Extracting Characteristics of Human-produced Video Descriptions

A thesis presented
by

Matěj Korvas

to
The Department of Computing and Information Systems
in partial fulfillment of the requirements
for the degree of
Master of Science

University of Melbourne
Melbourne, Australia
September 2012

Thesis advisor(s)
Timothy Baldwin
Manfred Pinkal

Author
Matěj Korvas

Extracting Characteristics of Human-produced Video Descriptions

Abstract

This thesis contributes to the SMILE project, aiming for video understanding. We focus on the final stage of the project where information extracted from a video should be transformed into a natural language description. Working with a corpus of human-made video descriptions, we examine it to find patterns in the descriptions. We develop a machine-learning procedure for finding statistical dependencies between linguistic features of the descriptions. Evaluating its results when run on a small sample of data, we conclude that it can be successfully extended to larger datasets. The method is generally applicable for finding dependencies in data, and extends methods for association rule mining for the option to specify distributions of features. We show future directions which, if followed, will lead to extracting a specification of common sentence patterns of video descriptions. This would allow for generating naturally sounding descriptions from the video understanding software.

Contents

Acknowledgments

I am most grateful to

Tim Baldwin
for his exceptional care, for wise, inspiring, and kind lead of my Masters project, as well as assistence with all technical issues that came up on the way.

Manfred Pinkal, Michaela Regneri, and Marcus Rohrbach
for inventing and working on such a great project, for taking me on board and helping me out.

Bobbye Pernice, Valia Kordoni
for administering my Masters studies, remembering most of my deadlines for me and remedies when I occasionally missed them.

Karolína Korvasová
for being with me all the time.

my and Karolína's family
for being with me when needed.

Ihor Tytyk, Adam Liška, Desmond Darma Putra, Yousef Kowsar, Mahtab Mirmomeni, Shaghayegh and Sarah Oveissi, and Sarah Erfani
for a great company.

Lea Frermann and Dominikus Wetzel
for warm welcome in Saarbrücken and altruistic help in carrying out my final duties to the University of Saarland.

John Bauer
for prompt handling of reported issues with the Stanford CoreNLP software suite.

A special thanks goes to the reader for *not* printing the thesis, thus saving our environment. Reading this thesis from the screen should be not only more sustainable, but also more convenient, with all the advantages of electronic documents.

Dedicated to Karolína.

Chapter 1

Background

1.1 Motivation

Meaning of words is one of the central issues of linguistics. Semanticists have long been occupied with the question what structure can describe meaning accurately. While the most successful approaches nowadays extract meaning from text corpora and capture it as vectors in an arithmetic vector space, we feel it does not reflect the actual nature of meaning very well. If you are asked to imagine an airplane, you usually recall a visual scene which is dominated by an airplane, rather than activating an appropriate vector of its characteristics including maybe its animacy or size. In reality, computational semantics does not even call components of such vectors by their name (*animacy* or *size*); they are just *the first component*, *the second component*, etc. Hence, even if one of the components corresponds to, say, *animacy*, this fact stays hidden to the semantic representation.

The present research contributes to a project called SMILE (Script Mining as Internet-based Learning). The SMILE project combines the areas of natural language semantics and computer vision. We aim for bridging the gap from visual to textual information by characterizing their relation. For that, we plan to use an intermediate representation which would describe meaning with regards to visual properties. We focus on videos as opposed to still images, and semantics of *scripts*.

Scripts are prototypical sequences of events that occur in everyday life. For example, we all know the script of cutting bread. It involves the actions of getting a loaf of bread, knife, and a cutting board, then cutting one or more slices off the bread, and finally putting all the things back to their place. More complex scripts include those for visiting a restaurant (note that there are different scripts for this, depending on the kind of the restaurant) or going to the cinema. It is important that scripts are *prototypical* sequences of events – that means that most people know scripts, they form a part of our collective knowledge. Therefore, they represent information which is usually implicit in human communication.

The goal of the SMILE project is to employ scripts as an additional cue to understand videos. The first step of the project was to gather a database of scripts. We have since been building a software system that identifies objects and actions in a video, exploiting the database of scripts. Although computer vision is already advanced in this task (or similar ones), we expect it

to benefit much from matching its output against the script database.

Once an interpretation for the input video is constructed based on the combined knowledge of computer vision and scripts, we would like to express the information in natural language. This is what we have investigated in the present Masters project – what natural language descriptions for videos should be like. In the description generation, pure information about contents of a video have to be classified into important / not so important information, and the most important chunks of information need to be transformed into words and sentences. The system needs to know at runtime how to do this – how to identify important information, and how to transform it into words. In this Masters project, we aim at extracting this knowledge from video descriptions that were created by humans.

Our work builds on three areas of research: a) analysing visual information; b) analyzing textual information; and c) learning the most probable association between elements of the two. Analysing visual information is a field too large, and a bit off-topic for this thesis, for us to review it here. Instead, we refer the reader to the comprehensive review of Aggarwal and Ryoo (2011). We review the other two areas in the following sections.

1.2 Analysis of Text

On the side of text analysis, we are mainly concerned by scripts and their unsupervised extraction from bodies of text. The notion of script was introduced by Schank and Abelson (1977). This work builds on previous experience of its two authors, bringing together artificial intelligence and social psychology to make one of the fundamental contributions to a newly formed discipline, coined cognitive science. The authors try to put a part of inherently human way of thinking into exact terms, and finally implement part of the theory they develop as a computer program. The specific knowledge they inspect is that of usual sequences of related events (scripts), and how people develop understanding of these.

Schank and Abelson had to specify scripts they wanted to work with, manually. Although they offer ways to specify only more generic entities and derive individual scripts from them, there has been need to extract script knowledge from bodies of text *without supervision*, for retrieving them in large enough amounts for low cost and adaptability to the input. Chambers and Jurafsky (2009) present an unsupervised approach to script extraction. Their approach is based on finding often occurring sequences of verbs (*narrative chains*) and related sequences of their actants which denote the same entity across the sequence (*coreference chains*). Authors show that learning the narrative chains jointly with semantic frames of their verbs is advantageous compared to performing the two tasks separately. This comparison is made using the cloze task, which requires gaps in a text to be filled in. Authors also report on top 20 extracted narrative schemata with respect to their confidence score: they achieved around 70 % accuracy of suggested frame fillers.

Following two papers are part of the SMILE project. They are both concerned with extracting script information from crowdsourced knowledge. The first one, Regneri, Koller and Pinkal (2010), describes how underlying data were obtained, and how they were clustered into a single temporal graph for each scenario. The data collection was performed using Amazon Mechanical Turk.[1] Its workers were given a scenario, and they were asked to provide a prototypical sequence of events that constitutes the scenario. Having gathered data in the form of *event sequence*

[1] http://www.mturk.com

descriptions (ESDs), Regneri, Koller and Pinkal clustered the elementary events across different ESDs for the same scenario by their similarity, using a *multiple sequence alignment* algorithm. The clustering then gave rise to a temporal graph, when temporal links implied by the original ESDs were projected into the clustered structure.

In a newer work, Regneri, Koller, Ruppenhofer and Pinkal (2011), the authors build on the earlier results by clustering also the participants of the events across ESDs. They combine structural cues and semantic similarity in an ILP (integer linear programming) framework to obtain the clustering of participants that satisfies maximum number of constraints. Their algorithm achieves F-score from 74 % to 89 %, depending on the evaluation metric.

Note that the two last mentioned papers demonstrate automatic crystallization of script knowledge from examples in natural language, but rely on high-quality examples to be specified on the input. The gap from an unconstrained body of text to a script is only bypassed by using Amazon Mechanical Turk. That implies that extracting more scripts is not for free; however, for a restricted domain, such as in our project, this approach yields the required results.

1.3 Mapping Visual Data to Text

The problem of matching visual data to text, either in the task of image/video annotation or text illustration, depends on advanced methods of both computer vision and text processing. Therefore, this task could be tackled only recently. Still, the results are only partial, relative to the whole way from video to adequate text description or from text to relevant videos or images. Successful recent approaches rely heavily on meta-information attached or related to pictures rather than actually naming the depicted objects. We first review two different approaches dealing with still images and then one approach which processes video.

The first work, Feng and Lapata (2010), models the relation of visual elements (*visiterms*) to words as direct. The authors use statistical methods to approximate this relation from BBC News Data corpus, which consists of newspaper articles with an attached figure (or more) and its caption. They annotate images with visiterms using the SIFT algorithm (Lowe, 1999). So they actually take key elements extracted by SIFT to be the visiterms. For the text, they assume it is generated by a two-level procedure: first, a topic is randomly chosen, and then the article is generated given the topic (the procedure is known as LDA, Latent Dirichlet Allocation). Then they take topic to be a hidden variable and learn the relation of articles (treated as bags of words) to topics, and also of images (treated as bags of visiterms) to those topics. They arrive at 20 % F1 score within top 10 suggested keywords, but are from the assumptions limited to extracting only sets of keywords as opposed to a fluent description. That indicates that a more sophisticated approach to modeling meaning is needed.

One such approach is represented by Farhadi et al. (2010). Here, the authors explicitly define the meaning (the hidden variable) to be a triple $\langle \text{object}, \text{action}, \text{scene} \rangle$. They exploit these triples for generating simple sentences in the form *object does action at scene*. Again, they learn statistically the two relations, one of text and meaning and another one of meaning and image. In their case, though, for the structured nature of meaning, relations are learned for each of object, action, scene, and between them. They define *tree F1-measure*: given a taxonomy, each predicted entity and corresponding gold standard entity are compared as nodes in the taxonomy. They are identified with the path in the taxonomy tree from their node to the root, path being a set of edges,

and F1 score is computed between the two paths. In this measure and with a corpus of 1000 images they created, they achieve about 40 %–50 % F1 scores for each of objects, actions and scenes. This work is encouraging as it shows a way to building sentential descriptions with a good accuracy. It remains a question for us how well it could be applied to the domain of video.

In a problem conceptually similar to ours, Gupta, Srinivasan, Shi and Davis (2009) automatically extract what they call *storylines* and what in fact corresponds to the notion of script as defined by Schank and Abelson, 1977. The authors take a corpus of recordings of 39 baseball matches and learn to distinguish different actions in the game as well as possible sequences in which they can occur. Because there is no universal linear sequence of actions that can happen in baseball, they capture the structure (storyline) using an AND-OR graph. They use several computer vision techniques to extract features from videos and define several soft constraints to calibrate the complex relation of video segmentation, actions appearance models and the storyline. The best model is then constructed iteratively using an algorithm similar to the EM-algorithm. They achieve impressive results for their problem, with both precision and recall of action recognition around 80 %–90 % and the storyline being very much like what a human would construct. Although they contributed in the same direction as we plan to, their research context is different. Whereas they base their visual model on recognizing medium scale movements (of players running across the field), we aim for analyzing human motion in more detail, in terms of joints and bones. We also focus more on the linguistic side of the problem, not only generating more natural texts, but even being guided by pre-extracted script information. Still, we can draw much inspiration from this work.

Finally, we review the recent work of Rohrbach, Regneri et al., 2012, which is also part of the SMILE project. It explores visual recognition of composite events, an area that has rarely been tackled. This work already joins the forces of computer vision and script information. To allow for these two paradigms to support one another, the computer vision step does not recognise whole events, but rather event attributes – elementary events and their participants. Attributes may be more reliably recognised; however, they generate a vast space of hypotheses for interpretation of the video. The space needs to be bound to feasible dimensions, and this is achieved by restricting the hypotheses by the means of scripts. The advantage of breaking the recognition problem into recognition of single attributes is two-fold: not only do they serve as an interface to information coming from the scripts, but they can also be compared and reinforce or reduce the confidence of each of them being recognised correctly. The authors consider two types of links between extracted attributes: contextual (an attribute occurring in subsequent frames), and co-occurrence (different attributes occurring at the same time). The reported results are decent, considering difficulty of the task, and this work shows that combining computer vision with computational linguistics is possible, and advantageous.

1.4 Other Related Work

We have identified various approaches to problems related to one we want to solve. Besides them, there is also a body of work with results relevant to us, but aimed at different problems. In the related field of software engineering, there has been research in specifications mining, c.f. (Shoham, Yahav, Fink & Pistoia, 2008). That work seems analogical to unsupervised learning of scripts from text on a given topic. Where they speak about code snippets, we would look at

sentences and paragraphs, and their resulting state automaton graphs seem promising as representation for our scripts, too. However, the transfer from the exact domain of programming languages to natural languages is far from straightforward and it is to be seen whether such formal approaches can be useful for the linguistic domain.

1.5 Plan for The Thesis

The rest of this thesis is structured as follows. In Chapter 2, we give a more complete overview of the SMILE project in Section 2.1, and introduce the important concept of alignment in Section 2.2. Chapter 3 describes the corpora that served as the source of linguistic data for our experiment. We also document how the corpora were preprocessed to facilitate subsequent learning of word aligners. Chapter 4 describes the central piece of the present research, i.e. extracting statistical dependencies between features of video descriptions. We describe how the theoretical problem specification is translated into a problem of statistical learning in Section 4.2. We then discuss possible approaches to the problem and present our method in Section 4.3. In Section 4.4, we address the issue of evaluating results of the system to present and discuss the results in Section 4.5. Section 4.6 provides discussion of the present approach, and suggestions for future work in regard of the core dependency classification problem. Finally, Chapter 5 outlines directions for future work on the project and describes what has already been carried out.

Chapter 2

Overview

In the Masters project documented by this thesis, we worked towards improving automated video analysis and understanding. Video understanding is a very intriguing, and challenging task, which brings together computer vision, artificial intelligence, and computational linguistics. Albeit each of the approaches has its own merits, uniting them to solve the task of video understanding is even more attractive.

Imagine we were to automatically analyse a video by applying only methods of visual object recognition. Visual recognition can be reasonably accurate when matching parts of the video input against a fixed dictionary of objects. Using visual context provided by the scene improves the accuracy, and using the change in the visual context as the video proceeds helps further refine the recognition results. We believe that humans use yet more information when recognizing visual percepts. To model this additional information for computer video recognition, we experiment with scripts – prototypical sequences of events from everyday life. Events hypothesized to be captured by a footage are matched against a set of scripts; the sequences of events that have a closely matching script to them are preferred as the interpretation over other candidate event sequences.

From the perspective of computational linguistics, we touched on the motivation for combining forces with computer vision in Section 1.1. Basically, having visual grounding for concepts whose denoting expressions we know provides us with a lot of information about the meaning of the expressions. In fact, computer vision supplements linguistic analyses with information that would never be available otherwise, and that is crucial in human understanding of things. This is analogical to the situation described in the previous paragraph, where computer vision benefits from information obtained using methods of computational linguistics.

