restaurant.

campaign?". The (properly adjusted) system of S. Blair-Goldensohn et al. could give answers to these questions by analysing comments of internet users in the future.

2.9 Conclusion

In this work, we discussed the automatic sentiment summarizer, as proposed by S. Blair-Goldensohn et al. We explained their system and discussed some alternatives to the system parts they use, namely the sentiment classifier and the aspect extractor. While the system of S. Blair-Goldensohn et al. produces good results for the local service review summarization task, it would be interesting to know how their system can be improved by using a good sentiment lexicon or different aspect extraction techniques. Finally, we believe that with a different static aspect extractor and an appropriate initial lexicon set for the sentiment classifier, their system might be used for other tasks, such as discussion analysis or media response analysis.

References

1 Sasha Blair-Goldensohn, Kerry Hannan, Ryan McDonald, Tyler Neylon, George Reis, and Jeff Reynar. Building a sentiment summarizer for local service reviews. *WWW workshop on NLP in the information explosion era*, 14:339–348, 2008.

2 Pratima More and Archana Ghotkar. A study of different approaches to aspect-based opinion mining. *International Journal of Computer Applications*, 145, 2016.

3 Tony Mullen and Nigel Collier. Sentiment Analysis using Support Vector Machines with Diverse Information Sources. *Empirical Methods on Natural Language Processing*, 4:412–418, 2004.

4 Cataldo Musto, Giovanni Semeraro, and Marco Polignano. A comparison of lexicon-based approaches for sentiment analysis of microblog posts. *Information Filtering and Retrieval*, 59, 2014.

5 Ye Yuan and You Zhou. Twitter Sentiment Analysis with Recursive Neural Networks. *CS224D Course Projects*, 2015.

3 Measuring the Coherence of statements

Julia Kastner

Abstract

The first part of this report deals with probabilistic coherence measures and their application to a set of different examples, like in [3]. After that, we will give an example for the application of coherence measures, namely opinion maps and discuss some interesting features for the coherence function used in opinion maps.

3.1 Introduction

Suppose we have two (or more) sets of statements made by people and we want to find out which set is more coherent.

Many researchers have already looked into this and developed so-called *measures of coherence*. These measures take a set of propositions and assign it a real number representing the set's degree of coherence.

The authors of [3] evaluate different measures of coherence by applying them to two example cases. The first one is the already existing robber case by Schupbach, the second one is a slight modification of this case. Both are examples constructed in a way such that there are two sets of propositions one of which would intuitively be more coherent than the other.

The authors of [3] focus mainly on *probabilistic* coherence measures. These are measures that only take probabilistic information on the propositions into account. First, they introduce two naïve measures that are easy to understand, and later on they also test support based coherence measures.

This report is structured as follows: Simliar to [3], we will first look at overlap and deviation measures and apply them to Schupbach's robber case in section 3.2. We will also introduce our own modified version of Schupbach's robber case and apply the measures to this case. In section 3.3, we will look at a different form of coherence measures discussed in [3], namely mutual support measures. After investigating the example of an incoherent set of statements given in [3], we will again contribute a modified version of this example and look at the performance in this case.

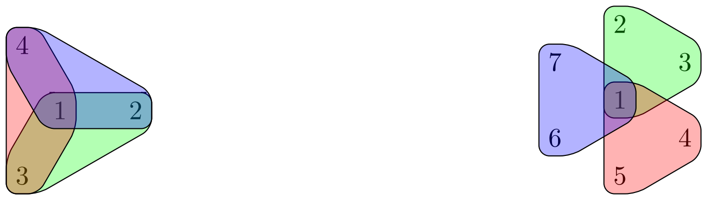
In section 3.5, we will discuss an application of coherence measures, namely opinion maps. After having given an overview over how Opinion Maps are constructed out of sets of opinions in section 3.5.1, we will take a closer look at desirable properties for coherence measures used for Opinion Maps in section 3.5.2.

Let us start by examining overlap and deviation measures in the following section.

3.2 Overlap and Deviation Measures

3.2.1 Naïve Measures

In this subsection we will look at the naïve measures of coherence introduced in [3] and apply them to Schupbach's robber case (example 1). We will then introduce our own modified version of the robber case in example 2 in order to further investigate the naïve overlap measure.



(a) The witnesses' statements in the case investigated by the first police department. **(b)** The witnesses' statements in the case investigated by the second police department.

Figure 1 Illustration of Schupbach's robber case.

The first two measures of coherence that are introduced in [3] are so-called naïve measures of coherence. They work by simply evaluating the probability of all the statements in a set being true at the same time and compare that to the probability of any statement being true or the product of the probabilities of all statements. The first naïve measure of coherence is the *naïve deviation measure.*

Let $S = \{A_1, \ldots A_n\}$ be a set of proposition. The naive deviation measure $\mathcal{D}$ is defined as:

$$\mathcal{D}(S) = \frac{P(A_1 \& \ldots \& A_n)}{P(A_1) \cdot \ldots \cdot P(A_n).}$$

This measure of coherence takes into account how the conjunction of all propositions in S relate to the product of their probabilities. For independent statements, the probability of the conjunction is the same as the product of probabilities, therefore $\mathcal{D}$ is 1 in this case.

Another naïve measure of coherence is the *naïve overlap measure* $\mathcal{O}$. For a set $S = \{A_1, \ldots, A_n\}$ of propositions it is defined as follows:

$$\mathcal{O}(S) = \frac{P(A_1 \& \ldots \& A_n)}{P(A_1 \vee \ldots \vee A_n)}$$

We now want to apply the naïve measures to Schupbach's robber case:

▶ **Example 1** (Schupbach's robber case)**.** In a bank robbery, there are eight suspects (numbered 1-8) and three witnesses. Each of the witnesses makes a statement to the police. In the following, we can see the witnesses' statements:

1. The first witness claims that the robber was either suspect 1, 2 or 3.
2. The second witness claims that the robber was either suspect 1,2 or 4.
3. The third witness claims that the robber was either suspect 1, 3 or 4.

Another police department in a different city is dealing with a similar case. There has also been a robbery and they have also been able to find eight suspects and three witnesses.

1. The first witness claims that the robber was either suspect 1,2 or 3.
2. The second witness claims that the robber was either suspect 1,4 or 5.
3. The third witness claims that the robber was either suspect 1,6 or 7.

What we can see in this example is that in both cases, the witnesses can agree that the first suspect might have been the robber. However, in the first case, each pair of two witnesses can also agree on another suspect whereas in the second case, this is not the case. For further illustration, see fig. 1. Intuitively, the first set of statements in example 1 is more coherent than the second.

For the naïve deviation measure $\mathcal{D}$ we have got the following (we will only write the number of the suspect instead of the statement „suspect x was the robber“):

$$\mathcal{D}(S_1) = \frac{P(1)}{P(1 \vee 2 \vee 3) \cdot P(1 \vee 2 \vee 4) \cdot P(1 \vee 3 \vee 4)}$$
$$\mathcal{D}(S_2) = \frac{P(1)}{P(1 \vee 2 \vee 3) \cdot P(1 \vee 4 \vee 5) \cdot P(1 \vee 6 \vee 7)}$$

We can see that the numerator is in both cases $P(1)$ but the denominator is different. In the case where all of the suspects are equally likely to have committed the crime, this means that $\mathcal{D}(S_1) = \mathcal{D}(S_2)$. Therefore, the naïve deviation measure does not follow the intuition that the first set of witness's statements is more coherent than the other.

In the second example, the calculations are as follows:

$$\mathcal{O}(S_1) = \frac{P(1)}{P(1 \vee 2 \vee 3 \vee 4)}$$
$$\mathcal{O}(S_2) = \frac{P(1)}{P(1 \vee 2 \vee 3 \vee 4 \vee 5 \vee 6 \vee 7)}$$

In general, the probability $P(1 \vee 2 \vee 3 \vee 4 \vee 5 \vee 6 \vee 7)$ is greater or equal than the probability $P(1 \vee 2 \vee 3 \vee 4)$ simply because the second set is a subset of the first. Therefore $\mathcal{O}(S_1) \geq \mathcal{O}(S_2)$ which fits our intuition that the first set is more coherent than the second one.

We can see that for this example, at least the overlap measure works, however, it does so due to the higher numbers of overall possible suspects in the second case. We therefore want to introduce our own version of the robber case where the overall number of suspects mentioned in the first police department is the same as in in the second. The property that the witnesses in the first police department make more coherent statements (each pair of witnesses can agree on two suspects) stays.

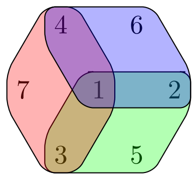
▶ **Example 2** (Robber case with equal numbers of suspects). Same as before, the police department is looking for a robber and has found eight suspects and three witnesses. These are the statements of the witnesses:
1. The robber was either suspect 1, 2, 3 or 5.
2. The robber was either suspect 1, 2, 4 or 6.
3. The robber was either suspect 1, 3, 4 or 7.

In the second part of the example, the witnesses' statements stay the same.
1. The robber was either suspect 1,2 or 3.
2. The robber was either suspect 1,4 or 5.
3. The robber was either suspect 1,6 or 7.

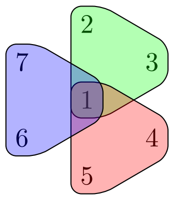
An illustration for the robber case with equal numbers of suspects can be seen in example 2. In the following we will look at how the naïve overlap measure performs in this new version of the robber case for equal probabilities:

$$\mathcal{O}(S_1) = \frac{P(1)}{P(1 \vee 2 \vee 3 \vee 4 \vee 5 \vee 6 \vee 7)} = \frac{\frac{1}{8}}{\frac{7}{8}} = \frac{1}{7}$$
$$\mathcal{O}(S_2) = \frac{P(1)}{P(1 \vee 2 \vee 3 \vee 4 \vee 5 \vee 6 \vee 7)} = \frac{\frac{1}{8}}{\frac{7}{8}} = \frac{1}{7}$$

As the results are the same for both cases, the naïve overlap measure does not comply with the intuition that the first police department has found witnesses who made more coherent statements than the second.



(a) The witnesses' statements in the case investigated by the first police department in example 2.



(b) The witnesses' statements in the case investigated by the second police department from example 2.

Figure 2 Illustration of the robber case where both sets of witnesses name the same number of suspects in total.

This means that there is room for improvement for both of the naïve measures. In the following, refined versions of these two naïve measures will be discussed and applied to the robber case.

3.2.2 Refined measures

One problem with the naïve measures of coherence introduced earlier is that they do not take into account whether or not subsets of a set of propositions are coherent. A suggestion for fixing this is measuring the coherence of all subsets of a given set of propositions. For any set of statemenst S, the refined measures are defined as follows:

$$\mathcal{D}^*(S) = \frac{1}{2^{|S|} - |S| - 1} \sum_{\substack{S' \subset S \\ |S'| \geq 2}} \log_{10} \mathcal{D}(S')$$

$$\mathcal{O}^*(S) = \frac{1}{2^{|S|} - |S| - 1} \sum_{\substack{S' \subset S \\ |S'| \geq 2}} \mathcal{O}(S')$$

In the first case, the logarithm is used in order to prevent small denominators (which means large values in total) from pulling the average towards ∞. For $\mathcal{O}$ this is not necessary since the denominator is only one probability instead of a product of probabilities.

We can now look into how these measures work with the robber case examples.

The refined deviation measures $\mathcal{D}^*$ is calculated as follows.

$$\begin{aligned}\mathcal{D}^*(S_1) = \ & \frac{1}{4}\Bigg(\log_{10}\frac{P(1)}{P(1 \vee 2 \vee 3) \cdot P(1 \vee 2 \vee 4) \cdot P(1 \vee 3 \vee 4)} + \log_{10}\frac{P(1 \vee 2)}{P(1 \vee 2 \vee 3) \cdot P(1 \vee 2 \vee 4)} \\ & + \log_{10}\frac{P(1 \vee 3)}{P(1 \vee 3 \vee 4) \cdot P(1 \vee 2 \vee 4)} + \log_{10}\frac{P(1 \vee 4)}{P(1 \vee 2 \vee 4) \cdot P(1 \vee 3 \vee 4)}\Bigg) \\ \mathcal{D}^*(S_2) = \ & \frac{1}{4}\Bigg(\log_{10}\frac{P(1)}{P(1 \vee 2 \vee 3) \cdot P(1 \vee 4 \vee 5) \cdot P(1 \vee 6 \vee 7)} + \log_{10}\frac{P(1)}{P(1 \vee 2 \vee 3) \cdot P(1 \vee 4 \vee 5)} \\ & + \log_{10}\frac{P(1)}{P(1 \vee 4 \vee 5) \cdot P(1 \vee 6 \vee 7)} + \log_{10}\frac{P(1)}{P(1 \vee 2 \vee 3) \cdot P(1 \vee 6 \vee 7)}\Bigg)\end{aligned}$$

We can se that the fact that any pair of two witnesses can name an additional suspect is taken into account here, as the probability of the disjunction is used as the numerator of the fraction for the first set of statements.

If all the suspects are deemed equally likely to have committed the crime, we can fill in the probabilities as the following[1]:

$$\begin{aligned}
\mathcal{D}^*(S_1) &= \frac{1}{4}\left(\log_{10}\frac{\frac{1}{8}}{\frac{3}{8}\cdot\frac{3}{8}\cdot\frac{3}{8}} + \log_{10}\frac{\frac{2}{8}}{\frac{3}{8}\cdot\frac{3}{8}} + \log_{10}\frac{\frac{2}{8}}{\frac{3}{8}\cdot\frac{3}{8}} + \log_{10}\frac{\frac{2}{8}}{\frac{3}{8}\cdot\frac{3}{8}}\right)\\
&= \frac{1}{4}\left(\log_{10}\frac{64}{27} + \log_{10}\frac{16}{9} + \log_{10}\frac{16}{9} + \log_{10}\frac{16}{9}\right)\\
&= 0.281\\
\mathcal{D}^*(S_2) &= \frac{1}{4}\left(\log_{10}\frac{\frac{1}{8}}{\frac{3}{8}\cdot\frac{3}{8}\cdot\frac{3}{8}} + \log_{10}\frac{\frac{1}{8}}{\frac{3}{8}\cdot\frac{3}{8}} + \log_{10}\frac{\frac{1}{8}}{\frac{3}{8}\cdot\frac{3}{8}} + \log_{10}\frac{\frac{1}{8}}{\frac{3}{8}\cdot\frac{3}{8}}\right)\\
&= \frac{1}{4}\cdot\left(\log_{10}\frac{64}{27} + \log_{10}\frac{8}{9} + \log_{10}\frac{8}{9} + \log_{10}\frac{8}{9}\right)\\
&= 0.055
\end{aligned}$$

If we assume that each suspect is equally likely to have committed the crime, we get a new result for $\mathcal{D}^*$, namely that it is actually greater for S_1 and therefore correctly represents our intuition of which set of statements is the most coherent. We can also see that the refined deviation measure takes into account that in the first case, each pair of two witnesses can agree on two suspects instead of one, as the refined deviation measure also looks at subsets of the complete set of statements.

