Institut für Visualisierung und Interaktive Systeme
Abteilung Intelligente Systeme
Universität Stuttgart
Universitätsstraße 38
D–70569 Stuttgart

Diplomarbeit Nr. 2973

Analysis and Visualization of Coreference Features

Stefanie Wiltrud Kessler

Course of Study:	Computer Science
Examiner:	Prof. Dr. Gunther Heidemann
Supervisor:	Dipl.-Inf. Andre Burkovski
Commenced:	October 15, 2009
Completed:	April 16, 2010
CR-Classification:	I.2.7, I.5.1, I.5.4

Contents

1 Introduction **7**

2 Coreference Resolution **9**
2.1 Markables and Links 10
2.2 Coreference versus Anaphora Resolution 10
2.3 Applications for Coreference Resolution 11
2.4 Why is it Hard 12
2.5 Machine Learning for Coreference Resolution 13
2.6 Visualization of Coreference 14

3 Principal Component Analysis (PCA) **17**
3.1 Graphical Explanation of PCA 18
3.2 Computation of Principal Components 19

4 Self Organizing Maps (SOMs) **23**
4.1 Training of SOMs 24
4.2 Visualization of SOMs 25

5 The Project SÜKRE **29**
5.1 Preprocessing 30
5.2 Link Generation 31
5.3 Feature Extraction 32
5.4 Visualization 32
5.5 Coreference Learner 32
5.6 Database Design 33

6 The Coalda Software **35**
6.1 Requirements 35
6.2 Conceptual Design 37
6.3 Architectural Design 39
6.3.1 I/O Libraries 40
6.3.2 ActionLists 40
6.3.3 UI Controls 41

7 Features for Coreference Resolution **43**
7.1 Markable Attributes and Link Features 43
7.1.1 Content and Comparison Features 44
7.1.2 Position and Distance Features 45
7.1.3 Syntactic 46
7.1.4 Grammatical 47
7.1.5 Semantic 48
7.1.6 Apposition 49
7.2 Features Used by Ng and Cardie 49
7.3 SÜKRE Feature Set 51

8 Feature Analysis and Results **53**
8.1 Exploration of the Feature Space 54
8.2 Results of PCA 56
8.2.1 PCA on the Ng Feature Vector Set 56
8.2.2 PCA on the Filtered Ng Feature Vector Set 59
8.2.3 PCA on the Filtered SÜKRE Feature Vector Set 61
8.3 Visualization with Coalda 63
8.3.1 Ng Feature Vector Set Visualized with Coalda 63
8.3.2 Filtered Ng Feature Vector Set Visualized with Coalda 65
8.3.3 Filtered SÜKRE Feature Vector Set Visualized with Coalda 65
8.4 Evaluation of Visualization 68

9 Conclusion **71**

Bibliography **73**

List of Figures

2.1 Example for Coreference (from the ARE Corpus) 9
2.2 Basic Terminology 10
2.3 Text-Based Visualization of Coreference 14
2.4 Chain-Based Visualization of Coreference 15

3.1 Example Data 18
3.2 2D Scatterplot of Example Data 18
3.3 Possible PCs for Example Data 20

4.1 Different Visualizations of a SOM on Data with 6 Variables, drawn with Matlab 26
4.2 U-Matrix, drawn with Matlab 27

5.1 Modules in the SÜKRE Project 29
5.2 Link Generation for Prelabeled Text 31
5.3 ER-Diagram of SÜKRE Database Design 33

6.1 Coalda GUI 38
6.2 Architecture of the prefuse Visualization Framework [HCL05] 39

8.1 Values for Feature PARANUM for the first 250 Feature Vectors 54
8.2 Values for Feature WORD_OVERLAP 55
8.3 PCA for Ng Feature Vector Set, plotted with The Unscrambler 57
8.4 Scores for PC1 for the first 250 Feature Vectors 58
8.5 PCA for Filtered Ng Feature Vector Set, plotted with The Unscrambler 60
8.6 PCA for Filtered SÜKRE Feature Vector Set, plotted with The Unscrambler . 62
8.7 Ng Feature Vector Set Visualized with Coalda 64
8.8 Filtered SÜKRE Feature Vector Set Visualized with Coalda 66
8.9 Filtered SÜKRE3 Feature Vector Set Visualized with Coalda 67

List of Tables

7.1 Attributes for Markables 44
7.2 Features Used by Ng and Cardie [NC02] 50

8.1 Comparison of Assignment and Gold Standard Label 68
8.2 Evaluation Results 69

Chapter 1

Introduction

When people talk about someone or something, they don't always use the full name to refer to this person or thing. For example the person *Bill Clinton* could be referred to by part of his name like *Clinton* or a description like *the president* or a personal pronoun like *he*. All these expressions still refer to the same person. This is called coreference. A human listener performs coreference resolution intuitively to understand what the other person is talking about. We need computers to perform coreference resolution as well, in order to understand a text in natural language, because this is the natural way people talk and write.

There has been a lot of work on coreference resolution using rule-based systems or machine learning systems. But as the task is a difficult one, there are still a lot of unresolved problems. One of the issues in the development of machine learning systems for coreference resolution is the limited amount of training data available. For annotating a text with coreference, full understanding of the text and a basic level of linguistic knowledge is needed. This makes it more difficult to annotate a corpus with coreference information than with other linguistic information. One goal of the Coalda software presented in this work, is to facilitate the annotation of text with coreference information.

The second goal of the software is to enable a researcher to explore the space of the features used for machine learning. This should help the feature engineer to see areas in the data where coreferent data is not clearly separable from other data. With this information he might be able to design new features which better solve that specific problem.

In this work, a new visualization for coreference has been developed and implemented in a software called Coalda. This new visualization is not centered on the text, but tries to visualize the feature space. As it is impossible to directly visualize a high-dimensional feature space, an indirect way has to be found. In Coalda, the feature space is visualized by training a Self Organizing Map (SOM) and visualizing this SOM. The software allows the user to interactively explore the map, label feature vectors with coreference information and recalculate the map with different settings.

This diploma thesis is part of the DFG-project SÜKRE. The development of the mentioned interactive visualization is one of the main goals of the project. Other research topics are the development of new semantic and global features. Semantic features attempt to capture

semantic information about the compatibility of the two markables that form the link. Global features are features that work on a partition of links that belong to one discourse entity.

The remainder of this document is split into three main parts. First, the theoretical basis of this work is explained in the first three chapters.

Chapter 2 contains basic definitions for coreference resolution as well as a small overview of the challenges of the problem, applications of coreference resolution and existing visualization approaches.

Chapter 3 provides a very short explanation of the Principal Component Analysis (PCA). The PCA is used later for the analysis of coreference feature vectors in chapter 8.

Chapter 4 describes the training method for Self Organizing Maps (SOMs) along with methods for the visualization of SOMs. A SOM is used in Coalda for the visualization of the feature space.

In the middle part, the implementation part of this thesis, Coalda and other modules in the SÜKRE project are presented.

Chapter 5 describes the context of this work as part of the DFG-project SÜKRE. The main research topics of the project are presented along with a brief description of the software components that are used by the visualization software.

Chapter 6 contains details about the implementation of Coalda. This includes requirements, a conceptual design and an overview of the architectural design.

Finally, in the third part, the developed feature sets are described and used in the evaluation of the new visualization.

Chapter 7 discusses different features for a machine learning coreference resolution system. Two different feature sets are defined. These sets will be used for the evaluation.

Chapter 8 contains the evaluation of the feature sets and the visualization. A PCA has been applied to both sets to explore the structure of the data. The data has then been visualized with Coalda and the resulting visualizations are evaluated.

Throughout this document, code, shell commands and file system paths are written in `fixed font`. Example sentences or expressions in natural language are written in *italic*.

Chapter 2

Coreference Resolution

Coreference resolution is an important task in natural language processing (NLP). Many applications rely on it to "understand" a text written in natural language. Coreference is something widely used in natural language texts, but it is very hard for computers to understand. The definition is simple: if two expressions in a text refer to the same discourse entity, they are coreferent.

In some cases coreference is simple to determine. It is easy to detect that the expressions *(Jordan King Hussein)*$_1$ and *(Hussein)*$_3$ in the example in figure 2.1 are coreferent, because one is a substring of the other. On the other hand intuitively two names that are not the same (like *Hussein* and *Clinton*) can never be coreferent.

For human readers the connection of *(the president)*$_4$ with the previously mentioned entity *(U.S. President Bill Clinton)*$_2$ is obvious. Although if expression 2 would only consist of *(Bill Clinton)*$_2$ we (as human readers) would need to know that he is (was) the president or the text would need to contain that information somewhere.

To resolve a pronoun like *(his)*$_5$ we also need the context to decide whether it refers to *Hussein* or *Clinton*.

> *The White House said on Monday* ***(Jordan King Hussein)***$_1$ *would meet* ***(U.S. President Bill Clinton)***$_2$ *in Washington on April 1 and denied that the Middle East peace process was unravelling.* ***(Hussein)***$_3$ *had been scheduled to meet* ***(the president)***$_4$ *on March 18, but* ***(his)***$_5$ *visit was postponed after a Jordanian soldier shot dead seven Israeli girls near the Israel-Jordan border on March 13 and after* ***(Clinton)***$_6$ *had knee surgery on March 14.*

Figure 2.1: Example for Coreference (from the ARE Corpus)

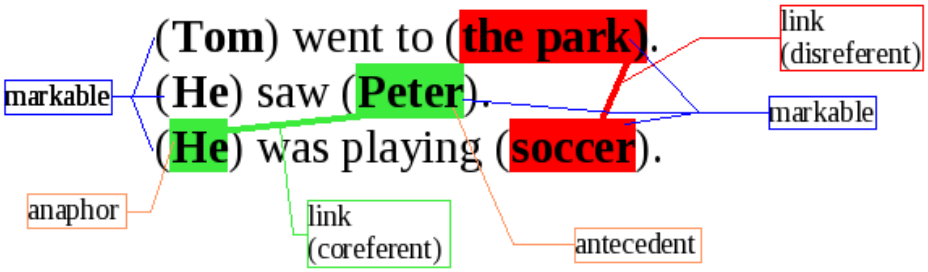


Figure 2.2: Basic Terminology

2.1 Markables and Links

An expression that might be coreferent to another expression is called a **markable**. The same thing is often called mention or coreference element in literature. All markables we are going to consider for coreference will be noun phrases. A **noun phrase** is a group of words in a sentence that can be replaced by a (pro-)noun. For example *the house* or *my small yellow old house that my father built when I was a kid* can be replaced by the pronoun *it*.

Often a markable A is coreferent with another markable B and B is coreferent with yet another markable C. Such sets of markables that refer to the same entity are called **coreference chains**.

The opposite of coreferent is **disreferent**. Two expressions that do not refer to the same discourse entity, but to two distinct entities, are disreferent.

A pair of markables that can be co- or disreferent is called a **link**. This link has a number of associated link features. A link can have a **label** that contains information about co-/disreference of that link. The text in figure 2.2 has two labeled links.

The first markable in a link (in order of appearance in the text) is called **antecedent**, the second one is called **anaphor**. We assume that normally a link is created by linking a markable (the anaphor) with other markables which occurred earlier in the text.

2.2 Coreference versus Anaphora Resolution

Coreference resolution is closely related to anaphora resolution, but it is not the same. Anaphora resolution treats only expressions that depend on another expression for interpretation. The task is to find out what that other expression is. Anaphora is not a symmetric relation (for example *he* may depend on *Peter* for interpretation, but *Peter* will never depend on the pronoun *he*).

The task of coreference resolution is to find expressions in the text that refer to the same discourse entity. This includes expressions that do not depend on other expressions for interpretation. Coreference is an equivalence relation, that means it is reflexive (every markable can be considered coreferent with itself), symmetric (if *he* refers to the same entity as *Peter*, of course *Peter* refers to the same entity as *he*) and transitive (if *he* is coreferent with *the student* and *the student* is coreferent with *Peter*, also *he* and *Peter* are coreferent).

Not all coreferent links are anaphoras, for example *Queen Elizabeth* and *The Queen Mother*. Neither expression depends on the other for interpretation, so they are not anaphoric. But they refer to the same entity and therefore they are coreferent. Also coreference relations can occur across documents., but anaphora cannot.

But there are also anaphoric relations where anaphor and antecedent are not coreferent. For example in the two sentences *The boy entered (the room). (The door) closed automatically.* the expression *The door* is an anaphor, because it depends on *the room* for interpretation. But these two expressions are not coreferent, because they don't refer to the same entity.

More details on the distinction between anaphora and coreference resolution can be found in [Ng02] and [Ela05].

2.3 Applications for Coreference Resolution

When writing a text in natural language, we nearly always use different expressions to refer to the same entity. Also the expressions used to refer to an entity may contain new information about that entity. In order to still be able to get all the information about one entity and for general understanding of a text, we have to know which expressions refer to which entities. It is also important for some of the applications to consider cross-document coreference to collect information about an entity from different sources.

A selection of NLP tasks where coreference resolution is important follows, for more information and references see [Ng02].

Question Answering is the task of answering a question in natural language based on a corpus in natural language. We need coreference resolution to be able to connect a sentence like *He was elected president in 1993* with the markable from the previous sentence *Clinton* in order to answer a question like *Who was president in 1993*.

The goal of Text Summarization is to produce a shorter version of a given text in natural language. This shorter version still has to contain the important facts about the main topic in the text. To solve that task, we need to know what the main topic of the text is. Typically this would be the entity most expressions in the text are coreferent with.

Information Extraction automatically extracts information from a given text in natural language and organizes it to machine readable structured information. To be able to add the information from different sentences to the correct entity and determine how many entities or events are talked about in a text we need coreference resolution.

Machine Translation is the automatic translation from a text in one natural language to another natural language. Anaphora resolution is needed if there are differences in the language with regard to a feature of the antecedent. To be able to translate *Das ist meine Brille, sie ist kaputt* from German, where *Brille* is feminine and singular, to English *These are my glasses, they are broken* where *glasses* are neuter and plural, we need to know what *sie* refers to. The naive translation of *sie* with *she* as in *These are my glasses, she is broken* would not be understandable.

2.4 Why is it Hard

For coreference resolution many different sources of knowledge are needed. This ranges from morphological information like Part-Of-Speech tag (POS tag) to semantic information like semantic class. Every knowledge source has its level of correctness and may introduce wrong information. Some knowledge is very hard or expensive to compute (like semantical information) and has a high error level.

Even if some linguistic constraints indicate that two markables cannot corefer, sometimes they can. For example *the assassination of her bodyguards* (singular) can corefer with *these murders* (plural) or *Das Mädchen* (*the girl*, neuter) with *sie* (*she*, feminine) [HKS09].

Often an expression refers to something that has been in focus for the last sentence. *Tom went to the park. He saw Peter. He was playing soccer.* In the second sentence there is a focus shift from *Tom* to *Peter*, so the pronoun *he* in the last sentence probably refers to *Peter* (also because we have no information about *Tom* playing soccer earlier). But if the last sentence would be *He was happy* it would not be obvious even to a human reader. Focus is hard to determine. Centering theory [Ela05, Ng02] tries to track entities in focus, but this is a very hard problem in itself.

Sometimes a lot of world knowledge is required to correctly determine the antecedent of a noun phrase. For example *(The boys) were kidnapped by (masked men). After (they)*$_1$ *blindfolded ... After (they)*$_2$ *were released ....* The first *they* refers to *masked men*, the second to *the boys*. This text is understandable for us humans only because we know that kidnappers blindfold their hostages while the hostages are the ones to be released [Ela05]. World knowledge is very difficult to formalize and it is very hard to determine what exactly is the information we need. This is also the case when coreference depends on the time a text was written or similar information. Whether *Obama* and *the president of the United States* corefer, depends on the result of the election and the time the text was written.

Some categories of noun phrases are harder to resolve than others. Proper names are easiest because coreference can be very well recognized with string matching and alias matching. Common nouns are harder, especially definite noun phrases (for example *the door* or *the president*). Not all definite noun phrases are anaphoric, some already uniquely identify an entity. That means the first problem is to determine the anaphoricity of a noun phrase. The distance from a definite noun phrase to its antecedent can be greater than the distance of a

pronoun to its antecedent. Definite noun phrases may also refer to entities not in focus while pronouns mostly do [Ela05, SGCR09].

Additionally, there are markables which are not coreferent with any other markable that has occurred in the text onto this point. These "singletons" can be markables that are self-explaining or the only mention of an entity in the text. A second kind of singletons are markables that are the first mention of the entity in the text (if we consider only links of markables with markables that occurred earlier in the text). Adding anaphoricity determination to a system would help to save computation time, because for singletons no antecedents have to be searched. It would also prevent errors, because any antecedent to a non-anaphoric noun phrase is certainly wrong.

To determine anaphoricity of a noun phrase is a difficult problem. Ng and Cardie [NC02] use a feature that indicates the anaphoricity of a markable. This feature is calculated by a separate anaphoricity classifier. On the other hand Luo [Luo07] argues, that the determination of anaphoricity is a part of coreference resolution. He uses two models, where one determines the most probable antecedent for an anaphoric markable and the other determines whether a markable is a singleton.

In the training of classifiers for coreference it is important to keep in mind that coreference is a very rare relation. The vast majority of links are not coreferent. The MUC-6 corpus for example contains only 2% positive instances [Ng02].

2.5 Machine Learning for Coreference Resolution

Machine learning algorithms are often categorized in two basic categories, supervised learning and unsupervised learning. Supervised learning algorithms work on data that has been labeled with the class information. From this data the algorithm creates a classifier that can predict the class of a new input. Unsupervised learning works on unlabeled data and attempts to find structures that are inherent in the data itself.

Applying a supervised learning algorithm to the coreference resolution problem means to classify every link between two markables as coreferent or disreferent. Such a link-based classification system is the decision tree classifier used by Soon [SNL01] and later Ng and Cardie [NC02].

But coreference resolution can also be viewed as a clustering task. Clustering is an unsupervised learning method. The inherent structure of the data that needs to be found is the group of markables that refer to each entity mentioned in the text. Cardie and Wagstaff [CW99] cluster markables using a manually defined distance metric over a set of link features.

For all of these machine learning algorithms a set of features is needed. The features are used to measure the similarity of the two markables forming a link or to calculate the probability that this link is coreferent. A subset of the feature set used by Ng [NC02] has been re-implemented for this work and is explained in detail in section 7.2.

The White House said on Monday ***(Jordan King Hussein)***$_1$ *would meet* ***(U.S. President Bill Clinton)***$_2$ *in Washington on April 1 and denied that the Middle East peace process was unravelling.* ***(Hussein)***$_1$ *had been scheduled to meet* ***(the president)***$_2$ *on March 18, but* ***(his)***$_1$ *visit was postponed after a Jordanian soldier shot dead seven Israeli girls near the Israel-Jordan border on March 13 and after* ***(Clinton)***$_2$ *had knee surgery on March 14.*

Figure 2.3: Text-Based Visualization of Coreference

Coreference is a transitive relation. If a markable A is coreferent with B and B is coreferent with C, then A also has to be coreferent with C. Properties like this cannot be enforced by any system that works on links only. Other restrictions for a valid entity are for example that an entity cannot consist only of pronouns. Apart from not using links where both markables are pronouns (which creates other problems) such restrictions cannot be expressed only in link-features [HKS09].

To deal with such problems, recent systems for coreference resolution work on clusters, not links [CGB08]. This allows to formulate "global" features that take all available information about an entity into account.

The SÜKRE project uses a link classifier as well as a cluster-based classifier as described in section 5.5.

2.6 Visualization of Coreference

There are different methods of visualization for coreference. Visualization can center on the text, on coreference chains or on the feature space.

The most intuitive visualization of coreference is text-based visualization. The text is shown, the markables are marked and a coreference chain is connected by lines (as in MMAX [MS01]) or the entity is identified with numbers (as in CorefDraw [HBTM01]) or colors (as in GATE [CMBT02]). This does not show the features or the feature space and what links are similar to others. It is also limited by the paper and the number of coreference lines/colors one can understand. This makes it difficult to analyze large chains or inter-document coreference or many links at ones.

Figure 2.3 contains the example sentence used earlier in this chapter visualized with text-based visualization. Every markable is denoted by parenthesis, and the entity it is referring to is identified with a unique ID. The Coalda Software offers a simple text-based visualization for the feature vectors associated with one node of the SOM only, to make the text of the links available to the user.