The goal of the project goes beyond understanding videos. We aim for generating natural language descriptions of the recognized events. While the descriptions could be produced using sentence templates, filling them with single phrases according to what object was recognized in the video, we want to deliver a more flexible, and thus more natural result. We work towards characterising what human descriptions of videos tend to look like, so that they can be approximated by a computer video recognition system.

In this chapter, we will first describe the SMILE project, as the context of our work, in more detail. We will then introduce some important notions regarding alignment, which will be applied later in Sections 3.2.5, and 5.1.

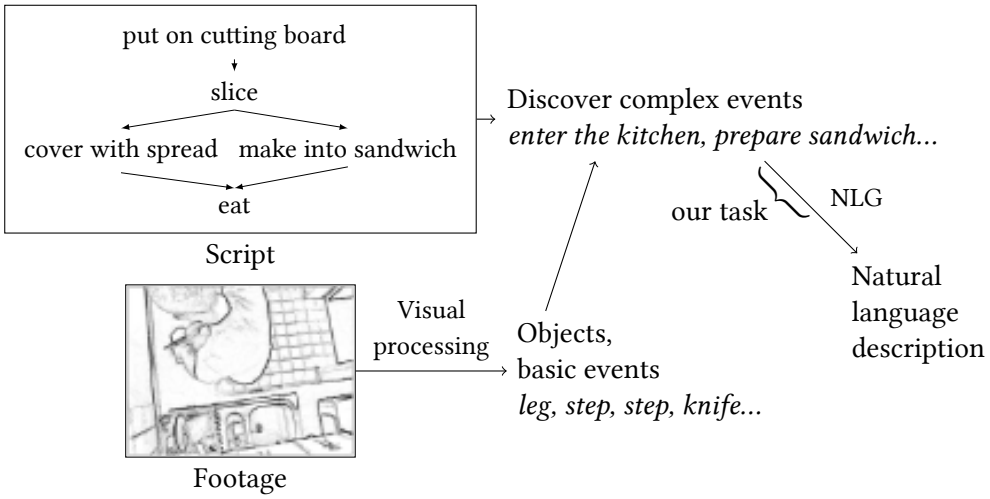

Figure 2.1: Schema of the umbrella project.

2.1 High-level Plan of The Overarching Project

The present research is meant to contribute to a project which we briefly introduce in this section. The project is called SMILE (Script Mining as Internet-based Learning) and it is conducted jointly by researchers in computer vision and computational linguistics at The University of Saarland. The goal of the project is to implement software that would be able to understand videos from the domain of food preparation. Such software could find practical use, e.g. writing down recipes based on a footage capturing a cook performing them, but more importantly, it would show a possible way for other applications of video understanding. (See Aggarwal and Ryoo (2011) for a review.)

In the following paragraphs, we provide a high-level overview of the software that the projects aims to develop. Figure 2.1 illustrates the basic organisation of the software components. There will be two information inputs to the software, script information and a footage. The script information is a static input, meaning it serves as a permanent resource for the software; whereas the footage is the input that gets its corresponding piece of output assigned.

First, the footage needs to be processed using computer vision, which identifies elements shown in the video, on a rather low level of description. We assume that the output of computer vision consist of elementary recognised objects and events.

Here comes the principal motivation for this interdisciplinary project – to take the output further towards more abstract description, which would be understandable to humans. Instead of a sequence of elementary objects and events, such as *leg*, *(one) step*, *(a second) step*, *knife* etc., we thus want to recognise, for instance, the action of *entering the kitchen* and *taking the knife*. In order to be able to recognise such more complex events, script information is required.

Scripts (Schank & Abelson, 1977) by definition describe usual sequences of events that constitute everyday (complex) events, such as visit in a restaurant. Hence, they provide exactly the missing link from the computer vision output to high-level descriptions. In fact, as Rohrbach, Regneri et al. (2012) have shown, this link can be exploited even before processing the output of computer vision. It can aid the object recognition step to select feasible candidates for the

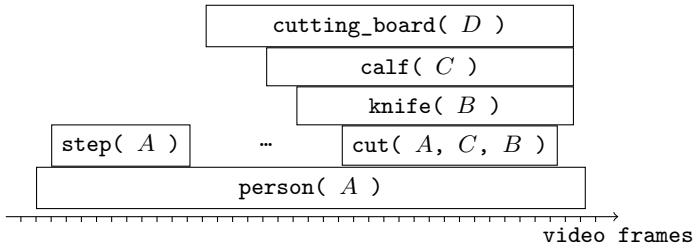

Figure 2.2: An example of the assumed form of computer vision output.

recognised objects or events.

Finally, once the complex events are recognised, they need to be output in an appropriate form. Because we aim at making the extracted information available for a human end user, it would be ideal to output the information in the form of a fluent natural language text.

2.1.1 Video And Script Input

For there to be a real hope for success of the project, the input domain needed to be restricted. The core investigators decided the domain should be food preparation in the kitchen. This choice is well suited to both computer vision processing (it is restricted enough) as well as the application of scripts (food preparation involves a rich variety of scripts).

The script information needed on input to the system was obtained as described in Regneri, Koller and Pinkal (2010) and Regneri, Koller, Ruppenhofer and Pinkal (2011). The necessary video input data have been gathered in an artificial kitchen built for that purpose, as described in Rohrbach, Amin, Andriluka and Schiele (2012).

2.1.2 Nature of Information during The Process

Let us now discuss briefly the nature of information transferred between components of the software system. Figure 2.2 illustrates the outcomes of object and event recognition. Of course, there will be more information extracted from the video, such as position of each recognised object; however, we want to emphasize that there are elementary objects (*cutting board*, *knife*) and elementary events (*step*, *cut*) that occupy a certain span of video frames each. Moreover, there are events or objects recognised wrongly (*calf* instead of *loaf of bread*), which there will presumably be many more in reality. We assume the object recognition will keep track of individual instances of objects, so that its output will imply that, for instance, the person who *makes a step* is the same as the one who *cuts [the bread] with the knife*. However, this assumption is not necessary, as this information can be partly recovered using the scripts.

An example of a script, in a simplified version of Schank and Abelson's notation, is shown in Figure 2.3. The script consists of two parts: the script *header*, and its *body*.[1] The header defines the name of the script and roles figuring in it. The script body, to be read from top to bottom, then defines interactions of the roles that constitute the script. These interactions are expressed

[1] The body of the script, according to Schank and Abelson, should be a conceptual dependency (CD) graph of events and states. Due to the restricted domain and for the sake of simplicity, here we assume the same template for the CD structure of all scripts, where the events always lead to a state that enables the subsequent event.

Name:	Preparing bread
Roles:	cook **c** *(bread **b**, knife **k**, cutting board **p**, spread **s**)*

c PTRANS **b** TO **p** **c** GRASP **k**
c EXPEL **k** TO **b**
c PTRANS **k** TO **s**
c PTRANS **s** TO **b**
c INGEST **b**

Figure 2.3: An example script.

in terms of a very restricted grammar. (Although Schank and Abelson perfected this restricted grammar, we might use a different one in the SMILE project, according to needs of the computer vision component.) In the first line of the body of the example script, we write the two actions besides each other to show that their order is irrelevant. Both of them need to be completed, though, before the action below them can be commenced.

In a procedure central to the SMILE project, the output of computer vision gets combined with the script information to give rise to structured information describing the video. For now, it suffices to assume that the procedure selects scripts that apply to the footage, and *instantiates* them (or, *unifies* the roles in the scripts with the objects recognised in the video). A fraction of the set of unification equalities could look like this:

$$\mathbf{c} = A \qquad \mathbf{k} = B$$
$$\texttt{calf} \sim \texttt{bread} \qquad \mathbf{p} = D$$
$$\mathbf{b} = C \qquad \mathbf{s} = \textit{none}$$

Here, A–D refer to objects in the video recognition output (as shown in Figure 2.2), bold letters refer to roles of the script (Figure 2.3), and words typeset in monospace font (`calf` and `bread`) are predicates.

The above example of unification would assign the recognised objects (and their respective actions) to the roles in the script as expected. Furthermore, it fixes the misrecognition of the bread loaf as a calf, using restrictions implied by the script. Because there figured no spread **s** in the hypothetical video, that role does not get instantiated, and all its related actions in the script have to be ignored.

Once a procedure like the above described unification takes place, the video is understood and the software has captured it as raw structured information. An example of widely used formalism for capturing structured information is shown in Figure 2.4. It is an *attribute-value matrix* (or AVM for short) – on the left are names of attributes and their values are on the right. Each value on itself can be structured, which is captured again using an (embedded) AVM. Small squared numbers co-index information that refer to the same object. The figure shows two AVMs – we assume there to be many AVMs at this point of processing, describing different states that occurred in the video.

The final step of processing is generating natural language text from the raw information. The irrelevant information has to be pruned, and the rest organised into sentences. Following is

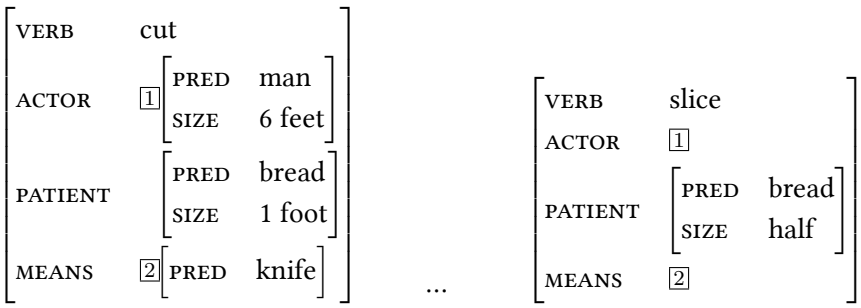

Figure 2.4: An example of the assumed input format for NLG.

an example final output of the system:

Example 1.

A man came to the kitchen.
He took a loaf of bread from the cupboard.
He cut two slices off the bread.
He put the bread back and left the kitchen.

2.1.3 Challenges

By carefully studying the above example output (Example 1), we will now find implications that the desired output has for inner workings of the system.

The first aspect we should note is the *lexical choice*. For instance, in saying

A *man* came to the kitchen.

the speaker had to select from a range of options, including *man*, *chef*, *man six feet tall*, *Barack Obama* etc. In this case, the choice was rather simple. Other cases may be not so clear. Lexical choice is a job of the NLG step, and we will be examining properties of different expressions that can alternate in the same position of video descriptions.

The issue of lexical choice is tightly connected to the issue of *complex events* identification. However, recognising complex events is a matter of understanding, rather than one of linguistic knowledge. It falls under the step of combining scripts with the output of video recognition. And it is an important step to do, since the resulting output for the same video as the one described in Example 1 would otherwise look like this:

Example 2.

A man is walking.
He stops.
He is in the kitchen.
(...)
He cuts a slice off the bread.
He cuts a slice off the bread.
(...)

One last issue we mention here is *tracking discourse referents*. The reader could have noted that in the example output, the same person is referred to first as *a man*, and later only as *he*. All other objects also need to be tracked, so that they can be introduced with the indefinite article (*a loaf*) and later referred to with the definite article (*the bread*). This apparently simple behaviour requires rigorous handling of the discourse structure throughout the system, ideally starting with computer vision, and ending with NLG.

The topic for my Masters project fits into the whole project in the NLG stage. Because NLG is a complex problem, subject to intensive research, the Masters project aims to prepare the grounds for implementing a NLG system in the special context of this video describing application. Concretely, our goal is to characterise *patterns of video descriptions* created by humans. Such characterisation can then aid the NLG component in its effort to generate sentences that sound naturally.

We will now turn attention to the notion of alignment. This is an important device for our analysis of parallel corpora from which we learn to characterise the video descriptions. Only when parallel descriptions (descriptions of the same clip) are word-aligned, can we compare their structure word by word, or, in our case, phrase by phrase.

2.2 Alignment

Alignment is the task of establishing a correspondence relation between linguistic units of *parallel texts*. A number of different texts are said to be parallel iff they convey the same meaning. Two linguistic units are considered to correspond iff they express the same or similar information in the parallel texts they belong to. Although the task of alignment is well specified for any number of parallel texts greater than or equal to two, we will only apply it to the basic case of two parallel texts, hence we need not discuss the general case any further. Even though we happen to work with resources that give us multiple parallel texts, we will treat them piecewise, as a number of *bitexts* (i.e. pairs of parallel texts). For instance, a text available in 10 parallel versions will be broken into $\binom{10}{2} = 45$ bitexts. Also, while alignment can be performed with parallel texts in different languages as well as with texts in the same language, we will only need the latter for our experiments, namely aligning English to English. Even if we employ tools used for SMT (statistical machine translation), we still apply them to the English-to-English "translation" problem. This does no harm, rather the opposite, because monolingual translation is simply much easier a problem than full cross-lingual translation.

The above definition of alignment is generic as to what one takes to be the linguistic unit. Because we speak about correspondence with respect to the conveyed information, the smallest alignable unit is the morpheme, i.e. the smallest linguistic unit that bears some meaning. Usually, however, the task is specified with larger units, most often with words or sentences.

Knowing word alignments of a parallel text is very useful for SMT. They instantiate the hypothetic relation of cross-lingual word correspondence which is traditionally captured by multilingual dictionaries, allowing for the SMT system to induce such a translation dictionary for itself. As opposed to traditional dictionaries though, the word translation dictionaries used in SMT have a confidence assigned to each entry, approximately describing the frequency with which the source word corresponds to the target word.

In the present work, we employ alignment in Section 3.2.5, and Section 5.1. The former

is an extensive experiment with aligning sentences using code developed from scratch. We refine the aligment of parallel corpora as part of their processing. In Section 5.1, we document how word aligner was trained from parallel corpora introduced in Chapter 3. Word alignment is a prerequisite for extending the key experiment (Chapter 4) to larger data.

The task of word alignment typically assumes that the given bitext is already sentence-aligned. This way, the selection of target words that can be aligned with any given source word is restricted enough, thus algorithms usually used for word aligning are still computationally feasible while delivering good results. If a bitext is to be word-aligned when it is not yet sentence-aligned, an alignment algorithm is first run on the sentence level before word alignment is tackled. Since we are going to apply both sentence alignment and word alignment in the present work, we describe the two problems in more detail, including references to standard approaches and terminology in forthcoming sections.

2.2.1 The Other Dimension of Alignment

Before we delve into word vs. sentence alignment, we should look at another important issue regarding the linguistic unit used for alignment. While the distinction of morpheme vs. word vs. sentence is measured along the linear dimension of the text, we can also consider another dimension, from surface linguistic forms through to a deep representation of their meaning. The latter dimension is traditionally thought of as depicted in Figure 2.5, the Vauquois triangle. The left vertex of the triangle represents the source text, the right vertex represents the target text. By following the edges to the top vertex, one gets to a deeper representation of the texts. At the top vertex, the texts are represented by their pure meaning. Because the texts are assumed to have the same meaning, the source and target edges meet there.