The refined overlap measure for the original robber case can be seen in the following:

$$\begin{aligned}
\mathcal{O}^*(S_1) = \quad & \frac{1}{4}\left(\frac{P(1)}{P(1\vee 2\vee 3\vee 4)} + \frac{P(1\vee 2)}{P(1\vee 2\vee 3\vee 4)}\right.\\
&\left. + \frac{P(1\vee 3)}{P(1\vee 2\vee 3\vee 4)} + \frac{P(1\vee 4)}{P(1\vee 2\vee 3\vee 4)}\right)\\
\mathcal{O}^*(S_2) = \quad & \frac{1}{4}\left(\frac{P(1)}{P(1\vee 2\vee 3\vee 4\vee 5\vee 6\vee 7)} + \frac{P(1)}{P(1\vee 2\vee 3\vee 4\vee 5)}\right.\\
&\left. + \frac{P(1)}{P(1\vee 2\vee 3\vee 6\vee 7)} + \frac{P(1)}{P(1\vee 4\vee 5\vee 6\vee 7)}\right)
\end{aligned}$$

We can see that it is also taken into account here that all subsets of two can agree on an additional suspect in the first case as the numerator of the summands representing sets of two contains the probability of the disjunction of the two suspects.

When all suspects are equally likely to have committed the crime, the refined overlap measure can be calculated as follows:

$$\begin{aligned}
\mathcal{O}^*(S_1) &= \frac{1}{4}\left(\frac{\frac{1}{8}}{\frac{4}{8}} + \frac{\frac{2}{8}}{\frac{4}{8}} + + \frac{\frac{2}{8}}{\frac{4}{8}} + \frac{\frac{2}{8}}{\frac{4}{8}}\right) = \frac{1}{4}\left(\frac{1}{4} + \frac{2}{4} + \frac{2}{4} + \frac{2}{4}\right) = \frac{7}{16} = 0.438\\
\mathcal{O}^*(S_2) &= \frac{1}{4}\left(\frac{\frac{1}{8}}{\frac{7}{8}} + \frac{\frac{1}{8}}{\frac{5}{8}} + \frac{\frac{1}{8}}{\frac{5}{8}} + \frac{\frac{1}{8}}{\frac{5}{8}}\right) = \frac{1}{4}\left(\frac{1}{7} + \frac{1}{5} + \frac{1}{5} + \frac{1}{5}\right) = \frac{13}{70} = 0.186
\end{aligned}$$

This means that the refined overlap measure follows the intuition that the witnesses' statements in the first case are more coherent than in the second case.

It is now interesting whether the refined overlap measure has the same problem as the naïve overlap measure when it comes to the number of suspects suspected by the witnesses.

[1] In the appendix of [3] $\mathcal{D}^*$ is 0.312 for the first case and 0.162 in the second case. The authors have been contacted.

If we apply the refined overlap measure to example 2, we get the following results:

$$\begin{aligned}\mathcal{O}(S_1) =& \frac{1}{4}\left(\frac{P(1)}{P(1\vee 2\vee 3\vee 4\vee 5\vee 6\vee 7)} + \frac{P(1\vee 2)}{P(1\vee 2\vee 3\vee 4\vee 5\vee 6)}\right.\\ &\left.+\frac{P(1\vee 3)}{P(1\vee 2\vee 3\vee 4\vee 5\vee 7)} + \frac{P(1\vee 4)}{P(1\vee 2\vee 3\vee 4\vee 6\vee 7)}\right)\\ =&\frac{1}{4}\left(\frac{\frac{1}{8}}{\frac{7}{8}} + \frac{\frac{2}{8}}{\frac{6}{8}} + \frac{\frac{2}{8}}{\frac{6}{8}} + \frac{\frac{2}{8}}{\frac{6}{8}}\right) = \frac{1}{4}\left(\frac{1}{7}+\frac{1}{3}+\frac{1}{3}+\frac{1}{3}\right) = \frac{2}{7} = 0.286\\ \mathcal{O}^*(S_2) =& \frac{1}{4}\left(\frac{P(1)}{P(1\vee 2\vee 3\vee 4\vee 5\vee 6\vee 7)} + \frac{P(1)}{P(1\vee 2\vee 3\vee 4\vee 5)}\right.\\ &\left.+\frac{P(1)}{P(1\vee 2\vee 3\vee 6\vee 7)} + \frac{P(1)}{P(1\vee 3\vee 4\vee 5\vee 6\vee 7)}\right)\\ =&\frac{1}{4}\left(\frac{\frac{1}{8}}{\frac{7}{8}} + \frac{\frac{1}{8}}{\frac{5}{8}} + \frac{\frac{1}{8}}{\frac{5}{8}} + \frac{\frac{1}{8}}{\frac{5}{8}}\right) = \frac{1}{4}\left(\frac{1}{7}+\frac{1}{5}+\frac{1}{5}+\frac{1}{5}\right) = \frac{13}{70} = 0.186\end{aligned}$$

We can see that the refined overlap measure also works for the robber case with equal numbers of suspects suspected by all witnesses together.

The naïve and refined measures are not the only measures of coherence discussed in [3]. They also evaluate support based coherence measures. These will be discussed in the following.

3.3 Mutual Support Measures

For mutual support measures, there is a different approach. The general idea of these types of measures is to look at disjoint subsets of a set of propositions and measure to what extent each subset supports the other one. The basic formula for these measures is therefore:

$$C_{\mathfrak{s}}(S) = \sum_{\substack{S',S''\neq\emptyset\\ S',S''\subset S\\ S'\cap S''=\emptyset}} \frac{\mathfrak{s}(\wedge S', \wedge S'')}{l}$$

where $l = \sum_{i=1}^{n-1}\binom{n}{i}(2^{n-1}-1) = (2^n-2)\cdot(2^{n-1}-1)$ which is the number of pairs (S', S'') summed over.

What we can already see without even knowing what $\mathfrak{s}$ looks like, is that similar to the refined measures, mutual support measures take the coherence of subsets into account by looking at how these subsets support each other. This also means that support within a subset is regarded if the subset has at least two elements and can therefore be separated into two smaller subsets.

We will now look into what these support measures $\mathfrak{s}$ could look like.

The first possibility is the prior-posterior difference. It is defined as follows:

$$d(B, A) = P(B|A) - P(B)$$

This support measure takes the two sets of propositions A and B and looks at the difference between the conditional probability for B given A and the probability for B alone. Note that this is 0 when A and B are independent. It is also easy to see that this value ranges between -1 and 1 because probabilities range from 0 to 1. As C_d is basically the average over all d, C_d also ranges from -1 to 1.

The next type of support measure has a similar idea. Except in this case, we don't take the difference between two probabilities but the ratio instead. This measure is called the *prior-posterior ratio* and is defined as follows:

$$r(B, A) = \frac{P(B|A)}{P(B)}$$

This measure ranges from $-\infty$ to $+\infty$ and is 1 for independent sets.

The *counterfactual difference* is based on the idea of looking at the conditional probability of B given A versus B given $\neg A$. It is defined as follows:

$$s(B, A) = P(B|A) - P(B|\neg A)$$

Again, since probabilities range from 0 to 1, the counterfactual difference s ranges from -1 to 1. Therefore C_s also ranges from -1 to 1. For independent sets A and B, the probabilities for B given A and B given $\neg A$ are the same and therefore the counterfactual difference is 0.

Another counterfactual measure is the *counterfactual ratio*. It is defined as follows:

$$j(B, A) = \frac{P(B|A)}{P(B|\neg A)}$$

The counterfactual ratio ranges from $-\infty$ to $+\infty$. Therefore D_j also ranges from $-\infty$ to $+\infty$.

Other support measures evaluated by the authors of [3] are relative distance, factual support and firmness. We will omit those for brevity.

What all of the support measures evaluated in [3] have in common is that they need $P(B|A)$ which is defined as follows:

$$P(B|A) = \frac{P(B\&A)}{P(A)}$$

This can be a problem when we want to measure the coherence of inconsistent sets. For an inconsistent set A, $P(A)$ is 0 and therefore $P(B|A)$ is not defined. However, it can be interesting to evaluate the coherence of inconsistent sets. We will take a closer look at those in the next section

3.4 Coherence of inconsistent sets

After having seen the coherence of consistent sets, we now want to examine how we can measure the coherence of inconsistent sets. In order to do that, we will first look at some examples of inconsistent sets. After that, we will discuss the suggestions for improving coherence measures that are made in [3].

3.4.1 Examples of inconsistent sets

In the following, we want to look at examples of inconsistent sets of different coherence. Example 3 is the example that [3] gives for two inconsistent sets with different coherence. This example again has two sets of witness statements. As the numbers of total named suspects are again different for the two sets, we will again construct our own example based on the modified robber case, in order to examine the influence of the number of named suspects.

The authors of [3] give the following example for inconsistent sets:

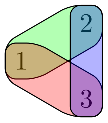
▶ **Example 3.** Just like in example 1, we have eight suspects and three witnesses. In the first case, the three witnesses make the following statements to the police:

1. The robber was either suspect 1 or 2.
2. The robber was either suspect 2 or 3.
3. The robber was either suspect 3 or 1.

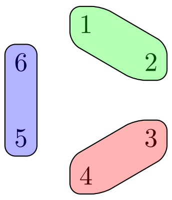
Again, there is another police department investigating a similar case with also eight suspects and three witnesses who make the following statements:

1. The robber was either suspect 1 or 2.
2. The robber was either suspect 3 or 4.
3. The robber was either suspect 5 or 6.

In this example, the witnesses claims are not consistent. However, we can see that in the first case, each pair of two witnesses is consistent whereas this is not the case in the second case. This is illustrated in fig. 3 Therefore, it is interesting to see how this reflects in the coherence measures. The authors then apply all of the coherence measures discussed so far to



(a) The witnesses' statements in the case investigated by the first police department in example 3.



(b) The witnesses' statements in the case investigated by the second police department from example 3.

Figure 3 Illustration of the modified robber case by [3].

the modified robber case shown in example 3. Table 1 shows the results of [3] for example 1

Table 1 Performance of measures of coherence in the robber case and modified robber case when all suspects are equally likely to have committed the crime. A ✓ indicates that the measure reflects the intuition that the first set of claims is *more* coherent than the second. A × indicates that the measure does not comply with this intuition (i.e. the measure gives the same result for both sets or the first set has a lower coherence than the second). `NaN` means that there is a division by 0.

Measure	Robber case (see example 1)	Modified robber case (see example 3)
$\mathcal{D}$	×	×
$\mathcal{O}$	✓	×
$\mathcal{D}^*$	✓	×
$\mathcal{O}^*$	✓	✓
C_d	✓	`NaN`
C_r	×	`NaN`
C_s	✓	`NaN`
C_j	×	`NaN`

and example 3.

For $\mathcal{D}^*$, the problem is that the probability for an inconsistent set is zero, therefore $\mathcal{D}$ of that set is zero and the logarithm is $-\infty$. The value for the whole set therefore makes the average become $-\infty$ for both cases.

The support based measures have the problem that the probability for inconsistent sets is zero, therefore they are undefined for the case where all subsets with at least two elements are inconsistent. The sets as a whole being inconsistent is not a problem because the two subsets have to be disjoint and not empty. Therefore the summation is only over all sets of size one and two.

The authors of [3] suggest some solutions for this problem. These solutions will be discussed in the following section.

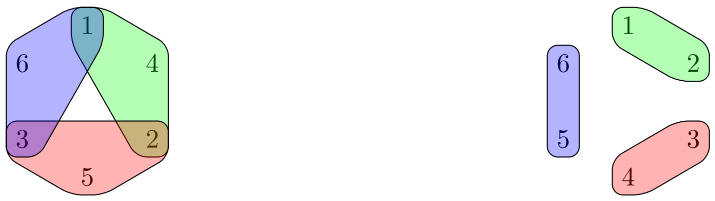
But first, we want to look at our own version of the modified robber case where the overall number of suspects named by the witnesses the same for the first and second case. An illustration of this version of the modified robber case can be seen in fig. 4.

▶ **Example 4.** Similar to example 3, there is a robbery with eight suspects and three witnesses. The witnesses make the following statements:

1. The robber was either suspect 1, 2, or 4.
2. The robber was either suspect 2, 3, or 5.
3. The robber was either suspect 1, 3, or 6.

A second police department investigates a similar case where the witnesses make the following statements:

1. The robber was either suspect 1 or 2.
2. The robber was either suspect 3 or 4.
3. The robber was either suspect 5 or 6.



(a) The witnesses' statements in the case investigated by the first police department in example 4.

(b) The witnesses' statements in the case investigated by the second police department from example 4.

Figure 4 Illustration of our own robber case example based on the modified robber case.

The calculation for $\mathcal{O}^*$ in this case goes as follows if the all the suspects are equally likely to have committed the crime.

$$\begin{aligned}\mathcal{O}^*(S_1) =& \frac{1}{4}\left(\frac{P(\emptyset)}{P(1 \vee 2 \vee 3 \vee 4 \vee 5 \vee 6)} + \frac{P(2)}{P(1 \vee 2 \vee 3 \vee 4 \vee 5)} + \frac{P(1)}{P(1 \vee 2 \vee 3 \vee 4 \vee 6)}\right.\\ &\left. + \frac{P(3)}{P(1 \vee 2 \vee 3 \vee 5 \vee 6)}\right)\\ =& \frac{1}{4}\left(\frac{0}{\frac{6}{8}} + \frac{\frac{1}{8}}{\frac{5}{8}} + \frac{\frac{1}{8}}{\frac{5}{8}} + \frac{\frac{1}{8}}{\frac{5}{8}}\right) = \frac{1}{4}\left(\frac{1}{5} + \frac{1}{5} + \frac{1}{5}\right) = \frac{3}{20} = 0.15\\ \mathcal{O}^*(S_2) =& \frac{1}{4}\left(\frac{P(\emptyset)}{P(1 \vee 2 \vee 3 \vee 4 \vee 5 \vee 6)} + \frac{P(\emptyset)}{P(1 \vee 2 \vee 3 \vee 4)} + \frac{P(\emptyset)}{P(1 \vee 2 \vee 5 \vee 6)}\right.\\ &\left. + \frac{P(\emptyset)}{P(3 \vee 4 \vee 5 \vee 6)}\right) = 0\end{aligned}$$

We can see that, again, the set that is intuitively more coherent, also has the higher value for $\mathcal{O}^*$.

In the following, we will discuss the improvements suggested in [3]. We will start with improvements for $\mathcal{D}^*$.

3.4.2 Improving $\mathcal{D}^*$

The authors of [3] first inspect the measure $\mathcal{D}^*$. They point out that due to the fact that this method uses the logarithm, which is $-\infty$ for 0 which is $\mathcal{D}$ of an inconsistent set. Therefore, $\mathcal{D}^*$ can be tipped off towards $-\infty$ for any inconsistent set. The authors of [3] then first make clear that not using the logarithm brings other problems which is why the logarithm was used in the first place. Therefore they suggest the following adaptation of $\mathcal{D}$:

$$\mathcal{D}_{\#}(S) = 1 - 2^{\mathcal{D}(S)}$$

Note that this ranges from 0 to 1 where 0.5 indicates neutrality. They then define a refined measure based on $\mathcal{D}_{\#}$ as follows:

$$\mathcal{D}^*_{\#} = \frac{1}{2^{|S|} - |S| - 1} \sum_{\substack{S' \subset S \\ |S'| \geq 2}} \mathcal{D}_{\#}(S')$$

The authors then show that this measure works for inconsistent sets as well because now the minimal value assigned to an inconsistent set is 0 instead of $-\infty$. Therefore, the average of all subsets is not pulled down by the inconsistent subsets. In the following, we will discuss possible improvements for support based measures.