A step in the direction of more abstract visualization is chain-based visualization. The chains from the example text are visualized in 2.4. Coreference chains are shown and the user can click on the chains to see the markables that form the chain. This representation only serves

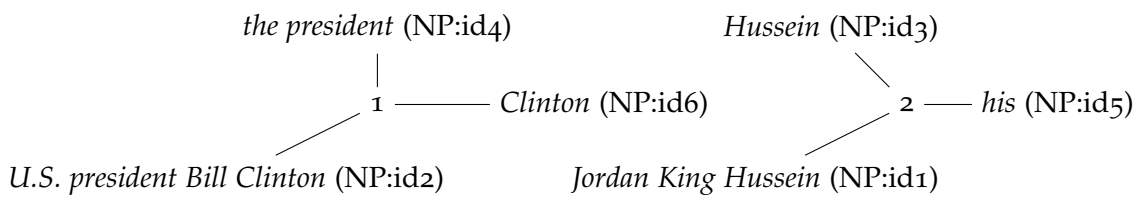


Figure 2.4: Chain-Based Visualization of Coreference

to visualize and navigate the result space, not the feature space. Additionally, the markables are shown without context which makes it hard to judge if a link is correct or not.

Witte and Tang [WT07] use this kind of visualization in their graphical representation of results obtained by a machine learning coreference resolution system. The visualization can be explored by users for browsing of recognized coreference chains and error detection and analysis. They represent coreference chains by Topic Maps and OWL Ontologies and use existing tools for the visualization of document navigation. For the error analysis a manually defined ontology is used. The extracted chains are compared to a gold standard and put in several classes in the ontology like 'correctChain' or 'hasNPmissing'.

In feature space visualization the visualization does not center on the text, but on the feature space. As it is impossible to directly visualize a high-dimensional feature space, an indirect way has to be found. In the Coalda software the feature space is visualized by training a SOM and visualizing this SOM (see chapter 6).

Chapter 3

Principal Component Analysis (PCA)

In this chapter a very short explanation of the Principal Components Analysis is given. The PCA is used for the analysis of coreference feature vectors in chapter 8.

The idea of PCA is to reduce the dimensionality of a data set and at the same time keep most of the information that is in the set. To achieve this, the original variables are transformed into a new coordinate system that is spanned by so-called principle components (PCs). The PCs can be thought of as ”hidden variables” that cannot be directly observed, but are responsible for the structure of the data. The number of PCs that are needed to explain the structure of the data is smaller than the number of variables in the original data set.

These PCs are ordered by the amount of variation of the data they explain. All PCs are uncorrelated with each other, unlike the original dimensions. Geometrically the first PC can be thought of as the vector on which the projection of the data causes smallest loss of information (smallest sum squared error). After the projection on this axis, the data has maximum variation. The second PC has the second-smallest loss of information while being orthogonal to the first PC and so on.

The data can be plotted in the new coordinate system of the first two or three PCs. It is now easier for humans to see the inherent structure of the data and find clusters or anomalies. Additionally, correlations between variables can be found. The PCA will also show which variables are important for the model.

The PCA was first introduced by Karl Pearson in 1901. In 1933 it was re-invented by Harold Hotelling. But the method became widely known (and used) only when computers allowed the computation of principle components without a lot of effort. It is today often used as a tool in exploratory data analysis in many areas.

x	y	class
1	2	a
2	2	a
5	8	a
7	7	a
5	3	a
13	16	b
11	14	b
14	15	b
16	12	b
18	19	b

Figure 3.1: Example Data

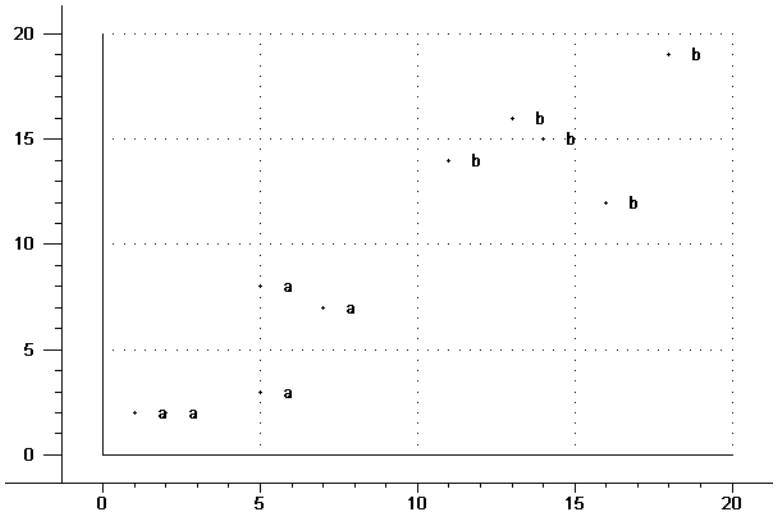


Figure 3.2: 2D Scatterplot of Example Data

3.1 Graphical Explanation of PCA

To show with an example[1] what the PCA can do, we take the data in table 3.1. This data is only two-dimensional, so we are able to plot even the original data. In the plot in figure 3.2 we see that the data forms two clusters, the two classes that are labeled with *a* and *b*.

The PCA tries to draw a line through the data so that the difference between the clusters becomes apparent. The best line to achieve this is a line, where the distance from each data point to the line is minimal. This distance is the information we lose when we use the projection on the line as the only information.

If we draw a line just at random, as for example the line in 3.3(a), it doesn't give the desired result. The projections of all data points are very close together and we cannot see the two clusters we expect to see. The sum of all distances from all points to this line is big, this means that a lot of information is lost.

The line in 3.3(b) fits the data much better than the first line. It does minimize the distance from all points. It also maximizes the variation in the data, the data points are far away from each other and we can clearly distinguish the two clusters. This line is the first principal component of the data. We have reduced the dimensionality of the original data set from two dimensions to one and still kept most of the information. The information we have lost is not relevant for us in understanding the structure of the data.

To find a second principal component, we have to find a line orthogonal to the first one. In case of the example data, there is only one possibility. If we had a data set with more dimensions, we would again have to choose a line orthogonal to the first line, that maximizes variation and minimizes the information loss.

[1] This example can be found in more detail in [Kes07]

The coordinates of the original data can now be transformed to the new coordinate system and the data can be plotted in this new coordinate system.

3.2 Computation of Principal Components

Mathematically doing a PCA corresponds to finding the eigenvectors of the covariance matrix of the data. The first PC is the eigenvector with the largest eigenvalue, the second PC with the second largest and so on.

Before doing a PCA the data has to be centered. To center the data, the mean of the variable is subtracted from every variable. The mean μ_X of the variable X where the sample i has the value X_i is defined as

$$\mu_X = \frac{\sum_{i=1}^{n} X_i}{n}$$

and denotes the ”middle point” of the data set.

The next step is to calculate the covariance matrix of the data. The variance $var(X)$ is a measure of the spread of data in a variable. It is the mean ”distance” (deviation) of the samples from μ_X. The formula is for discrete samples

$$var(X) = \frac{\sum_{i=1}^{n}(X_i - \mu_X)^2}{n}$$

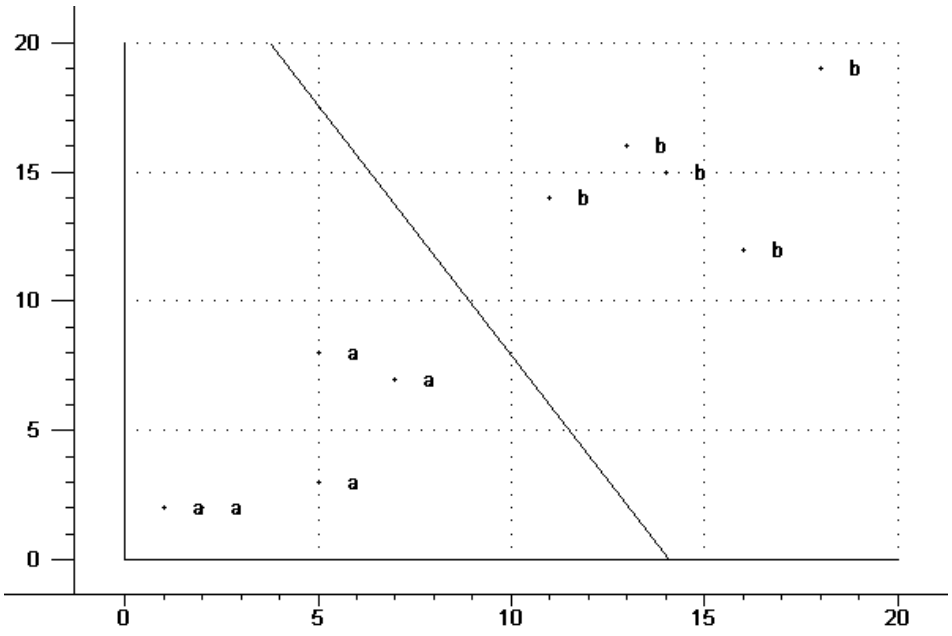
Mean and variance are one-dimensional. To find out how much two variables change with respect to each other, the covariance of these two variables can be calculated. The covariance of a variable with itself is always its variance. If the covariance is positive, the variables are correlated and increase together. If the covariance is negative, the variables are negatively correlated, if one variable decreases the other one increases. If the covariance is zero, the variables are independent of each other. The formula for calculating the covariance of variables X and Y is

$$covar(X, Y) = \frac{\sum_{i=1}^{n}(X_i - \mu_X)(Y_i - \mu_Y)}{n}$$

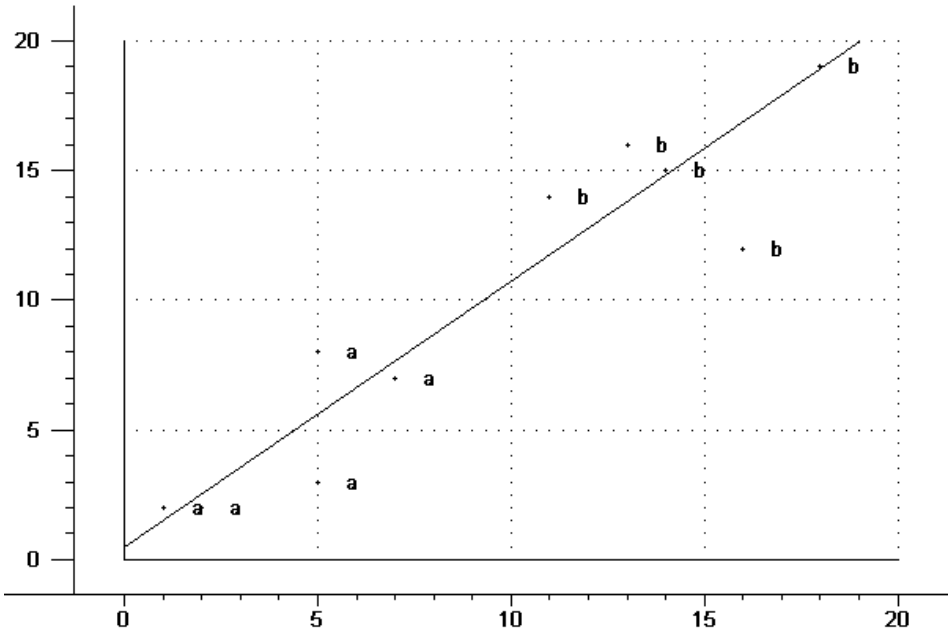
The covariances between all variables are often displayed in a matrix, this is called the covariance matrix. The covariance matrix is a square and symmetric matrix.

Once the covariance matrix of the data set is calculated, we have to find the eigenvectors of this covariance matrix. A vector is an eigenvector of a given matrix, if the multiplication of the vector with the matrix results in an integer multiple of the original vector. This means that the vector does not change its direction in the vector space, only its magnitude. Eigenvectors can only be found for square matrices. All eigenvectors are orthogonal to each other. Often eigenvectors are normalized to have a length of one.

Every eigenvector has a corresponding eigenvalue. This is the amount by which the original eigenvector is scaled by the multiplication with the matrix.



(a) Line where Information Loss is Big



(b) Line where Information Loss is Small

Figure 3.3: Possible PCs for Example Data

Mathematically this means a vector $\vec{v}$ is an eigenvector of matrix A if

$$A\vec{v} = \lambda\vec{v}$$

where λ is the eigenvalue corresponding to $\vec{v}$.

The calculation of eigenvectors is done by numerical methods. The values v_i of the eigenvector $\vec{v}$ in the original dimensions are called loadings.

The eigenvectors are then ordered by their eigenvalues. The eigenvector with the biggest eigenvalue is the first principal component, the one with the second biggest eigenvalue the second principal component and so on.

Finally, we only need to convert the coordinates of the data in the original space to coordinates in the new eigenvector space. To do this, we put the eigenvectors we want to use in the eigenvector matrix. In this matrix, the eigenvectors are the columns. The first column contains the eigenvector with the biggest eigenvalue, the last column the one with the smallest eigenvalue.

We map the data points onto the eigenvectors by multiplying the transpose of the eigenvector matrix with the data matrix. The data in the new coordinate system can then be plotted. The values of the data points in every PC are called scores.

To summarize, to do a PCA for the data matrix D with p original variables we have to

1. Calculate the mean for each of the p original variables and center the data.
2. Calculate the covariance matrix for all original variables.
3. Find the eigenvectors of the covariance matrix.
4. Sort the eigenvectors by their eigenvalues and select $k \ll p$ eigenvectors we want to use for display.
5. Calculate the new coordinates of the data points in the eigenvector space.
6. Plot the data in the new coordinate system.

Mathematical proofs of the properties of PCA can be found in [Jolo2]. For more practical applications consult [Keso7].

Chapter 4

Self Organizing Maps (SOMs)

This chapter will introduce Self Organizing Maps which are used in this work for the visualization of coreference links, as explained in chapter 6.

A Self Organizing Map (SOM) is a neural network (non-linear unsupervised learning algorithm) first described by Teuvo Kohonen in 1982 [Koh82]. It is a method to map a high dimensional feature space $\mathbb{R}^{in}$ onto a low dimensional output space $\mathbb{R}^{out}$ (for visualization typically three dimensions or less). A SOM preserves the topological properties of the feature space, so that feature vectors that are close together in the input feature space will be close together as well in the output space.

Motivation for this type of neural networks comes from the human brain, where the information from neighbouring visual inputs is processed by neighbouring regions in the visual cortex [Roj96].

A SOM consists of a grid of map units (also called nodes or neurons). This grid has the dimension of the output space $\mathbb{R}^{out}$. For a two-dimensional output space the grid is usually hexagonal or rectangular. Every node has a fixed place on this grid and fixed neighbours. These neighbourhood relations are never changed.

Additionally every node has a weight vector of the same dimension as the input data vectors ($\mathbb{R}^{in}$). These weight vectors are adapted in training so that the nodes of the SOM change their place in the feature space to get closer to the data points. Also the weight vectors of the neighbouring nodes are adjusted, so that nodes near to each other on the grid are also near to each other in the feature space.

Because the nodes change their places in feature space to get closer to the data points, at the end of training most nodes will be in areas where there are many data points. Only a few nodes will be in areas where there are few data points. This means, not only the topology of the data is approximated by the SOM, but also the data density distribution.

4.1 Training of SOMs

For the training algorithm of SOMs, two functions that monotonically decrease with training time t are required. One is the learning coefficient $\alpha(t)$, the other is the neighbourhood function $h_{ij}(t)$ which describes the neighbourhood of node j. Neighbourhood is defined on the neighbourhood relations of the lattice. Both, learning coefficient and neighbourhood function, get smaller with training time.

A typical choice for the neighbourhood function is $h_{ij}(t) = exp(-\frac{||\vec{w}_i - \vec{w}_j||^2}{2\sigma(t)^2})$ where $\sigma(t)$ is the radius of the neighbourhood and decreases with time (Gaussian function).

The learning coefficient $\alpha(t)$ controls how much the current input influences the training. At the beginning of the training it is big. Then it decreases with time.

We also need a distance metric for the feature space $\mathbb{R}^{in}$. This is usually the euclidean distance, but could be any distance metric.

At the beginning of the training, all weight vectors are initialized. If there is no prior knowledge about the data, the weights are initialized at random.

The training algorithm of a SOM has the following steps:

1. Chose a random input vector $\vec{x}$ (with dimension of the feature space $\mathbb{R}^{in}$) from the data points.
2. The input $\vec{x}$ is given to all nodes of the SOM. Calculate the distance $d(\vec{w}_i, \vec{x})$ for all nodes i and take the node k with the minimum distance (the node nearest to $\vec{x}$). It is called Best Matching Unit (BMU).
3. Update all nodes, the update formula for a neuron i with weight vector $\vec{w}_i$ is

   $$\vec{w}_i(t+1) = \vec{w}_i(t) + h_{ik}(t)\alpha(t)(\vec{x} - \vec{w}_i(t))$$

   where k is the BMU. This means, that the node k is now pulled in the direction of the input $\vec{x}$. The neighbours are also pulled in that direction. Nodes nearer to the BMU are adapted more than nodes further away in the neighbourhood. Weight vectors of nodes outside the neighbourhood are not changed at all.
4. Increase t (which decreases $\alpha(t)$ and $h_{ij}(t)$).
5. Repeat for a new input vector until limit of iterations is reached or other $\alpha(t)$ is smaller than a specified threshold value.

At the beginning of training, when $\alpha(t)$ and $h_{ij}(t)$ are still big, the net is only roughly adjusted to the data. Once the values get smaller, the weight vectors of single nodes are fine tuned.

Training can also be done in batch mode, where the whole data set is presented to the net before any adjustments are made. The new weight vector of a node is then calculated with the adjustments that would have been made for all the nodes that have this node as a BMU

[VHAP00]. The advantage of this processing is, that it is deterministic. For the same data set the resulting map is always the same. This makes the result of the training reproducible.

A lot of literature on SOMs exists. For a short explanation with some examples for the application of SOMs consult [ZG93] or [Imo08]. For a more in depth discussion use [RMS90].

4.2 Visualization of SOMs

Visualizing the SOM in output space is easy. The output space has a small dimension and the neighbourhood relations are fixed. The result would be a grid and every map with the same topology looks the same. But what we want to visualize is the feature space as represented by the SOM. This feature space is still high dimensional, and so it is hard to visualize.

The easiest way of visualizing the feature space is to show the weight values of the SOM map units in every dimension in feature space on component planes (see figure 4.1(a) on the next page). For a limited number of dimensions it is also possible to show the different values in these dimensions in one map unit with barchars or other statistics (see figure 4.1(b) on the following page). In the same way one can use colors or numbers to visualize the number of feature vectors having one map unit as BMU.

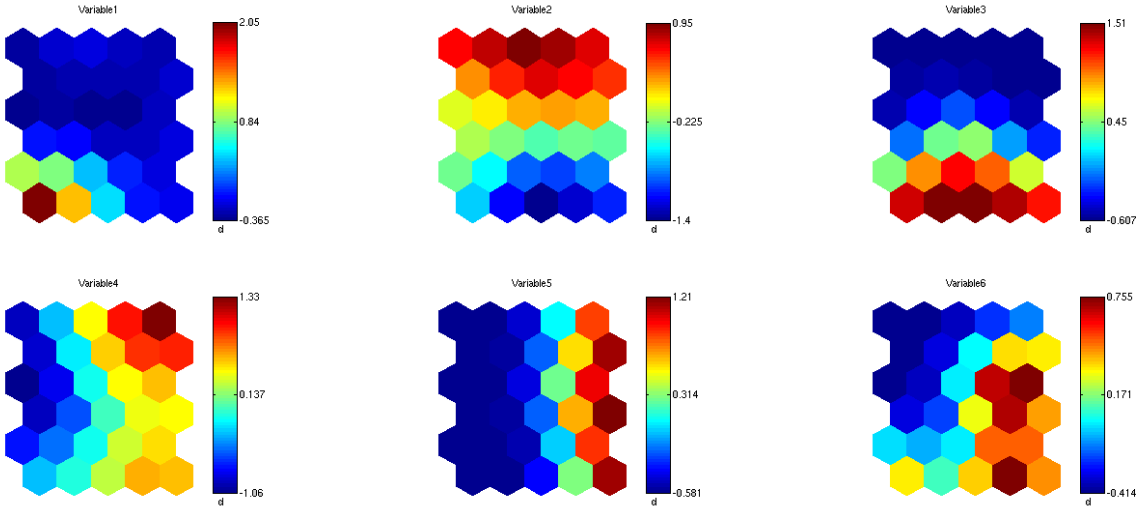
One can also visualize the similarity of nodes to each other by coloring [Ves99]. Similar nodes get similar colors. If done in a single dimension this gives the component plane. For two or three dimensions one can take the dimensions as two/three dimensions of the RGB-space. For more than three dimensions the weight vectors need to be projected on the RGB-space.

SOMs can also be visualized by a projection in 2D or 3D. The two or three dimensions to be shown can be dimensions of the feature space or the first two/three principal components. An example of a projection in 2D is shown in figure 4.1(c) on the next page.