This metaphor was originally meant to represent the problem of translation, while we employ it for the problem of alignment. In translation, one searches to find a way in the Vauquois triangle from a given source sentence to the corresponding target sentence. In alignment, on the other hand, we start from both a source and a target sentence to meet in the middle. Not the target sentence, but the correspondence relation is expected on the output.

There are different paths through the triangle that can be taken to find the alignment. The solution can take the shallow way, along the base of the triangle, thus directly establishing pairwise correspondences of the surface forms. As the other extreme, it could go deep on both sides (which means "high" in terms of the triangle) until it reaches the top vertex, where the correspondence is trivial. The alignment, as a correspondence of surface units of the two texts, can then be established by following the path back from the top vertex back down to the two surface versions.

Unfortunately, reaching the top vertex from either side seems practically impossible, and the progress upwards gets harder and less reliable as one is getting further from the surface. On the other hand, the extreme of following the surface is much more popular. In SMT nowadays, systems of this kind actually perform one of the best. For SMT, using deeper representations implies performing an according amount of analysis on the source side, and generation on the target side. For alignment, only analysis on both sides is needed. Luckily, first few steps of analysis are nearly deterministic, perhaps until WSD (word sense disambiguation). Hence, performing such a basic analysis helps shorten the distance for the horizontal step with introducing only little noise by analysis errors.

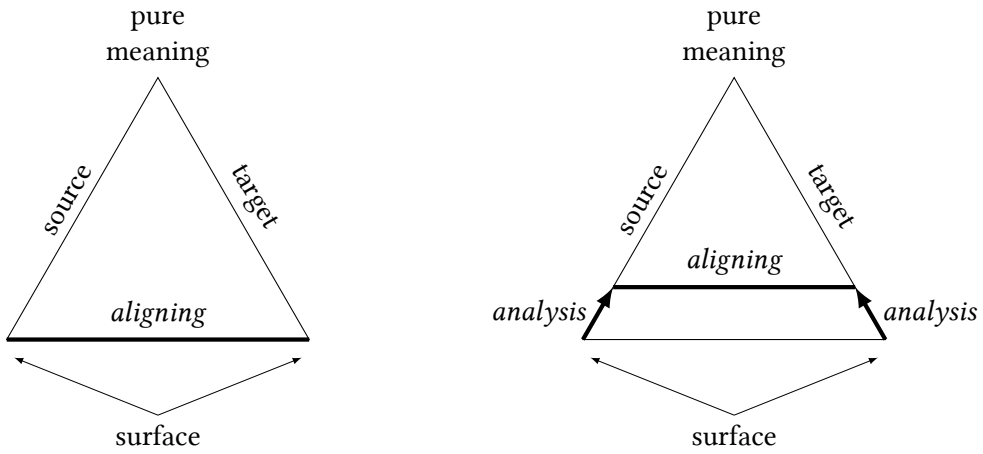

Figure 2.5: Vauquois triangle for the task of alignment. The central step in alignment is the *horizontal step*, the very aligning of linguistic units at a certain level of abstraction. This machine learning task can be performed directly with the surface forms (left); or, it can be applied to more abstract representations (right). In the latter case, the horizontal step is shorter, meaning that the correspondence relation at this level of abstraction is more regular. Note that aligning on the surface level here does *not* mean the text needs not be analyzed at all; we assume it to be already tokenized.

2.2.2 Monotone Alignments

Regarding the horizontal step in the alignment, the problem can be re-articulated in precise mathematical terms. The source and the target text are sequences of linguistic units (e.g. words or sentences), call them S and T, respectively, where $S = (s_1, s_2 \dots s_m)$ and $T = (t_1, t_2 \dots t_n)$. The task is then to determine the correct relation of $\{s_i : 1 \leq i \leq m\} =: S'$ and $\{t_j : 1 \leq j \leq n\} =: T'$, i.e. output the correct $A \subseteq S' \times T'$. Having defined what alignment is, we can provide a definition of *monotone alignment*.

Definition 1 (monotone alignment). Let us have texts S and T and an alignment $A \subseteq S' \times T'$, with S' and T' as defined above. We say that A is *monotone* iff:

1. There exist a suitable equivalence σ on S' factorizing S' into continuous chunks, i.e.:

$$\forall i, k : [s_i]_\sigma = [s_k]_\sigma \rightarrow (\forall j : i < j < k \rightarrow [s_j]_\sigma = [s_i]_\sigma) \,,$$

   and an equivalence τ on T' with the same property.

2. The factorizations $S'/\sigma, T'/\tau$ are both compatible with A:

$$\forall i_1, i_2 \in S',\ j \in T' : [s_{i_1}]_\sigma = [s_{i_2}]_\sigma \rightarrow (\langle i_1, j\rangle \in A \leftrightarrow \langle i_2, j\rangle \in A)$$
$$\forall i \in S',\ j_1, j_2 \in T' : [t_{j_1}]_\tau = [t_{j_2}]_\tau \rightarrow (\langle i, j_1\rangle \in A \leftrightarrow \langle i, j_2\rangle \in A) \,.$$

3. The relation induced by A on $S'/\sigma \times T'/\tau$ is a monotonically increasing partial function:

$$\forall a_1, a_2 \in A : (a_1 = (s_{i_1}, t_{j_1}) \wedge a_2 = (s_{i_2}, t_{j_2})) \rightarrow ([s_{i_1}]_\sigma = [s_{i_2}]_\sigma \vee (i_1 < i_2 \rightarrow j_1 < j_2)) \,.$$

For the monolingual English parallel data we will work with, we observed that most sentence alignments were monotone. This is because all the texts express the same sequence of ideas and are very short, so there is little reason, and little room for sentence reordering. However, with alignment of smaller units including words, non-monotone alignments are often required, because information within a sentence can be moved around more freely. Clearly, the restricted problem of finding monotone alignments is easier than the more general problem of finding any kind of an alignment. However, making the monotonicity assumption is not always adequate.

2.2.3 Sentence Alignment

We start with sentence alignment, since it is conceptionally a simpler task than word alignment, and also because it is more important for understanding some of the following chapters.

Length-based Sentence Alignment

A classic approach to sentence alignment is Gale and Church (1991). They apply their algorithm to sentence aligning parallel texts in English, French, and German, establishing the sentence correspondence merely by measuring the sentence length. They hypothesize that corresponding sentences are likely to be of the same length, with the difference in lengths (multiplied by certain language-dependent coefficients) being normally distributed. They derive a statistic based on the length difference, δ, and show that its distribution is indeed close to the normal distribution in their data. Gale and Church then go on to implement an algorithm for finding a good sentence alignment with respect to the δ statistic.

With the longer texts they have (Swiss economic reports, and Canadian Hansards), they actually run the same algorithm twice – first to align whole paragraphs, and only then to align sentences within the paragraphs. Even so, the number of paragraphs in a text or the number of sentences within a paragraph would be prohibitively large if one allowed for arbitrary many-to-many alignments. Therefore, the authors restrict the possible building blocks for alignment (termed *beads* in Brown, Lai and Mercer (1991)) to at most 2 consecutive sentences on either side.

Having discussed the high-level anatomy of the algorithm and the δ statistic, providing a crude measure of sentence similarity, we can look at the algorithm itself. It is an application of dynamic programming. Given two texts, a table is allocated with one column for each sentence boundary in the first text, and one row for each sentence boundary in the other text. It is then filled in using existing previous values in the table (up to 2 cells back in either dimension) plus the δ statistic of aligning the sentences spanning from the previous table cell to the current one. The best alignment can then be read off from the table starting from its last cell.

We should note that the δ statistic was based on the length of sentences *in characters*, not in tokens (words). The authors compare the two options and conclude that measuring in characters yields much better results. They suggest the reason to be that each sentence has more characters than words, and with higher numbers, the standard deviation is relatively smaller. When we use length-based sentence alignment in our experiments, we will compare the two options for measuring the length to see whether they will yield similarly different results also for our preprocessed data.

Sentence Alignment based on Lexical Overlap

The meager needs of the length-based approach are its main strength. They mean the algorithm is applicable to virtually any pair of languages. Gale and Church, 1991 also show it has very good results, reporting as high as 94.2% and 97.3% accuracy for sentence alignment of English and French, and English and German, respectively. However, these minimal assumptions also imply the main limitation of the approach. It represents each sentence by a single number; rather than attempting to capture the information conveyed by a sentence, it merely approximates the *amount* of information. The approximation by length in characters builds on the assumption that information is evenly distributed over characters of sentences in the given language, which is generally true. However, in the cases where Gale and Church's program gives the wrong answer, it is probably because there is a lack of cues about the information contained in the given sentences, not because of suboptimal processing of the cues it looks at. To make possible better results than those achievable by length-based approaches, there were developed another class of approaches – those representing sentences by their lexical content.

The basic idea used in the approaches based on lexical information is measuring sentence similarity by their lexical overlap, usually taking the sentence length into account, too. Lexical-based algorithms then use the lexical (and length) similarity information for individual units of those to optimize the total similarity for all the aligned units.

A good example of a lexical-based alignment algorithm is introduced by Melamed, 1999. The paper also provides a more elaborate overview of different approaches to alignment than we need here; the curious reader is referred there. Melamed introduces a number of useful concepts regarding alignment, and we will briefly review them in the following section for we will need them later in Section 3.2.5.

2.2.4 Formal Notions regarding Alignment

Although the definitions in this section introduce terms originally defined in Melamed (1999), we often use the same terms with a slightly different meaning. Still we think it better to use the same terms, since in essence, they denote the same things.

Any particular problem of alignment is always specified with respect to a bitext space, so we start by defining this one. For an example of a bitext space, see Figure 2.6.

Definition 2 (bitext space). For a given bitext consisting of texts S and T, the *bitext space* $\mathcal{B}(S, T)$ is a rectangle of all pairs of indices to characters or tokens of the two parallel texts. Whether it is characters or tokens that are indexed for a particular bitext space, can be distinguished by saying in what units *length is measured*.

Definition 3 (point of correspondence (PC)). *Point of correspondence* $\langle i, j \rangle$ within a bitext $\mathcal{B}(S, T)$ is a pair of indices either of elements of S and T, respectively, or of boundaries between elements of S and T, respectively. Boundaries of a sequence $(x_1 \ldots x_n)$ are indexed from 0 through to n.

The last definition deserves some elaboration. As defined earlier in this section, we understand a text to be a sequence of linguistic units. In the general specification of alignment of S and T, we consider any relation between the sets of S and T to be a valid candidate for alignment. In that case, the PCs based on *indices of elements* form the minimal sufficient set for selecting any alignment from. However, we may restrict the alignment to the kind often seen with sentence

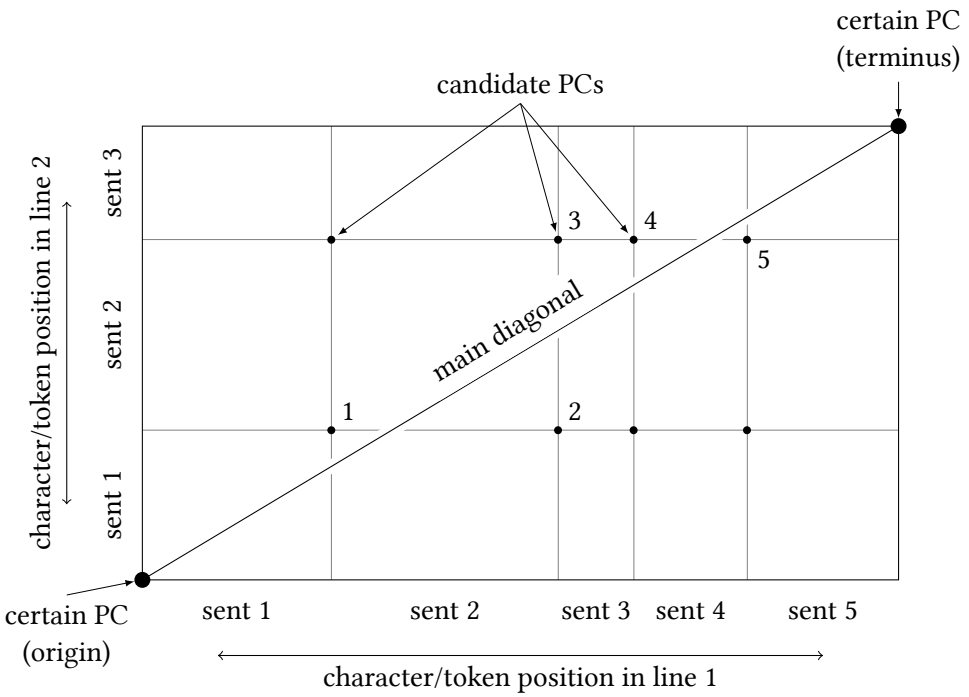

Figure 2.6: An example of a bitext space. On the x-axis is the first text, the other is on the y-axis. Both the axes can be measured in characters or in tokens. The bitext space then consists of all the pairs of character (resp. token) index pairs, and is represented by the rectangle.

In this example, the linguistic unit for alignment is a sentence and PCs are pairs of boundary indices. The first PC is termed *origin*, the last one *terminus*, and the line connecting them *main diagonal*. The main diagonal is important, because PCs closer to it are empirically more likely to belong to the true alignment, especially so in the monolingual case. The origin and the terminus are certain PCs for they must be in any valid alignment. The candidate PCs numbered from 1 to 5 denote the sequence of PCs in the order they are considered by Algorithm 2 (Section 3.2.5, page 35).

alignment, namely monotone alignment. In this case, many of the general alignments are not permissible anymore. In fact, exactly all permissible alignments can be captured as a set of boundary index pairs ordered into a non-decreasing sequence, and then *boundary indexing* is desirable.

We have defined the space of solutions, the bitext space[2], and its elements, points of correspondence. Now we need to state what the right solutions look like and how to evaluate any suggested solution. While we could be exact and define all the necessary concepts precisely, the definitions would take a few pages without being crucial for the present work. Instead, we give a cursory overview of these concepts and refer the reader to Koehn (2007, pp. 113–119), and Melamed (1999) for more details.