3.4.3 Improving support based measures

As already noticed, the support based measures have the problem that due to the probability of an inconsistent set being 0, there are divisions by zero which make the measure undefined for inconsistent sets. The authors therefore make suggestions what value to assign a for a support measure when it is undefined. The suggestions are the following:

- assign the minimal support value
- assign the neutral support value
- assign minimal support value for inconsistent hypotheses and neutral value for inconsistent evidence
- only include defined summands in calculation

Assigning the minimal degree of confirmation is suggested because inconsistencies usually have a negative impact on the overall measure.

However, the authors also argue that inconsistent evidence neither proves nor disproves a hypotheses and therefore they suggest assigning a neutral value instead. They had also considered assigning the maximal degree of confirmation but then abandoned this idea due to the fact that inconsistencies should not boost the degree of coherence of a set.

The authors then also suggest a mix of the first and second measure in which a neutral value is assigned for consistent hypotheses with inconsistent evidence and the minimal value is assigned for inconsistent hypotheses.

The last suggestion is to only include defined values in the calculation of the average value.

The results from these adaptations can be seen in table 2.

After having seen some coherence measures and their performance in certain examples, we want to look at an application of coherence measures, namely opinion maps.

Table 2 Performance of support based measures in the modified robber case with different kinds of adaptions.

Measure	minimal value	neutral value	mixed	ignore undefined values
C_d	✓	✓	✓	✓
C_r	✓	✓	✓	✓
C_s	✓	✓	✓	✓
C_j	✓	✓	✓	✓

3.5 Application: Opinion Maps

3.5.1 Introduction to Opinion Maps

Opinion maps are a way to visualize different viewpoints in a certain debate [2]. This can for be useful when one wants to create an overview of different opinions and viewpoints of a debate. In an opinion map, there are different „countries“ that represent similar opinions (for example, in a debate about vegetarianism and veganism, vegans and vegetarians could each get their own „country“). After having stated their opinion by agreeing or disagreeing to different statements, a participant of the debate can also see their position on the map and so-to-speak their „nationality“ in the debate. An example of what an opinion map could look like is shown in fig. 5. We now want to look at how this visualisation works.

First, we want to look into how we can represent opinions of people in a form that a computer can understand. In our case, an *opinion* will be a set of statements that a participant has either agreed or disagreed with [2]. It is of course important that there is a sufficiently large and diverse set of statements so that each position can be represented by a subset of these statements. How to find a suitable set of statements is not topic of this report.

The next step is to form a graph out of these positions. Each position is a vertex in the graph and similarity between positions is represented by the edges of the graph. This means, each vertex is a set of statements.

The graph is then embedded in the plane (note that such a graph is not necessarily planar but can still be embedded into the plane with edge crossings). Graph drawing is an interesting problem on its own, that shall not be discussed in this report.

After that, a clustering algorithm is applied in order to group the different positions together. Each cluster of the graph will later represent a country on the map. We will discuss how to get the edge weights in section 3.5.2.

The last step is to compute a voronoi diagram. This means that each vertex is assigned the piece of the plane that is closer to the vertex than to any other vertex. The regions assigned to each vertex are then coloured in the colour of the cluster the vertex belongs to.

The result is a map where each opinion is a point on the plane, and similar opinions are grouped together as countries. Figure 5 shows how such a map might look like for the veggie debate, note that it is however hand-drawn and not the result of an actual experiment.

The challenge is now to find a measure for comparing two opinions. We therefore want to look into support based measures for opinion maps.

3.5.2 Support based Measures for Opinion Maps

We now want to look into how the edge weights in an opinion map can be calculated. The difference to the coherence measures discussed in section 3.2 and section 3.3 is that we need a function that measures the mutual coherence of *two* sets instead of the internal coherence

Figure 5 Possible opinion map for the veggie debate – hand drawn. Image Source: Sophie von Schmettow, [2]

of only one set. This section will be mainly based on [1], but we will introduce our own example of two sets in order to measure their coherence and we will also look into possible requirements for getting the desirable properties of a mutual coherence measure listed in [1].

In the following, an opinion will be a set of statements. For simplification we will assume that

The basic formula for *mutual coherence* is the following

$$\mathrm{MutCoh}_\tau(\mathcal{A},\mathcal{B}) := \frac{1}{2^{k_\mathcal{A}+1}}\sum_{X\subset\mathcal{A}}\mathrm{Conf}_\tau(X,\mathcal{B}) + \frac{1}{2^{k_\mathcal{B}+1}}\sum_{X\subset\mathcal{B}}\mathrm{Conf}_\tau(X,\mathcal{A})$$

where Conf_τ is a suitable Bayesian confirmation measure indicating to which extent Y confirms X [1]. It is interesting to know properties of $\mathrm{MutCoh}_\tau$and which criteria Conf_τ should fulfil in order for $\mathrm{MutCoh}_\tau$to have these properties. This will be investigated in the following.

In [1], the Kemeny-Oppenheim measure is used as the confirmation measure. It ranges between -1 and 1 where 0 is neutral. In the following we will assume that the confirmation measure has the same range. Desirable properties for MutCoh_τ listed in [1] are the following:

- MutCoh_τequals 1 for equivalent positions
- $\mathrm{MutCoh}_\tau$is negative for inconsistent positions
- MutCoh_τequals 0 for independent positions

It is easy to see that since MutCoh_τ is the average over Conf_τ for all subsets of $\mathcal{A}$ and $\mathcal{B}$ respectively, Conf_τ has to be 1 for all cases in order for MutCoh_τ to be 1. This means that for any subset of $\mathcal{A}$ and $\mathcal{B}$ Conf_τ has to be 1 which we can interpret as that the other set implies the subset. For equivalent sets it makes sense to expect this.

For inconsistent positions, MutCoh_τ is desired to be negative. There are multiple ways by which this could be achieved. The easiest case would be that all of the summands are negative. However, an inconsistent position can have a consistent subset as we have seen earlier in example 3 where each subset of the first set was in itself consistent. A similar thing could happen for two positions where $\mathcal{A}$ is not supported by $\mathcal{B}$ but subsets of $\mathcal{A}$ are supported by $\mathcal{B}$. This will be illustrated in the following example:

▶ **Example 5.** Similarly to example 3, example 2, and example 1, we want to look at sets of statements. In this case, there are two sets of statements $\mathcal{A}$ and $\mathcal{B}$, each of which consists of the disjunction of multiple smaller statements (like the witnesses who claimed that one out of a selection of suspects was the criminal).

$$\mathcal{A} = \{\{1 \vee 2 \vee 3 \vee 5 \vee 6\}, \{1 \vee 3 \vee 4 \vee 5 \vee 6\}, \{1 \vee 2 \vee 4 \vee 5 \vee 6\}\}$$
$$\mathcal{B} = \{\{1 \vee 2 \vee 3 \vee 4 \vee 5\}, \{2 \vee 3 \vee 4 \vee 5 \vee 6\}, \{1 \vee 2 \vee 3 \vee 4 \vee 6\}\}$$

In each case, each of the sets as a whole can agree on three numbers, namely $1, 5$, and 6 for $\mathcal{A}$ and $2, 3$ and 4 for $\mathcal{B}$. The sets they can agree on are disjoint. However, each subset of each set can agree on at least one additional number that is also in the conjunction of the other set. Therefore a confirmation measure could result in positive values for any non-empty subset of each of the sets $\mathcal{A}$ and $\mathcal{B}$. Therefore it may not always be given that sets that are in total not consistent result in a negative value for $\text{MutCoh}_\mathcal{T}$.

For independent positions $\mathcal{A}$ and $\mathcal{B}$, it is desirable that $\text{MutCoh}_\mathcal{T}$ is 0. Confirmation measures that are based on the difference between X given Y and X given not Y or X alone, this is usually the case as the two probabilites will be the same.

3.6 Conclusion

In this seminar write-up, multiple probabilistic coherence measures have been discussed. We have looked at how [3] investigated the coherence measures' compatibility with their intuition on which set is more coherent in Schupbach's robber case. Furthermore, we have added our own version of the robber case where the number of suspects that have been suspected by at least one witness is the same for the first case and the second case. We have seen that in this case the naïve overlap measure does not work according to our intuition. We have then furthermore looked at the modified robber case example from [3] which shows that inconsistent sets can differ in coherence as well. We have seen that for this example, most of the introduced coherence measures do not work. Therefore we have also taken a look at the improvements suggested in [3]. We have seen that there are multiple ideas for how one can work around the divisions by zero in the support measures.

After having seen multiple coherence measures applied to theoretical examples, we have also taken a look at a practical application, namely Opinion Maps. We have seen how coherence measures can be used in order to calculate the edge weights of a graph of opinions. We have also seen the algorithm used for creating a map from this opinion graph. We have then looked into desirable properties for the mutual coherence measure used for calculating the edge weights and have seen that not all of these properties are trivial to accomplish.

References

1 Gregor Betz. Calculating degrees of justification on dialectical structures.
2 Tamara Mchedlidze Gregor Betz, Michael Hamann and Sophie v. Schmettow. Mapping the veggie debate.
3 Mark Siebel and Michael Schippers. Inconsistency as a touchstone for coherence measures. *THEORIA. An International Journal for Theory, History and Foundations of Science*, 30(1):11–41, 2015.

4 Stemmatology or ’Who copied from whom?’

Rebecca Seelos

Abstract

This report wants to show how algortihmic methods can be used in the field of literature, more specifically in the field of textual criticism that deals with the relations between several versions of the same text. The core structure that we want to compute is the stemma, which can be seen as a graph of relationships. This report will go into history of the use of algorithmic methods and then presents two algorithms to use in stemmatology, focusing on the latest algorithm that uses strucural expectation-maximization.

4.1 Introduction

Over the course of human history, a text had many forms and many ways of distribution, beginning with oral distribution over handwritten texts to print. To pass a text on by speech obviously leads to changes the fastest. This, most of us have probably witnessed in the popular childrens game *Chinese Whispers*. But also the hand-written distribution of a text leads to a certain extent of evolution. The history of a text is defined by various versions that were copied from the original at some point, or copied from each other, where one version of a text is just a small snap-shot in the history of the text [2].
For these reasons, the critical edition established itself that, given different versions of the same text, tries to set the different snap-shots into a relation chronologically and/or decendancy-wise. The desire, given several different versions of a text, to find a common ancestor or most ancient version that is supposed to be the true content of text - when the original is long lost has existed for a long time. Today, this problem belongs to the disciplines under the name of textual scholarship, or more precisely textual criticism that deals with the origins of a text.
Stemmatology, which is sometimes called stemmatics or stemmatic analysis, is the part of textual criticism and tries to represent the relationships of surviving variants of a text and their evolution of different versions of a text in a tree. The structure, that displays the system of relation between the texts is called *stemma*. A formalised form of stemmatology as part of textual criticism was first introduced by Karl Lachmann.

4.1.1 The Stemma

A stemma is basically a graph representing the relations - in terms of similarities and differences due to copying one from another - between discovered and hypothetical versions of a given text. In an ideal world, and intented by Karl Lachmann, this graph is a binary tree, that has the orignal version on the text in the root, followed by the *archetype*, placing all discovered versions of the text in its leaf nodes (see Fig.1) and the interior nodes representing hypothetical versions of the text.
In reality several problems arise when a stemma is created for a text: The first problem is to have only one surviving variant where a stemma is obviously not possible. The second problem is that a version of the text can be copied by more than two scribes, which destroys to property of the tree being binary. The third problem is called *contamination*. This very common problem arises if scribes copied from two or more originals that are not identical and

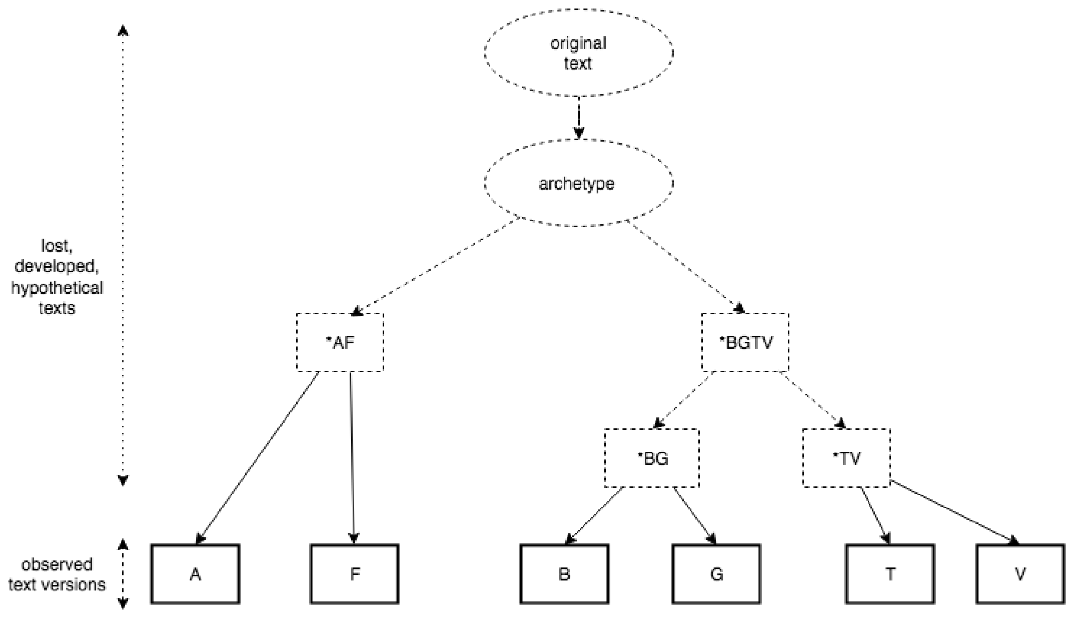


Figure 1 Schematic representation of a stemma, as a binary tree in which the original text is on the top and all the extant versions of the text are in the leaf nodes; reproduced from [2]

leads to te tree structure being destroyed as the condition of two vertices being connected by excactly one path can be violated.
Therefore creating a realistic stemma is more complicated and is not a tree most of the time. For reasons of simpler communication the terms *root*, *leaf* connected to trees are used regardless. Fig. 2 shows an example for a real stemma for the Julian of Norwich, *Showing of Love* manuscripts.