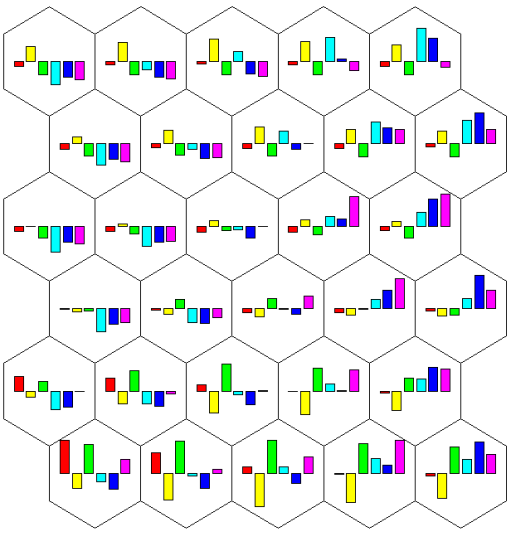
The most common visualization for SOMs is the U-matrix (unified distance matrix) [US90]. The U-matrix displays the distance of a node to its neighbour nodes. To prevent the vector distance from being influenced by a dimension with high values, the values of all dimensions should be normalized.

The U-matrix is shown in a grid with the same topology as the map. If the map is hexagonal, the U-matrix will contain hexagonal cells. There is one cell for every node at the place that node has in the input space. Additionally for every edge between two neighbouring nodes there is a cell at the location of the edge. In a hexagonal grid a cell for a node i will be surrounded by six cells for the edges.

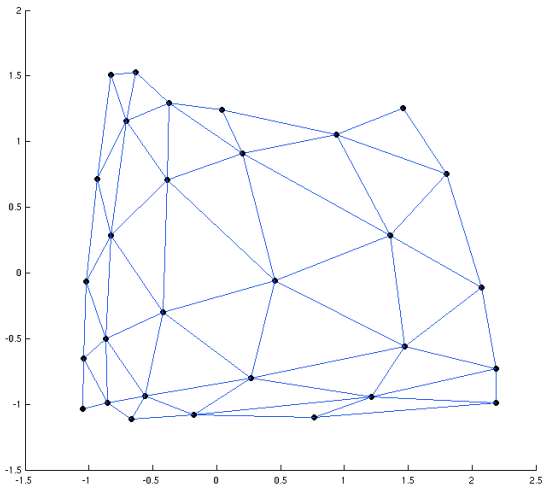
For every cell the U-matrix value is calculated. The U-matrix value for an edge is the distance in feature space of the two nodes of that edge. Any distance metric may be chosen, normally the Euclidean distance is used. The U-matrix value of a node can be set arbitrarily, but is normally set on the mean value of all edges from this node. The result is an overview of the distance structure of the map in feature space.



(a) Component Planes



(b) Barcharts



(c) SOM Projection in 2D

Figure 4.1: Different Visualizations of a SOM on Data with 6 Variables, drawn with Matlab

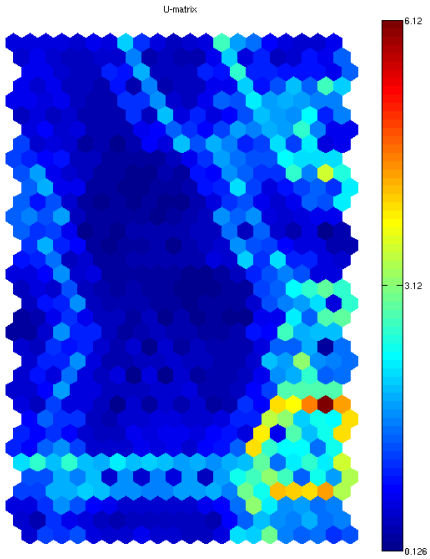


Figure 4.2: U-Matrix, drawn with Matlab

The U-matrix value of a cell is visualized with different colors. Figure 4.2 shows an U-matrix as generated by Matlab. Blue color means low U-matrix value, red color means high U-matrix value.

So the blue areas are nodes that are close together, that means they form a cluster. Nodes with a high U-matrix value (red) are very far away from their neighbouring nodes, the long (red) edges are the cluster boundaries. As the SOM preserves the topological properties of the original data, this means that there is also a cluster respectively cluster boundary in the original feature space.

In the U-matrix it is impossible to see how many feature vectors are near a specific node or how close the feature vectors are. The P-matrix [Ult03a] shows the density of the data around the nodes. It can be any density estimation, normally the Pareto Density Estimation (PDE) is used. The PDE calculates the number of feature vectors in a hypersphere (Pareto sphere) around a node. At every node the density of data around this node is displayed in the P-matrix. Neurons with a large P-matrix value are located in regions of the feature space with high density, nodes with small P-value are in regions with few data points. The P-matrix visualizes the data density structure of the data.

It is possible to combine U-matrix and P-matrix in the U*-matrix [Ult03b]. The idea is that in areas with high data density the distances between nodes should be valued less than in areas with low density, where they are always cluster boundaries. The U*-matrix value for edges is the same as the U-matrix value. The U*-matrix value $U^*(n)$ for the node n is $U^*(n) = U(n) \cdot \frac{P(n)-mean(P)}{mean(P)-max(P)}$, where $U(n)$ is the U-matrix value, $P(n)$ is the P-matrix value and $mean(P)$ and $max(P)$ are the mean and maximum P-values.

Chapter 5

The Project SÜKRE

The project "SÜKRE" is a project funded by the Deutsche Forschungsgemeinschaft (German Research Foundation). "SÜKRE" stands for "**S**emi**ü**berwachte **Kor**eferenz **E**rkennung" ("semi-supervised coreference resolution"). The participants are the Institute for Natural Language Processing and the Institute for Visualization and Interactive Systems of the University of Stuttgart. The project has started in September 2009 and will go on for two years. This diploma thesis is a part of the SÜKRE project.

The goals of the project are detailed in [HKS09]. One main topic is the development of an interactive visualization for coreference features. This visualization should facilitate the semi-supervised annotation of large amounts of training data. This training data can then be used to explore new features. Features that will be explored in the course of the project are semantic features and global features. Semantic features attempt to capture semantic information about the compatibility of the two markables that form the link (see section 7.1.5). Global features are features that work on a partition of links that belong to one discourse entity.

As the topic of this work is in the visualization of link features, this chapter will only give an overview about the part of the project that deals with link features. It will not talk about the global features although they are a big research topic of the project. But a visualization similar to the one developed here could be used for visualization of global features.

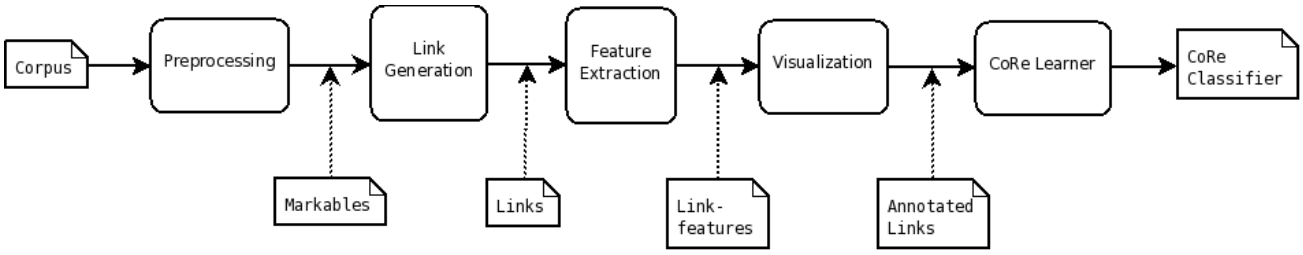


Figure 5.1: Modules in the SÜKRE Project

In figure 5.1 on the previous page you see on overview about the modules in the project. Every module takes as its input the information generated by the preceding modules.

The preprocessing, link generation and the CoRe Learner module are developed in the Institute for Natural Language Processing. The visualization module is work of the Institute for Visualization and Interactive Systems. The feature extraction for link features is work of both institutes in cooperation.

In the following each module is described in more detail. At the end, in section 5.6, the design of the database which is used by all modules is described.

5.1 Preprocessing

The preprocessing is different for prelabeled and unlabeled text. Goal of the project is to work on unlabeled text that is to be labeled with the help of the visualization. For the start of the project and the first analysis, labeled text has been used (namely the corpora of ARE, ACE 2005 and MUC-6).

For prelabeled text the words and labeled markables are extracted from the text and also all the word and markable features that are available. For the MUC-6 corpus this is POS-tag and number. All other markable features are extracted as they would be from unlabeled text.

Coreference information in the MUC-6 corpus is labeled in the form of coreference chains. The markables are extracted from these chains. So only markables that are coreferent with something are extracted. So a lot of noun phrases that would be markables if we processed the unlabeled text are not considered at all.

For unlabeled text more preprocessing is needed. After tokenization, sentence boundary detection and POS tagging, markables need to be extracted. As we consider only noun phrases as markables and every noun phrase could be a markable, the easiest way to do this will be to use the output generated by a parser. A parser will also provide POS-tags. The preprocessing for unlabeled text is currently in development.

At the end of this step all the information about the corpus is contained in four tables of the database. The first table contains all sentences from all documents (table ”sentences”). In an other table every word from every sentence is saved along with its features (table ”words”).

The markables that are extracted from the text are saved (table ”markables”) along with a potential entity that could be contained in that markable (table ”entities”). These entities are only based on noun clusters and proper names found in the markables.

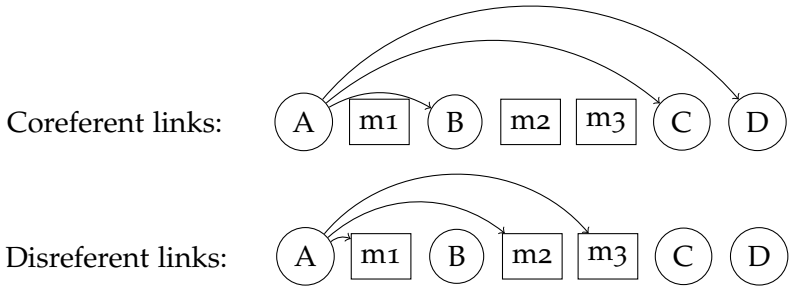


Figure 5.2: Link Generation for Prelabeled Text

5.2 Link Generation

After the markables have been extracted, they have to be put together to be pairs of markables (links). The link generation for prelabeled text is visualized in figure 5.2.

For prelabeled text, the markables of a conference chain are linked to create positive training samples. For a chain containing the markables A, B, C, D (in order of appearance in the text) the following links will be created. First A is linked with all other markables of the chain, creating the links A-B, A-C and A-D. The same is then done for B and the rest of the markables.

To create negative training examples, every markable of a chain is paired with markables of other chains that come close after them in the text. The generation is stopped as soon as the same number of disreferent links has been created that there are coreferent links from that markable. At the moment no links of markables in different documents are created.

In this step also a filtering process is done to reduce the number of links presented to the user. Filters can be defined by the user. The idea is that links that are certainly disreferent are filtered out. This could include for example links where one markable spans the other or links where a reflexive pronoun is linked with a markable from a different sentence.

The filter can also be used to limit the links down to a certain category. It might be useful to limit the distance of the two markables that can form a link or to consider only coreference inside one document. After a first step where these links are labeled, more links could be included in a second labeling step.

The result is the link table that contains the two markables that form the link and a label for that link. For labeled data the label is taken of the gold standard provided by the corpus. The confidence value of this label will always be 100. For unlabeled data the label will initially be set to unknown. At this point some prelabeling could be introduced where some 'very probably coreferent' links could be marked. This could be links where there is an exact string match of the markables or other heuristics. This label would help in labeling by giving the user some starting points to check.

5.3 Feature Extraction

For every link, a set of features is computed based on the attributes of the markables and words and other knowledge sources. The link features are not added to the link table, but they are kept in a separate feature vector table. This is because a link can have various feature vectors with different features calculated on this link.

This module is developed by both institutes in cooperation. At the moment it consists of two components, one developed in the Institute for Natural Language Processing, the other in the Institute for Visualization and Interactive Systems.

The first component calculates basic features like String match, edit distance or agreement on grammatical attributes. Features can be defined with the help of regular expressions.

The second component calculates a set of features that cannot be defined by regular expressions. These include for example the features for acronyms, apposition or non-linear features.

The feature extraction module is very modular so that new ideas for features can be tested easily. This is especially relevant for the semantic features to be developed in the project. A parser is already used for semantic role labeling and tests on parse tree features. WordNet is used for information about semantic class. Other sources for new features could be search engine distance or Wikipedia.

5.4 Visualization

The visualization takes the feature vectors from the database, maps them to a SOM and displays the SOM to the user. The user interactively explores and labels the data. The labels are added to the link table in the database. It is possible for the user to select a subset of the feature vectors to visualize and to recalculate the SOM for this subset. Also the user can chose the features he wants to use for the visualization.

Calculations are saved in the calculations table of the database.

This module is described in more detail in the next chapter, chapter 6 on page 35.

5.5 Coreference Learner

The data labeled with the visualization module can now be used to train a link-based classifier for coreference resolution. As feature set the same features can be used as for visualization or different features. This is because some features might be useful for visualization but not for training and vice versa.

At the moment the implemented machine learning algorithms are Support Vector Machine, a Naive Bayes Classifier and Regression (Ordinary Least Squares).

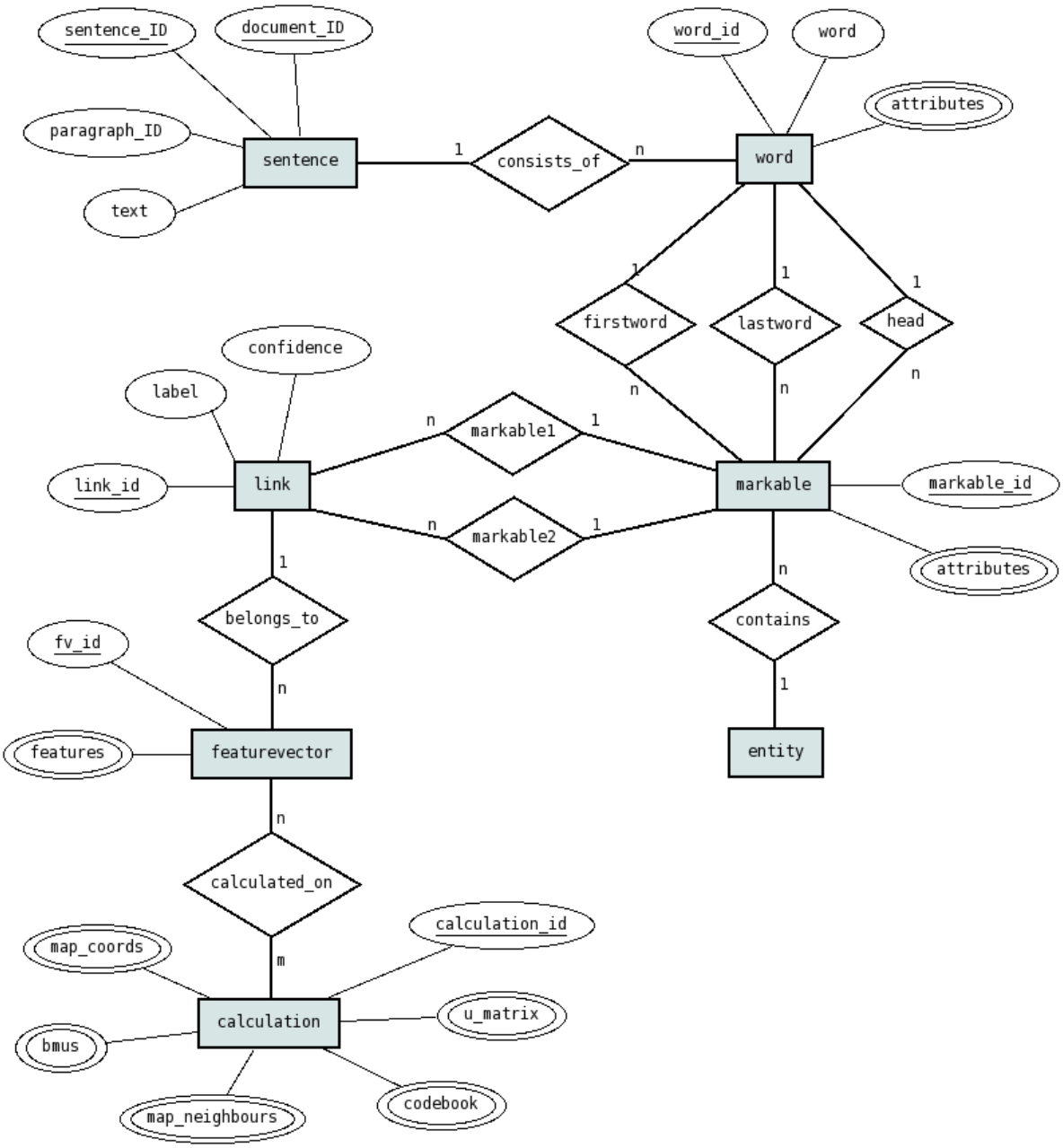


Figure 5.3: ER-Diagram of SÜKRE Database Design

5.6 Database Design

All modules of the software operate on a common database. This database is implemented as a Postgres database for the modules developed in the Institute for Visualization and Interactive Systems. The other components use the same design, but work on files.

An ER-diagram of the design of the database is shown in figure 5.3 on the previous page.

The **sentence** table contains all sentences from all documents in the corpus with document ID, paragraph ID, sentence ID and the content of the sentence.

Sentences consist of a number of words. The **word** table contains all words from all documents of the corpus. For every word it contains the word itself as text, a unique word ID, the document ID, paragraph ID, sentence ID and a list of attributes. The attributes can be any kind of linguistic information like POS tag, number, gender or semantic class. The attributes are saved as an array, not in separate columns to keep the design flexible for changes in the number and type of attributes. Punctuation marks also count as words.

The **markable** table contains all the markables of all documents. They also have a unique ID. The words the markable consists of are contained indirectly as references to the first and the last word of the markable. Also the ID of the head is saved. As the words are numbered in order of the text, the ID of the last word in a markable has to be bigger than the ID of the first word and the head has to be somewhere in between.

Markables also have a list of attributes such as number, gender, etc. Often the markable attributes will just reflect the attributes of the head. But sometimes the markable attributes can differ from that, for example a markable could be plural even if the head is singular (the markable *a cat and a dog* would have the singular word *cat* as a head after our preprocessing, but the grammatical number of the markable in total is plural).

The markable table contains a reference to a potential entity that could be contained in the markable. This **entity** table contains a first approximation of entities that could be found in the text with the word IDs of their start and end word.

A word can be part of multiple markables if the markables overlap. This should only be the case if one markable is embedded in the other one. This is the case for example in the markable $\textit{(the president of (America)}_2\textit{)}_1$.

A link is always formed by two markables. Supposedly the same link is only generated once. A markable can be part of any number of links. The **link** table contains a unique link ID for every link, a reference to the two markables that form the link and a label (coreferent or disreferent or unknown) and the confidence value (between 0 and 100) of that label.

For one link any number of feature vectors can be generated. It would make sense to calculate different feature vectors with different sets of used features. The resulting **feature vector** table contains a unique feature vector ID, the ID of the link the features belong to and a list of features.

Finally the **calculation** table contains the SOMs calculated for the visualization. Every calculation has a unique ID. A number of vectors and matrices with information about the SOM is stored along with the IDs of the feature vectors that have been used as input in this calculation.

Chapter 6

The Coalda Software

The new visualization that has been developed in this work is implemented in the Coalda software. Coalda stands for Coreference Annotation of Large Datasets. This chapter contains the specification of requirements for the visualization software, a conceptual design of the visualization and a high-level architectural design.

The conceptual design is the first design of the software intended for the user, not the programmer. It describes the functions and the basic structure of the software. It leaves open how the single functions are to be implemented. In contrast, the architectural design is directed at the programmer and already contains some information about the structure of the modules and the technology used.

6.1 Requirements

Requirements are listed in categories following the the VAI standard of documentation [Caro6]. The following categories are used:

RF Functional requirements list the functions the system has to execute (”what” does the system do).

RU User requirements list the desired options for the user interface (”how to access” what the system does).

RP Performance requirements denote minimum requirements in terms of time and space (”how fast”). These are always measurable.

RO Operational requirements define file formats, operating systems and other resources that are to be used.

Requirements are marked as ”essential” if they are a crucial part of the system, ”desirable” if they would add a considerable enhancement to the system and are considered important and ”optional” if their realization would add functionality to the system that is merely nice to have.