According to the definitions above, any alignment is a set of PCs. The PCs constituting the true alignment are called *true points of correspondence* (TPC), and the true alignment itself is called the *true bitext map* (TBM). If PCs are taken to be based on indexing of elements, we can evaluate the accuracy, precision, or recall of any given candidate alignment against a gold standard alignment by comparing them as sets of PCs. A small adjustment is appropriate, though – since any valid alignment contains both the origin and the terminus, these should not count neither towards none of the scores. The evaluation is somewhat more complicated if PCs are based on indexing of element boundaries. In that case, the alignments must be interpreted as aligning the chunks of text between consecutive PCs, rather than PCs themselves. If there is just one gold standard alignment, evaluation scores can be computed as usual, only with the beads between PCs rather than PCs themselves. The most complicated case is evaluation with PCs based on boundary indexing and with multiple gold standard alignments. This case is detailed in Figure 2.7.

Example 3. This example illustrates how an alignment would be evaluated for the complex case of boundary indexing and multiple gold standard alignments (TBMs). All the alignments are shown in Figure 2.7.

In this hypothetic case, we view the two notions of precision and recall in a different context. When evaluating precision, we consider the *beads* induced by the alignments and compare sets thereof. (Beads are the objects represented by the hatched rectangles for the TBMs.) The beads induced by the candidate alignment are origin–A, A–B, B–C, C–D, D–E, and E–terminus. Three of these five beads are compatible with a TBM (B–C, D–E, and E–terminus), hence the precision is $3/5 = 60$ %. We understand a bead to be compatible with a TBM iff after removing zero or more PCs from the TBM, it contains the bead. Note that we do not require the beads to be compatible with the same TBM – the beads B–C and D–E come from different TBMs.

For recall, we view the problem as a classification of PCs – whether they should be selected or not. We then compare the set of selected PCs barring the origin and terminus (i.e. $\{A, B, C, D, E\}$ in this case; call them *inner PCs*) to the minimal set of inner true PCs covering maximum number of the selected PCs. The maximum number of the selected PCs that can be covered by some of the TBMs in our case is 4, for all of B, C, D, and E belong to some of the TBMs, while A does not. There is one minimal set of TBMs covering all of B, C, D and E, namely both the TBMs. That implies the minimal set of inner TPCs, which has the size 8, in turn implying that the recall is $4/8 = 50$ %.

The accuracy of the alignment must be evaluated differently to the precision. It is easy

[2]In some cases, not all subsets of the bitext space are considered to be valid solutions. For instance, for aligning monotonically while prohibiting to align anything to an empty fragment, each solution has to contain the origin, the terminus, and must be strictly monotone (as a function).

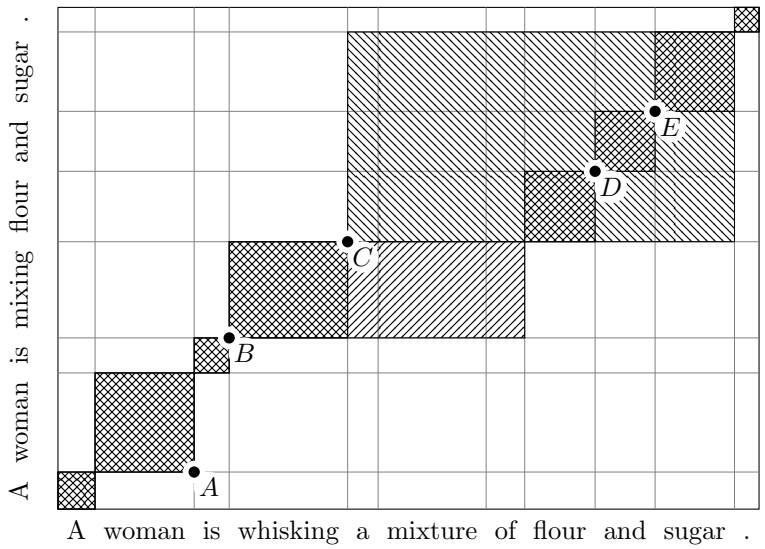

Figure 2.7: Evaluation in case of multiple gold standard alignments (or, TBMs). We align words in this example, since it is hard if not impossible to find a similar example for sentence alignment. The assumption of a monotone alignment remains, though, as the PCs reside at pairs of indices of *boundaries*, not the units themselves. The two TBMs are represented by the rectangles with two types of hatching. The candidate alignment comprises PCs A, B, C, D and E, plus the obligatory origin and terminus.

to achieve 100% precision (with the trivial alignment {origin, terminus}). Unlike precision though, accuracy is not complemented by another metric; it should reflect non-perfect alignments by assigning them a lower score. Therefore, accuracy would be computed by comparing the set of *exactly* recovered beads ($\mathcal{B}_R$) to the set of beads of the minimal number of TBMs covering all of $\mathcal{B}_R$. In our example, the beads B–C and D–E are the only ones recovered exactly (E–terminus should have been split). The minimal number of TBMs covering them are again both the TBMs, comprising 10 beads in total. Therefore, the accuracy is 20 %.

2.2.5 Word Alignment

As noted above in the general introduction to alignment, the problem of word alignment is an essential one in SMT. In contrast to sentence alignment, there are no shortcuts for solving this problem like comparing the length of words (whereas aligning sentences by comparing lengths of sentences gave good results, viz Section 2.2.3). Usually, the only input is a parallel sentence-aligned corpus. The traditional word alignment algorithms, which are still used widely nowadays, are based on plain counting co-occurrences of words from one side (language) of the corpus with words from the other side.

In our view, word alignment is a harder task than sentence alignment, for the following reasons:

1. Sentence alignment can be restricted to monotone alignments in some cases (including our monolingual alignments) and still yield quite accurate results. In many other cases, it is close

to monotone. Word alignment typically abounds in complex reorderings.

2. Sentences can be analyzed into single tokens, allowing for lexical-based alignment methods (Section 2.2.3). Although one can employ a similar heuristic for word alignment, namely prefer aligning words to the same or similar words, even in a multilingual setting (cf. the approach of Melamed (1999)), that cannot work as a universal strategy.

Because the word alignment algorithms rely on co-occurrence counting and they solve a harder problem than sentence alignment, we want to emphasize again that it is important to split a parallel corpus into chunks as small as possible before passing it on to word alignment.

IBM models

To our knowledge, the need for word alignment was first formulated by Brown, Cocke et al., 1988. This is pioneering work in SMT, proposing a generative approach to translation, detailing its central part, namely extracting a probabilistic dictionary. They proposed a model of translation including word translation probabilities and word fertility, making only preliminary experiments with word reordering (all these terms are explained below). Couple of years later, they organized the ideas into one framework, and thus laid the basics for contemporary SMT. This framework, presented in Brown, Pietra, Pietra and Mercer (1993), consists of five models, now called "IBM models".

IBM models model how a sentence from the source language gets translated word by word into a sentence in the target language. They range from very simple to complex and potentially very accurate ones. With increasing complexity of the model, it captures more sophisticated translation phenomena, but also training and applying the model becomes increasingly complex. The reason why there are five models, and typically most of them get used in any SMT problem, is that each simpler model can be used to initialize parameters for the more complex one, making use of the more complex models feasible.

IBM Model 1 is the starting point. It only includes *word translation probabilities*, i.e. probabilities in the form $p(e|f)$ where e is a target language (English) word, and f is a source language (French) word. It asserts that each target word is aligned to (was translated from) exactly one source word or the imaginary NULL word. Each sentence pair is assumed to be governed by a word alignment satisfying this assumption, however the alignment is hidden to us. If we knew the alignment, it could be used to maximize the word translation probabilities; and if we knew the word translation probabilities, we could estimate the probability distribution over word alignments. This is an instance of a problem with a hidden variable that is best solved using the EM (estimation-maximization) algorithm (Dempster, Laird & Rubin, 1977).

IBM Model 1 is trained using the EM algorithm, under the assumption that alignment of each target word is uniformly distributed over all options (i.e. all the source words and the NULL word). IBM Model 2 is also trained using the EM algorithm, but makes the alignment of each word conditioned on positions of both the source and target word, and on the lengths of both the sentences.

IBM Model 3 substantially changes the generative story from the previous models. Most importantly, it includes the notion of *word fertility*. Given a source and target sentence, each source word is assumed to have some fertility, which is a natural number (including 0). The fertility value determines how many target words are aligned to the source word. To keep the possibility for

target words to have no corresponding words in the source sentence, the next generative step is insertion of NULL words. The lexical translation, $p(e|f)$, turning all the source words into target language words, is followed by the final step, *word reordering* or *distortion*. Because this model cannot be factored into probabilities for individual target words (but rather it needs to always be evaluated with a complete alignment during training), EM algorithm is not applicable in its original form. Instead of performing full maximization in the M-step, the parameters are maximized only based on a sample of the most likely alignments.

While IBM Model 3 employed a simple *absolute reordering model* (the same as IBM Model 2), the next model gets more realistic in this aspect. The position of a target word in the sentence in reality tends to be influenced much more by the position of, say, the word translated from the preceding source word, than by its absolute position. At the same time, it depends on what kind of a word it is and what kind was the preceding source word (whether a noun, an adjective, etc.). This is needed for instance to capture the phenomenon of different order of a noun and its attributes in English and in French, as in *International Phonetic Alphabet* vs. *Alphabet phonétique international*. Similarly to the previous model, IBM Model 4 needs to only sample likely alignments for training.

Finally, IBM Model 5 fixes deficiencies present in models 3 and 4, where the probability of all possible target sentences does not add to 1. This model is the most complex one. However, it is often *not* used, in favour of IBM Model 4, which is already good enough for practical purposes.

To conclude our discussion of the IBM models, we should say their main goal was translation, word alignments being a byproduct. Nowadays, though, they are mainly used just for finding word alignments, notably so in phrase-based SMT. The curious reader can find a textbook exposition of IBM models in Koehn (2007, pp. 81–113).

Berkeley jointly trained models

In this section, we briefly introduce a word alignment tool we actually used in our experiments. It is presented in Liang, Taskar and Klein (2006), the motivation for it relying largely on *symmetricity* of the alignment. Symmetricity here refers to the way the alignment was obtained, not to a property of the alignment itself. The attentive reader will have noticed that the IBM models are inherently asymmetric – they are directed from one language to the other one. Furthermore, they are restricted to consider only one-to-many alignments, hence the set of permitted alignments for one direction is different from the set of permitted alignments for the other direction. Assuming the true alignment can be a combination of alignments obtained for the two directions, it is typically computed exactly that way – IBM models are trained in both directions, asked to each predict an alignment for the given sentence pair, and these two alignments are somehow combined. The combination can range from intersection of the two to their union; typically, it is something in the middle, obtained using heuristics.

Because the IBM models are iterated on alignments other than those output as final, their training is missing on the potential of symmetrising. They use wrong alignments for the whole training, and find out only when symmetrised on a single piece of test data. Then it is too late to change the model. This line of thought was behind the development of the Berkeley aligner. It has two major characteristics:

1. It accounts for symmetrisation between training iterations.

2. It is simpler than the more complex IBM models.

The Berkeley aligner employs a cascade of increasingly complex models, similarly to the IBM models. By default, it trains the following models: IBM Model 1, IBM Model 2, HMM model (Vogel, Ney & Tillmann, 1996). The last one has not been mentioned before; its distinguishing property is the *relative reordering model* – it includes the probability of a target word position conditioned on the position of the translate of the previous source word. Importantly, models for both directions are trained simultaneously. After completing the M-step of the EM algorithm[3] for both the directions, their results are put together and effectively symmetrised in the E-step. The authors showed that this procedure outperforms even the most sophisticated IBM models (namely, IBM Model 4).

[3]In fact, it is not an EM algorithm anymore. It performs conceptually the same steps (estimation followed by maximization), but the estimation is only approximated. It is not even guaranteed to improve the objective function, hence the algorithm is not guaranteed to converge. Nevertheless, it works well in practice.

Chapter 3

Resources

In this section, we provide an overview of and characterize data resources that were used in this research project. We introduce the following corpora:

1. Microsoft Research Video Description Corpus (MSR VDC) provides us with human-made video descriptions. This is the corpus of sentences we ultimately aim to characterize. It is also used in learning to word-align.

2. Multiple Translation Arabic (MTA) and Multiple Translation Chinese (MTC) corpora are parallel monolingual corpora that we use to learn to word-align.

3. Gazetteers and other proper name lists were used in truecasing of MTA and MTC.

3.1 Microsoft Research Video Description Corpus (MSR VDC)

Microsoft Research Video Description Corpus (hereafter, MSR VDC), served as the principal data resource for our investigations. Its authors give some of its characteristics and document how it was obtained in Chen and Dolan, 2011. Here, we will describe properties of the corpus relevant to our research and document how we preprocessed it.

3.1.1 Characteristics

MSR VDC features over 122K one-sentence descriptions of short video clips, written by human workers at Amazon Mechanical Turk.[1] The descriptions are based on 2,089 different video clips, which implies that each clip received a decently high number of descriptions. Hence this corpus is a very good resource of parallel sentences, and we take advantage of it when creating training data for word alignment.

MSR VDC is also an ideal corpus for the purpose of our present experiment. Firstly, it consists of one-sentence video descriptions, and secondly, a large portion of the video clips come from the domain of food preparation. We thus work with sentences, many of which could be the ideal output of the target software system.

[1] http://www.mturk.com

1. A man is driving a car.
2. A man is doing a magic trick with cards.
3. A man is playing a guitar.
4. A woman is peeling a potato.
5. "A man kicks a soccer ball into a goal, and a soccer ball knocks down a camera and tripod, barely missing the man who jumps out from behind it."

Table 3.1: Sample sentences from MSR VDC.

We list a few example sentences from the corpus in Table 3.1. These illustrate not only the constitution of the corpus itself, but also what kind of videos are described in the corpus. The first sentence is description of one of the shortest videos present in the corpus (2 seconds), whereas the second one shows a description of the longest video (49 seconds). Following is the most frequent sentence in the corpus (used 176 times), compared to 36 occurrences of the most frequent sentence from the domain of kitchen (number 4). Note that large portion of the corpus describes videos either about a performing musician (such as the third one in Table 3.1), or about work in the kitchen. The last sentence shown in Table 3.1 is one of the longest sentences present in the corpus.

Before starting to use it, we split the corpus into three parts – training, test known, and test unknown. They purpose is the following:

train This is the portion that we use for our experiments. It is meant for training any later machine learners when someone builds on the present work.

test – known This is the portion of the corpus meant for testing machine learners on data that are familiar to them. Clips with corresponding descriptions in the "test – known" have majority of their descriptions included in the "train" portion of the corpus, and none in the "test – unknown" portion. "Test – known" is also meant to be taken into account in training machine learners, even though not as regular training data. For example, we use it for word aligning in that we construct the global vocabulary from both the "train" and "test – known" parts.

test – unknown This portion is reserved for future purposes, especially for evaluating any trained systems on data that may be new to them, and still from the target domain. We do not use these descriptions nor any parallel descriptions for the same clips in any way, and suggest anyone who does a research related to the present work to not use it for training either.