4.2 Algorithmic Methods for Stemmatology

4.2.1 Relationship to Phylogenetics

In 1991 a *Textual Criticism Challenge* was posted on several network bulletin boards by Peter Robinson. The challenge was to establish a kind of stemma for 44 manuscripts of and Old Norse narrative called *Svipdagsmal* by statistical or numerical methods. Apparently an evolutionary biologist called Robert J. O'Hara used a phylogenetic method and the results were remarkable at that time which probably marks the starting point of this development[1]. Phylogenetic analysis was used on stemmatic data and as a matter of fact the results were quite fruitful [11].
Phylogenetics are part of biology and bioinformatics, and research how different species have evolved from each other. A *phylogenetic tree* - a tree that shows the phylogenetic relationship of species by splitting up the species into different branches starting from a common ancestor - is the result of a phylogenetic analysis.
Of course there are some differences between phylogenetic analysis and stemmatology. In

[1] http://bmcr.brynmawr.edu/1992/03.03.29.html

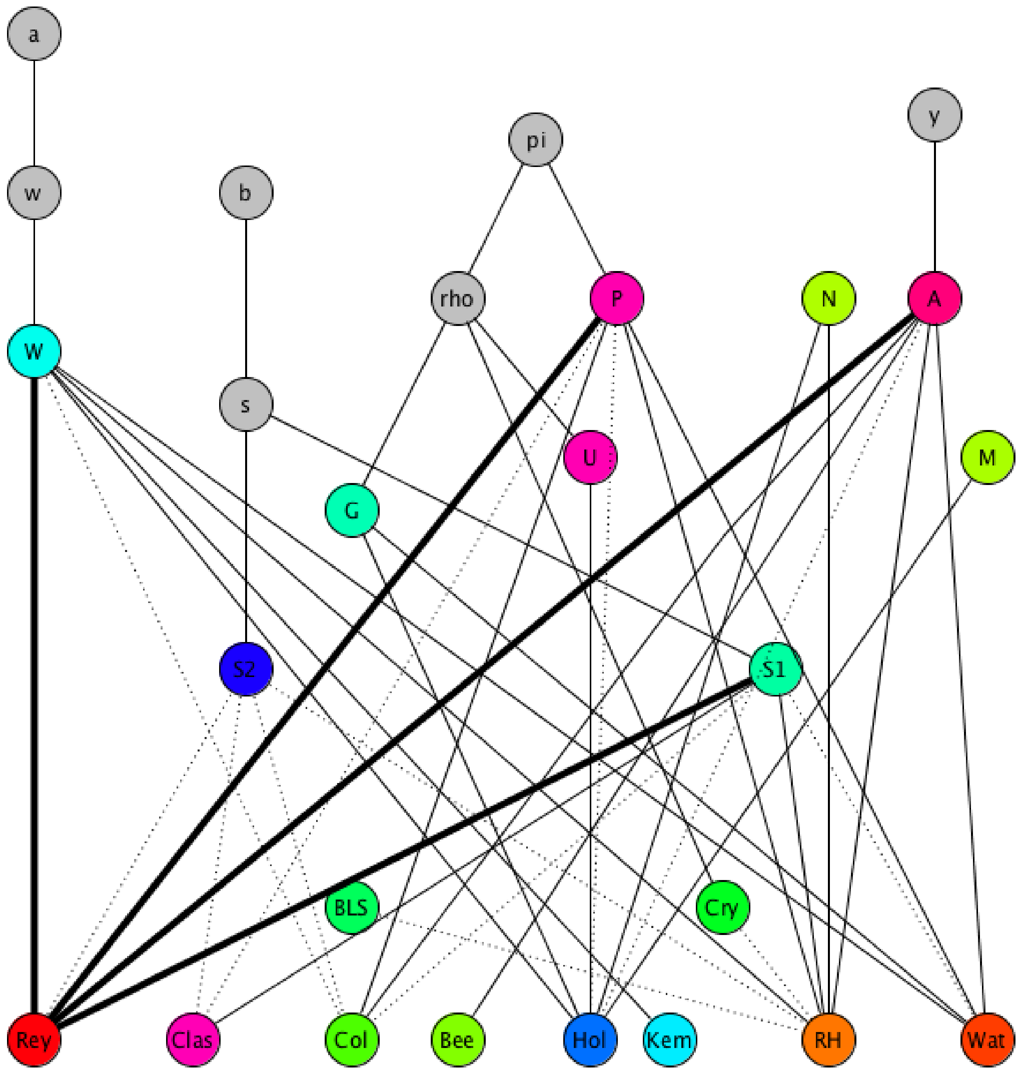


Figure 2 Example for a realistic stemma, not a tree. The hypothetical (unovserved) versions of the tradition are shown in grey, coloured nodes represent the observed versions. Severeal archetypes exist, the extant versions are also placed at 'internal' nodes, contamination can be seen at almost all of the 'leaf' nodes (one case of contamination is highlighted with thick lines here) and most versions are copied more than one time. This figure was reproduced from http://www.umilta.net/julsismelindex.html.

phylogenetics the alphabet that encodes the dna is limited to 4 DNA nucleotides, 20 amino acids or 64 codon triplets. In stemmatology on the other hand the alphabet can possibly be anything, and is defined by the texts that are analyzed.
In this paper we the presented algortihms are based on three different categories of phylogenetic algorithms:

1. **distance-matrix based** (example: *Neighbour Joining* [10] - in this algorithm the different versions of a sequence are based on a matrix that stores the fraction of mismatches between each pair of sequences (diagonal elements are zero) and then in each step joins the two sequences with the smallest distance through a new unobserved node and then calculating the new matrix including the new node.)
2. **parsimony based** (example: *Phylogenetic Algorithm Using Parsimony* [PAUP] - this method finds a tree that minimizes the total number of character changes.)
3. **model based** (example: *Structural Expectation-Maximization for Phylogenetics* [5] [SEMPHY] which is based on expectation-maximization that is used to fit the data to a given structure, in this case a tree representing the probability distribution, as a model for the phylogenetic tree.)

Most of the algorithms for phylogenetic analysis make the following assumptions about phylogenetic trees that are more unlikely to be true for stemmas: firstly they assume all observed nodes are leaf nodes, secondly most computer-generated phylogenetic trees are *bifurcating* trees, meaning all internal nodes have either zero or two decendants. [11].

4.2.2 RHM

The first algorithm for stemmatology was proposed by Roos, Heikkilä and Myllimäki in 2006. RHM computes a graph $G = (V, E)$, V corresponding to the text variants, $G = (V, E)$, based on their compressed information thus tries to find regularities in the observed texts and using these to set them into context.
The method is based on maximum parsimony algorithms for phylogenetics - a tree that is maximally parsimonious has minimal differences between the connected nodes. For an approximation of the complexity of a text, the `gzip` compression algorithm is used to compute a numerical number to measure the amount of information in a given text. String compression means that for strings x, y the amount of information is calculated as

$$C(x \mid y) = length(compressed(concatenated(x, y))) - length(compressed(x))$$

. In the end a maximum likelihood criterion is used to select a tree. Minimizing he total information cost approximates maximizing the likelihood. This tree at first is undirected and is converted to a directed graph.
As an input aligned text files are needed. The output (see Fig.6 **(c)**) however still has the same restrictions as the output generated by the former phylogenetic algorithms: trees are still bifurcating and all observed nodes are still leaf nodes. [8]

4.2.3 Semstem

The second algorithm for stemmatic analysis is called *Semstem*, which stands for *Structural Expectation-Maximization in Stemmatology.* It was proposed by Roos and Yuan in 2011.
This algorithm is based on a phylogenetic algorithm called *Semphy* that was mentioned above.
Expectation-Maximization algorithms in general deal with sets of incomplete data, in this

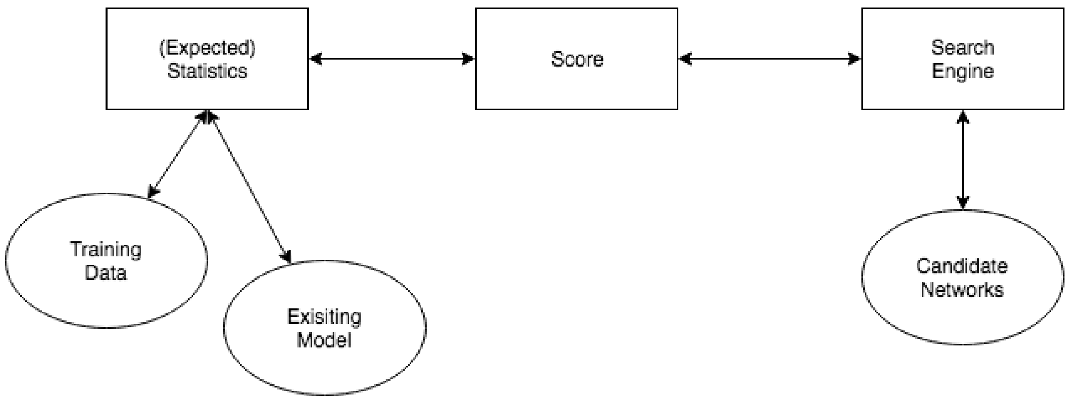


Figure 3 Schema of structural EM

case we want to find the unobserved manuscripts on the base of the extant versions of a text that were collected. The term *structural* EM means that in addition to estimating the data to best explain the given data it tries to fit the data into a structure, e.g. finding the best matching network for the training set. In this case a *belief network*, also called bayesian network is used as a model. These networks have two components pair $B = \langle G, \Theta \rangle$ and Θ where G is a directed acyclic graph where each node represents a random variable representing the structure and is a collection of local interaction models that describe the conditional probability of each vertex given it's parents in the graph. [4]
The schematic representation of a general structural EM algorithm is shown in Fig. 3) and shows that there are three components: a statistical model, a score function and a search engine [6]. The statstical model uses the training data and an existing model and then uses the score function to evaluate them where as a search engin evaluates new candidate networks using the same score function. If a candidate network is suitable a new iteration will use this as an existing model.
As for the input, all the texts in the different manuscripts need to be aligned in a way that each manuscript has the same number of positions and the same position in different manuscripts corresponds to the reading of the same word - if it exists.
The model for the evolution of texts is denoted as a uniform transition matrix for each position r in the aligned text meaning that all diagonal elements (therefore the probability of the position to stay the same) are $1 - \alpha$ and every other entry is

$$\frac{\alpha}{k_r - 1},$$

where $0 < \alpha < 1$ and k_r is the number of observed unique readings - e.g. the number of different words in position r.
The statistical model for a stemma T is defined as a set of nodes $X_1, ..., X_{M+N}$ and a set of edges $(i, j) \in \{1, ..., N + M\}^2$ where $N + M$ is the number of nodes, where $X_1, ..., X_N$ are the observed manuscripts and $X_{N+1}, ..., X_M$ are corresponding to the undiscovered/hypothetical manuscripts.
The two important functions for the algorithm are the *expected log-likelihood* of the new tree compared to 'best' tree and the *weights of potential edges* between nodes i,j.
For the case of the set containing undiscovered manuscripts, a number of $M = N - 2$ latent nodes $(X_{N+1}, ..., X_{N+M})$ is assumed (this results from the use of the NJ algorithm).
For an arbitrary tree structure, the expected log-likelihood is

$$Q(T : T_t) = E\left[L_T(X_1, ..., X_{N+M}) \mid X_1, ..., X_N, T_t\right].$$

Here $L_T(X_1, ...)$ is the logarithm of the probability of data X_1 to X_{N+M} given the stemma T. So the expextation value here denotes which log-likelihood the data has given the observed nodes and the new tree T_t (see if-statement in Algorithm 1).
This means that for every new tree, it is measured if the conditional expected value of the log-likelihood of all nodes (observed or unobsorved) given all the observed nodes and T_t.
The weights of potential edges between nodes i, j (for the complete data set) are defined as

$$w_{i,j} = \sum_{r=1}^{n} \sum_{(x,y)\in\Sigma^{(r)2}} P(X_i^{(r)} = x, X_j^{(r)} = y \mid X_1, ..., X_N, T_t)\frac{p_{x\to y}}{p_y}.$$

Here the $P(\dots)$ denotes "the conditional expectation of the indicator function"[9].
The first step of the Semstem-alogorithm is to initialize the tree T_0 using the neighbour joining algorithm. Therefore a matrix of relative distances between the texts is used for the initialization. Also, a initial likelihood $Q(T_0 : T_0)$ is calculated. Both values act as the initial max-values (Fig. 4 **(a)**).
Next comes the iteration of the expectation and maximization steps. The main-loop of the algorithm for Semstem in pseudocode is shown in Algorithm 1. It can be seen that in the expectation step (*E-step*) the weight function $w_{i,j}$ is used to calculate the weights between all the pair of nodes (Fig. 4 **(b)**).
The maximization step (*M-step*) uses the algorithm of Kruskal to determine a minimum spanning tree (MST) in the graph produced by the E-step (Fig. 4 **(c)**). The last step is to return tree T_{max} after a certain amount of iterations [9].
The following pseudocode shows the main loop of the Semstem method:

Algorithm 1 : Expectation-Maximization in Semstem

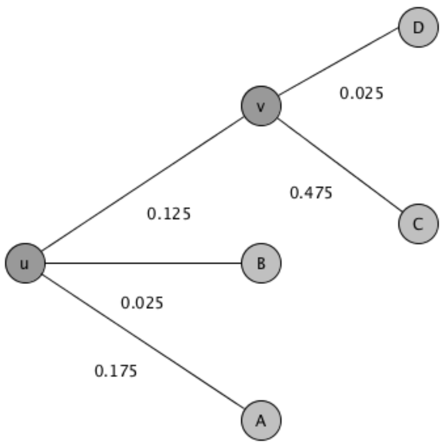
```
repeat
    E-step: compute the weights $w_{i,j}$ for all pairs of nodes i, j under tree $T_t$; (Step 2)
    M-step: find a new tree $T_{t+1}$ by the MST algorithm; (Step 3)
    if $Q(T_{t+1} : T_t) > Q_{max}$ then
        let $T_{max} = T_{t+1}$;
        let $Q_{max} = Q(T_{t+1} : T_t)$;
    end
    let $t = t + 1$;
until $T_{t+1} = T_t$ or $t > t_{max}$;
```

Fig. 4 shows a little demonstration of how the algorithm works. The example starts with the distance matrix of four hypothetical texts A, B, C and D. For the initialization step I created a table of fictional relative distances between the texts:

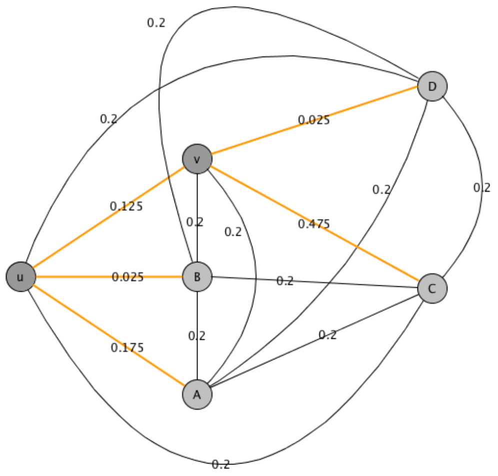
$$\begin{array}{c|cccc} & A & B & C & D \\ \hline A & 0 & 0.2 & 0.8 & 0.3 \\ B & 0.2 & 0 & 0.6 & 0.2 \\ C & 0.8 & 0.6 & 0 & 0.5 \\ D & 0.3 & 0.2 & 0.5 & 0 \end{array}$$

This is used as an input for the NJ algorithm an the output is the initial tree T_0. (The result of the first step is displayed in Fig.4 **(a)**). I calculated the initial tree using an online-tool

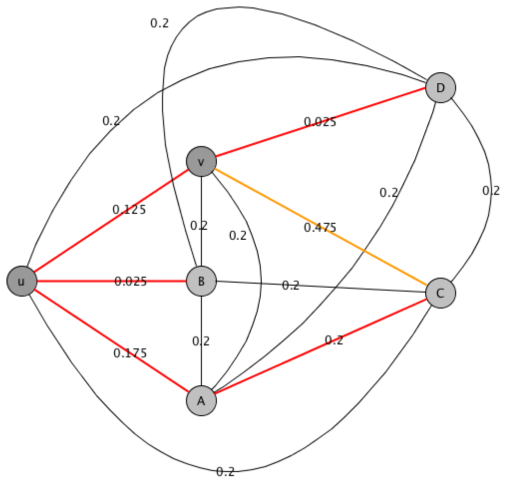
that implements the NJ algorithm[2].
For simplicity, in the expectation step I set all the new weights (that are normally calculated using $w_{i,j}$ as stated before) to 0.2, which lies in the range of the old values. (The result of the first step is displayed in Fig.4 **(b)**). Stochastic perturbations are added to reduce the chance of getting stuck at a local optimum. These consist of the addition of Gaussian noise to the elements in the weight matrix $(+\epsilon_{i,j})$



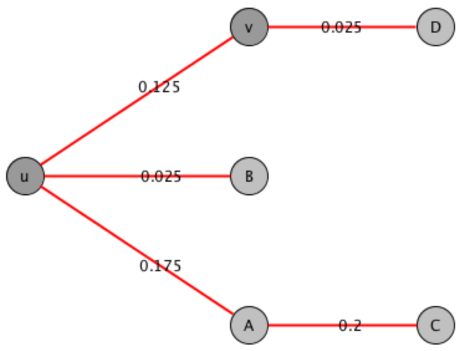
(a) Result of the initialization step



(b) Result of the expectation step



(c) Result of the maximization step



(d) Output after one iteration

Figure 4 Four steps of a simple example for the Semstem algorithm (for the fictional texts A, B, C, D).