Functional Requirements

RF 1 The input data to be visualized are feature vectors belonging to links (pairs of markables) [essential]

RF 2 The feature space is visualized using a SOM [essential]

RF 3 The structure of the SOM is visualized using the U-matrix [essential]

RF 4 The number of feature vectors associated with a node is shown by colors or labels [desirable]

RF 5 Colors are used to visualize the weights of the nodes in the different dimensions of feature space [desirable]

RF 6 The user can assign a label to one or several nodes of the SOM [essential]

RF 7 The user can assign a confidence level to the label [essential]

RF 8 The visualization provides a ”zoom” to inspect an area of the SOM in more detail [essential]

RF 9 Departing from the abstract SOM visualization the user is able to access the text that belongs to the links [essential]

RF 10 The user can choose the data to be visualized [essential]

RF 11 The user can choose the features used to create the visualization [essential]

RF 12 The user can see the features used to create the visualization [essential]

User Requirements

RU 1 Selected nodes are highlighted in a color different from the other nodes [essential]

RU 2 Nodes are colored differently according to the selected field (dimension in the feature space or U-matrix value) [essential]

RU 3 Labels from pre-labeled data can be shown [essential]

RU 4 The zoom view will open in a new tab [desirable]

RU 5 The visualization can be dragged using the mouse [desirable]

RU 6 The visualization can be used by a linguist without knowledge about SOMs [essential]

Performance Requirements

RP 1 The visualization works with calculations based on up to 40000 feature vectors [essential]

RP 2 The number of nodes that can be visualized is up to 1900 [essential]

Operational Requirements

RO 1 The software runs under Linux (Fedora 9) [essential]

RO 2 The software is written in Java [essential]

RO 3 The data is loaded from the project database and uses the database format specified for the project [essential]

RO 4 The labels assigned by the user in the visualization are written into the project database using the database format specified for the project [essential]

6.2 Conceptual Design

The software to be developed is called Coalda. Two main goals serve as guidelines for its design. These goals are to allow the user to interactively label links with coreference information and to explore the space of the features used.

The coreference visualization to be implemented in Coalda is based on visualizing the feature space. As it is impossible to directly visualize a high-dimensional feature space, an indirect way has to be found. In Coalda, the feature space is visualized by training and visualizing a SOM. The visualization is based on the U-matrix.

The SOM will be visualized as a graph. The map units of the SOM are nodes of the graph and the neighbourhood relations in the output space are the edges. Distances between nodes are visualized by using the U-matrix value as the color for nodes and edges.

Figure 6.1 on the following page shows a screenshot of the Coalda GUI. The graph of the SOM is displayed in the upper left of the GUI. When a user clicks a node, detailed information about this node is displayed in the lower part of the screen. This includes the weights of the node in the different input dimensions (left) as well as the feature vectors associated with the node (middle) and the text belonging to the feature vectors (right).

The text corresponding to the feature vectors is shown in a simple text-based visualization. A markable is enclosed in square brackets []. The number right after the closing bracket is the ID of the feature vector the markable belongs to. There can be multiple IDs for one markable. If the feature vector is labeled as coreferent, the ID is green. If the feature vector is labeled as disreferent, the ID is red. The text visualization component will be provided by Andre Burkovski.

The color of the nodes is initially the U-matrix value. The color can be changed to represent the weight of the node in a selected feature. It can also be changed to show the percentage of coreferent feature vectors in this node, if there already are labeled feature vectors.

Every node has a node-label which is initially the ID of the node. This node-label can contain the number of feature vectors associated with this node. It can also indicate how many of these feature vectors are labeled as co- or disreferent.

Size of the nodes is influenced by the number feature vectors associated with this node.

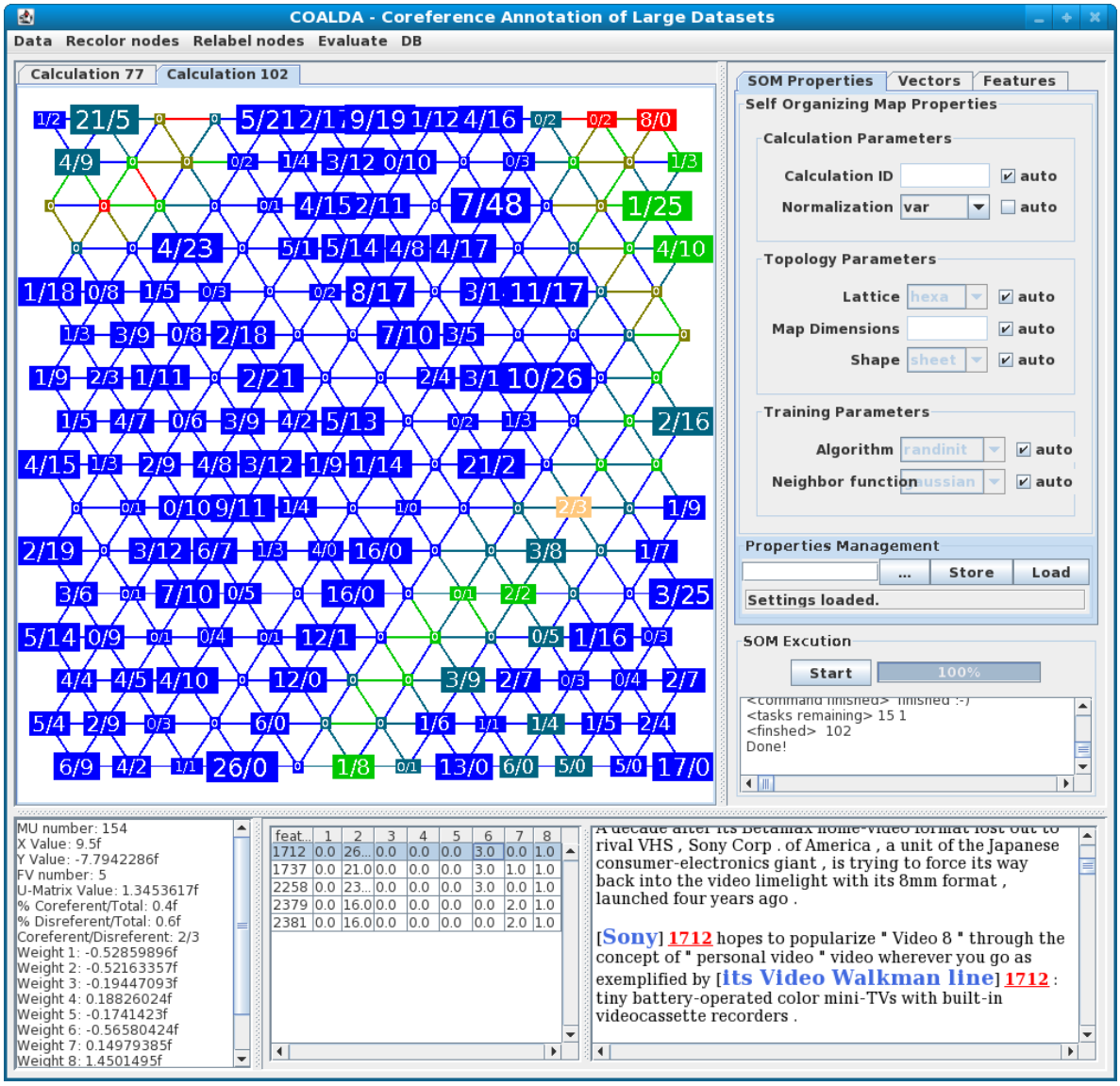


Figure 6.1: Coalda GUI

With a double click on a node, all feature vectors associated with that node can be assigned a label. The label can have a confidence value. It is not possible to assign a label to a single feature vector.

If a user wants to see a part of the SOM in more detail, he can either zoom in on one node or select several nodes and recalculate a SOM for this selection. In both cases a new SOM is calculated for the feature vectors associated with the selected node(s) and the result of the calculation is visualized in a new tab.

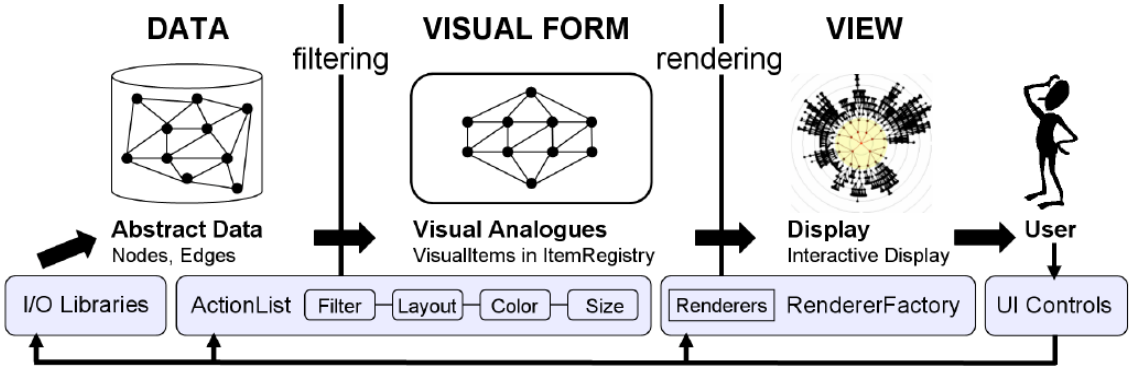


Figure 6.2: Architecture of the prefuse Visualization Framework [HCL05]

The SOM can also be recalculated for all the feature vectors with a different configuration of the SOM or with a subset of the features used for the first calculation. These settings can be changed in the panel at the right hand side of the screen.

The calculation of the SOM will be performed by a Matlab SOM server. The communication with the Matlab SOM server will be implemented by Andre Burkovski.

6.3 Architectural Design

The visualization is based on the prefuse visualization framework[1] for Java. The architecture of this framework as presented in [HCL05] is shown in figure 6.2.

The data to be visualized is taken from some source and converted to prefuse abstract data items. The data items are converted to visual items by adding the information necessary for visualization. Action lists can influence the visual appearance of visual items, for example the size or the color. Actions can also change the layout.

Visual items are drawn on the display by a renderer. UI controls can be added to the display to make the visualization interactive. These UI controls can trigger changes in any part of the system.

The parts to implement and to adapt for the Coalda software are I/O Libraries, Action Lists and UI Controls. Other components are taken from default prefuse libraries, for example the renderer or the UI Control for dragging.

[1] `http://prefuse.org/`

6.3.1 I/O Libraries

The modules that implement I/O libraries work on the data in the project database. The data to be loaded is of three types: data for calculations, feature vectors and labels. Consequently, a module for each type has been implemented. Data from calculations and feature vectors only needs to be loaded. Labels need to be loaded and written back into the database. The modules to be implemented are the following:

CalculationImport Takes the data out of the database and converts it to a prefuse abstract data item. The data to be imported is the result of a calculation. This includes the nodes of the SOM and for every node the U-matrix value, the weights in the different input space dimensions and the associated feature vectors. For edges only the U-matrix value needs to be imported.
Input: Data from calculation table in the project database.
Output: Prefuse abstract data items.

FeatureVectorImport Responsible for loading the feature vectors associated with a node and the text belonging to the feature vectors.
Input: Data from feature vector table in the project database.
Output: Screen display of feature vectors and text.

LabelImport Responsible for getting the label of one or several feature vectors.
Input: Data from link table in the project database.
Output: Label of feature vector(s).

LabelExport Writes the label a user assigned to the feature vectors associated with this node into the data base.
Input: User Input.
Output: Entry in link table in the project database.

Learner The SOM is calculated by Matlab[2] using the SOM Toolbox for Matlab[3] [VHAP00]. The implementation of the classes for the connection to Matlab as well as the classes for the configuration of Matlab was provided by Andre Burkovski in the package `somserver`. All these classes are contained in the module `Learner`. Once a calculation is finished, the actual loading is done by the module `CalculationImport`.
Input: SOM Configuration.
Output: Calculation ID to be loaded.
Uses: `CalculationImport`

6.3.2 ActionLists

Action lists are used to determine the appearance of visual items on the display. Action lists can influence layout, color, node-label and size. It can be specified when an action list should be run. The action lists to be defined are the following:

[2] `http://www.mathworks.com/products/matlab`
[3] `http://www.cis.hut.fi/projects/somtoolbox/`

LayoutAction The basic layout is a two-dimensional grid based on the positions of the SOM nodes in output space.
Runs after: `CalculationImport`

ColorAction The color of an edge is its U-matrix value. The color of a node is determined by the field the user has selected. Initially this field is set to be the U-matrix value. Selected nodes are highlighted in a color different from that of other nodes.
Runs after: `CalculationImport`, `Selection`, `RecolorNodes`

NodeLabelAction The node-label of a node is determined by the field the user has selected. Initially this field is set to be the ID of the node.
Runs after: `CalculationImport`, `RelabelNodes`

SizeAction Size of nodes is partially determined by the space the label requires. Other than that, size is determined by the number of feature vectors associated with a node.
Runs after: `CalculationImport`

6.3.3 UI Controls

Every UI module implements a single functionality. Several UI functionalities can be triggered by the same user action. User actions can trigger changes in any part of the system. The controls to be implemented are the following:

Selection When a user selects a node by clicking on it, the color of this node is changed to indicate that the node is selected. The nodes of the previous selection are not highlighted anymore.
Trigger: Click on any node.
Result: Current node is highlighted.
Uses: `ColorAction`

Focus When a user selects a node by clicking on it, additional information is displayed. This includes information about the node, about the associated feature vectors and the text associated with the feature vectors.
Trigger: Click on a node that has associated feature vectors.
Result: Information about current node is displayed.
Uses: `FeatureVectorImport`

Labeling When a user double clicks a node, he can assign a label to all feature vectors associated with this node. This label has an associated confidence value also assigned by the user.
Trigger: Double click on any node.
Result: All feature vectors associated with the current node are assigned the label chosen by the user with the confidence value chosen by the user.
Uses: `LabelExport`

SOMZoom This control contacts the Matlab SOM server, which calculates a new SOM for the feature vectors of the node the user has zoomed in on. Zoom does not work on edges or nodes with no feature vectors associated. The visualization of the new calculation

opens in a new tab.
Trigger: Zoom in on a node that has associated feature vectors.
Result: A new tab opens containing the visualization of a SOM calculated with the feature vectors associated with the current node.
Uses: `Learner`

Recalculate Contacts the Matlab SOM server, which calculates a new SOM with the current settings. Modified settings can affect the feature vectors, the features or the SOM configuration used for the calculation.
Trigger: Click button `Start`.
Result: A new tab opens containing a new SOM calculated with the current settings.
Uses: `Learner`

RecolorNodes Updates the color of all nodes according to the selected field.
Trigger: Select different field in menu `Recolor`.
Result: All nodes are colored according to their value in the selected field.
Uses: `ColorAction`

RelabelNodes Updates the node-label of all nodes according to the selected field.
Trigger: Select different field in menu `Relabel`.
Result: All nodes are labeled according to their value in the selected field.
Uses: `NodeLabelAction`

LoadCalculation Loads a specific (previously stored) calculation from the database.
Trigger: Select menu item `Load` from menu `Data`.
Result: A new tab opens containing the calculation to be loaded.
Uses: `CalculationImport`

Chapter 7

Features for Coreference Resolution

Most machine learning systems for coreference resolution work upon a set of link features. These link features are calculated on a link which is a pair of markables. The link features in turn rely on attributes of the markables that need to be extracted as a part of the preprocessing or the feature generation.

The markable attributes planned for the SÜKRE system are described in section 7.1. In section 7.2 the link features used by Ng and Cardie [NC02] are presented. Part of this feature set was re-implemented as a starting point and benchmark. Section 7.3 describes the SÜKRE feature set that was developed for this work.

7.1 Markable Attributes and Link Features

Markable attributes can be put in different categories. Some work on the surface representation of the markables, for example the content words of the markables or their position in the text. Others make use of linguistic processing in different levels ranging from Part-of-speech tag (POS tag) to semantic class.

Table 7.1 on the next page provides an overview of all attributes for markables that are used in the system. These attributes are explained in the following, ordered by category. Following the explanation of the markable attributes in every category, link features are discussed that use the attributes of the current category.

Several of the markable attributes and link features discussed in the following make use of the entity belonging to a markable. An entity is defined as the first noun cluster found in the markable. If no noun cluster is found, the markable contains no entity. Entities are extracted as a part of preprocessing.

Category	Name	Description
Content	words	All content words
	head	Head word
	premodifiers	Prenominal modifiers
Position	firstword_id	ID of the first word of the markable
	lastword_id	ID of the last word of the markable
	head_id	ID of the head of the markable
	sentence_id	ID of the sentence of the markable
	paragraph_id	ID of the paragraph of the markable
	document_id	ID of the document of the markable
Syntactic	postag	Part-of-speech tag (POS tag)
	type	Type of markable (pronoun, proper name or common NP)
	pro_type	Type of pronoun, none if markable is no pronoun
	c_type	Type of common NP, none if markable is no common NP
Grammatical	number	Grammatical number
	gender	Grammatical gender
	animacy	Animacy
	role	Grammatical relation
Semantic	semcls	Semantic class (from WordNet)
	semrole	Semantic role

Table 7.1: Attributes for Markables

7.1.1 Content and Comparison Features

Comparison features work on the words the markable consists of (the content words). When two markables contain exactly the same words, this can be a very strong indicator for coreference.

In the category ”content” we have the following markable attributes:

words This attribute contains all words of the markable. This includes punctuation (commas, quotes and so on).

head The head of a phrase is the ”most important” word of the phrase that is modified by the other words of the phrase. It determines the syntactic category of the phrase (if the head is a noun, it is a noun phrase). A phrase can always be replaced by its head. In case of a noun phrase the head is a noun or pronoun.

In this system the head is calculated as the last word in the first noun cluster in the phrase or the last word of the phrase if it does not contain a noun cluster (motivated by [Sano2]). In the noun phrase *new, innovative and revolutionary machine learning techniques for coreference resolution* the head would be *techniques*, as it is the last word in the first noun cluster (consisting of *machine learning techniques*). This generates errors, for

example in *John Thrasher, Tower's top video buyer* the word *Tower* is extracted as head while the correct word would be *Thrasher*. However, those cases are few.

premodifiers Prenominal modifiers are all words that occur before the head and are adjectives, gerunds, past participles or other nouns [Mit09]. In the previous example *new, innovative and revolutionary machine learning techniques for coreference resolution* the prenominal modifiers would be *new innovative revolutionary machine learning*. The conjunction *and* is not included. Instead of taking all words before the head, words before the start of an entity can be chosen. This is useful for markables containing proper names.

To compare the content of two markables, exact matches, substring matches or the edit distance of the two markables can be calculated. String matches can be performed on the head of the markable, for the prenominal modifiers or for the whole markable. Comparison features are often used in combination with the type of the markable.

Another link feature on content words is to check if one markable is an acronym of the other. For the construction of an acronym, the first letter of every word in the entity that is found in the markable is used.

7.1.2 Position and Distance Features

The distance of two markables in the text can be measured in different levels of granularity. The position of the markable in the corpus is saved in the following attributes:

firstword_id ID of the first word in the markable. Word IDs are unique.

lastword_id ID of the last word in the markable. This can be the same as the first word, if the markable consists of only one word.

head_id ID of the head of the markable (see category content for how the head is determined).

sentence_id ID of the sentence of the markable.

paragraph_id ID of the paragraph of the markable. Often, related information is combined in one paragraph which leads to the expectation of a lot of coreferences to be inside a paragraph.

document_id ID of the document of the markable.

Distance is an important feature for coreference resolution. In a study by McEnery et. al. [MTB97], most anaphora (95,92%) in the UCREL[1] Anaphoric Treebank have their antecedent less than five sentences from the referent position.

Distance is most important for the resolution of pronouns. Reflexive pronouns are always coreferent with a markable in the same sentence. For common noun phrases distance is not as important and it is even less important for proper names.

[1] University Centre for Computer Corpus Research on Language

Distance is calculated only on the absolute distance of markables in terms of words, sentences, paragraphs and documents. Alternatively, it could be calculated as the number of intervening markables [Luo07].

7.1.3 Syntactic

Syntactic attributes are based on the syntactic information about a word. This is primarily the POS tag assigned by a tagger. Based on the POS tag and content words, different categories of markables can be defined.

postag The Part-Of-Speech tag is extracted from the corpus. The MUC-6 corpus uses the TreeTagger[2] tagset, a variation of the Penn-Treebank tagset.

type A markable can be a proper name, a pronoun or a common noun phrase. A markable is a pronoun if it consists of one word which has a pronoun POS tag. A markable is a proper name, if its head is tagged as a proper name. In this definition, a markable is still a proper name if the name is preceded or followed by other words, as in the beforementioned *John Thrasher, Tower' s top video buyer*. A phrase where nothing can be determined is always treated as a common noun phrase.
Possible values: {name, pronoun, common}

pro_type If a markable is a pronoun, it can be a personal pronoun (like *he, him*), a possessive pronoun (like *his*) or a reflexive pronoun (like *himself*). Additionally the pronoun *it* can be pleonastic, meaning it is semantically empty and does not refer to anything as in *it was raining*. Other words tagged as pronouns that are not in one of these categories get the type "other". Non-pronouns have the type "none".
Possible values: {personal, possessive, reflexive, pleonastic, other, none}

c_type A common noun phrase can be definite (starting with *the*), demonstrative (starting with *this, that, these, those*), quantified (starting with a quantifier like *every, some, all, most, many, much, few, none*), indefinite (starting with *a, an*) or unknown.
Possible values: {definite, demonstrative, quantified, indefinite, unknown, none}

Syntactic link features can simply compare the type of markables in the link or be used in combination with other features.