We split the corpus into the three portions following these simple rules:

1. Sentences number 42,363 and later form the "test – unknown" part.
2. For the rest of the corpus, we proceed clip by clip. For the first, third, fifth etc. clip, we put the last 3 of its descriptions into the "test – known" portion, and the rest to "train". For the

second, fourth etc. clip, we put only the 2 last clips to the "test – known" portion, the rest to "train".

This results in the three parts of the corpus having the following number of descriptions: In the "% of MSR VDC" column, we compare the number of corresponding descriptions to the total number of descriptions in the *used part of MSR VDC* only.

	# sents	% of MSR VDC
train	37,400	83.9%
test – known	4,960	11.1%
test – unknown	2,224	5.0%

Table 3.2: The split of MSR VDC sentences into "train", "test – known", and "test – unknown".

3.1.2 Filtering Descriptions by Quality

Since authors of the descriptions were asked to write them in the language of their choice, 30% of the descriptions are in other languages than English. However, we only looked at the English descriptions. Furthermore, Chen and Dolan manually evaluated some of the responses gathered from the workers and marked those that had a high quality. Let us call such (English only) descriptions *clean*. The other descriptions are marked as *unverified*. However, during processing of the corpus, we collected *unverified* descriptions from workers who also produced *clean* descriptions. Let us call such descriptions *unverified, author mixed*. We discovered that there is virtually no noticeable difference in quality of these descriptions and the *clean* ones, hence we added them to the set of *clean* descriptions and used both.

Despite the fact that the *clean* descriptions were deemed to be correct English by Chen and Dolan, and the *unverified* ones with *author mixed* were produced by someone whose later descriptions were deemed *clean*, we checked the set of sentences we would process manually and cleaned them ourselves. We identified 7 sentences that were not English and other 2 defect sentences (which explains the difference between 33,855 + 10,742 and 44,590 in Table 3.3; all the 9 sentences were *clean*, i.e. marked as English and clean in the corpus). We further ran a spellchecker (Hunspell v1.3.2) on the corpus and corrected typos it identified, and occasionally corrected other errors, randomly discovered.

part of the corpus	number of items
video clips	2,089
all descriptions	122,665
descriptions marked English	85,550
of which *clean* descriptions	33,855
of which *unverified* descriptions, *author mixed*	10,740
descriptions used for experiments	44,584

Table 3.3: Overview of Microsoft Research Video Description Corpus (MSR VDC)

3.1.3 Corpus Preprocessing

We analysed the corpus to abstract from the surface to more general level of linguistic description, in order to facilitate word alignment, as motivated in Section 2.2.1. We applied Stanford CoreNLP and NLTK to obtain lemmatised and stemmed versions of the original sentence, with substituted coreferring expressions, and annotated with syntactic parses. The procedure was parallel to the processing of the corpora described in the following section, so we refer the reader to the text on 27 and in Section 3.2.4 for a detailed description.

3.2 Multiple Translation Arabic Corpus (MTA) And Multiple Translation Chinese Corpus (MTC)

Multiple Translation Arabic Corpus (MTA) and Multiple Translation Chinese Corpus (MTC) are corpora of English paraphrases. MTA and MTC comprise independent English translations of Arabic and Chinese short newswire articles, respectively. Although the corpora authors identify for each (English) article in the corpus which Arabic or Chinese article it originated from, we ignore all aspects of the original texts. We only look at the multiple English translations, very much like in the case of MSR VDC where video clips were the source (that we ignored), and their descriptions were the multiple translations into English (which we used as a parallel corpus).

We describe the corpora together in one section for several reasons:

- They were created by the same procedure.
- In the result, they are of a similar nature.
- We preprocessed them together in the same way.
- We used them for the same purpose.

For what purpose we used the corpora was outlined in the preceding paragraph. For the other three points, we will touch on them in sequence.

MTA and MTC consist of translations created by several translators and machine translation (MT) systems. Because the quality of the machine translations is much lower than that of the translations produced by humans, we restrict ourselves to using only the human produced translations. MTA was published in two parts (Walker et al., 2003; Ma, 2005), MTC in four (Huang, Graff & Doddington, 2002; Huang, Graff, Walker et al., 2003; Ma, 2004, 2006). All parts have the same format but they were generally created by different translation teams and MT systems. Regarding the format, the corpus is composed of a number of files, one file for each translated version of an original text. All versions of the same text have the same number of lines, and the corresponding lines always come from the same segment of the source text, i.e. they are asserted to convey the same meaning. When referring to these corpora, we will use the term *line* for the lines as just described.

Tables 3.4 and 3.5 provide basic figures about MTA and MTC, respectively.[2] "Number of regular sentences" states the number of sentences other than headlines and special endings

[2]The field "total" in Tables 3.4 and 3.5 in the row "number of translators" is parenthesized, because some teams or MT systems that participated in creating different parts were actually the same or similar.

	part 1	part 2	total
number of translators			
human	10	4	(14)
MT	3	3	(6)
number of texts			
source texts	141	100	241
human translations	1,410	400	1,810
number of lines used	10,430	2,652	13,082
number of sentences used	13,747	3,192	16,939
regular sentences	11,645	2,850	14,495
number of line pairs	46,935	3,978	50,913
number of sentence pairs	47,999	4,034	52,033

Table 3.4: Overview of Multiple-Translation Arabic (MTA) Corpus

	part 1	part 2	part 3	part 4	total
number of translators					
human	11	4	4	4	(23)
MT	6	6	0	11	(23)
number of texts					
source texts	105	100	100	100	405
human translations	1,155	400	400	400	2,355
number of lines used	10,923	3,512	3,740	3,676	21,851
number of sentences used	13,578	4,584	4,935	4,819	27,916
regular sentences	13,035	4,248	4,606	4,339	26,228
number of line pairs	54,615	5,268	5,610	5,514	71,007
number of sentence pairs	58,743	5,667	6,202	5,993	76,605

Table 3.5: Overview of Multiple-Translation Chinese (MTC) Corpus

(explained later; cf. Table 3.8). "Number of line pairs" refers to lines that correspond to each other in meaning, as implied by the corpus, and "number of sentence pairs" refers to pairs of regular sentences that correspond to each other in meaning, as implied by our best performing sentence aligner. Table 3.6 specifies the grand totals for MTA and MTC.

To underline the different nature of sentences from MTA and MTC in contrast to descriptions from MSR VDC, we provide Table 3.7. The table shows the lengths of sentences, short for MSR VDC and much longer for MTA and MTC; and the variability of the vocabulary, small for VDC and larger for MTA and MTC.

The aim of preprocessing MTA and MTC was to abstract somewhat from the surface forms of tokens to make subsequent alignment easier. The rationale is discussed at more length in Section 2.2.1. The following paragraphs detail all the preprocessing steps we have done.

We obtained the corpora from Trevor Cohn, who kindly provided them in a pre-tokenized form. But since the tokenisation in Cohn's copy of the corpora was imperfect in places, we tweaked

	# of texts		# of lines	# of sentences	# of sentence pairs
	source	translations			
MTA	241	1,810	13,082	16,939	52,033
MTC	405	2,355	21,851	27,916	76,605
sum	646	4,165	34,933	44,855	128,638

Table 3.6: Total numbers for MTA and MTC. We only consider human-produced translations in this table.

	tokens	tok/sent	types
MSR VDC	388,204	8.71	4,687
MTA+MTC	1,072,386	23.91	20,070

Table 3.7: Size of MSR VDC vs. MTA and MTC in words.

the tokenisation first. Figure 3.1 (A) shows an example sentence[3] from this corpus. We chose this sentence such that it contains several tokenisation errors typically encountered in the data, although they are generally not as frequent as here.

The next step was truecasing.[4] After that, we segmented lines, which often comprise two or three sentences, into individual sentences. After these steps, we obtained sentences like those in Figure 3.1 (B) and (C)[5] (the numbers on the left indicate the sentences into which the original line was split).

Then, we processed the sentences in their original context using Stanford CoreNLP for POS-tagging (Toutanova & Manning, 2000; Toutanova, Klein, Manning & Singer, 2003), lemmatisation, named entity recognition (NER) (Finkel, Grenager & Manning, 2005), parsing (Klein & Manning, 2002), and coreference resolution (Raghunathan et al., 2010; H. Lee et al., 2011). For this, we used a patched 2012-03-09 release of the software. Most of the information obtained by the CoreNLP analyses were actually used for sentence and word aligning, including lemmata, parses, and resolved coreference chains. POS tags and NER were exploited by subsequent CoreNLP components.

We made one more step beyond lemmatisation, namely stemming of the lemmata using the NLTK (Bird, Klein & Loper, 2009) implementation of the Porter stemming algorithm (Porter, 1997).

Finally, for every two corresponding lines, we sentence-aligned them. For the lines shown in Figure 3.1 (B) and (C), they got aligned in the following way. The first sentence of (B) (0 *New York Times* -) was matched against a pattern for special sentence starts, hence it was not aligned with other regular sentences. It is probably a mistake that the same words in the latter line are considered a regular sentence, even though they are formally really written so. In any case,

[3] line 7 of the MTA file `part1.ahd/artb_535.tok`

[4] Here, by the term *truecasing*, we mean lowercasing letters in the word-initial position if the word is not a proper name.

[5] This is another version (by another translator) of the same original Arabic fragment, from the file `part1.ahg/artb_535.tok` of MTA.

(A)

New York Times - A British government document published yesterday reported that in the days preceding the September 11 terrorist attack on the United States* , Osama Bin Laden indicated that he was " about to carry out a massive attack against America"._____Actually says " United Nations " in the original

(B)

0 New York Times -
1 A British government document published yesterday reported that in the days preceding the September 11 terrorist attack on the United States * , Osama Bin Laden indicated that he was " about to carry out a massive attack against America " .
2 _____ actually says " United Nations " in the original .

(C)

1 New York Times .
2 a document published by the British Government yesterday evening (Thursday) reveals that in the days before the attacks of September 11 , Osama Bin Laden made it clear that he intends to launch a major attack against the United States .

Figure 3.1: An example of lines from MTA/MTC in different phases of preprocessing.

the two last sentences of (B) vs. the two sentences of (C) form the bitext for sentence alignment. In all our sentence alignment algorithms, they get aligned correctly as a single 2-2 bead.

As the result, we created a corpus of pairs of corresponding fragments (one or more sentences) annotated with stems and parse trees. Thanks to the original corpora providing many parallel versions of the texts, we got a decent number of pairwise sentence correspondences (128,638 pairs, created from 44,855 sentences).

3.2.1 Tokenisation Tweaking

The corpus, as provided by Trevor Cohn, was reasonably well tokenized. For instance, the negation particle *n't*, as in *won't*, was split off from the auxiliary verb, and punctuation was separated into individual tokens, distinguishing between the sentence-final fullstops (separated) and fullstops used with abbreviations (kept together with the abbreviation; e.g. *Ltd.* or *U.S.*). However, the punctuation was from time to time left attached to the preceding, or even the following word (cf. Figure 3.1 (A)).

We iteratively searched the corpus for tokenisation errors and developed a script based on regular expression substitution, to fix those errors. Apart from fixing obvious errors, we hypothesized additional token divisions that might lead to extracting interesting lexical correspondences in later stages. An example is splitting numeric values on thousand delimiters, so that e.g. *US$400,000* is split into *US $ 400 ,000*.

3.2.2 Truecasing

This step was the next natural thing to do on the way up the Vauquois triangle, in the way of stripping idiosyncratic irregularities. We especially aimed to resolve whether the initial capital

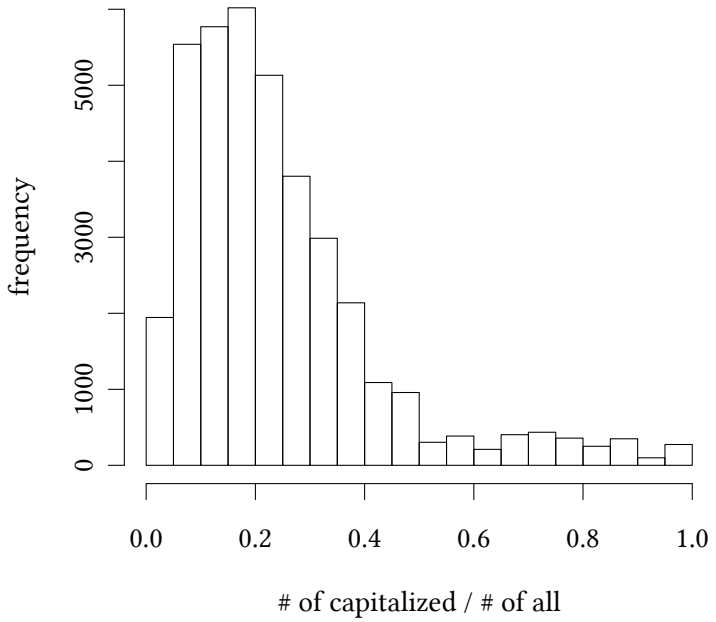

Figure 3.2: Histogram of the ratio of capitalized to all words in a sentence

letter in a sentence belongs to a proper name or another word, lowercasing it in the latter case. Besides that, the corpora contained in part headlines, i.e. sentences with each open-class word capitalized. Cf. Figure 3.2 – it shows that ratio of the number of capitalized words to the number of all words is centred around the mode (approx. 0.2), corresponding to common sentences with the first capitalized letter and a few proper names; however, a much smaller but still non-zero probability mass is far from the mode, centered about approx. 0.75, corresponding to headlines. That illustrates that headlines constitute a non-negligible part of the corpus. For that reason, we set a threshold on the ratio (although the histogram would suggest the two cases border at the ratio of 0.5, we set the threshold to 0.6 to select headlines with higher precision), and for sentences with the ratio exceeding the threshold, every word was checked for lowercasing. For sentences with the ratio below the threshold, only the initial word and words following a punctuation mark were considered.

To decide whether a word is (part of) a proper name, we used dictionaries of proper names. Each candidate word for lowercasing is lowercased, unless it is listed in one of the dictionaries. The dictionaries we used are described in Section 3.3. Since actual proper names appearing in MTA and MTC were often not listed in the static dictionaries, for each text, we additionally built a list of phrases assumed to be proper names and used it as another dictionary for versions of the text.[6]

[6]By versions of one text, we mean different translations of one original Arabic/Chinese text.

The truecasing script, whose pseudocode is provided as Algorithm 1, was run in two iterations. In the first iteration (lines 5–9), only the static dictionaries were used to lowercase candidates . In the second iteration (lines 10–12), both the static dictionaries and the customized one (built in the first iteration) were merged and used to lowercase the text.