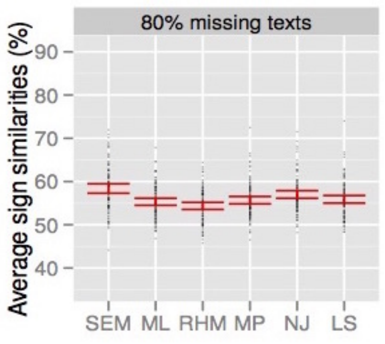
[2] http://www.trex.uqam.ca/index.php?action=trex&menuD=1&method=2

4.2.4 Comparison

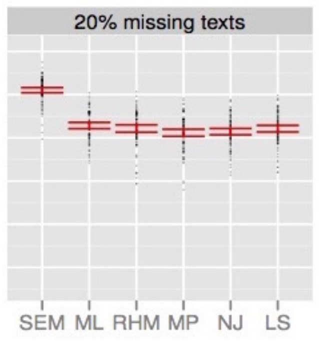
To show how the different techniques for computational stemmatology have evolved, a artificial tradition of *Notre Besoin de consolation est impossible à rassasier* (short *Notre Besoin*) will be used. The use of artificial traditions was established by different authors as a tool to assess the quality of stemmatic analysis as in artificial traditions the correct stemma is known and can be used for comparison [7] [1].
Notre Besoin was published as an essay in a swedish newspaper in 1952, therefore is more of a modern text. Artificial means that the text was copied in a controlled environment and a correct stemma is known [1]. The artificial tradition used here contains 13 versions of *Notre Besoin* where the longest version has 1035 words and one version is contaminated [7].
When this tradition was introduced in 2004 Macé et al. used the following questions to evaluate the different methods for stemmatology [1]:

1. Are groups and clusters identified correctly?
2. Are the distances between the manuscripts indicated?
3. Is the output restricted to a binary tree?

In the course of the *Computer-Assisted Stemmatology Challenge*[3] a evaluation script was developed to evaluate the different methods based on their output. The script that was published by Roos et al. can be accessed athttps://www.cs.helsinki.fi/u/ttonteri/casc/stemma.html.
The goal of the challenge was to collect and evaluate different methods for stemmatological analysis. In the course of the challenge different artificial traditions were collected to provide a tool for comparing the different methods.



(a) RHM output



(b) Semstem output

Figure 5 Two sets of scores of different algorithms, (interesting for us are SEM = Semstem, RHM, and NJ = Neighbour Joining) using the average sign distance measure on stemmas for*Notre Besoin*. The left shows the scores if in the data, 80 % of the text is missing; the right shows the scores if 20% of the text is missing.

For the comparison of the results a quality measurement called *average sign distance* is used. This method uses triples (A, B, C) of nodes in two graphs to compare and calculate if the distance between A and B is less, equal or greater than the distance between A and C in the first graph, and compares them to the distances between (A, B, C) in the second graph.

[3] https://www.cs.helsinki.fi/u/ttonteri/casc/

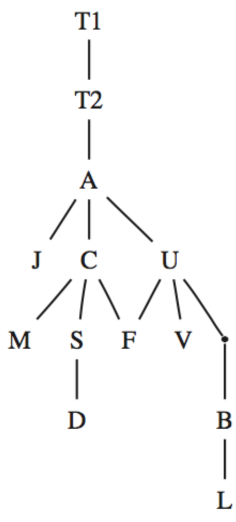
Correct stemma	Proposed stemma	u(A,B,C)
d(A,B)<d(A,C)	d'(A,B)<d(A,C)	1
d(A,B)<d(A,C)	d'(A,B)=d(A,C)	$\frac{1}{2}$
d(A,B)<d(A,C)	d'(A,B)>d(A,C)	0
d(A,B)=d(A,C)	d'(A,B)<d(A,C)	$\frac{1}{2}$
d(A,B)=d(A,C)	d'(A,B)=d(A,C)	1
d(A,B)=d(A,C)	d'(A,B)>d(A,C)	$\frac{1}{2}$
d(A,B)>d(A,C)	d'(A,B)<d(A,C)	0
d(A,B)>d(A,C)	d'(A,B)=d(A,C)	$\frac{1}{2}$
d(A,B)>d(A,C)	d'(A,B)>d(A,C)	1

Table 1 All possible configurations for the score function $u(A, B, C)$ used to compare two graphs.

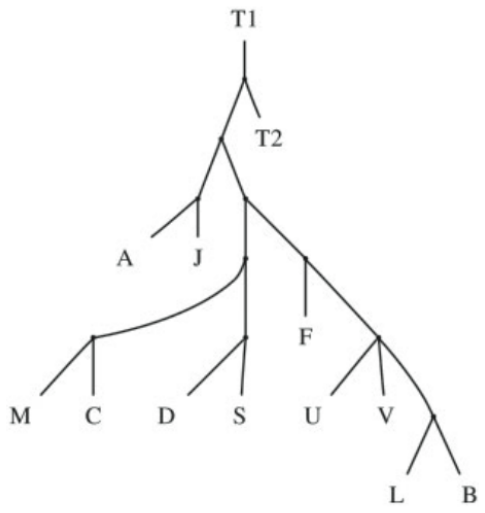
The score function is visualized in Tabular 1 (reproduced from [7]) in the form of all possible values that it generates.The possible scores vary between 50% and 100%, with 50% meaning that there are no edges at all.

The following results for different algorithms based on *Notre Besoin* were published in a comparative study following the Computer-Assisted Stemmatology Challenge. Fig. 6 **(a-c)** shows the results of the study. The correct stemma for the artificial tradition used for comparison is shown in Fig. 6 **(a)**. The stemma produced by the phylogenetic *Neighbour Joining* algorithm (Fig. 6 **(b)**) produces a score of 77.389%. The stemma produces by *RHM* (Fig. 6 **(c)**) is quite the same with a score of 76.9%. The problem with the presented output of Semstem is that it is produced from a set of texts where 10% of the data is missing. This means it can't be compared to the others as simple as that.

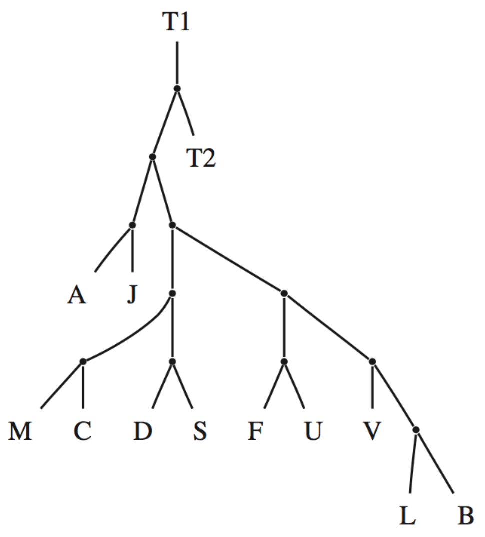
The comperative study was published in 2009 so Semstem is not yet concluded, but Roos et al. also used the average sign distance to evaluate Semstem when it was published. In Fig. 5 the scores for two different experiments are shown still using the set of *Notre besoin*. Fig. 5 **(a)** shows the results on a set where 80% of the text is missing. Fig. 5 **(b)** shows the results on a set where 20% of the text is missing.
In Fig. 5 **(a)** all the scores are close to a random distribution of nodes which is to be expected when most of the data is missing. In 5 **(b)** the scores for RHM and NJ are around 70% matching our results for the complete data set. The score for *Semstem* with 20% of the data missing is around 80%. Therefore having the best result in this case. This suggests that in a complete data set it would also perform better than the other two algorithms.



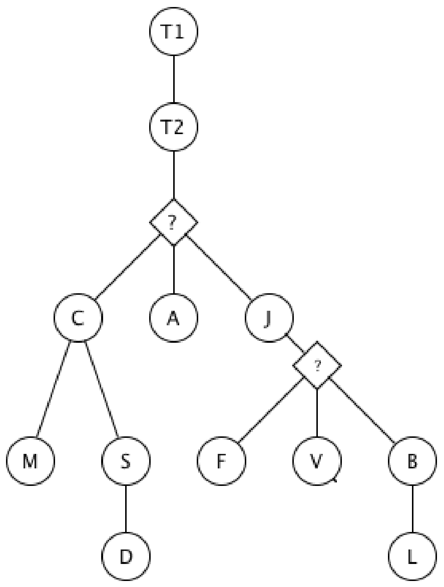
(a) Correct stemma



(b) NJ output



(c) RHM output



(d) Semstem output

Figure 6 Four different stemmas for a artificial tradition of *Notre besoin*. (a) Shows the correct stemma (as this is a *artificial tradition* it is known who copied which text)

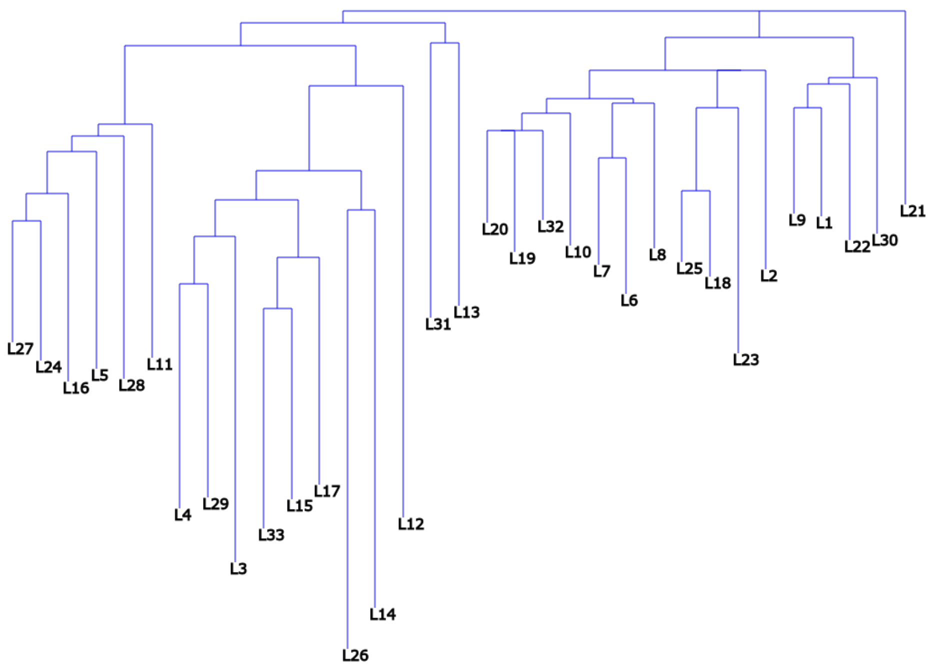


Figure 7 Phylogenetic tree of the 33 chain letters, computed using the NJ method.

4.3 Further Application – Chain Letters

The use of stemmatic analysis is not restricted to medieval texts. In 2003 Bennett et al. published a phylogenetic anlaysis of chain letters. For this they compared 33 versions of a chain letter that promises great fortune to the ones that pass it on (- all 33 chain letters can be found at https://cs.uwaterloo.ca/~mli/chain.html).
For determining the phylogenetic tree, they used a compression-based phylogenetic Algorithm. [3] Even if the purpose of this study was, among other things, to make phylogenetic trees more intuitive to students using letters in natural language instead of genetic sequences. The phylogenetic tree of the chain letters is shown in Fig. 7. It was reproduced from [3] using the distance-matrix that is given at https://cs.uwaterloo.ca/~mli/matrix33.
As this is also a sort of literature, in my opinion a stemma for these chain letters seems to be an interesting study object. It can be assumed that there is very little contamination here as one person only copies one chain letter. And due to the letters being shorter and their distribution is recorded better it seems a good way to focus on the stemma itself here and the respective algorithms.

4.4 Conclusion

In my opinion the problem of creating a stemma shows very well how computational methods can get involved in the humanities and how problems from biology can be help to solve problems in Literature. Over the course of time the methods that were used changed from phylogenetic methods to methods specifically for stemmatology and the use of digital methods in the field of literature is growing slowly but steadily, also starting to develop its own methods to better suit its needs.

As of now, Semstem seems to be state-of-the-art for stemmatic analysis, improving some of the issues that came with the use of phylogenetic algorithms. It is still not perfect though, as seen in the comparison section.
Phylogenetic methods, on the other Hand, are developing too and can not be ignored for the future research.
One problem that remains is how to measure the quality of the used methods and their respective stemmas. While there are obvious differences in the outputs of different methods this is rather hard to specify in a scientific way. While in a early evaluation the main tool was to ask questions that would intuitivly evaluate the methods Roos et al. presented the average sign distance method in 2009 where the methods are based on the graph similarity of the stemmas. This would give a percentage of similarity but, in my opinion, is not very intuitive at first.
It should be interesting to see how this field develops in the future along the field in evolutionary biology. Roos et al. even suggested that there is the possibility of a field like computational literary studies.

All in all this is a good example of how algorithms can be used in the humanities, even in established fields like stemmatology where the traditional methods have a long history and where the use of a certain method is widely discussed - not only computational methods but stemmatological methods in general.

References

1 Philippe Barret, P. Robinson, and Caroline Macé. Testing methods on an artificially created textual tradition. 2004.

2 Thomas Bein. *Textkritik*. 2009.

3 Charles Bennett, Ming Li, and Bin Ma. Chain letters and evolutionary histories. 2003.

4 Nir Friedman. The bayesian structural em algorithm. In *Proceedings of the Fourteenth conference on Uncertainty in artificial intelligence*, 1998.