The type of a markable can be important in determining if a link can be coreferent. Most pronouns are anaphoric, for example in the MUC-6 corpus the probability that a pronoun is anaphoric is estimated between 75% (1st/2nd person pronouns) and 99,6% (3rd person pronouns) [HKS09]. But if the antecedent is a pronoun and the anaphor not, they can never be coreferent [BR08].

If two markables have an exact string match, the category of markables is important to determine whether the markables are coreferent. If both markables are proper names, the probability that they are coreferent is very high. On the other hand, if both markables are pronouns, this means nothing more than they have the same gender and number.

[2] http://www.ims.uni-stuttgart.de/projekte/corplex/TreeTagger/

7.1.4 Grammatical

The best known grammatical attributes are gender, number and person. There are other grammatical attributes like animacy, case or mood. All grammatical grammatical word attributes have semantical content that influences syntax. For example if the subject of a sentence has number plural, the verb must also occur in the same number. The possible values for each category are determined by the grammatical classes that exist in the language. The attributes presented here are for the language English.

number The MUC-6 corpus contains information about number and it is extracted from there.
Possible values: {singular, plural, unknown}

gender The grammatical gender of a word does not necessarily have to be the same as the biological gender, but it mostly is. Gender information is not included in the MUC-6 corpus. It could be determined with word lists for pronouns and common nouns. For proper names a check for a title like *Mr.* (gender = male) or *Mrs.* (gender = female) or some indication that it is an organization like *Inc.* (gender = neuter) can give the gender information. If nothing is present, a list of countries (neuter), common first names or celebrities could be consulted [BRo8]. The addition of gender extraction to the SÜKRE preprocessing is planned for the future.
Possible values: {masculine, feminine, neuter, unknown}.

animacy Animacy indicates how much ”alive” something is. There are different possible categorizations, for English a simple one will be sufficient. The categories are to be determined with the help of WordNet. This attribute is planned for the future.
Possible values: {deity, person, animal, plant, thing, unknown}

role Grammatical relation (role) refers to the role a markable plays in the syntactic structure of a sentence. We have used BitPar[3] to obtain parse trees. Currently ”subject” is extracted as the only grammatical role.
Possible values: {subject, none}

In a link, the grammatical attributes of both markables normally have the same value. Thus, grammatical link features typically check for agreement of grammatical attributes. In regular cases, non-agreement means that the link cannot be coreferent. A markable with gender female like the pronoun *she* cannot be coreferent with a male name like *Peter*.

But these constraints are not always hard, in some exceptional cases two markables can be coreferent even if some attributes do not match. An example is the female pronoun *she* that can be coreferent with the neuter *the moon*. This is why grammatical link features should normally be treated as features and not used as a filter.

The grammatical roles of two markables do not have to agree for them to be coreferent. But often the subject of one sentence is also the subject of the sentence in which it is referenced by another markable.

[3]http://www.ims.uni-stuttgart.de/tcl/SOFTWARE/BitPar.html

7.1.5 Semantic

In many cases, looking at comparatively shallow attributes only, like those presented until now is not enough. To decide if markable three in *(They)$_1$ saw (the Alps)$_2$ as (they)$_3$ flew over Zurich* is coreferent with markable one or markable two, we need semantic information. Possible sources for semantic information are semantic role labeling, WordNet or Wikipedia [PS06].

semcls We have used WordNet as a source for semantic class information. The semantic class of the head is extracted as semantic class of the markable.

semrole Semantic role labeling can be done with a parser. Semantic roles are not the same as grammatical roles. Grammatical roles are determined by syntactic rules, while semantic roles capture, which of the arguments is the agent of the action or what other relation they have to the action. A grammatical subject can have varying semantic roles. In the sentence *Peter opened the door*, the markable *Peter* is grammatical subject and the agent of the action. In the passive construction *the door was opened by Peter*, the new subject is *the door*, but *Peter* is still the agent.

The easiest semantic feature is to check if the semantic attributes of the two markables forming the link are the same. This can be interesting for the attribute semantic class, if we define the semantic classes in a way that the classes are mutually exclusive. Then, two markables from different semantic classes (like for example person and company) cannot corefer.

With the help of WordNet it can be determined whether two markables are synonyms, antonyms or hypernyms [BR08]. Two words are synonyms if they have the same meaning, for example *rock* and *stone*. Words with only a very small difference in meaning (quasi-synonyms) are often regarded as synonyms for most applications. If one word is the opposite of the other, these words are antonyms, for example *big* and *small*. If word A (for example *tree*) includes word B (for example *apple tree*), A is a hypernym of B.

An interesting feature when using WordNet is the distance of two markables in WordNet. One such WordNet distance has been implemented. For the calculation all hypernyms of the head words of both markables are retrieved. Then the number of elements in the intersection of all hypernyms of both words is calculated, divided by the number of elements in the union of hypernyms of both words. The result is close to 1 if nearly all hypernyms are the same. It is 0 if there is no common hypernym.

A problem with using semantic information from WordNet is, that we have to deal with the word sense disambiguation problem. This, too, is a hard problem that is far from being solved. Common solutions are to use only the first sense of the word, or all senses without disambiguation.

Semantic features are a part of what is to be explored in the course of the SÜKRE project. Some ideas include the development of path features that compare the dependency path between markables or selection restrictions [HKS09].

7.1.6 Apposition

Two noun phrases form an apposition if the second noun phrase directly follows the other, separated by a comma, and modifies the first noun phrase. The phrase *Clinton, the president of the US,* is an apposition. The second noun phrase *the president of the US* modifies the first (*Clinton*) and gives more information about the same person. Apposition is an important feature, because two markables in an appositive construction are normally coreferent.

Not all constructions of two noun phrases directly following each other are appositions. Consider *Stuttgart, Germany,* as an example where the two markables are not coreferent. As a heuristic rule, two markables form an apposition if the second noun phrase contains an article [CW99] and one of them is a proper name [SNL01].

Apposition can be viewed as a feature of a markable (the second markable of an apposition is appositive) or a link feature (two markables form an apposition). This feature is planned to be implemented in the SÜKRE system in the future.

7.2 Features Used by Ng and Cardie

To have a baseline to compare to, we have re-implemented in our system part of the features used by Ng and Cardie in their 2002 paper [Ng02]. Their original feature definitions are shown in table 7.2 on the following page.

The values of these features evaluate to 'Y' (yes) or 'N' (no) for features that are based on one markable. Features based on two markables can have values 'C' (compatible), 'I' (incompatible) or 'NA' (not applicable). As we need numbers to use the features with a SOM, we have used 1 for 'Y' and 0 for 'N'. If the possible values for a link feature are only 'C' and 'I', 'C' is 1 and 'I' is 0. If the feature can evaluate also to 'NA', 'C' is 1, 'NA' is 0 and 'I' is -1. This follows the reasoning that 'NA' is a sort of in-between state between compatible and incompatible and thus the distance to both other values should be the same.

In contrast to the original feature definition, here the classification of noun phrases as pronoun, proper name or common noun depends only on the POS tag of the head. This difference affects only markables where the head is a proper name, but there are other words in the markable besides the name. There should be only few such markables.

For the first three features, the original feature definition seems to compare every word of the markable. Here only the head is compared.

The features that have been re-implemented along with the changed result values are the following:

PRO_STR 1 if the heads of both markables are pronouns and the heads are the same string; else 0.

PN_STR 1 if the heads of both markables are proper names and the heads are the same string; else 0.

Lexical		PRO_STR	C if both NPs are pronominal and are the same string; else I.
		PN_STR	C if both NPs are proper names and are the same string; else I.
		WORDS_STR	C if both NPs are non-pronominal and are the same string; else I.
		SOON_STR_NONPRO	C if both NPs are non-pronominal and the string of NP_i matches that of NP_j; else I.
		WORD_OVERLAP	C if the intersection between the content words in NP and NP is not empty; else I.
		MODIFIER	C if the prenominal modifiers of one NP are a subset of the prenominal modifiers of the other; else I.
		PN_SUBSTR	C if both NPs are proper names and one NP is a proper substring (w.r.t. content words only) of the other; else I.
		WORDS_SUBSTR	C if both NPs are non-pronominal and one NP is a proper substring (w.r.t. content words only) of the other; else I.
Grammatical	NP type	BOTH_DEFINITES	C if both NPs start with "the;" I if neither start with "the;" else NA.
		BOTH_EMBEDDED	C if both NPs are prenominal modifiers ; I if neither are prenominal modifiers; else NA.
		BOTH_IN_QUOTES	C if both NPs are part of a quoted string; I if neither are part of a quoted string; else NA.
		BOTH_PRONOUNS	C if both NPs are pronouns; I if neither are pronouns, else NA.
	role	BOTH_SUBJECTS	C if both NPs are grammatical subjects; I if neither are subjects; else NA.
		SUBJECT_1	Y if NP i is a subject; else N.
		SUBJECT_2	Y if NP j is a subject; else N.
	linguistic constraints	AGREEMENT	C if the NPs agree in both gender and number; I if they disagree in both gender and number; else NA.
		ANIMACY	C if the NPs match in animacy; else I.
		MAXIMALNP	I if both NPs have the same maximal NP projection; else C.
		PREDNOM	C if the NPs form a predicate nominal construction; else I.
		SPAN	I if one NP spans the other; else C.
		BINDING	I if the NPs violate conditions B or C of the Binding Theory; else C.
		CONTRAINDICES	I if the NPs cannot be co-indexed based on simple heuristics; else C. For instance, two non-pronominal NPs separated by a preposition cannot be co-indexed.
		SYNTAX	I if the NPs have incompatible values for the BINDING, CONTRAINDICES, SPAN or MAXIMALNP constraints; else C.
	ling. prefs	INDEFINITE	I if NP is an indefinite and not appositive; else C.
		PRONOUN	I if NP is a pronoun and NP is not; else C.
	heuristics	CONSTRAINTS	C if the NPs agree in GENDER and NUMBER and do not have incompatible values for CONTRAINDICES, SPAN, ANIMACY, PRONOUN, and CONTAINS PN; I if the NPs incompatible values for any of the above features; else NA.
		CONTAINS_PN	I if both NPs are not proper names but contain proper names that mismatch on every word; else C.
		DEFINITE_1	Y if NP starts with "the;" else N.
		EMBEDDED_1	Y if NP is an embedded noun; else N.
		EMBEDDED_2	Y if NP is an embedded noun; else N.
		IN QUOTE_1	Y if NP is part of a quoted string; else N.
		IN QUOTE_2	Y if NP is part of a quoted string; else N.
		PROPER NOUN	I if both NPs are proper names, but mismatch on every word; else C.
		TITLE	I if one or both of the NPs is a title; else C.
Semantic		CLOSEST_COMP	C if NP is the closest NP preceding NP that has the same semantic class as NP and the two NPs do not violate any of the linguistic constraints; else I.
		SUBCLASS	C if the NPs have different head nouns but have an ancestor-descendent relationship in WordNet; else I.
		WNDIST	Distance between NP i and NP j in WordNet (using the first sense only) when they have an ancestor-descendent relationship but have different heads; else infinity.
		WNSENSE	Sense number in WordNet for which there exists an ancestor-descendent relationship but have different heads; else infinity.
Pos		PARANUM	Distance between the NPs in terms of the number of paragraphs.
Other		PRO_RESOLVE	C if NP is a pronoun and NP is its antecedent according to a naive pronoun resolution algorithm; else I.
		RULE_RESOLVE	C if the NPs are coreferent according to a rule-based coreference resolution algorithm; else I.

Table 7.2: Features Used by Ng and Cardie [NC02]

WORDS_STR 1 if the heads of both markables are common noun phrases and the heads are the same string; else 0.

WORD_OVERLAP Number of words in the intersection between the content words of both markables.

MODIFIER Number of words in the intersection between the prenominal modifiers of both markables.

PN_SUBSTR 1 if the heads of both markables are proper names and the intersection between the content words of both markables is not empty; else 0.

WORDS_SUBSTR 1 if the heads of both markables are common noun phrases and the intersection between the content words of both markables is not empty; else 0.

BOTH_DEFINITES 1 if both markables start with *the*; −1 if neither start with *the*; else 0.

BOTH_PRONOUNS 1 if both markables are pronouns; −1 if neither are pronouns; else 0.

BOTH_SUBJECTS 1 if both markables are grammatical subjects; −1 if neither are grammatical subjects; else 0.

SUBJECT_1 1 if markable 1 is a subject; else 0.

SUBJECT_2 1 if markable 2 is a subject; else 0.

AGREEMENT 1 if the markables agree in both gender and number; −1 if they disagree in both gender and number; else 0.

SPAN 1 if one markable spans the other; else 0.

PRONOUN 1 if markable 1 is a pronoun and markable 2 is not; else 0.

DEFINITE_1 1 if markable 1 starts with *the*; else 0.

PARANUM Distance between the markables in terms of the number of paragraphs.

SENTNUM Distance between the markables in terms of the number of sentences.

7.3 SÜKRE Feature Set

A lot of the features of Ng and Cardie are composite features where different information is combined in one feature. The idea for the SÜKRE feature set was to create a feature set with more or less the same information, but decoupled. So a feature like PRO_STR ”1 if both NPs are pronominal and are the same string; else 0.” will be split into two binary features where the first feature is ”Both NPs are pronominal” and the second ”Both NPs are the same string”.

The second goal for this feature set was to eliminate as many binary features as possible. Therefore we tried to convert as many binary features as possible to similar non-binary features. These features are marked with an asterisk (”*”).

The following features have been implemented:

PARANUM Distance between markables in paragraphs.

SENTNUM Distance between markables in sentences.

SPAN 1 if one markable spans the other, else 0.

SPAN_REL* Span relative to size of bigger markable, $spannedWords/max(length(m_1), length(m_2))$.

BOTH_PN 1 if both markables are proper names; else 0.

BOTH_PRO 1 if both markables are pronouns; else 0.

BOTH_COMMON 1 if both markables are common nouns phrases; else 0.

TYPE_1* Type of markable 1: pronoun (1), proper name (2), common noun (3), else (0).

TYPE_2* Type of markable 2: pronoun (1), proper name (2), common noun (3), else (0).

HEAD_STRINGMATCH 1 if the heads of both markables are the same string; else 0.

HEAD_SUBSTRINGMATCH 1 if the head of one markable is a substring of the head of the other; else 0.

EDIT_DISTANCE* Edit distance between the heads of both markables.

ACRONYM 1 if both markables are proper names and one markable is an acronym of the other; -1 if they are proper names, but no acronyms; else 0.

WORD_OVERLAP Number of words in the intersection between the content words of both markables.

WORD_OVERLAP_REL* Number of words in the intersection between the content words of both markables relative to size of bigger markable, $wordOverlap/max(length(m_1), length(m_2))$.

MODIFIERS_OVERLAP Number of words in the intersection between the prenominal modifiers of both markables.

MODIFIERS_OVERLAP_REL* Number of words in the intersection between the prenominal modifiers of both markables relative to size of bigger markable, $modifierOverlap/max(length(m_1), length(m_2))$.

DEFINITE_1 1 if markable 1 starts with *the*; else 0.

DEFINITE_2 1 if markable 2 starts with *the*; else 0.

INDEFINITE_1 1 if markable 1 starts with *a* or *an*; else 0.

INDEFINITE_2 1 if markable 2 starts with *a* or *an*; else 0.

SUBJECT_1 1 if markable 1 is a grammatical subject; else 0.

SUBJECT_2 1 if markable 2 is a grammatical subject; else 0.

AGREEMENT_NUMBER 1 if both markables agree in number; else 0.

WORDNET_D* Distance between the heads of both markables in WordNet.

Chapter 8

Feature Analysis and Results

This chapter presents an evaluation of the visualization developed in this work. For this evaluation the MUC-6 corpus was used. This corpus was created for the coreference task of the sixth Message Understanding Conference[1] held in 1995.

The data used for the analysis of the feature space and most of the evaluation is based on a subset of the MUC-6 corpus. For this subset, document one of the corpus has been selected. It contains 43 sentences, 1244 words and 177 markables. These markables are paired to form 2098 links in the link generation module. 1034 of the links are coreferent, 1064 are disreferent.

For these 2098 links two different sets of feature vectors have been calculated. The first set was calculated using the reimplementation of the features used by Ng and Cardie (see section 7.2). It will be called Ng feature set in the following. The second set was calculated using the SÜKRE feature set (described in section 7.3).

Before evaluating the effectiveness of the visualization software, the data used in the tests needs to be analyzed. First, the values of all feature vectors for each extracted feature have been plotted in order to look for obvious errors or irregularities. Any findings are reported in section 8.1. Additionally, a Principal Component Analysis (PCA) has been done on both sets. The results are discussed in section 8.2.

Both these steps have been using the program The Unscrambler[2] by CAMO Software AS. All graphics in these two sections were created by that program.

Finally, both feature vector sets have been used as an input for the Coalda Software. The resulting visualizations and an evaluation of the result are discussed in section 8.3. A quality measure for the visualization has been introduced to compare the different visualizations. This measure uses the SOM to approximate human classification. The results for all feature vector sets can be found in section 8.4.

[1] http://cs.nyu.edu/cs/faculty/grishman/muc6.html
[2] http://www.camo.com/rt/Products/Unscrambler/unscrambler.html

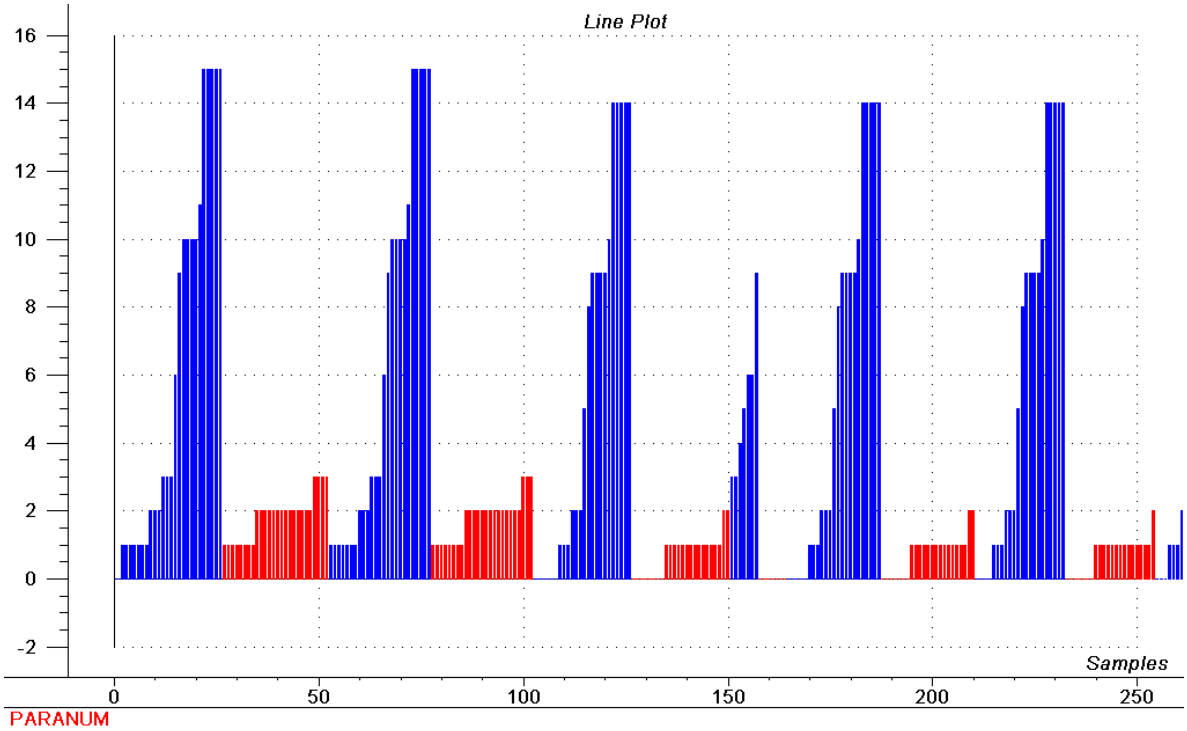


Figure 8.1: Values for Feature PARANUM for the first 250 Feature Vectors

8.1 Exploration of the Feature Space

The features used for the two feature vector sets are similar, at least for the features where irregularities have been found. If not stated otherwise, the features discussed here are found in both feature vector sets. Checking a feature for irregularities has been done by plotting the values of all feature vectors for this feature. This has been done for every feature, one feature at a time.

The first thing that caught the attention were the values for the distance features. Figure 8.1 shows the values for the first 250 feature vectors in the feature PARANUM (distance of markables in terms of paragraphs). The values for SENTNUM (distance of markables in terms of sentences) behave the same way.