The words to be considered for truecasing were selected differently for headlines and other lines. If a line consisted of more than 60% capitalized words (this ratio having been discussed above), and if it had at least 4 words, it was considered a headline. For headlines, all words were considered for lowercasing, while for other lines, it was only words that could have been written with an uppercase initial because of a preceding punctuation. By comparing samples from the corpora, we found that words can get capitalized after most punctuation marks, apart from a few exceptions (“,” (comma), “’” (apostrophe), “/” (slash), and “\” (backslash)).

Having identified the words that might need lowercasing (on lines 15–18), we checked in sequence whether they belong to a capitalized phrase within the line, that is listed among the known proper name phrases (line 20). If they did not belong to such a phrase, they got lowercased.

The `getPropPhrases` function scans a text for maximally long sequences of words that follow the format of a capitalized phrase. In English orthography, it is possible to identify capitalized phrases from having each open-class word and the first word capitalized. That is the definition we used for recognizing them in this function.

We do not consider truecasing a critical step of preprocessing, so we were content when the script normalized most of the capitalized words, not making silly mistakes, even though it was far from perfect.

3.2.3 Sentence Tokenisation

We split lines into sentences using a straightforward Perl script, based on regular expressions. There was only one non-standard issue we had to deal with. With corpora based on newswire articles, many lines are introduced by a so-called dateline – a header specifying details where the news report comes from and when it came into being. The following is a typical example of a sentence starting with a dateline:

Example 4. *AL Ghardagah (Egypt) 5 - 5 (AFP)* - a ship displaying the Maltese flag caused damages , initially estimated at four million dollars , to the coral reefs in the Red Sea , according to a confirmation by the Egyptian police on Tuesday .

Because each sentence was subsequently fed into the Stanford parser, which is trained to parse single sentences, it was desirable to split off datelines, to avoid confusing the parser. In spite of the datelines varying wildly in their form (cf. Table 3.8), they are very restricted as to the content. Due to that, we were able to enumerate a few types of content typical for headlines (the place, the date, the agency, and the reporter), characterize them by regular expressions (except for the place) and frame these component regular expressions into one expression. As result, we could recognize datelines in the corpora with high accuracy. Table 3.8 also shows two examples of other special sentence-like segments, which are not datelines, but rather appear at the end of lines. Since these were correctly split off already by the general part of the sentence segmenting script, we handled them specially only later in the alignment refining phase.

Algorithm 1: Truecasing of MTA, MTC

Input: tokenized corpus files: TokFiles; proper name lists: PropLists; list of exceptions: Exceptions
Output: truecased corpus files

```
// Read the proper name lists.
PropNames ← ⋃ PropLists \ Exceptions
// Organize the input files by discourse and version.
Catalogue ← doCatalogue( TokFiles)
foreach Discourse in Catalogue do
    // Extract the special dictionary for this discourse.
    DiscPropNames ← ∅
    foreach Version in Catalogue [ Discourse ] do
        OnceTruecased ← truecase( Version, PropNames)
        PropPhrases ← getPropPhrases( OnceTruecased)
        DiscPropNames ← DiscPropNames ∪ PropPhrases
    DiscPropNames ← DiscPropNames \ Exceptions
    // Truecase using both the general list and the special one.
    foreach Version in Catalogue [ Discourse ] do
        SecondTruecased ← truecase( Version, PropNames ∪ DiscPropNames )
        output SecondTruecased into the right file

function truecase( Text, PropPhrases)
    foreach Line in Text do
        // If the line is a headline,
        if length( Text) ≥ 4  && # of capitalized words > 0.6 · # of words then
            PhraseStarts ← Line // Consider all words for lowercasing.
        else // if the line is not a headline,
            PhraseStarts ← {Word : Word is first or follows specific punctuation
                                   && lowercaseable( Word)}
        foreach Word in PhraseStarts do
            if Word is not in a Phrase in Line, s.t. Phrase ∈ PropPhrases then
                substitute Word in Line by lowercase( Word)
    return Text

function getPropPhrases( Text)
    return set of maximal capitalized phrases in Text

function lowercaseable( Word)
    return Word has exactly one capital letter or digit
           && Word starts in a capital letter
```

(AFP , Beijing , Jan. 21st)
(AFP report of June 27 from San Francisco)
(Durbi)
(Xinhua , Shenzhen , Jan 25 , Li Nanling , Li Xiangzhi report)
(news flash)
(reporter : Zhou Ruipeng) --
(summary of news reports from Hong Kong) -
// Washington Post // : -
Abu Dhabi , April 13 , (Xinhua) -- (Xin Weidong reports)
Abu Dhabi October 6 / Xinhua /
Al Ayn-Emirates 11 - 2 AFP -
Algeria , 17 / 02 (AFP)
April 2
- end of the story -
/ to be continued /

Table 3.8: An example of special segments in MTA and MTC that were treated like separate sentences. The last two examples are found at the end of lines, the other at the start.

3.2.4 Coreference Resolution

We decided to include coreference resolution into our preprocessing pipeline, as it seemed relatively simple and reliable to perform for the short texts of MTA and MTC. With this assumption – that it does not err much – it can be very useful for later stages of alignment (both sentence and word alignment). The reason is that the semantics of the referent is almost opaque in the lexical terms of pronouns – their meaning is expressed by their antecedent, not by the pronoun itself. When we later try to learn which lexical items correspond in meaning, pronouns would present only noise. Therefore, we substitute pronouns, and pronouns only, with their non-pronoun antecedents. Because the antecedents are typically not single words but larger syntactic constructions, we extract the whole construction and put that in place of the pronoun. Note that coreference resolution follows syntactic parsing, so we deal with parse trees when substituting coreferents. At the same time, we want to keep the parse trees consistent with the surface structure, as the parse trees are later used as an additional source of information for word aligning. This somewhat complicates the problem in some cases, when the antecedent does not form one syntactic phrase. In those cases, we effectively substitute the whole parse forest of the antecedent in the place of the pronoun.

Another issue we dealt with was topological ordering of the graph of coreference links – in some cases (28 instances in 21 files – 5 ‰ of all files), expressions were analysed as referring to a phrase that directly includes them.[7] We disregarded such coreferences suggested by Stanford CoreNLP. For a graph without cycles, the topological ordering was then necessary to prevent

[7] For instance, the following sentence contains two examples of the simple circular coreference: *t'ching Kaishiang , who refused to disclose the identity of the two men , said that* he *and* his *colleagues in the hospital have been working for the past few years on technology to rebuild sexual organs .* The pronouns *he* and *his* are resolved as coreferring with the whole phrase *he and his colleagues in the hospital.*

```
0 New York Times -
1 A British government document published yesterday reported that in the days preceding the
        September 11 terrorist attack on the United States * , Osama Bin Laden indicated that
        he was " about to carry out a massive attack against America " .
2 _______ actually says " United Nations " in the original .
```

```
(ROOT
 (S
  (NP
   ...
  (VP
   (VBD reported|VBD|report)
   (SBAR
    (IN that|IN|that)
    (S
     (PP
      ...
     (, ,|,|,)
     (NP
      (NNP Osama|NNP|Osama)
      (NNP Bin|NNP|Bin)
      (NNP Laden|NNP|Laden))
     (VP
      (VBD indicated|VBD|indic)
      (SBAR
       (IN that|IN|that)
       (S
        (NP
         (PRP
          (NP
           (NNP President|NNP|Presid)
           (NNP Bush|NNP|Bush))))
        (VP
         ...
 (. .|.|.))
(S
 (NP
  (NN _______|NN|_______))
 (ADVP
  (RB actually|RB|actual))
 (VP
  (VBZ says|VBZ|say)
  (NP
   (`` ``|``|``)
   (NP
    ...
 (. .|.|.)))
```

Figure 3.3: An example of an analysed sentence just before sentence alignment. Parts of the tree are elided and the last line of the left column is repeated for better readability.

substituting in a phrase that still contains an unresolved pronoun.

Figure 3.3 shows an example sentence (the same one as in Figure 3.1 (B)) just before refining the sentence alignment, which is the next step after coreference resolution. The sentence is stored as a PTB-style[8] parse tree, with the terminal symbols expanded to include POS-tags and stems. You can note in Figure 3.3 that the pronoun *he* (following *Osama Bin Laden indicated that*) got (incorrectly) substituted by the syntactic substructure for *President Bush*. Also note that the first sentence from Figure 3.1 (B) (numbered 0) is missing here – that is due to it matching a pattern for datelines.

[8] PTB stands for Penn Treebank.

3.2.5 Alignment Refining

Most lines in MTA and MTC, barring the previously mentioned special segments, consisted of just one sentence. One line in several conveyed an amount of information for which most the translators used two sentences; and only occasionally were there lines comprising three sentences or more. Hence, we aimed to find a rather simple algorithm for sentence alignment, tailored to this unusually restricted problem. We implemented two sentence alignment algorithms – a length-based one, and one based on the lexical overlap. We describe them in the next two sections, and evaluate them afterwards. For both of them, we assume a monotone alignment of the sentences, and eschew aligning to an empty segment. In accordance with earlier observations from Section 2.2.4, we let PCs (points of correspondence) reside at indices of *sentence boundaries*, rather than indices of sentences themselves.

Length-based Alignment

Following the ideas of Gale and Church (1991), introduced in Section 2.2.3, we implemented a sentence alignment algorithm based on comparison of the sentence length (). Similarly to Gale and Church's approach, we give preference to aligning fragments which have similar lengths. Our program can be parameterized to measure sentence length in characters or tokens, and we evaluate both versions. In both cases, let m denote the length of the first text (one along the horizontal axis in bitext space pictures), and n the length of the other text, for our exposition in this section.

Algorithm 2 shows the pseudocode of the algorithm, and we will explain it in words in this paragraph. Given a bitext, the algorithm considers in each recursive call candidate PCs closest to the main diagonal. It starts with the PC closest to the origin (line 3) and always continues towards the main diagonal (as exemplified by the sequence of numbered PCs in Figure 2.6 on page 16; handled by lines 15–16 in the pseudocode) until it hits the end of the bitext in either dimension. For each candidate PC considered, it evaluates a statistic describing how close it is to the main diagonal. It then selects the candidate PC with the highest statistic (call it *pivot*; line 10 or 18), provided the value is not less than a threshold ("MIN_COSINE_STATS", empirically set to 0.93). After that, it recursively analyses the two bitext subspaces, from the origin to the pivot (line 24), and from pivot to the terminus (line 27), provided they still contain some candidate PCs. The resulting alignment is then composed of all the PCs that were selected for the pivot at some time during the run of the algorithm, plus the implied origin and terminus.

The primary purpose of the statistic for evaluating closeness of candidate PCs to the main diagonal is not the distance from the main diagonal per se, but rather the balance of lengths of aligned fragments. By selecting any PC, we split the bitext into two beads – and the lengths of the fragments aligned within each of the beads are the lengths we need to be as similar as possible. This is the quantity we try to capture with the statistic.

For this reason, we need something less obvious than Euclidean distance (of the PC from the main diagonal). The cosine statistic, applied widely in computational semantics, is a better choice. However, we opt for a measure slightly more sophisticated than that, with regards to the following argument.

This argument is based on Figure 3.4, showing the bitext space normalized to the $[0, 1] \times [0, 1]$ square, with O being the origin, T the terminus, and P_1, P_2, and P_3 different candidate PCs.

Algorithm 2: Length-based sentence aligning

```
Function lenAlign
Input: array of lengths of fragments of text 1: LensX; ditto for text 2: LensY
Output: alignment as a set of PCs
Almnt ← [ Point( 0, 0 ) ]                          // Start with the origin.
Candidates ← [ ]                                   // sequence of candidate PCs
CandX ← 1;   CandY ← 1                             // coordinates of the current PC
NFragsX ← length( LensX);   NFragsY ← length( LensY)
TotLenX ← sum( LensX);   TotLenY ← sum( LensY)
// Compute CosStats for the PCs around the main diagonal.
while CandX < NFragsX && CandY < NFragsY do
    RatioX ← LensX[ CandX ]/TotLenX
    RatioY ← LensY[ CandY ]/TotLenY
    if RatioX = RatioY then                        // perfect candidate
        BCandX ← CandX;   BCandY ← CandY
        break
    CosineStats ← CosStats( RatioX, RatioY )
    if CosineStats > MIN_COSINE_STATS then
        Candidates.append( Candidate( CandX, CandY, CosineStats ) )
    // Find the next candidate: which direction to the main diag?
    if RatioX > RatioY then CandY ← CandY + 1
    else CandX ← CandX + 1
// If any eligible candidates were found,
if BCandX not defined && Candidates not empty then
    // choose the best one (pivot) according to their CosStats.
    BCand ← best of Candidates wrt CosineStats
    BCandX ← BCand.CandX;   BCandY ← BCand.CandY
if BCandX is defined then          // If any good candidates were found,
    if BCandX = 1 || BCandY = 1 then  // If the first bead is atomic,
        Almnt.append( Point( 1, 1 ) )              // Add the pivot PC.
    else // if the first bead is not atomic, recur.
        SubAlmnt ← lenAlign( LensX [ 0:BCandX ], LensY [ 0:BCandY ] )
        Almnt.extend( SubAlmnt)
    // Similarly for the second bead - if not atomic, recur.
    if BCandX < TotLenX − 1 && BCandY < TotLenY − 1 then
        SubAlmnt ← lenAlign( LensX [ BCandX :TotLenX ],
                             LensY [ BCandY :TotLenY ] )
        Almnt.extend( SubAlmnt)
Almnt.append( Point( TotLenX, TotLenY ) ) // Conclude with terminus.
return Almnt
```

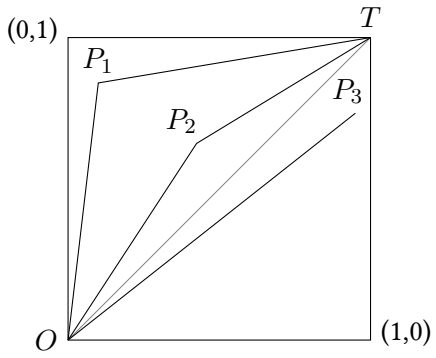

Figure 3.4: Cosine similarity as a measure of closeness to the main diagonal. The point P_1 is far from the main diagonal, and it has the cosine statistic 0.64. The point P_2 is closer to the diagonal, so it has a better cosine statistic – 0.95. The point P_3 is the closest, but induces an unbalanced bead, and hence is dispreferred. Its cosine statistic is 0.83. (All the decimal numbers are rounded.)

If we compare distances of the PCs from the diagonal by the angle POT, P_1 is the furthest, P_2 is closer, and P_3 is the closest. (The same holds true for Euclidean distance.) However, the beads induced by P_3 are unbalanced – although the first one (between O and P_3) is very good, the lengths of the two fragments forming the other bead (between P_3 and T) are very different. Ideally, both the beads should be balanced. For that reason, P_2 should get a better score than P_3. We achieve that in a principled manner by taking also the PTO angle into account, and combining the cosine statistic for both the angles.