5 Nir Friedman, Matan Ninio, Itsik Pe'er, and Tal Pupko. A structural em algorithm for phylogenetic inference. 2002.

6 Nir Friedmann. Learning belief networks in the presence of missing values and hidden variables. 1997.

7 Teemu Roos and Tuomas Heikkilä. Evaluating methods for computer-assisted stemmatology using artificial benchmark data sets. 2009.

8 Teemu Roos, Tuomas Heikkilä, and Petri Myllymäki. A compression-based method for stemmatic analysis. 2006.

9 Teemu Roos and Yuan Zhou. Analysis of textual variation by latent tree structures. In *2011 IEEE 11th International Conference on Data Mining*, 2011.

10 Saitou and Nei. The neighbor-joining method: a new method for reconstructing phylogenetic trees. 1987.

11 Matthew Spencer. Phylogenetics of artificial manuscripts. 2004.

5 Storyline Visualization

Sihan Ren

Abstract

As a useful visualization method, storyline visualization illustrate the character relationships in a story by tracing the story evolution in a visual way. However, while generating such a visualization layout, we may encounter some problems, which have a negative influence on the readability: line crossing, wiggle lines, long wiggle distance and much unnecessary white space. These problems deteriorate the readability and aesthetic appearance of layout. According to the problems, corresponding optimization algorithm is applied to get a better result. Storyline Visualization is widely applied in many fields, it reveals the relationships among entities along with their change.

5.1 Background

Everyone has told a story. When someone is telling a story, he must organize the relationships among characters of the story in his mind deliberately or unconsciously. With simple stories, people can handle it without problem. But when a story gets more complicated, especially when the number of characters increases drastically and the relationships among them change frequently over the time, it can get challenging to tell a well-organized story, which can be easily understood. To solve this problem R. Munroe made a hand drawn illustration of the movie *Jurassic Park* as in Figure 1. This hand drawn layout is the inspiration of storyline visualization. In this layout, each line represents a character. A line going from left to right along the horizontal timeline represents the lifespan of the character in this story. When some characters are having a session with each other, their lines will be grouped together. However this kind of layout by hand is time consuming and not unified with different editors. Therefore [5] proposed storyline visualization method to track the evolution of stories.

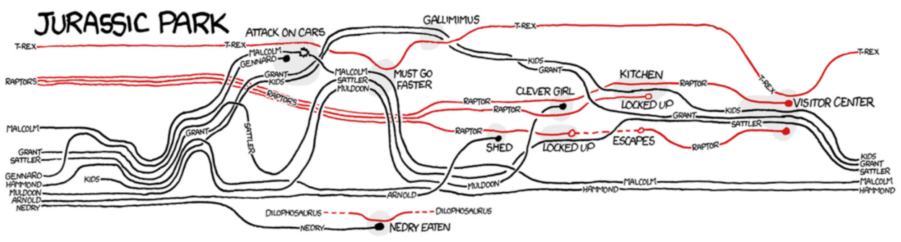


Figure 1 hand-drawn illustration from XKCD [6]

5.2 Model and Metrics

5.2.1 Model

The storyline problem is defined as follows. Let $C = \{1, ..., n\}$ be a set of characters, $T \subset \{[s, t] \mid s, t \in \mathbb{N}, s \leq t\}$ be a arbitrary time interval within the story, and $L = \{1, ..., m\}$

be the locations in the story. *Storyline* is a quadruple $S = (C, T, L, E)$, which is defined by a set of characters C, who meet during closed time intervals T, in the Location L. We call a meeting of characters at some time interval in some location an *session*, and the set of sessions is denoted as $E \subset L \times C \times T$. In each session $E_i = ([s_i, t_i], L_i, C_i) \in E(1 \leq i \leq m)$ the closed time interval $[s_i, t_i] \in T$ means the duration of this session, the symbol of location $L_i \subset L$ denotes the place of this session, and a subset $C_i \subset C$ of characters denotes the involved characters in this session (naturally, a character cannot participate in two overlapping events). A *storyline visualization* is a 2D visual representation of S such that

- characters are drawn as x-monotone lines placed in some vertical order for each point in time, where the x-axis represents time;
- O_i is defined as the line representing the i-th character, $y_j(O_i)$ be the y-coordinate of line O_i at time j;
- at each time point there exists a vertical line called time frame;
- during each session $E_i = ([s_i, t_i], L_i, C_i)$, lines representing characters in C_i should be grouped within some small vertical distance δ_{in} of each other, and otherwise the characters should be separated by some larger vertical distance $\delta_{out} > \delta_{in}$;
- all sessions within the same location should be contained in a polygon. In order to distinguish each location, adjacent polygons are always colored with different color.

Figure 2 offers a simple example of a session in a storyline. The second session E_2 in the storyline happens from the time frame s_2 to t_2 in the location L_2, and it involves the characters $C_2 = \{2, 3\}$.

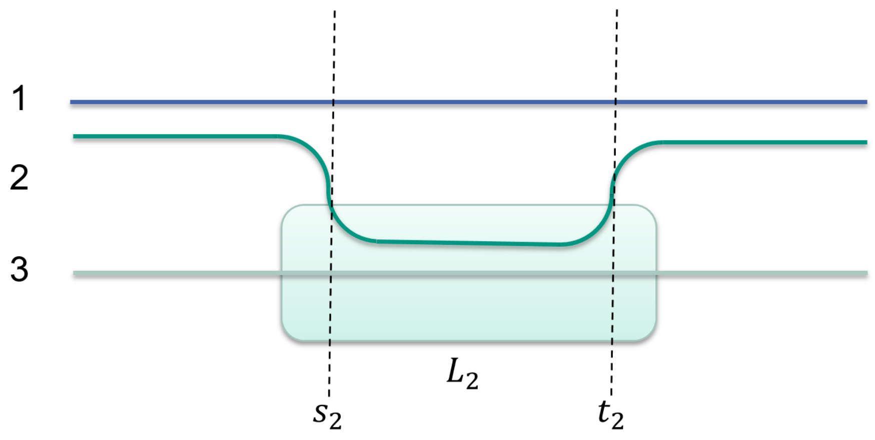


Figure 2 A simple example of a session in a storyline

In order to generate this kind of visualization, we need some information about those character relationships, namely the quadruple $S = (C, T, L, E)$. Such information is organized in a session table and a location tree. Every element in the session table $a_{i,j}$ has a session ID, to which the character i belongs at time j, that's to say, it tells us at every time frame, which characters are in the same session. Table 1 shows a simple example of a session table, where characters i_2 and i_3 have a session at time j_2, and characters i_1 and i_2 are in the same session at time j_3. The location tree represents all locations and all session IDs occurring at each location. With the session table and the location tree as input, we may apply visualization methods to get the initial drawing, which makes the relationship information visible in a picture.

Table 1 An example of session table

	j_1	j_2	j_3
i_1			SID_2
i_2		SID_1	SID_2
i_3		SID_1	

5.2.2 Metrics

However, it's not so easy to generate a readable layout, some problems would appear with a cursory layout. When some lines cross others, it results in line crossing, which makes it hard to recognize the lines clearly. As we will see, the number of line crossings can be reduces by reordering the lines at each time frame. Secondly, many lines could wiggle frequently, which leads to visual discontinuities. Then we should align as many lines, in order to make the most line segments straight. In addition, the wiggle distance and white space have also an influence on a compact and balanced layout, so the layout should be also compacted. In a word, optimization metrics are: line crossing, line wiggle, wiggle distance and white space. And the pipeline of the optimization is ordering, alignment and compaction, as shown in Figure 3. Formally, they can be defined as follows.

▶ **Definition 1** (Line crossing). As defined, $y_j(O_i)$ is the y-coordinate of line O_i at time j. $\forall m, n \in C, \forall t \in T$. If $y_t(O_m) > y_t(O_n)$ and $y_{t+1}(O_m) < y_{t+1}(O_n)$ then there exists a *line crossing* between lines O_m and O_n from time t to $t+1$.

▶ **Definition 2** (Line wiggle *and* Wiggle distance). $\forall m \in C, \forall t \in T$. If $y_t(O_m) \neq y_{t+1}(O_m)$, then the line O_m has *wiggle* from time t to $t+1$, $d_w = |y_{t+1}(O_m) - y_t(O_m)|$ is the *wiggle distance.*

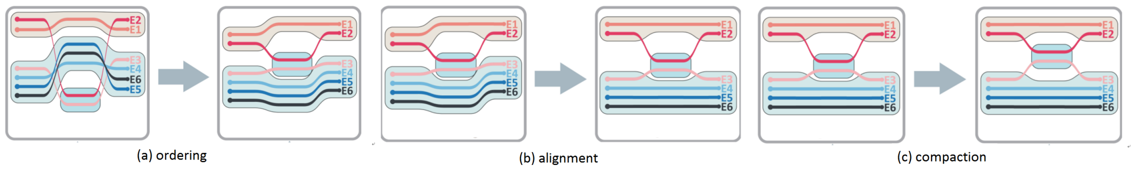


Figure 3 Different operations for the metrics: (a)ordering for minimizing the line crossing; (b)alignment for reduce wiggle lines; (c)compaction for reducing wiggle distance and white space [5]

5.3 Optimization algorithms

5.3.1 Odering

In order to reduce the line crossings, it is necessary to arrange the orders of lines properly, that's to say, we should decide wisely which lines should lay above while which lines should lay below. We can divide the ordering problem into two parts: 1) sorting the location nodes; 2) ordering the session and entity nodes within each location.

5.3.1.1 Sorting the location nodes

Here we treat locations as nodes with all sessions happening there, and each session contains all involved characters. The basic idea is that we first place and fix the node with the largest number of characters, and place the remaining nodes in a position that introduces a minimum crossing number. As in Figure 4, it involves three locations A, B and C in total. We assume that A and C are already fixed as in Figure 4.c , which provides C three candidate positions. We calculate the crossing numbers in each candidate position and find the best position with the least crossing numbers. This kind of sorting happens every time when a new location appears, then all existing locations along with the new locations are reordered and keep their orders till the next new location appears.

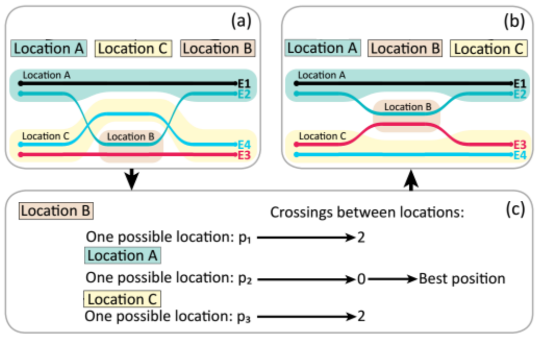


Figure 4 An example of sorting locations[5]

5.3.1.2 Ordering the lines within location

After sorting the locations, the orders of characters' lines within a location should be calculated. At each time frame, all lines degrade into nodes. These nodes are representation of the characters at each time frame. The change of line orders is actually also the change of node positions at each time frame. Thus, this problem is transformed to order the node positions at each time frame. The basic idea is that we treat the ordering at the first time frame as a reference and calculate the ordering at the next time frame, this step will be repeated until the ordering at the last frame is calculated. Afterwards we treat the ordering at the last time as reference and calculate the ordering at the last time frame. The iteration is conducted forwards and backwards and will stop when the line crossing number is already acceptable or the maximum iteration number is reached. In each step of iteration, every two adjacent time frames, the problem of ordering points positions can be further abstracted as the two-layer crossing problem, which can be solved by several methods.

We can firstly transform the problem to the crossing problem in a two layered digraph by concerning each adjacent two time frames. A two layered digraph is a bipartite digraph $G = (L_1, L_2, E)$, which consists of disjoint sets L_1 and L_2 of vertices and a set $E \subseteq L_1 \times L_2$. All vertices of each layer lie on a vertical line. In each two layered digraph Barycenter method [8] can be used to solve this problem. The definition of Barycenter is described as follows.

▶ **Definition 3** (Barycenter). The *barycenter* of any vertex $u \in L_2$ is the average of the y-coordinates of of its neighbors. That is, $\forall u \in L_2$:

$$BCenter(u) = \frac{1}{deg(u)} \sum_{v \in N_u} y_1(v)$$

where $deg(u)$ means the degree of u, and N_u means all neighbor vertexes of u in the first layer L_1.

For example, $\exists u \in L_2$ has three neighbors in the first layer L_1, v_1, v_2, and v_3. Its barycenter is exactly the average of the y-coordinates of the three neighbors. Abstractly seen, the barycenter describes where the neighbors of a vertex concentrate. According to the Barycenter method, all nodes are placed on their neighbour's barycenter, which introduce less line crossings.

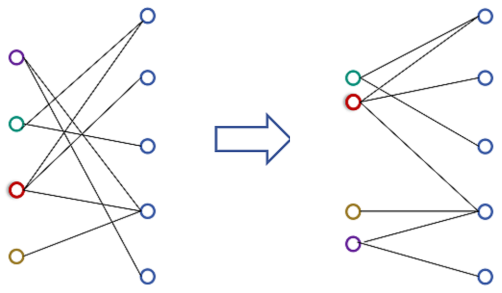


Figure 5 Using Barycenter method to reduce crossing lines.

The median method [3] is similar to the barycenter method. The Median is defined as:

▶ **Definition 4** (Median Center). Given a two layered digraph $G = (L_1, L_2, E)$, the *median center* of any vertex $u \in L_2$ is the median of the y-coordinates of its neighbors. If the neighbors of u are $v_1, v_2, ..., v_j$, with $y_1(y_1) < y_1(v_2) < ... < y_1(v_j)$, then we define $med(u) = y_1(v_{\lfloor j/2 \rfloor})$

According to the median method, all nodes are placed on median of their neighbours, which would also introduce less line crossings.

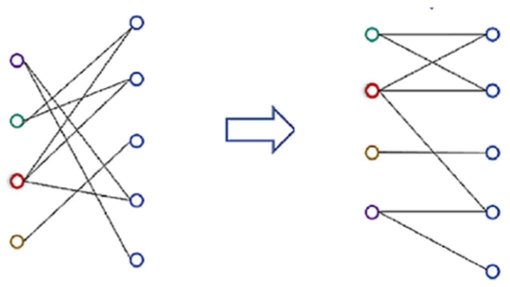


Figure 6 Using Median method to reduce crossing lines.

Finally, the penalty minimization method [8] is also a good alternative to be applied for solving this problem. This method cares more about the relationship among vertices in the same layer. To illustrate this method, some related concepts should be declared.