There are small subsets of feature vectors where this feature starts with a low value, then the value increases for the following feature vectors. This pattern is repeated for the next subset. Also on a global level the maxima of the subsets are sorted descending. The colors in figure 8.1 represent the classes, coreferent feature vectors are blue, disreferent feature vectors are red. The overall values are higher for coreferent feature vectors.

This is related to the method of link generation described in section 5.2. Coreferent links are generated by linking all members of a coreference chain with each other. Beginning with the first markable in order of appearance in the text, every markable is linked with all the other

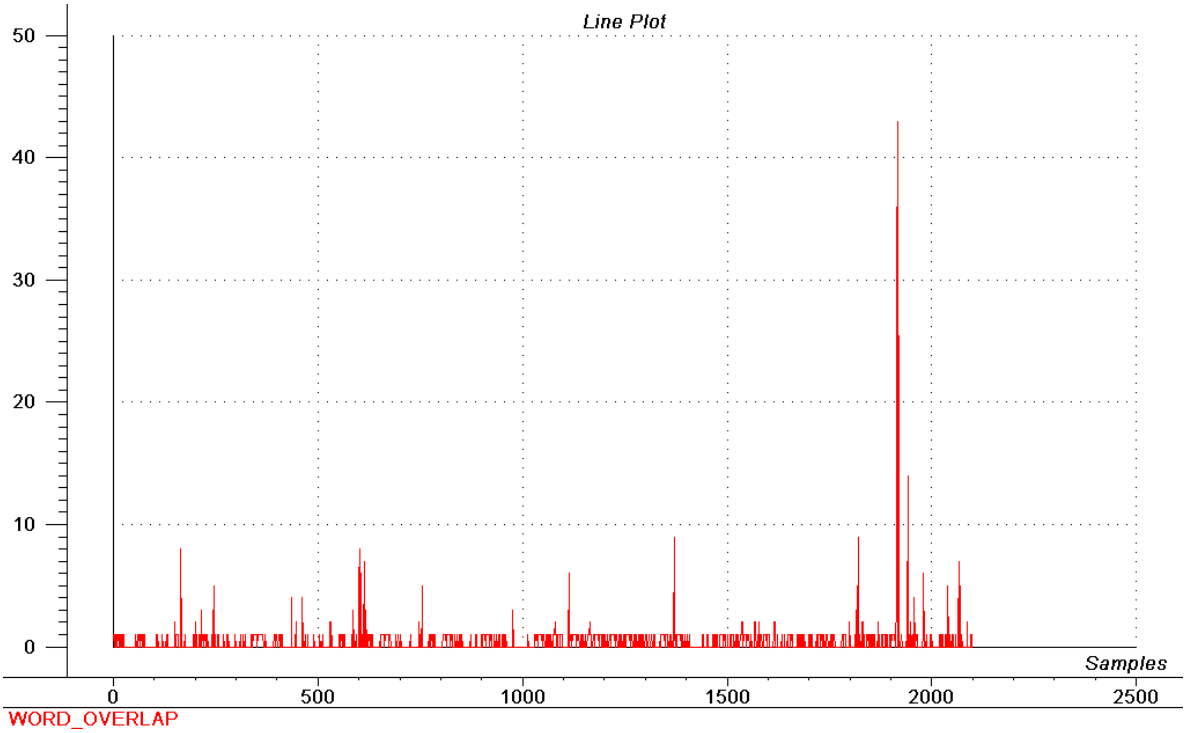


Figure 8.2: Values for Feature WORD_OVERLAP

markables in the same chain. As the markables are sorted by order of appearance in the text, this accounts for the sorting of the subsets of coreferent links. Disreferent links are generated by linking the markable in question with the next few markables in the text that doesn't belong to the same coreference chain. The same sorting by distance is achieved. But because there are more disreferent markables close to the markable, the overall distance values for disreferent links are smaller.

A similar phenomenon is responsible for the observed overall sorting of values. The markables are used for link generation in the order of appearance in the text. For a markable that occurs late in the text, the distance for the link of this markable with the last member in the text cannot be as big for a markable from the first few sentences.

The next irregular values appeared while looking at the feature WORD_OVERLAP in figure 8.2. For a few feature vectors the values are huge. The values prove to be correct, the markables are very large and as one spans the other, they do have a lot of words in common. The feature vectors having these huge values are the feature vectors number 1914, 1915 and 1918, to a lesser extent 1942 and 1370.

The feature AGREEMENT in the Ng feature vector set has three possible values. If the markables agree on number and gender the value is 1, if they agree on only one it is 0 and if they disagree on both is is -1. Because only number is implemented, the value for gender agreement is always 1 and the only possible values for AGREEMENT are 1 (they agree on

number) and 0 (they don't). Half of the feature vectors have the value 1 for AGREEMENT, probably because 60% of the markables are singular.

The feature ACRONYM in the SÜKRE feature vector set also has three possible values. It should be 1 if both markables are proper names and one markable is an acronym of the other, -1 if they are proper names, but no acronyms, and 0 in all other cases. There is no feature vector where ACRONYM has the value 1. This is correct, because there is no link where one markable is an acronym of the other. So the possible values for ACRONYM are -1 (both markables are proper names) and 0 (the markables are not both proper names).

There are a lot of correlated features, for example all the distance features are correlated. Correlations between two features could be found by plotting the feature vectors in a 2D scatterplot of the two features, or by calculating the covariances for all pairs of features. This was not done, because correlated features will have the same loadings in the PCA and can be identified this way without additional effort.

8.2 Results of PCA

The PCA for all feature vector sets has been done with the software The Unscrambler. For every PCA done, there are two plots. The first one shows a projection of the feature vectors in the coordinate system spanned by first two principal components. This corresponds to a plot of the scores of the data. The second plot shows graphically the loadings for PC1 and PC2 in the original features. The bigger the absolute value on the axis for a PC, the greater is the influence of this feature for the PC.

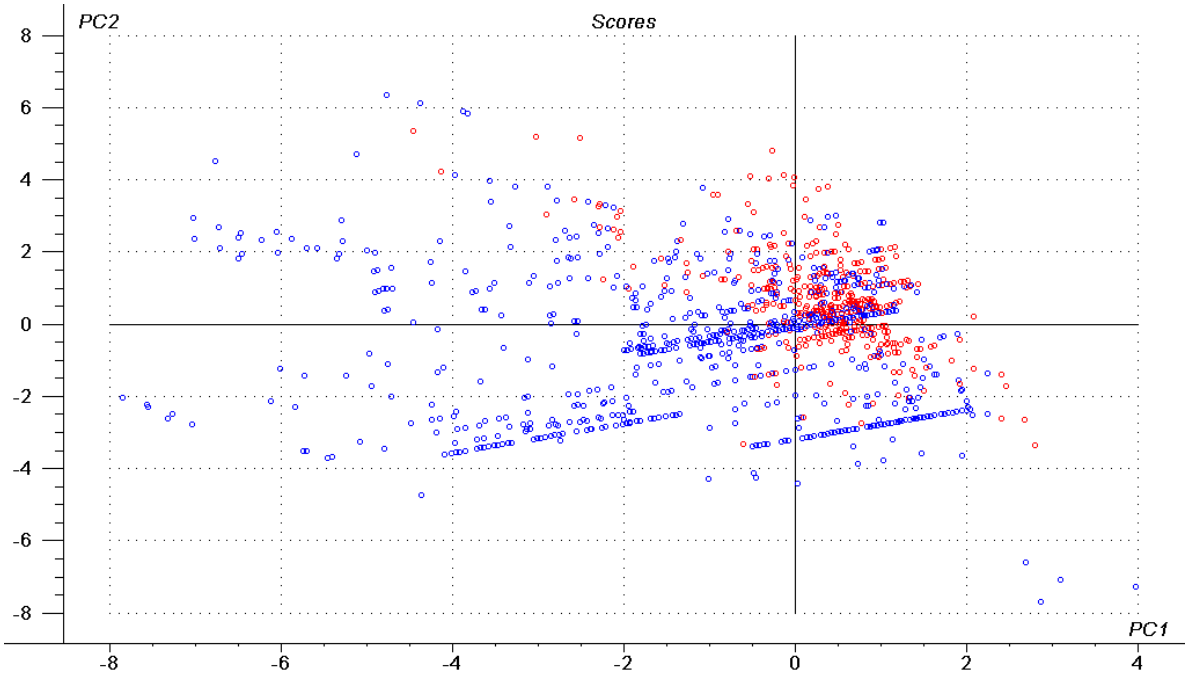
8.2.1 PCA on the Ng Feature Vector Set

If we perform a PCA on the feature vector set calculated with the features of Ng and Cardie, we get the result shown in figure 8.3 on the facing page.

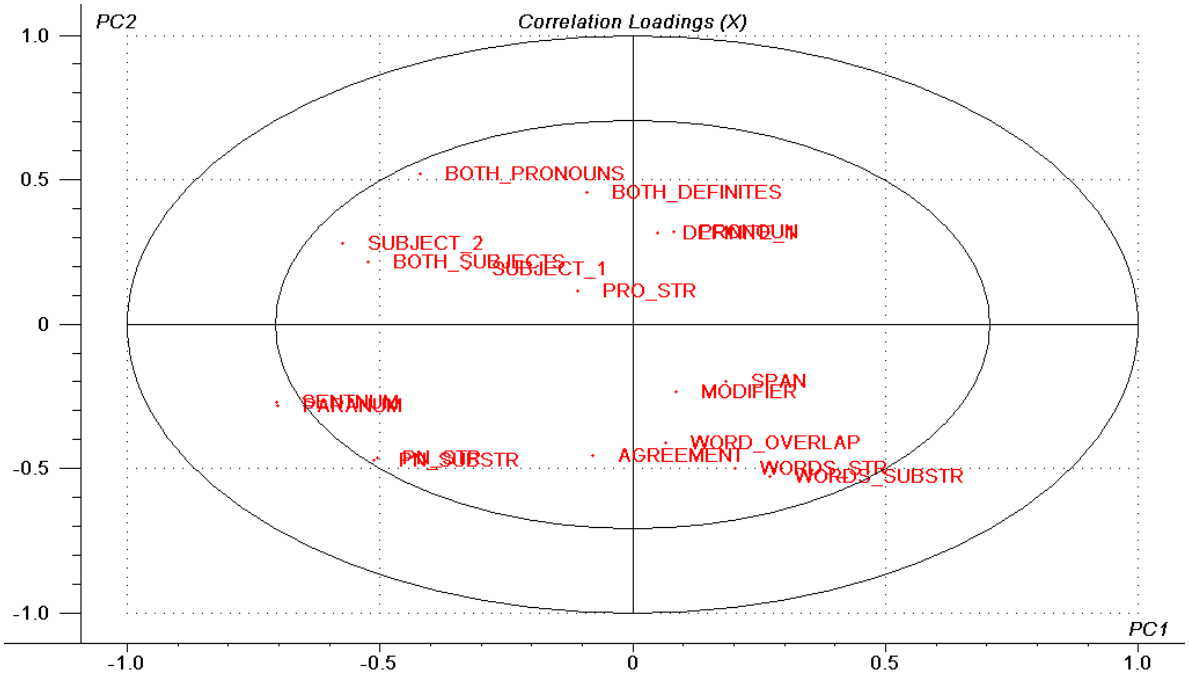
We note that on the left side of the diagram (negative value on PC1) there is almost only coreferent data (blue), while the right side seems very mixed. Also there are three clusters roughly separated.

The first PC is determined to a big extent by the distance features (PARANUM and SENTNUM). Figure 8.4 on page 58 contains the scores for PC1 for the first 250 feature vectors. Compare this figure to the values of PARANUM as shown for the first 250 feature vectors in figure 8.1 on page 54. For feature vectors where PARANUM is big, the score on PC1 is negative. For feature vectors where PARANUM is small, the score is positive.

This means that the most striking difference between coreferent and disreferent feature vectors in our feature vector set is the distance of the two markables that form the link. If we recall our observation from the data analysis, that is easily explained. Coreferent links are generated from markables in a coreference chain, no matter what their distance is. On the



(a) Scores



(b) Loadings for PC1/PC2

Figure 8.3: PCA for Ng Feature Vector Set, plotted with The Unscrambler

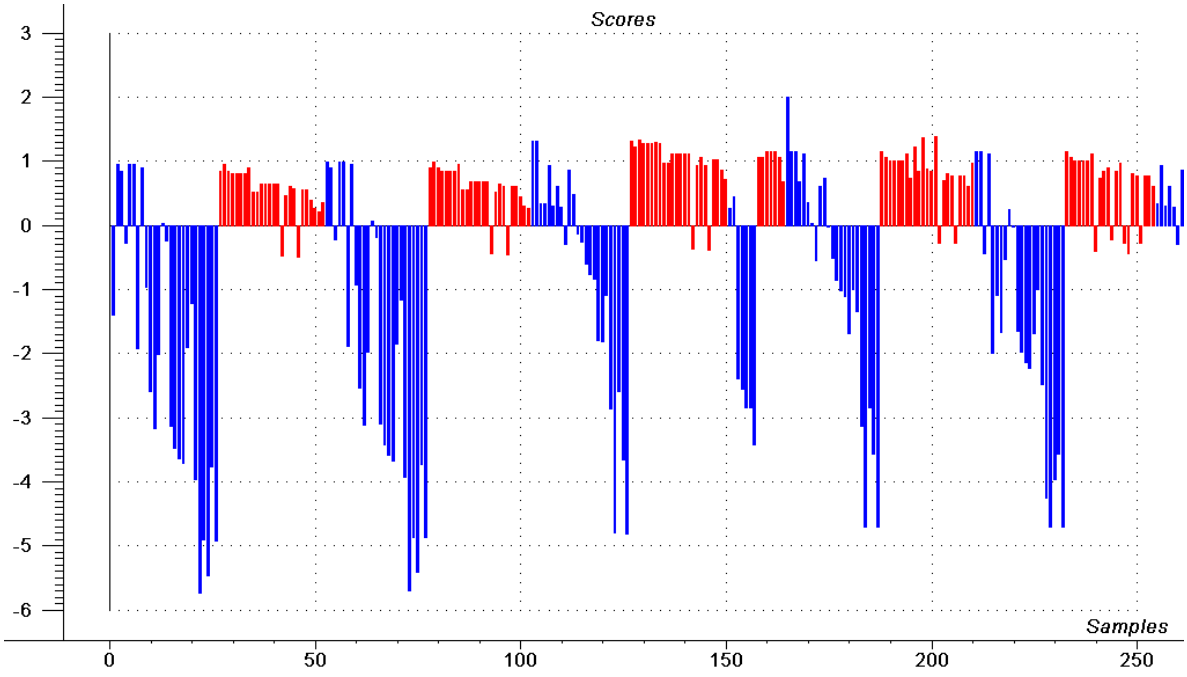


Figure 8.4: Scores for PC1 for the first 250 Feature Vectors

other hand, disreferent links are generated from linking a markable with other markables close to it.

This is not a result we can use for unlabeled data, so we will repeat the PCA in the next section using only the feature vectors belonging to links where the distance between the two markables is smaller than seven sentences.

Before doing that, let us find out what distinguishes the clusters from one another. The cluster on the lower left left in figure 8.3 (negative values on PC1 and PC2) is influenced by the features PN_STR and PN_SUBSTR. If we look at the values of the feature vectors forming this cluster, we see that these are all the feature vectors where both of these features have the value 1.

The cluster on the lower right (mostly positive values on PC2, negative values on PC1) is influenced by WORDS_STR and WORDS_SUBSTR. The feature vectors in this cluster have the value 1 for these two features.

We cannot see a cluster influenced by PRO_STR, because the influence this feature has on PC1 and PC2 is very low (in the loadings plot it is very close to 0).

We also see some outliers on the lower right. These are the feature vectors 1914, 1915, 1918 and 1370 with the very high values for WORD_OVERLAP that we noted earlier.

8.2.2 PCA on the Filtered Ng Feature Vector Set

If we repeat the PCA using only the feature vectors where the distance between the markables is smaller than seven sentences, we get the result in figure 8.5 on the next page.

The features with the biggest influence on PC1 and PC2 are PN_STR and PN_SUBSTR. The small cluster on the lower right of the scores plot (positive values on PC1, negative values on PC2) contains all feature vectors where one or both of these features have the value 1. This cluster consists almost exclusively of coreferent feature vectors, but there are a few disreferent feature vectors in between.

The disreferent feature vectors where PN_STR has value 1 are mostly created by errors in determining the head word. For example the word *Tower* is extracted as head for the markable *John Thrasher, Tower' s top video buyer* (see section 7.1.1). So there would be an exact match with the head of a different markable *Tower*, but the two markables are not coreferent.

The distance features SENTNUM and PARANUM still have a relatively big influence on PC1, but this does not result in any visible separation of coreferent and disreferent links.

An influence on PC1 and PC2 is also given by the features WORDS_STR and WORDS_SUBSTR. This feature creates the small cluster on the upper right (positive values for PC1 and PC2), it contains all feature vectors where one or both of these features have the value 1. This clusters contains mostly coreferent feature vectors, but also some disreferent feature vectors. A disreferent feature vector in this cluster would be the one belonging to the link of the markables *the blue table* and *the red table*. These markables are not coreferent, however the head words are identical.

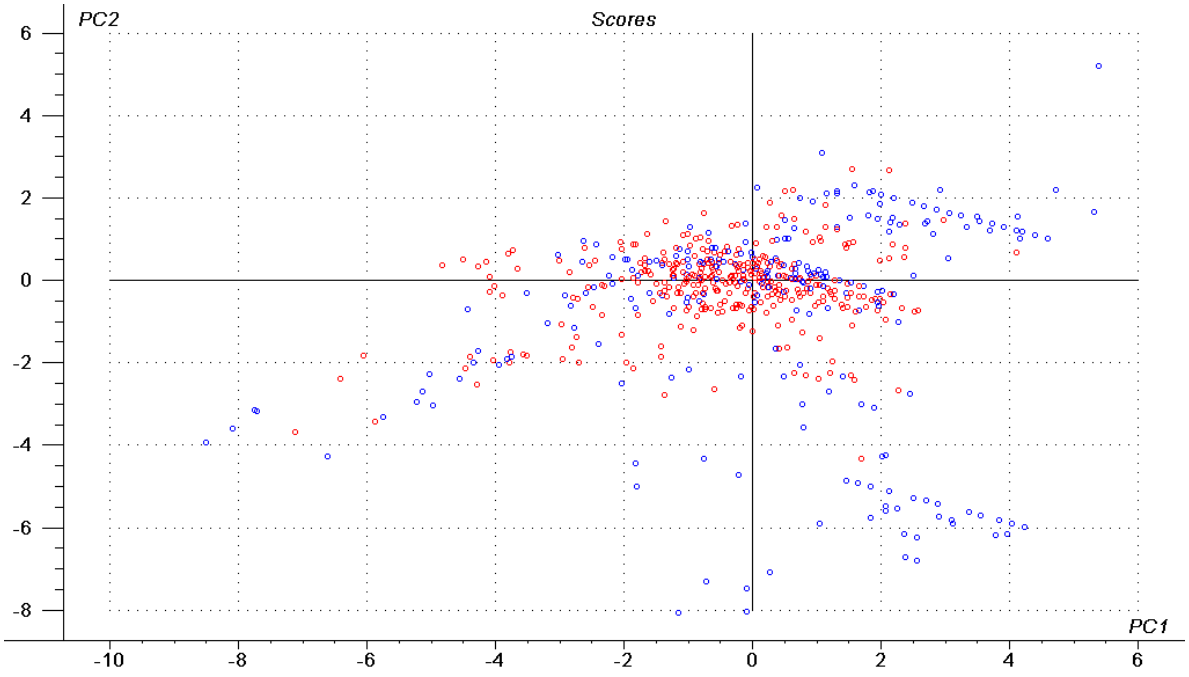
Between this cluster and the main cluster in the middle are some disreferent feature vectors where WORD_OVERLAP is not zero, but there is no string match of the head words. It is therefore difficult to define an exact border of the cluster.

At the far left hand side of the diagram there are feature vectors where both markables are pronouns (given by the feature BOTH_PRONOUNS), but they do not really form a distinguishable cluster and they are few. The same happens to the feature vectors where both markables are subjects (given by the feature BOTH_SUBJECTS). These can also be found on the lower left.