The formula for the measure we use is the following:

$$\begin{aligned} \textit{CosStat}(x,y) &= \cos\left((x,y)^T,(1,1)^T\right)\cdot\cos\left((x-1,y-1)^T,(-1,-1)^T\right) \\ &= \frac{x+y}{|(x,y)^T|\cdot|(1,1)^T|}\cdot\frac{(1-x)+(1-y)}{|(1-x,1-y)^T|\cdot|(1,1)^T|} \\ &= \frac{(x+y)\cdot((1-x)+(1-y))}{\sqrt{(x^2+y^2)\cdot((1-x)^2+(1-y)^2)}}\cdot\frac{1}{2}. \end{aligned} \tag{3.1}$$

(Note that we measure the cosine only after normalizing the length of the texts to keep the notion of angle equal for both of them; so the arguments to *CosStat* are the ratios $x = i/m$ and $y = j/n$.)

For the ideal candidate PC, *CosStat* is equal to 1, and the value always lies in the interval $[0, 1]$. The surface of the measure in the bitext space normalized to a unit square is shown in Figure 3.5 using its contours at several levels. It reaches the value of one exactly on the main diagonal. The figure justifies the claim that the statistic indeed captures closeness to the main diagonal as well as balancedness of the induced beads.

Alignment based on Lexical Overlap

The algorithm described in this section was the original algorithm we planned to use for sentence alignment, whereas the length-based algorithm described previously, we developed

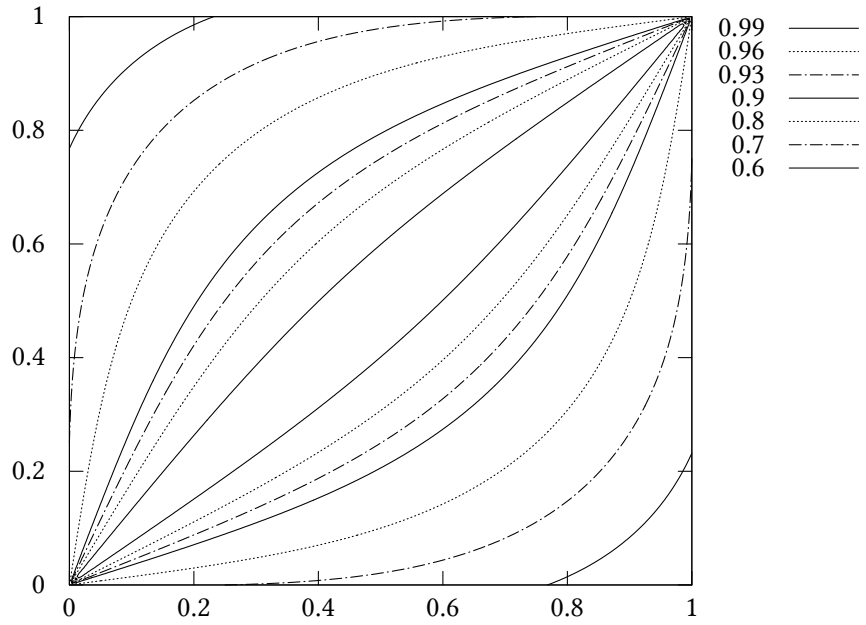

Figure 3.5: Map of the cosine statistic within the normalized bitext space. The figure shows contours of the function surface; the closer the contour is to the main diagonal, the higher the value of the statistic.

as a baseline. We were initially motivated to use this lexical-based algorithm for reasons listed as general advantages of lexical-based approaches in Section 2.2.3. The predominant motivation is that the aligner takes into account words themselves, not only their count, respectively their total length. Whether our expectations were right, is detailed further in Section 30.

We framed our suggested method as a dynamic programming algorithm. Because, as noted earlier, our alignment problem is unusually restricted, we could afford to lift restrictions on the length of beads. This resulted in an algorithm with complexity $\mathcal{O}(k^2 \cdot l^2)$ for each pair of lines, where k and l are the numbers of sentences in the two lines, respectively.[9]

The program iterates over all candidate PCs. For each PC $\langle i, j \rangle$, all previous candidate PCs $\langle i', j' \rangle$ are considered, i.e. all such that $i' < i \ \wedge \ j' < j$. (The inequality is sharp since we do not allow alignment to empty fragments.) For each of them, the *score* for aligning the fragment text_1[$i' : i$] with text_2[$j' : j$] is added to the best score of the PC $\langle i', j' \rangle$ to form a potential best score for the PC $\langle i, j \rangle$. By the notation Text[$a : b$], we understand the sequence of sentences of Text between the end of the a-th and the end of the b-th sentence. The highest of the potential best scores is saved with the PC $\langle i, j \rangle$ as the best score, together with the coordinates $\langle i', j' \rangle$ of the preceding PC that lead to the best score. After computing the best score for the last of the candidate PCs, the terminus, the pointers to the best preceding PCs give us the optimal alignment with respect to the scoring function.

[9]For this complexity estimate, we treat each computation of the Jaccard index for two sets as a constant-time operation.

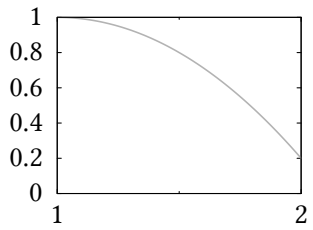

Figure 3.6: Penalty for high length ratio.

The scoring function we used compares sets of stems of the two fragments using the Jaccard index:

$$J(A, B) = \frac{|A \cap B|}{|A \cup B|}. \qquad \text{(Jaccard index)}$$

We represent each fragment by the set of stems for tokens it comprises, disregarding stopwords.

Besides computing the lexical overlap as just described, the scoring function also takes into account the lengths of the two fragments (in tokens), and it does so in three ways:

1. if the length ratio exceeds a threshold (set to 2 after observing the data), the function returns a very small value (e.g. 10^{-20});
2. if both the fragments are short (at most 10 words), they are favoured to be aligned onto each other and the function returns a higher value than the Jaccard index J (namely, $1 - 0.5 \cdot (1 - J)$);
3. in other cases, the length ratio of the two fragments is penalized by multiplying the Jaccard index by a penalty coefficient.

The penalty coefficient is a function of the ratio of fragment lengths. In spite of the usual assumption about penalties, the penalty is the higher, the smaller is the value of the function (plotted in Figure 3.6).

Evaluation

We evaluate the different alignment algorithms over a sample of 500 bitexts from MTA, which were manually aligned. We made the initial observation that our (lexical-based) alignment algorithm correctly resolved 10 out of 10 bitexts of the size 2×2 (i.e. two sentences on each side). We concluded that these alignment problems were, although not trivial (each has two valid solutions), still too easy. Thus we composed the gold standard sample only of bitexts larger than 2×2.

For the annotations, we followed a simple rule: if the same information, however unimportant, is expressed within the sentence $S1$ in the first text, and in the sentence $S2$ in the other text, $S1$ and $S2$ must end up within the same bead. Where the rule is applicable, it decides whether to split sentences between multiple beads, or whether to keep them grouped within a single bead. The first option, splitting the bead, would leave the subsequent processing[10] with smaller bitexts,

[10] In the experiments we have completed, subsequent processing consists solely of word alignment.

2 × 3	LEX			LEN-T			LEN-C		
	gold standard								
resolved as									
	**10**	1	0	**6**	0	0	**6**	0	0
	2	**309**	1	3	**307**	0	3	**307**	0
	5	1	**34**	8	4	**35**	8	4	**35**
complete	59%	99%	97%	35%	99%	100%	35%	99%	100%
beads	59%	99%	97%	35%	99%	100%	35%	99%	100%
TOTAL									
complete	97% (353/363)			96% (348/363)			96% (348/363)		
beads	98% (696/709)			97% (690/709)			97% (690/709)		

Table 3.9: Accuracies of sentence alignment algorithms for bitexts of size 2 × 3.

which would boost its efficiency. However, it would break some correspondences, and those could not be recovered anymore. We decided to follow the rule, because that means preferring (potential) correctness over speed. The rule had an indirect effect on the assessment of the algorithms too.

2 × 4	LEX			LEN-T			LEN-C		
	gold standard								
resolved as									
	23	0	0	**23**	0	0	**23**	0	0
	0	**12**	1	0	**12**	1	0	**12**	1
	0	0	**4**	0	0	**4**	0	0	**4**
complete	100%	100%	80%	100%	100%	80%	100%	100%	80%
beads	100%	100%	80%	100%	100%	80%	100%	100%	80%
TOTAL									
complete	98% (39/40)			98% (39/40)			98% (39/40)		
beads	98% (78/80)			98% (78/80)			98% (78/80)		

3 × 4	LEX				LEN-T				LEN-C			
	gold standard											
resolved as												
	11	0	0	0	**10**	2	0	0	**11**	2	0	0
	0	**3**	0	0	0	**1**	0	0	0	**1**	0	0
	0	0	**2**	0	0	0	**2**	0	0	0	**2**	0
	0	0	0	**2**	1	0	0	**2**	0	0	0	**2**
complete	100%	100%	100%	100%	91%	33%	100%	100%	100%	33%	100%	100%
beads	100%	100%	100%	100%	91%	33%	100%	100%	100%	33%	100%	100%
TOTAL												
complete	100% (18/18)				83% (15/18)				89% (16/18)			
beads	100% (51/51)				86% (44/51)				92% (47/51)			

Table 3.10: Accuracies of sentence alignment algorithms for bitexts of sizes 2×4 and 3×4.

3 × 3	LEX					LEN-T					LEN-C				
	gold standard														
resolved as															
	2	0	0	0	0	**2**	0	0	0	0	**2**	2	0	0	2
	0	**64**	0	0	4	0	**64**	1	1	6	0	**59**	1	1	4
	0	0	**1**	1	0	0	0	**0**	0	0	0	0	**0**	0	0
	0	0	0	**0**	0	0	0	0	**0**	0	0	0	0	**0**	0
	0	0	0	0	**7**	0	0	0	0	**4**	0	0	0	0	**4**
	0	0	0	0	0	0	0	0	0	1	0	3	0	0	1
complete	100%	100%	100%	0%	64%	100%	100%	0%	0%	37%	100%	92%	0%	0%	37%
beads	100%	100%	100%	0%	64%	100%	100%	50%	0%	37%	100%	92%	50%	0%	37%
TOTAL															
complete	94% (74/79)					89% (70/79)					82% (65/79)				
beads	96% (212/221)					93% (205/221)					86% (190/221)				

Table 3.11: Accuracies of sentence alignment algorithms for bitexts of size 3 × 3.

The results of the evaluation are shown in Tables 3.9, 3.10, and 3.11 for different sizes of the bitext. In each table, there are three columns corresponding to the three compared algorithms, and three horizontal parts providing different detail of the evaluation results. The three algorithms are coded LEX (lexical-based), LEN-T (length-based, measuring length in tokens), and LEN-C (length-based, measuring length in characters, excluding spaces).

The first horizontal section presents confusion matrices for the alignment as a classification task. Columns correspond to different gold standard annotations, and rows correspond to the classification results from the algorithm. Each row or column is labeled by a diagram of the alignment where each bead is shaded (similarly to Figure 2.7 on page 18).

The second horizontal section summarizes the data from the confusion matrices. For each gold standard alignment, it shows the accuracy of the algorithm. The row labeled "complete" refers to the accuracy of the alignment as a classification problem, neglecting the inner structure of alignments (i.e. an alignment exactly the same as the gold standard one counts as a hit, whereas any other counts as a fail). The row labeled "beads" refers to the accuracy of the task as finding the beads (the term *accuracy* was introduced in Example 3, page 17; i.e. each correctly recovered bead counts as a hit, and these are compared to the sum of beads of the gold standard alignments). The accuracy of complete alignments captures the overall performance of the algorithms; the accuracy of induced beads is perhaps more important, as it is the beads that are used in subsequent tasks and which need to be correct.

Finally, the third horizontal section of the tables summarizes the accuracy for each algorithm regardless of the gold standard classification. Again, the accuracy is evaluated separately for alignments as a whole, and for individual beads.

As the evaluation showed, all the three algorithms perform very well on these data. For most bitext sizes, their accuracy exceeds 95%, except for the 3×3 bitexts, where it is somewhat lower. Where the algorithms err, is often the bitexts with larger beads. For instance, the second alignment in the 3×4 bitexts (Table 3.10) or the third to fifth alignments in the 3×3 bitexts (Table 3.11) contain such beads. For the alignment of a 3×3 bitext consisting of a single bead (the fourth one in Table 3.11), none of the algorithms got it correct. In all such cases, the lexical-based algorithm handled the problem no worse than the length-based one. This is, the lexical algorithm recognizes correspondence links between words in different sentences, which allows it to group them under the same bead. In contrast, the length-based algorithm does not consider any cross-sentence links, in fact, it considers no correspondence links whatsoever, so it chooses to split beads more often. The tendency of the length-based algorithm to split more demonstrates itself clearly in the 3×3 type of alignments – the bitexts whose gold standard alignment contains a larger bead (the third to the fifth in Table 3.11) are more often resolved as three one-to-one beads () by the length-based algorithms. An example of a few bitexts where the lexical algorithm outperforms the length-based one is provided in Figure 3.7.

To get a better idea whether the problem is too easy for it to be worth implementing any alignment algorithm at all, we additionally evaluated a simple baseline. The *maximum frequency alignment* baseline selects for each bitext size the alignment it had most often in the gold standard data. The accuracy of the baseline, shown in Table 3.12, is 88% in the best case (beads accuracy), and

57% in the worst case (both beads' and complete alignment accuracy). The evaluation gives us confidence that a proper algorithm was indeed necessary.

a) Example sentences:

ahc

1 the volume of commercial exchange between the two countries witnessed a 72 % increase in the first seven months of this year compared to the same period of last year to reach 55 million dollars despite its regression since 1996 .
2 it is hoped that in the next years , the volume will reach the level it was at five years ago .
3 it reached 200 million dollars in 1996 .

ahf

1 trade between the two countries boosted throughout the first seven month of current year to reach 72 % , compared with same period last year .
2 bilateral trade reached the amount of 55 million dollars , and though it had witnessed severe decline since 1996 , it is hoped to reach in the coming years its average five years earlier .
3 in 1996 bilateral trade reached 200 million dollars .

ahk

1 trading relations between the two countries witnessed a rise of 72 % during the first seven months of the current year compared to the same period the year before .
2 they totaled 55 million dollars in spite of the fact that they had been receding since 1996 .
3 it is hoped that during the next few years they would attain their level of 5 years ago when they had reached 200 million dollars in 1996 .

b) Sample alignments:

ahc × ahf

	LEX	LEN-T	LEN-C
gold standard			
	1	0	0
	0	1	1

ahf × ahk

	LEX	LEN-T	LEN-C
gold standard			
	1	0	0
	0	1	1

ahc × ahk

	LEX	LEN-T	LEN-C
gold standard			
	1	**1**	**1**

Figure 3.7: Example of difference in alignment, as produced by the lexical-based, and the length-based algorithms. The example builds on three versions of the same line (line 4 from the files `part1.ah?/artb_522.tok` of MTA), translated by the translators **ahc**, **ahf**, and **ahk**.