▶ **Definition 5** (Interconnection matrix). Given two level bipartite digraph $G = (L_1, L_2, E)$, $\sigma_1 = v_1 v_2 ... v_m$ and $\sigma_2 = w_1 w_2 ... w_n$ denote the order of all vertexes in L_1, L_2, where there are totally m and n vertices. A *interconnection matrix* M is a $m \times n$ matrix whose rows and columns are ordered according to σ_1 and σ_2. The element (v_k, w_l) in M, denoted by m_{kl} is given by:

$$m_{kl} = \begin{cases} 1, & if(v_k, w_l) \in E \\ 0, & otherwise \end{cases} \tag{1}$$

▶ **Definition 6.** Given an interconnection matrix M, $r(v)$ denote the row vector in M, which also means the vertex in L_1, the number of crossings $k(r(v_j), r(v_k))$ produced only by the order of pair of row vectors $(r(v_j), r(v_k))$ is given by, in another word, when all other vertex in L_1 except (v_j, v_k) are ignored, and v_j precedes v_k, the number of crossings is given by $k(r(v_j), r(v_k))$:

$$k(r(v_j), r(v_k)) = \sum_{\alpha=1}^{q-1} \sum_{\beta=\alpha+1}^{q} m_{j\beta} m_{k\alpha}$$

where q means the number of vertex in L_2

$k(r(v_j), r(v_k))$ denotes the number of crossings caused by the pair order that v_j is placed before v_k. On the contrary, if v_k is placed before v_j, the number of crossings caused by the pair order can be represented by $k(r(v_k), r(v_j))$. Figure 7 gives a direct demonstration of $k(r(v_j), r(v_k))$. Take the vertices d and e for example. $k(r(d), r(e))$ is the crossing number, which results from the edge connecting with d and e, when d precedes e, like in M. According to this formula, we can work out that $k(r(d), r(e)) = 3$, which conforms to the number in the digraph.

Using the definition of $k(r(v_k), r(v_j))$, we can represent the total number of crossings in the bipartite digraph by concerning all the 2-combinations of vertex in L_1 as follows:

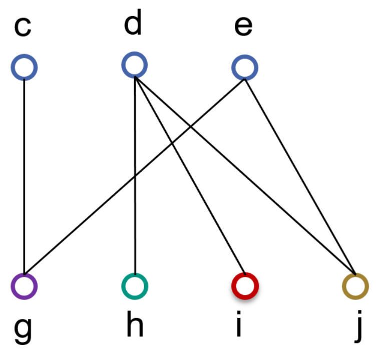
$$K(M) = \sum_{j=1}^{p-1} \sum_{k=j+1}^{p} k(r(v_j), r(v_k))$$

where p means the number of vertex in L_1. According to the pair order of v_j and v_k, one alternative of $k(r(v_j), r(v_k))$ and $k(r(v_k), r(v_j))$ will be chosen for the calculation of $K(M)$. That's to say, when we permute the orders of vertex in L_1, some of $p(p-1)/2$ terms will be replaced from one alternative to another because of the permuted order. In this way, we transform the problem of the optimal vertex's order to the problem to determine the set of orders in all the 2-combinations with some restrictions. Clearly if we take minimum for each term, i.e $mink(r(v_j), r(v_k)), k(r(v_k), r(v_j))$, the total number of crossings can be also be minimized. However there might not be any order of vertices which corresponds to the sum so obtained, hence some terms might have to take nonminima. Let $p(v_j, v_k) = k(r(v_k), r(v_j)) - k(r(v_j), r(v_k))$ be the *penalty*. To solve the crossing problem, some appropriate orders in all the two-combinations of vertices should be determined such that the sum of penalties induced by nonminimal terms is minimized.

▶ **Definition 7** (Strongly connected components)**.** In a directed graph G, if there are not only directed path from the vertex v_i to v_j, but also directed path from the vertex v_j to v_i, then the vertices v_i and v_j are strongly connected. A strongly connected subgraph is a subgraph, whose vertexes are all strongly connected with each other. The maximum strongly connected subgraph is then *strongly connected components*.

▶ **Definition 8** (Penalty digraph)**.** Given a interconnection matrix M, then consider the case where row order σ_1 is permuted under the fixed order σ_2. The *penalty digraph H* is defined by

$$H = (W, F, p)$$



	g	h	i	j
c	1	0	0	0
d	0	1	1	1
e	1	0	0	1

Figure 7 A two layered graph with its matrix realization

where

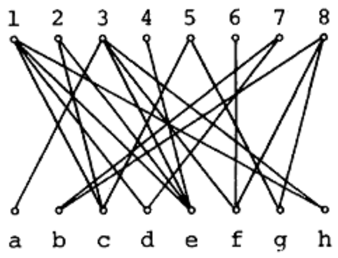
$W = \{v | v \in L_1\}$ is the vertex set

$F = \{(u, v) \in W \times W | k(r(u), r(v)) < k(r(v), r(u))\}$ is set of directed edges from u to v, which indicated that u precedes v such that fewer crossings are introduced than another order between them

$p : F \to \mathbb{N} = \{1, 2, ...\}$ is a weight function defined by $p(u, v) = k(r(v), r(u)) - k(r(u), r(v))$ for each $(u, v) \in F$

▶ **Definition 9** (Feedback arc set). A *feedback arc set* is a set of edges which, when removed from a directed cycled graph, leaves a directed acyclic graph. Put another way, it's a set containing at least one edge of every cycle in the graph.

In the penalty method, the penalty digraph H is firstly calculated. That's to say, for each $(u, v) \in W$, we are supposed to find $\min(k(r(u), r(v)), k(r(v), r(u)))$ and the difference between them. And then all strongly connected components in H are found[2]. In every component which contains more than two vertices, cycles should be eliminated by reversing directions of edges so that the sum of penalties is minimized. According to the revised digraph of H, the order of vertices in L_1 can be then determined. Let's take an example: Figure 8 shows a directed graph G along with its interconnection matrix.



$$g = \begin{array}{c|cccccccc|} & a & b & c & d & e & f & g & h \\ \hline 1 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 1 \\ 2 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 \\ 3 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 1 \\ 4 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 5 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 \\ 6 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 7 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 8 & 0 & 1 & 0 & 0 & 0 & 1 & 1 & 0 \\ \hline \end{array}$$

Figure 8 A two layered graph G with its matrix realization [8]

Step 1: According to the interconnection matrix, we are supposed to calculate its penalty digraph for $v \in L_1$, where $u \to v$ means $(u, v) \in F$, namely $k(r(u), r(v)) < k(r(v), r(u)))$ the number above arrows means the value of q, namely the difference between $k(r(u), r(v))$ and $k(r(v), r(u)))$. Figure 9 is the corresponding penalty digraph.

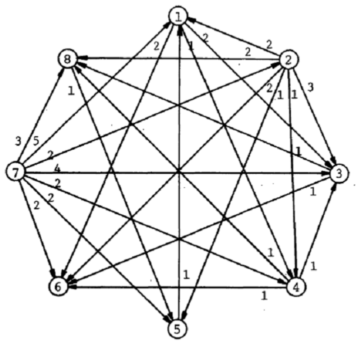


Figure 9 Penalty digraph H of $G[8]$

Step 2: With H, all the strongly connected components can be found: $\{1, 3, 4, 5, 8\}$ $\{2\}$ $\{6\}$ $\{7\}$

Step 3: The first strongly connected component $\{1, 3, 4, 5, 8\}$ has more than 2 vertices, its cycle should be found and eliminated. The connection of the the strongly connected component is demonstrated in Figure 10. Using the minimum feedback arc set algorithm, we can get the minimum feedback arc set solutions $v_5 \to v_1$ with penalty 1, the the minimum feedback arc set algorithm can be seen in [4].

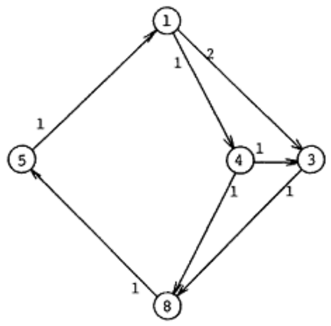
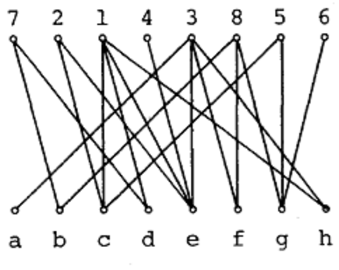


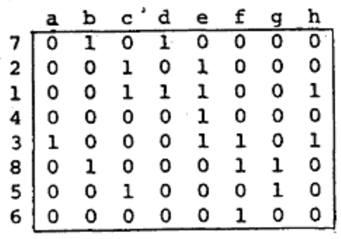
Figure 10 A strongly connected component in $H[8]$

Step 4: After reversing the direction from $v_5 \to v_1$ to $v_1 \to v_5$, we can find three orderings for the first layer L_1: $\{7, 2, 1, 4, 3, 8, 5, 6\}, \{7, 2, 1, 4, 3, 8, 6, 5\}, \{7, 2, 1, 4, 3, 6, 8, 5\}$. In another words, we find three paths going through every vertex in the reversed H. Consequently, we get the following map in the order of $\{7, 2, 1, 4, 3, 8, 5, 6\}$ along with its matrix. The number of crossing reduces from 69 to 48. The result is demonstrated in Figure 11.

5.3.2 Alignment

With the aim of minimizing the wiggle number, as many line segments as possible should be aligned. Line segments mean the parts of lines, which are divided by time frames.





	a	b	c	d	e	f	g	h
7	0	1	0	1	0	0	0	0
2	0	0	1	0	1	0	0	0
1	0	0	1	1	1	0	0	1
4	0	0	0	0	1	0	0	0
3	1	0	0	0	1	1	0	1
8	0	1	0	0	0	1	1	0
5	0	0	1	0	0	0	1	0
6	0	0	0	0	0	1	0	0

Figure 11 Result of the Penalty method[8]

Mathematically, it can be expressed as:

$$E_{align} = max \sum_{t=0}^{n} H(t)$$

Where $H(t)$ is the number of straight line segments between time frame t and $t+1$. As in Figure 12, the line segments between time frames t and $t+1$ are not straight at all, our goal is to make them straight, namely align as many line segments between each adjacent time frame t and $t+1$. The basic idea is to put the similar sessions at the same horizontal level. Assume there are two sessions a and c at time t, and there are three sessions a', b' and c' at time $t+1$, now these two session sequences at time t and $t+1$ should be matched.

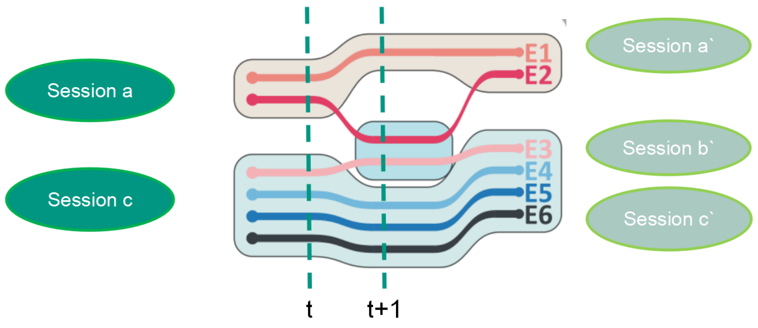


Figure 12 Illustration for alignment: At time t and $t+1$, sessions sequence ac and $a'b'c'$ should be compared and matched

Basically we can treat the problem of matching sessions between adjacent time frames as the longest common subsequence (LCS) problem. The longest common subsequence (LCS) problem is the problem of finding the longest subsequence common to both sequences. For example, the longest common subsequence of the sequences $ABGZDC$ and $QACBGD$ is $ABGD$.

In this scenario all sessions at two adjacent time frames are treated as two sequences and the longest common subsequence between them should be found. And then all matched session pair are going to be placed roughly at the same horizontal level, precisely at the place where between these matched sessions the most line segments are straight. In this case we can get the most straight segments between matched sessions. However letter strings can be easily compared whether they are the same. With sessions, a particular similarity used to compare sessions is defined as follows.

▶ **Definition 10** (Similarity between sessions)**.** Assume there are totally m sessions at time t, n sessions at time $t+1$, let l_i be the i-th session at time t, r_j be the j-th session at time $t+1$, the *similarity* between them is defined:

$$sim(l_i, r_j) = straight(l_i, r_j) + \alpha(1 - |\frac{i}{m} - \frac{j}{n}|)$$

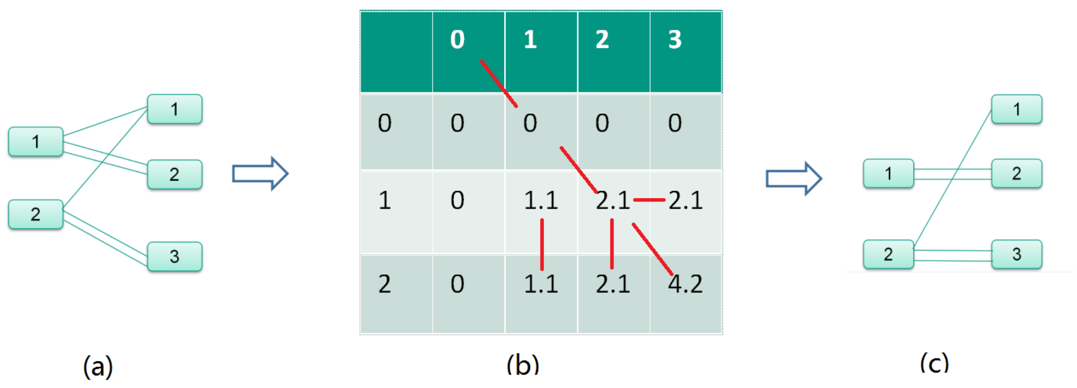
The fist term of the formula is the maximum number of straight segments we can get from l_i and r_j. We can fix one of these sessions and move another session vertically, in order to see when we can get the most straight segments. In that case, the number of straight segments is exactly the first term. The second term measures how similar their relative positions are in two frames. For example, there are both five sessions at time t and $t+1$, when the two sessions are both the first session at their own session sequence, then this term has the biggest value. On the contrary, when one is at first and another is at end, this term would contribute fewer.

In the process of matching session sequences, the match function is used, which is defined as follows:

$$match(i,j) = \begin{cases} 0, & i = 0 \vee j = 0 \\ max\{match(i-1,j-1) + sim(l_i, r_j), match(i,j-1), match(i-1,j)\}, & i \neq 0 \wedge j \neq 0 \end{cases} \qquad (2)$$

This match function describes how we match the i-th session at time t and j-th session at time $t+1$. Clearly it's a dynamic programming, the first part is the initialization, and the second part is update. $match(i-1,j-1) + sim(l_i, r_j)$ means the cumulative similarity if the two sessions are matched; $match(i-1,j)$ means the cumulative similarity if we match the session before i-th session with j-th session; Alike, $match(i,j-1)$ means the cumulative similarity if we match the i-th session with the session before j-th session. With the Max function we can see under which match mode we can get the most similarity.

In the period of matching, we are supposed to make a table to record the result of each matching. But we should notice that in the every field, what is recorded is not the similarity of this matching alone, but the similarity from beginning to this matching, so this is a cumulative process. Such dynamic programming fills a table, which records all matching results along with its origin. Then we trace backwards from the maximum value at end, and get the best match between these two "session sequences". An example is illustrated below.



	0	1	2	3
0	0	0	0	0
1	0	1.1	2.1	2.1
2	0	1.1	2.1	4.2

Figure 13 An example for Alignment:(a)original layout (b)match table (c)alignment result

As the example in Figure 13 shows, the left two sessions happen at some time frame t, and the right three sessions happen at the next time frame $t+1$. As seen in the table the start of the backtrace, namely the maximum is $match(2, 3) = 4.2$, and $match(2, 3)$ comes from $match(1, 2) + sim(l_2, r_3)$. Furthermore $match(1, 2)$ is obtained from $match(0, 1) + sim(l_1, r_2)$. In another word, the left first session l_1 is matched with the right second r_2 session, and the left second session l_2 is matched with the right third session r_3.