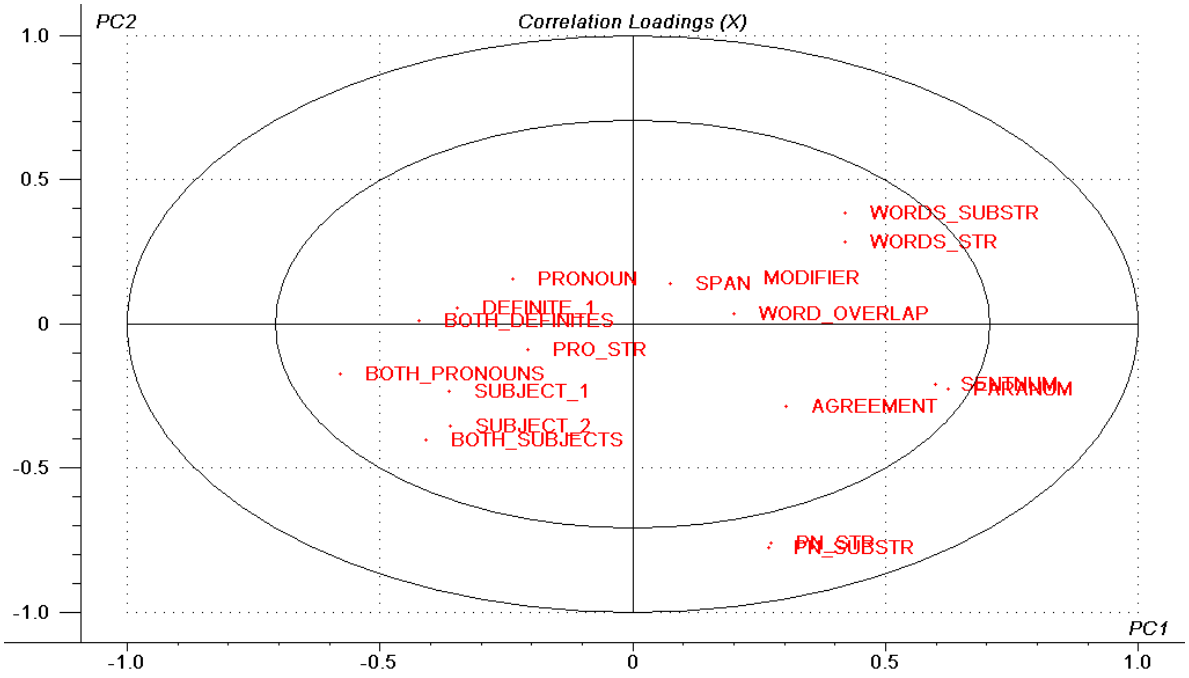
The outlier on the top right is feature vector 1370. The other outliers from the previous section have joined the cluster for WORDS_(SUB)STR.

If we look at the other PCs, there is no new cluster appearing, that cannot be seen when plotting the data only in the coordinate system of PC1 and PC2.

All in all we gained no new information from this analysis. The string matches in the features PN_(SUB)STR and WORDS_(SUB)STR are still the only ones able to create distinguishable clusters in the data.



(a) Scores



(b) Loadings for PC1/PC2

Figure 8.5: PCA for Filtered Ng Feature Vector Set, plotted with The Unscrambler

8.2.3 PCA on the Filtered SÜKRE Feature Vector Set

If we perform a PCA on the feature vectors generated using the SÜKRE feature set, we get figure 8.6 on the following page as a result. As in the previous section, only feature vectors where the distance between the markables is smaller than seven sentences have been used.

In figure 8.6(a), we clearly see two clusters, separated on PC1. PC1 is influenced by HEAD_STRINGMATCH, HEAD_SUBSTRMATCH, WORD_OVERLAP_REL (positive influence) and EDIT_DISTANCE (negative influence).

The cluster on the right hand side contains the all those feature vectors where there is a string match of some kind between the two markables. This includes all markables that have been in the clusters marked by PN_(SUB)STR and WORDS_(SUB)STR in the Ng feature vector set. Most feature vectors in this cluster are coreferent, but it also contains the same disreferent feature vectors as before.

PC2 is influenced to a large extent by SPAN and SPAN_REL. This is interesting, because in the Ng feature vector set, SPAN was one of the least significant features for PC1 and PC2. All feature vectors in the upper part of the figure belong to links where one markable spans the other.

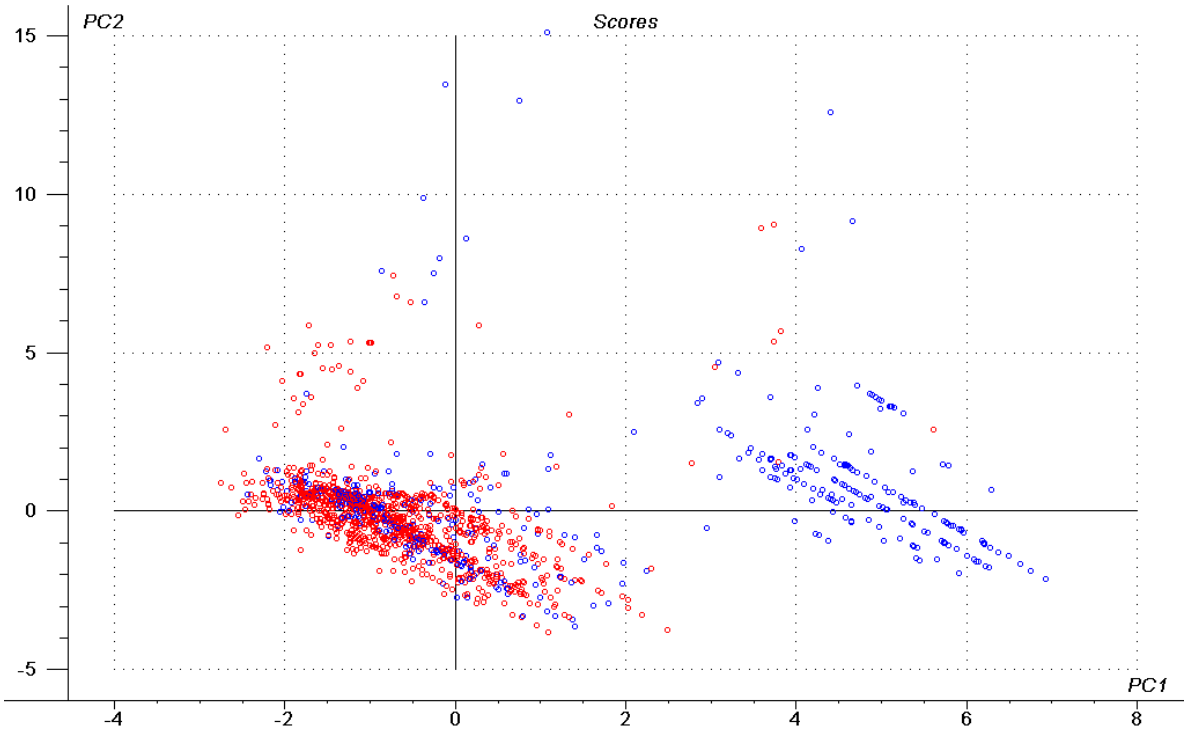
Normally, SPAN is viewed as a reliable indicator for disreference. Still, there are a number of coreferent links we see here. These are links where one of the markables covers two markables that form an apposition and the other markable is one markable of the two appositive markables. As an example, consider the link between markable 1 and 2 in *((Bill Clinton)$_1$, the president of the USA,)$_2$*. Such links are coreferent.

The outliers at the very top again are the feature vectors 1914, 1915 and 1918. Feature vector 1370 is the outlier more to the right.

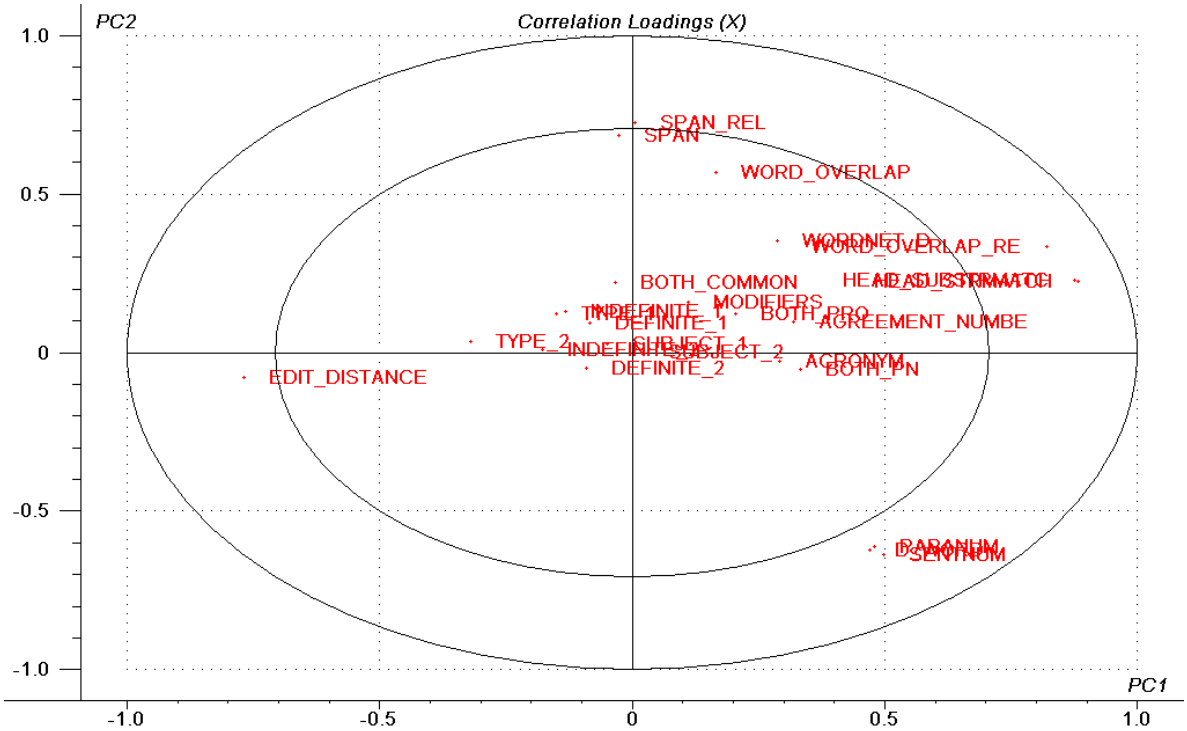
The cluster on the left hand side is very mixed and offers no more information about the structure of the data, at least when only plotting the coordinate system spanned by PC1 and PC2.

Looking at the other PCs, we see something interesting happening on PC4. PC4 is influenced by the features ACRONYM and BOTH_PN. If we plot the data in the coordinate system spanned by PC1 and PC4, both of the clusters we see in PC1 and PC2 are divided into two clusters each, depending on the value of the features ACRONYM and BOTH_PN.

As we recall the feature ACRONYM takes only the values -1 (both markables are proper names) and 0 (the markables are not both proper names) in the SÜKRE feature vector set. Thus, it does essentially the same as the feature BOTH_PN. The clusters that are formed by these features consequently represent the feature vectors where both markables are proper names. The clusters contain both coreferent and disreferent feature vectors.



(a) Scores



(b) Loadings for PC1/PC2

Figure 8.6: PCA for Filtered SÜKRE Feature Vector Set, plotted with The Unscrambler

8.3 Visualization with Coalda

After the data analysis, a short evaluation of the visualization developed in this work has been done. The visualization should help a user to label corefent and disreferent feature vectors and explore the feature space.

The effort needed for labeling a corpus with the software has not been evaluated directly. This can be done by comparing the time needed for labeling with Coalda to another visualization or without any visualization. Such an evaluation is planned as a part of the final evaluation of the SÜKRE project.

For a preliminary evaluation in this work, the two feature vector sets have been used as input for the software. The resulting visualizations have been examined whether clusters can be identified.

This section contains a number of screenshots from the visualization generated by Coalda. In these screenshots the colors represent the U-matrix value of the nodes. A low U-matrix value indicates that the node is close to its neighbouring nodes, a high value indicates that the node is far from the other nodes. The lowest U-matrix values are represented by the color blue, next comes steel blue, then light green (medium value), brown and the highest values are represented by the color red.

As we work with prelabeled text for the analysis, we have the labels of the links at our disposal. The node-labels in the visualization are the number of associated feature vectors of that node, the first number stands for the number of coreferent feature vectors, the second one for the number of disreferent feature vectors.

With the software we can also visualize the weights of all nodes for each feature (the component planes). Instead of presenting all these visualizations, in the screenshots the areas of the map, where the weight values for one feature are very high, have been marked with a circle and the name of the corresponding feature.

8.3.1 Ng Feature Vector Set Visualized with Coalda

In figure 8.7 on the next page, the visualization of the SOM trained on the Ng feature vector set is shown as it is created by Coalda. There are a few clusters that result from different features.

Taking the value of the feature PN_STR for example, we see, that there is a cluster in the lower right part of the SOM that consists of the feature vectors where PN_STR has the value 1. Because PN_STR is a good indicator for coreference, we see that all feature vectors in this cluster are coreferent.

Other features create clusters in the SOM where there are more disreferent feature vectors, for example SPAN. There are also clusters that are very mixed, like those created by BOTH_SUBJECTS or PRONOUN.

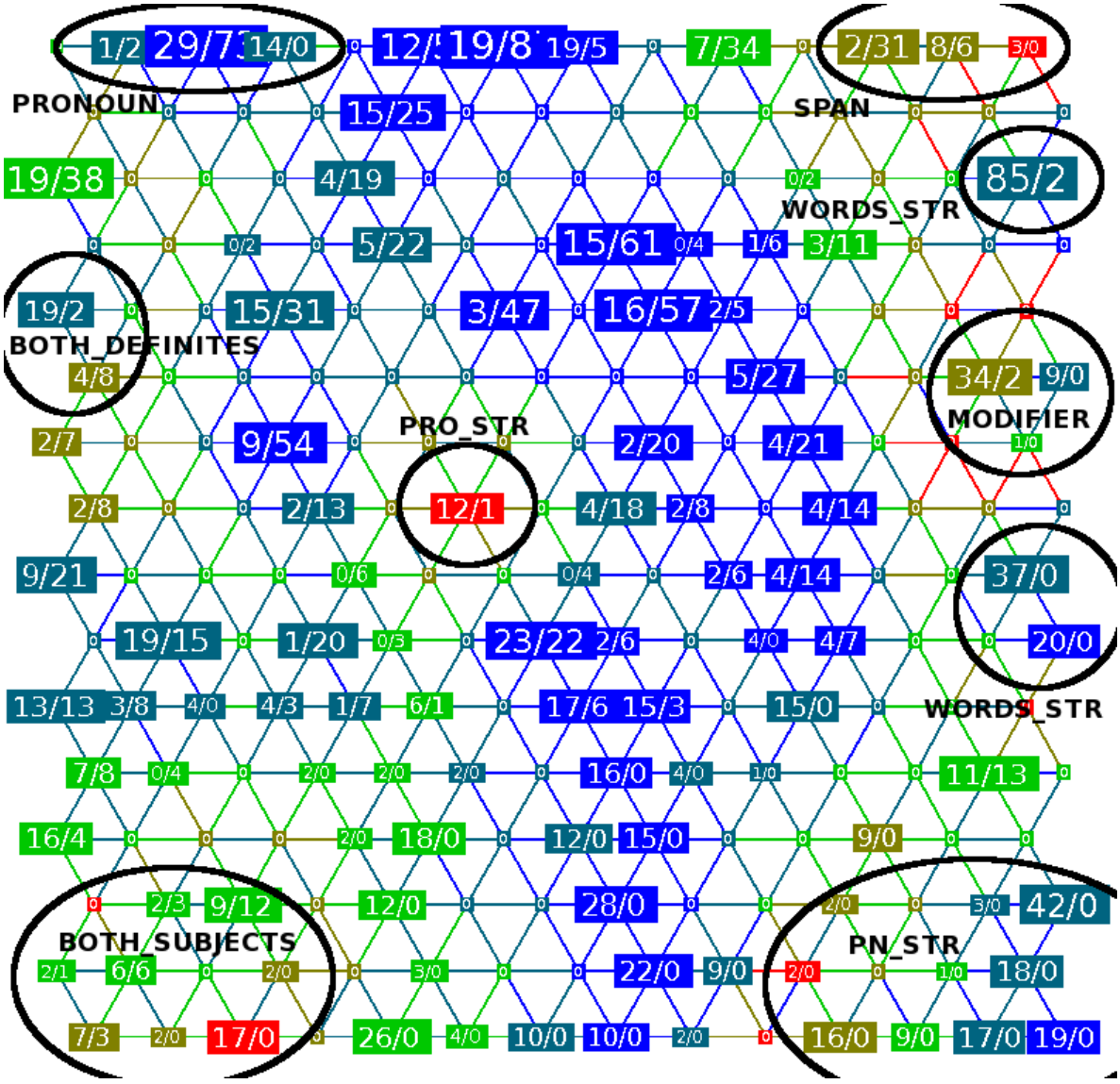


Figure 8.7: Ng Feature Vector Set Visualized with Coalda

It also becomes apparent which features are not suited for the visualization. The values of these features are spread out over the map and no clear clusters emerge. This is the case with AGREEMENT for example. AGREEMENT has the value 1 for half of the feature vectors. These can roughly be found in the right part of the map, but there is no clear cluster.

The input for the map has been all feature vectors. We noted in the PCA that all links with a big distance are corefent. When we look at the component plane for the feature PARANUM, we can see that the weights for the distance are bigger in the lower part of the map. Between the clusters for PN_STR and BOTH_SUBJECTS there are a lot of coreferent links. These are links with a very great distance between the markables.

8.3.2 Filtered Ng Feature Vector Set Visualized with Coalda

If we repeat the calculation using only the feature vectors where the distance between the markables is smaller than 7 sentences, as we did in the PCA, there is no real change to the map. The same clusters appear, although at different places. The coreferent cluster that contained all feature vectors with a high distance has vanished.

8.3.3 Filtered SÜKRE Feature Vector Set Visualized with Coalda

The visualization of the SÜKRE data with Coalda yields a result that is at first glance very similar to the previous visualization. It is shown in figure 8.8 on the following page. All features listed in section 7.3 have been used. Only feature vectors where the distance between the two markables is smaller than seven sentences have been used.

Some of the clusters visible are more or less the same clusters we saw in the visualization of the Ng feature vector set. These are for example the clusters for MODIFIER or SPAN. There is also a very clear cluster for HEAD_STRMATCH, similar to the cluster we had for PN_STR in the Ng feature vector set.

Other clusters are new, like the clusters for INDEFINTE_1 and INDEFINITE_2. Additionally there are some features that were contained in the Ng feature vector set, but did not create a strong cluster there, like DEFINITE_1 or SUBJECT_1.

In the Coalda software, one can easily select a subset of features to use for visualization. The SÜKRE2 feature set was created by removing all the binary features from the set that have a non-binary equivalent. The features that have been removed are BOTH_PN, BOTH_PRO, BOTH_COMMON, SPAN and WORD_OVERLAP. Again, the result is very similar. Other subsets also give very similar results and show no new clusters.

To make the clusters we have appear more clearly, we tried to use only the features that created clear clusters in the previous experiments. These are the features SPAN_REL, BOTH_PN, HEAD_STRMATCH, EDIT_DISTANCE, WORD_OVERLAP_REL, MODIFIER, DEFINITE_1, DEFINITE_2, INDEFINITE_1, INDEFINITE_2 and WORDNET_D. We called the subset SÜKRE3. The resulting map, shown in figure 8.9, indeed has clearer clusters. However, most of the features are not suited to create clusters consisting only of feature vectors with the same label. Still, having the cluster helps the user for labeling, because this would give him the possibility to treat all feature vectors of this type at the same time.

To obtain clusters where coreferent and disreferent feature vectors are more clearly separated, we need more features that are strong indicators for co- or disreference. Some of these features might only be relevant for a very small percentage of the feature vectors of a corpus. Thus they might not be very useful in machine learning for coreference resolution. But if these features formed clear clusters in the visualization, they would serve the same purpose as clusters with a lot of feature vectors for labeling.

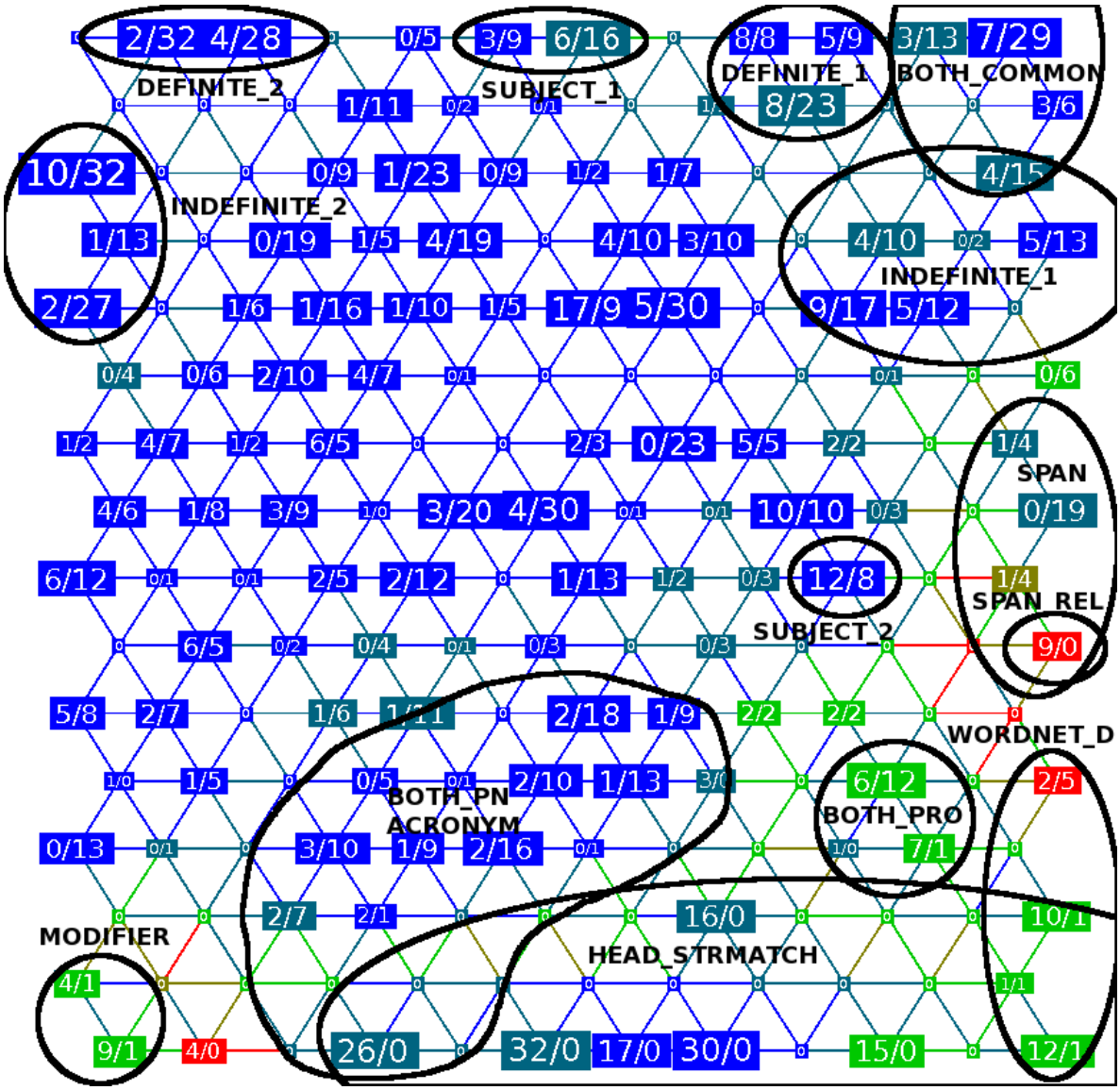


Figure 8.8: Filtered SÜKRE Feature Vector Set Visualized with Coalda

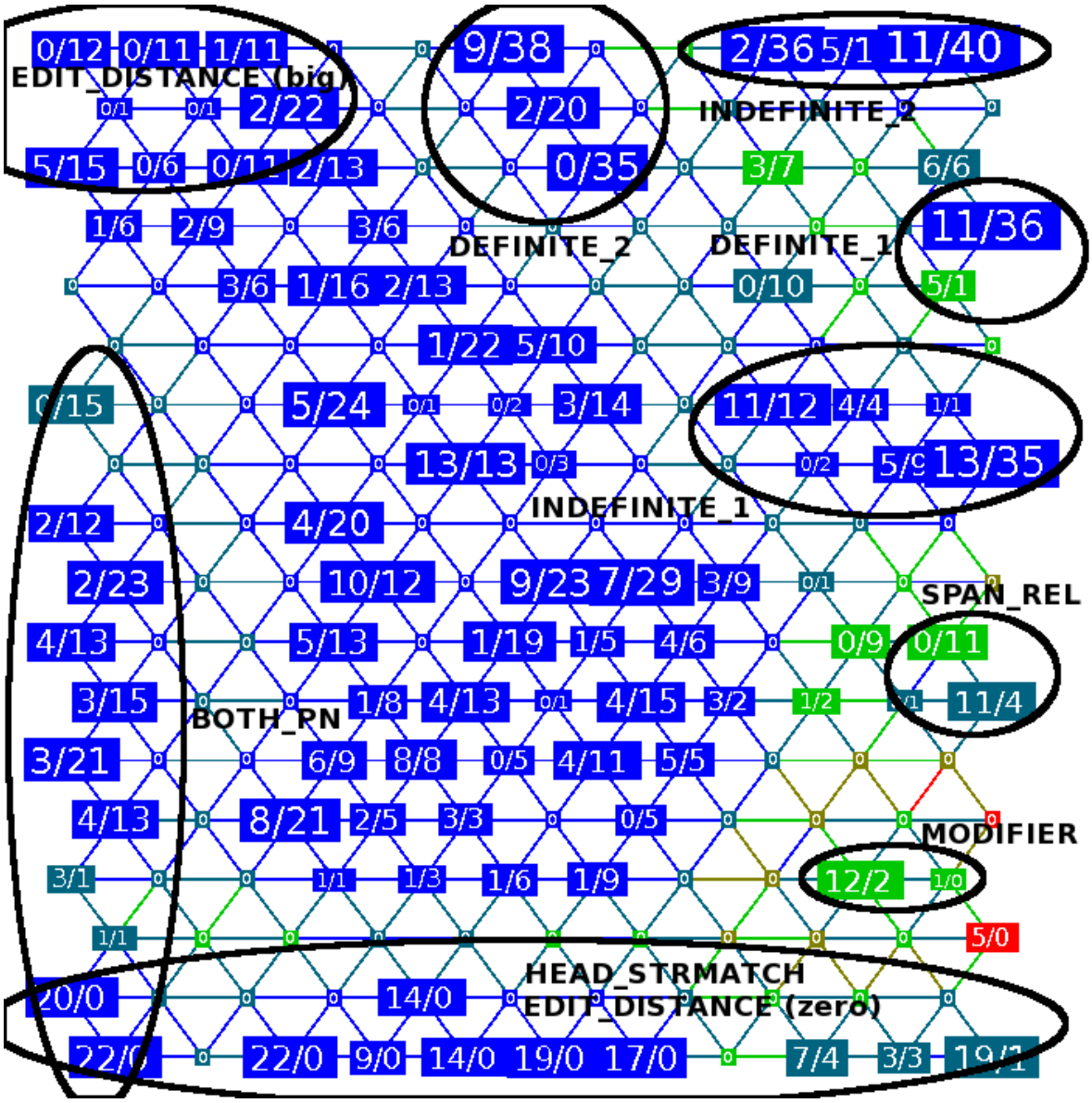


Figure 8.9: Filtered SÜKRE3 Feature Vector Set Visualized with Coalda

		Label in gold standard	
		coreferent	disreferent
Labeled as	coreferent	true positives (tp)	false positives (fp)
	disreferent	false negatives (fn)	true negatives (tn)

Table 8.1: Comparison of Assignment and Gold Standard Label

8.4 Evaluation of Visualization

In this section, a measure of quality for the visualization is introduced. This measure approximates human label assignments by treating the visualization as a classifier. Although automatic classification is not the intended use of the SOM, a high-quality clustering will make labeling easier for the user.

The calculated SOM has many settings that can be tuned. All results reported here have been made using the default settings. The size of the map has been automatically determined. If we manually increase the size of the map for any given calculation, the structure of the map essentially stays the same, only the feature vectors that formerly have been associated with one node are distributed over several nodes. Using this, we can arbitrarily increase all scores[3]. We have refrained from doing so, but further experiments with different (sensible) settings for the SOM could prove interesting.

To introduce a measure of quality for the visualization, we attempt to label the feature vectors with the SOM using the gold standard labels. For every node of the SOM, the label of all associated feature vectors is set to be the gold standard label of the majority of its associated feature vectors. So if a node has associated five coreferent feature vectors and seven disreferent feature vectors, all of the feature vectors would be labeled as disreferent.

After all feature vectors have been labeled, we compare the assigned labels to the gold standard labels. There are four different possibilities as listed in table 8.1. From these numbers we can compute precision and recall. Precision P indicates how many of the feature vectors labeled coreferent by the software are coreferent in the gold standard. Recall R indicates how many of the feature vectors that are coreferent in the gold standard have been labeled as such by the software.

$$P = tp/(tp + fp)$$

$$R = tp/(tp + fn)$$

To create one measure out of precision and recall, the F1-measure F_1 is computed. The F1-measure is the weighted harmonic mean of precision and recall.

$$F_1 = 2PR/(P + R)$$

The results for all calculations discussed in the previous sections are shown in the first part of table 8.2 on the facing page. The first column contains the feature set used to calculate the

[3]The limiting factor being the number of nodes we are able to store in the database.

Features	Filter	Documents	Precision	Recall	F1-measure
Ng	all	doc1	0.89	0.75	0.81
Ng	dist7	doc1	0.77	0.48	0.59
SÜKRE	dist7	doc1	0.81	0.60	0.69
SÜKRE2	dist7	doc1	0.82	0.58	0.68
SÜKRE3	dist7	doc1	0.81	0.55	0.65
Ng	all	all	0.78	0.65	0.71
Ng	dist7	all	0.75	0.52	0.62
SÜKRE	all	all	0.78	0.72	0.75
SÜKRE	dist7	all	0.74	0.58	0.65

Table 8.2: Evaluation Results

feature vectors used for the visualization. The column ”filter” indicates which feature vectors have been used, all (”all”) or only those where the distance between the two markables is smaller than seven sentences (”dist7”).

The analysis up to this point has been conducted only on a subset of the MUC-6 corpus, namely document one of that corpus. For this part of the evaluation, the experiments have been repeated for the whole MUC-6 corpus. The column ”documents” indicates on which set the scores were calculated, the subset (”doc1”) or all (”all”).

The results for using the whole MUC-6 corpus are similar to the results for the subset. The best results are obtained for the feature vector sets containing all feature vectors. This is due to the cluster of feature vectors where the distance features have a high value. This cluster consists of nodes where all feature vectors are coreferent. So this cluster generates only true positives. Their influence on precision and recall is high, because there are a lot of these feature vectors. The results for the filtered feature vector sets, where this cluster is missing, are considerably lower.

All SÜKRE feature sets have a higher F1-measure than the Ng set. This indicates that features developed for machine learning classifiers, like the Ng features, do not necessarily create good clusters in the visualization. The SÜKRE features were developed especially for the visualization and are better suited for clustering the data on the SOM. Replacing binary features by non-binary ones is also important for the SOM. These are promising results as a starting point for future experiments.

The evaluation results confirm that the visualization presented in this work is suitable to find clusters of feature vectors of similar type in the data. It thus helps the user in exploring the feature space and in evaluating the quality of the used features. Treating all feature vectors of one type at the same time, the user can efficiently label the feature vectors.

Chapter 9

Conclusion

In this work, a new visualization for coreference has been presented and implemented in Coalda. This visualization allows the user to interactively label links with coreference information and to explore the space of the features used to create the visualization.

The visualization works on feature vectors. To create these feature vectors, markables are extracted from a text in natural language. These markables are paired to form links. For every link, a feature vector is calculated based on a feature set. This preprocessing is done by modules of the SÜKRE project.

In Coalda, the resulting feature vectors are then mapped to a SOM and this SOM is displayed to the user. The SOM is visualized as a graph. The nodes of the SOM are nodes of the graph and the neighbourhood relations in the output space are the edges. Distances between nodes are visualized by using the U-matrix value of nodes and edges to color them.

The user can change the color of the nodes to represent the weights of the nodes in every input dimension. He can access the associated feature vectors for every node as well as the text belonging to the links. By doing this, the user can explore the feature space and discover the structure of the data.

To evaluate the influence a single feature or setting has on the SOM, the user can calculate a new SOM based on modified settings or on a subset of the feature vectors. It is also possible to zoom in on a node and get a new SOM calculated only on the feature vectors associated with that node.

For this work, two feature sets have been defined and the corresponding feature vector sets have been calculated. Both sets have been used as an input to the software and the resulting visualizations have been evaluated. In the development of the SÜKRE feature set, the visualization proved to be helpful for the design of new features.

The preliminary evaluation of the visualization shows promising results. The visualization allows the user to identify clusters of links that are influenced by a single feature. The feature vectors of this cluster can be considered as belonging to one type of links. When labeling, the user can treat all links of the same type at the same time.

Future Work

As this is a new approach to the visualization of coreference, there are still many ideas that can be included in the visualization. Some of these ideas are already part of the SÜKRE project specification.

One possibility to enhance the visualization is to include data density information. This could be done by calculating the P-matrix value and showing this value instead of the U-matrix value of the nodes. This corresponds to visualizing the U*-matrix.

Another idea from the specification is to visualize selected features as icons. Areas where the weight values of the nodes in this feature are high can then be identified by the icon (similar to the circles used in the screenshots of the evaluation).

To give the user a starting point for labeling, some links could be labeled before displaying the visualization. This prelabeling can be done by the user, or automatically using "reliable features" such as exact string match (nearly always coreferent) or span (nearly always disreferent) and marking the links where these features have the value `true`. In the visualization, the positions of the prelabeled links are then indicated.

Due to the transitivity of the coreference relation, as soon as the user has labeled some links, many other links are known to be coreferent or disreferent. If a user labels link A-B as coreferent when link B-C is coreferent, this entails that link A-C is coreferent. This could be used to automatically label all transitive links found in the SOM.

A major part of the SÜKRE project is the research on global features that work on a partition of links. With a few modifications, the visualization could probably be used to visualize partitions instead of links.

Acknowledgements

First of all, I need to thank Holger Rode for his support and his patience. I thank my supervisor Andre Burkovski, he did a good job and I hope I didn't annoy him too much with all my problems. Special thanks also go to my mother for looking at my data. Last but not least, thanks go to all those who read this work and helped me with suggestions: Christine Keller, Damian Philipp, Daniel Kauker, Dirk Coehne and Henning Rode.

Bibliography

[BR08] Bengtson, Eric ; Roth, Dan: Understanding the Value of Features for Coreference Resolution. In: *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, 2008 (Cited on pages 46, 47 and 48)

[Car06] Cardeñosa, Jesús: *Documentación Asociada a Cualquier Desarrollo del Grupo VAI*. 2006 (Cited on page 35)

[CGB08] Clark, Jonathan H. ; Gonzáles-Brenes, José P.: Coreference Resolution: Current Trends and Future Directions. In: *Language and Statistics II Literature Review* (2008) (Cited on page 14)

[CMBT02] Cunningham, Hamish ; Maynard, Diana ; Bontcheva, Kalina ; Tablan, Valentin: GATE: an Architecture for Development of Robust HLT Applications. In: *Proceedings of the 40th Anniversary Meeting of the Association for Computational Linguistics (ACL'02)*, 2002 (Cited on page 14)

[CW99] Cardie, Claire ; Wagstaff, Kiri: Noun Phrase Coreference as Clustering. In: *Proceedings of the Joint SIGDAT Conference on Empirical Methods in Natural Language Processing and Very Large Corpora (EMNLP/VLC)*, 1999 (Cited on pages 13 and 49)

[Ela05] Elango, Pradheep: *Coreference Resolution: A Survey*. Technical Report, University of Wisconsin Madison, 2005 (Cited on pages 11, 12 and 13)

[HBTM01] Harabagiu, Sanda M. ; Bunescu, Razvan C. ; Trausan-Matu, Stefan: CorefDraw: A Tool for Annotation and Visualization of Coreference Data. In: *Proceedings of the 13th IEEE International Conference on Tools with Artificial Intelligence*, 2001 (Cited on page 14)

[HCL05] Heer, Jeffrey ; Card, Stuart K. ; Landay, James A.: prefuse: a toolkit for interactive information visualization. In: *Proceedings of the SIGCHI conference on Human factors in computing systems*, 2005 (Cited on pages 5 and 39)

[HKS09] Heidemann, Gunther ; Kamp, Hans ; Schütze, Hinrich: *Antrag auf Förderung eines Forschungsvorhabens durch die DFG*. 2009 (Cited on pages 12, 14, 29, 46 and 48)

[Imo08] Imo, Johannes: *Visualisierung von Farbhistogrammen und Farbtexturmerkmalen zur Anwendung in CBIR-Systemen*, Universität Stuttgart, Diplomarbeit, 2008 (Cited on page 25)

[Jol02] Jolliffe, I.T.: *Principal Component Analysis*. Springer, 2002 (Cited on page 21)

[Kes07] Kessler, Waltraud: *Multivariate Datenanalyse für die Pharma-, Bio- und Prozessanalytik*. WILEY-VCH, 2007 (Cited on pages 18 and 21)

[Koh82] Kohonen, Teuvo: Self-organized formation of topologically correct feature maps. In: *Biological Cybernetics* (1982) (Cited on page 23)

[Luo07] Luo, Xiaoquiang: Coreference or Not: A Twin Model for Coreference Resolution. In: *Proceedings of NAACL HLT*, 2007 (Cited on pages 13 and 46)

[Mit09] Mitchell, Margaret: Class-based ordering of prenominal modifiers. In: *ENLG '09: Proceedings of the 12th European Workshop on Natural Language Generation*, 2009 (Cited on page 45)

[MS01] Müller, Christoph ; Strube, Michael: A Tool for Multi-Level Annotation of Language Data. In: *Proceedings of the 2nd IJCAI Workshop on Knowledge and Reasoning in Practical Dialogue Systems*, 2001 (Cited on page 14)

[MTB97] McEnery, A. ; Tanaka, I. ; Botley, S.: Corpus Annotation and Reference Resolution. In: *Proceedings of ACL Workshop on Operational Factors in Practical, Robust Anaphora Resolution for Unrestricted Texts*, 1997 (Cited on page 45)

[NC02] Ng, Vincent ; Cardie, Claire: Improving machine learning approaches to coreference resolution. In: *Proceedings of the Association for Computational Linguistics*, 2002 (Cited on pages 6, 13, 43 and 50)

[Ng02] Ng, Vincent: *Machine Learning for Coreference Resolution: Recent Successes and Future Challenges*. PhD thesis proposal, Cornell University, 2002 (Cited on pages 11, 12, 13 and 49)

[PS06] Ponzetto, Simone P. ; Strube, Michael: Exploiting Semantic Role Labeling, WordNet and Wikipedia for Coreference Resolution. In: *Proceedings of the main conference on Human Language Technology Conference of the North American Chapter of the Association of Computational Linguistics*, 2006 (Cited on page 48)

[RMS90] Ritter, Helge ; Martinetz, Thomas ; Schulten, Klaus: *Neuronale Netze. Eine Einführung in die Neuroinformatik selbstorganisierender Netzwerke*. Addison-Wesley, 1990 (Cited on page 25)

[Roj96] Rojas, Raúl: *Neural Networks*. Springer Verlag, 1996 (Cited on page 23)

[San02] Sang, Erik F. Tjong K.: Memory-Based Shallow Parsing. In: *The Journal of Machine Learning Research* (2002) (Cited on page 44)

[SGCR09] Stoyanov, Veselin ; Gilbert, Nathan ; Cardie, Claire ; Riloff, Ellen: Conundrums in Noun Phrase Coreference Resolution: Making Sense of the State-of-the-Art. In: *Proceedings of the ACL*, 2009 (Cited on page 13)

[SNL01] Soon, Wee M. ; Ng, Hwee T. ; Lim, Daniel Chung Y.: A Machine Learning Approach to Coreference Resolution of Noun Phrases. In: *Computational Linguistics 27, 4* (2001) (Cited on pages 13 and 49)

[Ult03a] Ultsch, Alfred: Maps for the visualization of high-dimensional data spaces. In: *Proceedings of the Workshop on Self Organizing Maps*, 2003 (Cited on page 27)

[Ult03b] Ultsch, Alfred: *U*-matrix: a tool to visualize clusters in high dimensional data*. University of Marburg, Department of Computer Science, Technical Report, Nr. 36, 2003 (Cited on page 27)

[US90] Ultsch, A. ; Siemon, H.P.: Kohonen's Self Organizing Feature Maps for Exploratory Data Analysis. In: *Proceedings Intern. Neural Networks*, 1990 (Cited on page 25)

[Ves99] Vesanto, Juha: SOM-Based Data Visualization Methods. In: *Intelligent Data Analysis* (1999) (Cited on page 25)

[VHAP00] Vesanto, Juha ; Himberg, Johan ; Alhoniemi, Esa ; Parhankangas, Juha: *SOM toolbox for Matlab 5*. Helsinki University of Technology, Technical Report A57, 2000 (Cited on pages 25 and 40)

[WT07] Witte, René ; Tang, Ting: Task-Dependent Visualization of Coreference Resolution Results. In: *Proceedings of the International Conference on Recent Advances in Natural Language Processing*, 2007 (Cited on page 15)

[ZG93] Zupan, Jure ; Gasteiger, Johann: *Neural Networks for Chemists*. VCH Publishers, 1993 (Cited on page 25)

All links were last followed on April 10, 2010.

Declaration

All the work contained within this thesis, except where otherwise acknowledged, was solely the effort of the author. At no stage was any collaboration entered into with any other party.

(Stefanie Wiltrud Kessler)