After all the sessions have been aligned, all matched sessions should be placed at the same horizontal level, precisely they should be so placed that the term *straight* between them get the maximum value.

5.3.3 Compaction

In order to generate a compact and balanced layout, the wiggle distance and white space need to be minimized. Mathematically, this can be expressed as follows.

$$min \sum_{i=1}^{n_e} \sum_{i}^{n_t-1} (y_{i,j} - y_{i,j+1})^2 + \beta \sum_{i=1}^{n_e} \sum_{i}^{n_t-1} y_{i,j}^2$$

where $y_{i,j}$ means the y-coordinate of the line representing the i-th character at time frame$j$. The first term in this formula represents quadratic summation of wiggle distance. Recall that wiggle distance describes how far the line segment of character i wiggles from time frame j to time frame $j + 1$. The second term represents the width of the whole layout with quoting the absolute y-coordinates of every entity at every time, indirectly shows unnecessary white space. That is, when white space increases, the layout would also become wider, which makes the second term greater.

However, while compacting the layout, the order and alignment of lines which we already achieve, must be preserved. In addition, the basic rule of storyline must also be obeyed: When characters are interacting, that's to say, when their line segments are in the same session, the distance among them must be less than the distance between sessions. Therefore we add some constraints to the minimization problem:

$y_{i_1,j} < y_{i_2,j}$, if $S_{i_1,j} \prec S_{i_2,j}$

$y_{i,j} = y_{i,j+1}$, if $S_{i,j} \leftrightarrow S_{i,j+1}$

$y_{i,j} - y_{i+1,j} = d_{in}$, if $SID(S_{i,j}) = SID(S_{i+1.j})$

$|y_{i,j} - y_{i+1,j}| \geq d_{out}$, if $SID(S_{i,j}) \neq SID(S_{i+1,j})$

Where $S_{i,j}$ is the line segment representing character i at time j. The first one shows the line order constraint. If the line order of $S_{i_1,j}$ is less than the line order of $S_{i_2,j}$, then segment $S_{i_1,j}$ must be placed above $S_{i_2,j}$; The second constraint is line alignment constraint, when the segment i is already aligned between time j and $j + 1$, its y-coordinate must be kept the same, that is, the straightness of the segment must be preserved; The last two constraints describe adjacency constraints. Hier SID means the ID of sessions. That two SID are the same means these segments are in the same session. When two segments are in the same session, their distance must be d_{in}, which is small and defined ahead. When two segments are in the different sessions, their distance must be greater than some threshold value d_{out}. In this way, different sessions can be distinguished clearly. It's clear that This is a quadratic programming problem, which can be solve by the Mosek package[1]. The Mosek package implements the state-of-art interior point method to solve such a quadratic programming problem.

5.4 Applications

5.4.1 Movie Jurassic Park

Three storyline layouts visualizing the movie *Jurassic Park* are generated respectively with the hand-drawn layout, the TM method and the StoryFlow method as in Figure 14. Compared with the hand-drawn layout, the other two layouts have less line crossings or wiggle numbers. In addition, the layout from StoryFlow is more balanced and compact. The Table 2 shows the total number of line crossings and wiggles[5]. In term of line wiggles, the three layouts are very similar, but in term of line crossings, the StoryFlow method is much better.

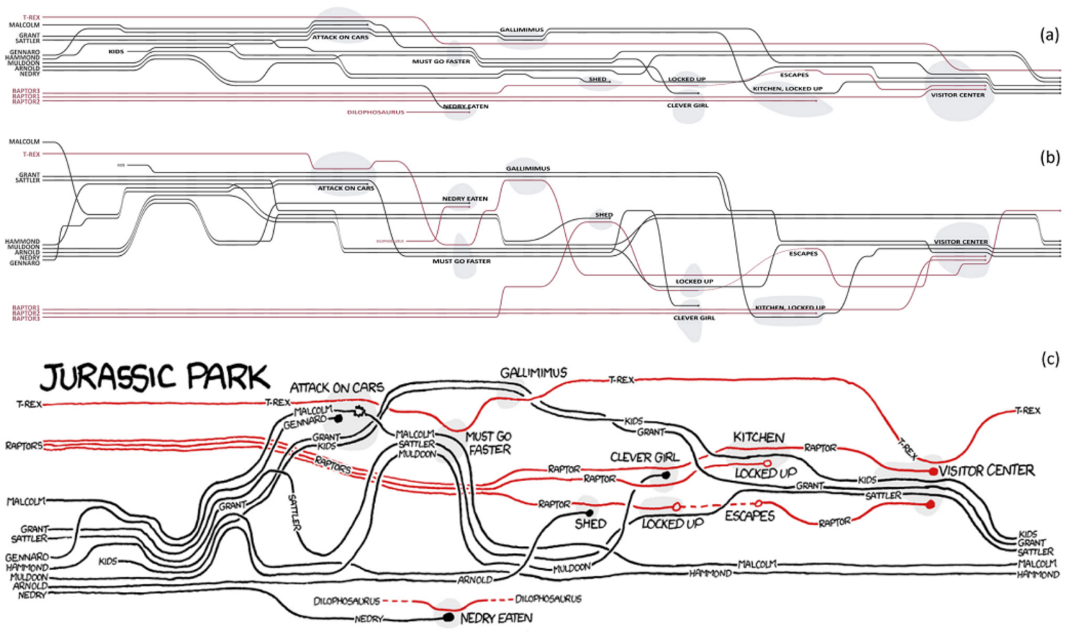


Figure 14 Storyline visualization of the movie *Jurassic Park* with three generation methods: (a)layout by StoryFlow[5]; (b)layout by TM method[9]; (c)hand-drawn illustration from XKCD[6]

Table 2 Comparison among three generation methods concerning line crossing and line wiggle

	StoryFlow	TM	Hand-drawn
Line Crossing	25	80	53
Line Wiggle	110	110	107

Concerning the usability of the visualization, feedback was collected from users with different professions in the movie area. They were firstly demonstrated how the visualization works and then they did free exploration: For film directors, storyline visualization can help understand the story evolution and enables a fast review of the shooting timetable. With its help, film directors can make a better decision on the shot order; Script adaptors can quickly see the overview of the storyline, which helps him decide, which characters or interactions are not necessary and can be deleted, while which characters or interactions can be added to support the story evolution; With storyline visualization, actors can trace their related scenes and see, with whom they will interact. This facilitates their preparation for their performance.

5.4.2 Twitter Data

Storyline visualization can also be applied to the field of data analysis.

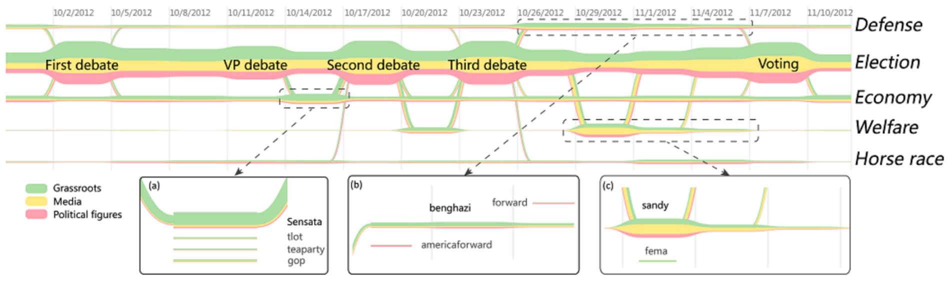


Figure 15 Visualization of twitter data during the 2012 US presidential election, which contains 900 opinion leaders and 63 time frames[5]

Figure 15 describes the change of public's attention on different topics during the 2012 US presidential election constructed based on tweeter data. In this research, about 900 opinion leaders are selected. They are divided into three groups according to their identities: Grassroots, Media and Political figures. All of their tweets during this time are collected, labeled with hashtags and classified into these five main topics. From left to right, their attention changes over time as shown on top, the important phases are also shown in the middle of the chart. On the right are five different topics. Each layer represents a topic. In each layer, there are three colored lines representing opinion leaders from different groups: green for grassroots, yellow for media and pink for political figures. The width of a layer at a time frame means the overall proportion of different opinion leader groups in this topic, namely the degree of attention.

Using the storyline visualization makes a big amount of information visual and easily readable. Firstly, all three opinion leader groups focused mainly on the Election topic. And the grassroots group was also interested in the Economy topic. In the election topic, there are five significant peaks, four of them were related to the presidential debates and the vice presidential debate, the last one was about voting. After the third debate, which was about foreign affairs the Libya issue, grassroots attention on the election topic decreased gradually and many of them switched their focus to the Defense topic. Furthermore, the grassroots mainly talked about the issue related with the hashtag “benghazi”.

From this chart, we can also see that there are two significant focus transitions from the Election topic: On October. 14, a large number of opinion leaders in grassroots group switched their attention from election topic to the Economy. By expanding the economy topic from October 14 to 16 and checking the involved hashtags, it's found the focus transition was caused by Sensata scandal involving Mitt Romney; On October 29, a lot of opinion leaders turned their attention from election to welfare. The most of them are from the media group. By expanding the welfare topic and checking the involved hashtags, it's found the the attention transition was resulted in the hurricane “Sandy”, which hit United States on October. When the hurricane weakened, their focus went back to the election because the election date was approaching.

As we can see, data analysis with storyline visualization assists to understand the change of trend of entities over the time.

5.4.3 Literary Analysis

The Storyline visualization can be also applied in the field of literary analysis. With time going by, most original Latin texts from the classical Roman era are already lost. Only replications of the original texts are handed down till now. However, because of the limitation of technology at that time, errors often appeared in the transcription or early printing process. Therefore, existing replications in the form of manuscripts and early printed editions, vary a lot. But with the help of the storyline visualization, we can reconstruct the original text, which is in the form of the critical edition.

First of all, there are some related concepts, which should be explained to help understand the application. Classic scholar refer to manuscripts and early printed edition as witnesses. Textual differences between witnesses are called variants. A scholar reconstructs the critical edition, by carefully choosing variants from the existing witnesses. The main text in a critical edition is accompanied by a critical apparatus, which includes variants from witnesses that the editor considered important during collation. Each word or phrase in the main text that has an entry in the apparatus is called a lemma.

By applying the storyline visualization method, we get figure 16, the Giarratano's critical edition of the classical Latin poem *Calpurnius Siculus.*

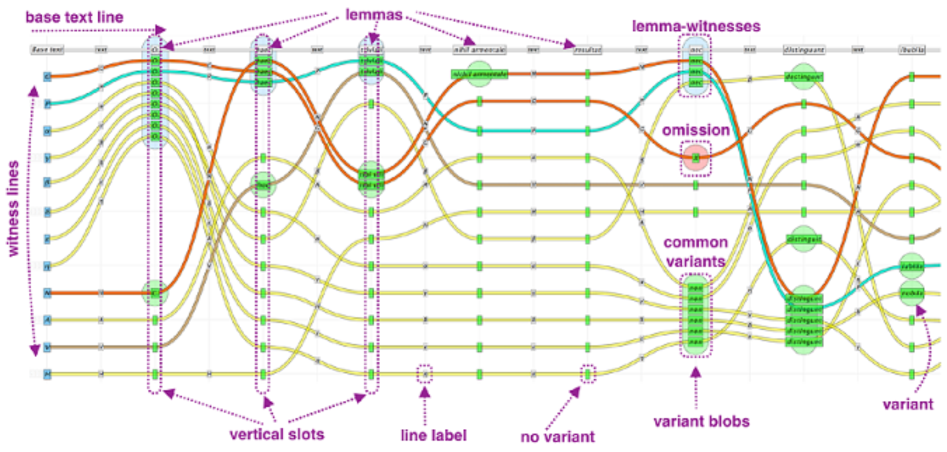


Figure 16 Storyline Visualization of Giarratano's critical edition of the classical Latin poem *Calpurniu Siculus*[7]

Unlike the previous two applications, where the baseline is the time line, which means, the lines going from left to right indicate their evolution with time. In the library analyse, the horizontal axis means the read order. Corresponding, lemmas are placed on the top from left to right as the read order. The first transparent line on the top represents text in a critical edition, other lines represent different witnesses, which are also labeled with their witness name. Under each lemma, the variants of this lemma in every witness are placed vertically. On each line, their label appears between each variants,which provides better traceability and readability of individual lines in the layout. When some witnesses have a common variant, their lines will be grouped together in a blob. Blob's colors green, blue and red represent three variant categories of same as lemma, variant and omission. Empty boxes indicate no textual variation. Like variants, each empty box is vertically aligned with a lemma and horizontally positioned on a witness line, depicting the absence of a variant in a particular witness, for a particular lemma.

With the storyline as a visualization tool, a scholar can clearly see the difference between witnesses under different lemma. It's also helpful to find the best pattern among these witnesses.

5.5 Conclusion

Storyline visualization reveals the relationship among different entities along with their change. As its name implies,it was initially introduced to trace the evolution of stories. In order to generate pleasing layouts, some problems in layouts must be solved: line crossing, wiggle lines, long wiggle distance and much unnecessary white space. Ordering is the first step in the optimization of layouts, where we sort firstly the session locations and then the entities within each location. This step can effectively reduce line crossings. Through alignment, wiggle number of lines would decrease by a large margin. Alignment uses the idea of the longest common subsequence. When each pair of the most alike sessions at adjacent time frames are put in the same horizontal level, lines can be then aligned with effect. At end, compaction should be executed with the aim of reducing the wiggle distance and unnecessary white space. In essence Storyline visualization visualizes the relationship among different entities over time. With this idea it can be used in other fields to provide profits, such as topic data analysis and literary analysis.

References

1 Mosek package. http://www.mosek.com.

2 Alfred V. Aho and John E. Hopcroft. *The Design and Analysis of Computer Algorithms.* Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1st edition, 1974.

3 Peter Eades and Nicholas C. Wormald. Edge crossings in drawings of bipartite graphs. *Algorithmica*, 11(4):379–403, Apr 1994.

4 A. Lempel and I. Cederbaum. Minimum feedback arc and vertex sets of a directed graph. *IEEE Transactions on Circuit Theory*, 13(4):399–403, Dec 1966.

5 Shixia Liu, Yingcai Wu, Enxun Wei, Mengchen Liu, and Yang Liu. Storyflow: Tracking the evolution of stories. *IEEE Trans. Vis. Comput. Graph.*, 19(12):2436–2445, 2013.

6 R. Munroe. Movie narrative charts. http://xkcd.com/657/.

7 Shejuti Silvia. Variantflow: Interactive storyline visualization using force directed layout. 2016.

8 Kozo Sugiyama, Shojiro Tagawa, and Mitsuhiko Toda. Methods for visual understanding of hierarchical system structures. *IEEE Trans. Systems, Man, and Cybernetics*, 11(2):109–125, 1981.

9 Yuzuru Tanahashi and Kwan-Liu Ma. Design considerations for optimizing storyline visualizations. *IEEE Transactions on Visualization and Computer Graphics*, 18(12):2679–2688, 2012.