	2×3	2×4	3×3	3×4
	17	**23**	2	**11**
	311	12	**64**	3
	35	5	1	2
			1	2
			11	
complete	86% (311/363)	57% (23/40)	81% (64/79)	61% (11/18)
beads	88% (622/709)	57% (46/80)	87% (192/221)	65% (33/51)

Table 3.12: Accuracies of sentence alignment for the maximum frequency alignment baseline.

	LEX	LEN-T	LEN-C	MFA
complete	**96.8**% (484/500)	94.4% (472/500)	93.6% (468/500)	81.8% (409/500)
beads	**97.7**% (1037/1061)	95.9% (1017/1061)	94.7% (1005/1061)	84.2% (893/1061)

Table 3.13: Summary evaluation of sentence alignment algorithms.

Table 3.13 summarizes the accuracy scores for all the three algorithms and the most frequent alignment baseline (MFA). The results confirm our expectation that the lexical-based algorithm would perform the best. However, the difference in performance of LEN-T and LEN-C is not what we expected. We would expect measuring in characters to lead to better alignments than measuring in tokens, according to the results reported in Gale and Church (1991). We hypothesize that we got the opposite result not only because we implemented a different alignment algorithm, but probably more importantly due to the characteristics of our alignment task. Our corpus differs from the Gale and Church's one in two important aspects:

1. The problems are very small, making it possible even for crude approaches to solve them accurately.

2. The language on both sides of the bitexts is the same. It is governed by the same grammar and tends to express same situations by similar syntactic constructions, comprising similar number of words.

The second aspect seems to be so salient in our task that it even overshadows the similarity in number of characters, judging by the evaluation of LEN-T vs. LEN-C. We hypothesize that different translators, authors of the parallel texts, were forced by the grammar to use similar syntactic constructions to express the same meaning, while they had more freedom in lexical choice. That would cause the number of words in corresponding sentences to be more stable than the number of characters they comprise. However, with this size of test data and such small difference in performance, we even cannot rule out that the observed difference between LEN-T and LEN-C be purely random.

3.2.6 Summary

We have described the way we preprocessed the Multiple Translation Arabic and Multiple Translation Chinese corpora. The preprocessing comprised fairly standard tasks. For tokenisation tweaking, truecasing, and sentence tokenisation, we devised custom scripts adapted to the specific nature of these corpora. We further applied Stanford CoreNLP tools to obtain deeper linguistic analyses, especially coreference resolution. After substituting pronominal coreferents with their antecedents, we refined the sentence alignment in the corpus using a lexical-based alignment algorithm. As a result, from original total of 35K lines contained in the two corpora, we obtained 128K pairs of sentences to use for training word alignment or paraphrase extraction.

3.3 Gazetteers And Other Proper Names Lists

This section lists resources we used as lists of proper names (or common names, which is the case of the lists of conjunctions and prepositions.) for truecasing of MTA and MTC (described in Section 3.2.2). We manually edited the resources to eliminate as many nouns homographic with common nouns as possible.

URL	**extracted lists**
http://www.rong-chang.com/namesdict/dictionary.htm	countries, first names, last names, U.S. presidents, U.S. states
http://www.pronouncenames.com/all_names/alpha/A	person names
http://www.muslimnames.info/	Arabic person names
http://en.wikipedia.org/wiki/Title	person titles
http://earth-info.nga.mil/gns/html/namefiles.htm	geographical names
http://geonames.usgs.gov/domestic/download_data.htm	towns in the U.S.
http://geography.about.com/library/weekly/aa030900a.htm	names of nations
http://www.english-grammar-revolution.com/list-of-conjunctions.html	conjunctions
http://en.wikipedia.org/wiki/List_of_English_prepositions	prepositions

Table 3.14: List of proper name lists used for truecasing of MTA and MTC.

Chapter 4

Discovering Relations between Sentence Constituents

The reader will recall that in the present work, we aim to measure relevant properties of human-produced descriptions of video content. This chapter is devoted to the first phase of a two-stage approach to the challenge. In this phase, we hand-annotate a small sample of data from MSR VDC (Microsoft Research Video Description Corpus; see Section 3.1) and investigate into the subsequent automatic processing of the manually extracted features. The goal for the second phase is automating the annotation (feature extraction), and this is the subject for the next chapter.

4.1 Motivational Example

As noted in Section 2.1, we want to be able to provide other project components with information on how different aspects of a video are important for its description. That is, we want to help guide the video description generation process to produce natural descriptions with a balanced content of different sentence constituents. There is certainly no single answer, not even for a concrete clip; there are rather a handful of approaches to distribute content over a sentence. For example, one video clip[1] has following, entirely different descriptions in MSR VDC:

Example 5.

1. A man is watching two bear cubs digging.
2. The baby bears dug in the dirt for insects.
3. Bears.

The above sample from the corpus, although tiny, embraces several linguistic phenomena that differ from description to description. The probably most striking one is the amount of detail provided in each description. The first sentence provides a high-level overview of the scene participants and actions, the second one expands on where and why the bears dug, whereas the last one barely mentions the main protagonists of the scene.

[1] with ID `uB9zRlV47qA` in the corpus

Another phenomenon evident in the above example is the shift of focus. While the first sentence is saying what the man is doing (he is watching bears), the second sentence ignores him and focuses directly on the bears. So does the third description, obviously. It is very typical for MSR VDC descriptions that their main actor is a person, as in the first sentence, often described using words like *a man*, *the lady* or *someone*. In this particular footage, the man is rather marginal. Thus we can attribute his getting into the focus to some prior expectation of a human in the clip.

The last outstanding property that we will mention is the verb form. While the first sentence describes the clip as if it is happening currently, using the present continuous tense, the second one described it only statically with the simple past tense. And the last example? No need for a verb whatever. – The varying verb form has a large impact on how the whole sentence sounds, and interacts with other properties of the sentence.

These three properties: the level of detail, the focus, and the verb form; will guide us in our search for a way to characterize the pattern of a sentence.

4.2 Feature Extraction

Since our goal is learning the relations between elements in a sentence by computer, we need to project each sentence into an exactly defined space of features to be able to handle it. This section discusses what features we used for sentence representation. The feature choice is critical for further analyses of the corpus, since any patterns we might find will be expressed solely in terms of these features. Unless otherwise noted, we will use dependency syntax in this section.

4.2.1 Anatomy of Feature Vectors

As verbs are central to a sentence, they are also central to our feature vectors. We partition each sentence into parts, each corresponding to a syntactic subtree rooted in a full verb node. In the experiment described in this chapter, we constructed feature vectors by simply concatenating component feature vectors for individual verbs of the sentence. Thanks to the sample of data for the experiment being fixed, the entire feature vectors could have a fixed length too – which is a prerequisite for statistical machine learning.

Let us now describe how the entire feature vector was constructed from the individual *verb feature vectors*. (Verb feature vectors will be described later.) We first need to introduce a notation for distinguishing verbs in a sentence. Firstly, we will denote main verbs in the sentence simply by their order in which they come. Thus the first main verb will be called "verb 1", the second main verb "verb 2", and so on. Secondly, subordinated verbs will have their name prefixed with the name of their nearest ancestor verb. The second part of their name will be their order within verbs with the same prefix, again according to the linear order of the sentence. The prefix and suffix will be separated by a period. So, for example, "verb 2.1" is a verb which is in the syntactic subtree of the "verb 2", within which it is first in linear order; "verb 2.1.3" would be the third verb below "verb 2.1". We will not represent the intervening path in the syntactic tree between a verb and its ancestor verb. It can be one edge, such as in constructions of control, or it can be a multiply nested structure with the subordinate verb expressing merely a property/activity of some unimportant subordinated noun.

The entire feature vector is constructed by concatenating the verb feature vectors in

order	type	description
1	real	Information content of the *verb*.
2	boolean	Is the verb in the *past* tense?
3	boolean	Is the verb in the *present* tense?
4	boolean	Is the verb in the *passive* mood?
5	boolean	Is the verb in the *progressive* aspect?
6	boolean	Is the verb in the *perfective* aspect?
7	boolean	Does the verb denote a *process*?
8	boolean	Does the verb denote a *culminated process*?
9	boolean	Does the verb denote a *point* event?
10	boolean	Is the verb *iterative*?
11	real	Information content of the *actor* of the verb.
12	real	Information content of the *patient* of the verb.
13	real	Information content of the *third actant* of the verb.
14	real	Information content of all *manner modifications* of the verb.
15	real	Information content of all *temporal modifications* of the verb.
16	real	Information content of all *local modifications* of the verb.
17	real	Information content of all *other adverbial modifications* of the verb.

Table 4.1: Anatomy of a verb feature vector.

preorder sequence: 1, 1.1, 1.1.1, 2, 2.1, 3. If a particular verb is not present in the sentence (for example, in a sentence with a single verb, only verb 1 is present), its features get the value of 0. Similarly for other constituents – if not present, their respective feature is set to 0. Each verb feature vector comprises 17 features, summarized in Table 4.1. In that table, when referring to a verb, it is always the respective verb for the verb feature vector. The table also uses a term we have not introduced so far, namely *information content*. It is described in detail in turn in Section 4.2.2. For now, it suffices to say that it is low for common words with general meaning, and high for very specific words.

In our experiments, we extracted the features for the corpus sample manually. We already designed the feature set bearing in mind that they would be extracted by a human, at least in the first phase of experiments. These are reasons why the features are rather abstract and presumably hard to extract by machine. If we find that we get reasonable results using manually extracted features, then, and only then, it makes sense to proceed to automating the feature extraction, probably introducing some approximation.

4.2.2 Information Content

From the three properties (the level of detail, the focus, and the verb form), only for the verb form is it evident how it is reflected by our features, concretely, in features 2–10 in Table 4.1. We capture the other two properties, at least partially, using *information content*.

The application of information content (hereafter, IC) for NLP was introduced by Resnik

(1995). It is defined with respect to an IS-A taxonomy of concepts as:

$$\mathrm{IC}(c) = -\log_2 p(c), \tag{IC}$$

where c is a concept from the hierarchy and p the probability of it occurring in a text (its normalized frequency). Because each concept is also an instance of each of its subsuming concepts, when counting the frequencies to estimate p, each (disambiguated) word counts towards multiple concepts. That ensures that more specific concepts get higher IC.

We based our measurement of IC on the WordNet InfoContent 3.0 dataset[2] (Pedersen, Patwardhan & Michelizzi, 2004), the concept hierarchy of WordNet 3.0 (Fellbaum, 1998) with precomputed values of IC. We used the version with IC based on the British National Corpus.

Assume we have a sentence and want to measure the level of detail it provides. It is reasonable to interpret "level of detail" as the IC of the sentence. One could argue that a sentence can have large IC even if it does not provide much detail, provided it (briefly) mentions many different things. That is a valid argument; but for our targeted application of describing a clip, the total amount of available information is fixed. Therefore, constructing a sentence with a higher IC means specifying more details, rather than including more unspecific information.

Assume we have a video clip description and we want to determine how much it details different aspects of the clip. It is natural to distinguish aspects of the clip by separating parts of the sentence. For example, we can expect the subject to describe the main character of the clip. By following this line of reasoning, we arrived at the features numbered 1 and 11–17 in Table 4.1. We claim that they correspond to the level of detail that the sentence (or, the verb phrase) provides for different aspects of the video.

We believe that shift of the focus partially translates into the IC of different sentence constituents, too – focused constituents should tend to have some minimal IC. However, the IC does not tell the whole story regarding the focus. As seen in Example 5, the sentence focus is a function of multiple factors, including the order of main verbs and the order of their arguments.

Example 5 (repeated).

1. A man is watching two bear cubs digging.
2. The baby bears dug in the dirt for insects.

It could also be indicated by specific constructions, such as the cleft construction; for instance:

Example 6.

4. It is under the roots of a big tree that the two baby bears are digging.

To properly account for the phenomenon of focus shift, descriptions of the same clip need to be word-aligned. Only then can we discover the difference in the focus for the descriptions 1 and 2. We need to know that the words *two bear cubs digging* correspond to *the bears dug* in order to be able to say that the focus on *man (...) watching* is different from the focus on *baby bears [having] dug*. Moreover, if we want to make conclusions about the focus based on any single sentence, we should have the sentence compared to a *standard sentence template* for the particular

[2] Available from http://www.d.umn.edu/∼tpederse/Data.

clip. For instance, in the above example, we would designate digging as the main event, leading to a standard sentence template like *bears dig*. In the experiment described in this chapter, we did not include such features. However, we planned to analyse shift of the focus later in the project. How this could be done, is discussed later in Section 4.6.

We will now describe how the IC values were obtained for each constituent. Since there is no clear way of extending the definition of IC to syntactic structure, we abstracted from the structure of the constituent and treated it as a bag of words. Furthermore, due to limitations of the WordNet InfoContent dataset, we had to discard all words except for nouns and verbs. We consider this a reasonable approximation of the ideal quantification of the IC borne by a phrase, since the meaning of a phrase is usually easy to guess just from its nouns and verbs. To be precise, the IC of a phrase is computed as the sum of ICs of the verbs and nouns constituting the phrase. The words are treated as tokens, i.e. the IC of each word is counted in as many times as the word appears.

For each word, either a noun or a verb, we collect the IC of all its WordNet senses (call the set of senses, or synsets, of a word w, $S(w)$). We did not attempt word sense disambiguation (WSD), although that would be the clean way to estimate the IC of a word. We preferred not to linger with this task, partly because it is not crucial for completion of the experiment, and partly because we did not have high expectations regarding the performance of WSD with the limited context provided by the short sentences from MSR VDC. Instead, we obtained the IC estimates by combining the IC values for all possible senses s of the word w, weighted by the estimated probability that w occurs in the sense s:[3]

$$\mathrm{IC}_{\mathrm{used}}(w) = -\sum_{s \in S(w)} \frac{p(s)}{\sum_{s \in S(w)} p(s)} \cdot \log_2 p(s). \tag{4.1}$$

4.3 Classification of Relations

In this section, we will discuss ways of discovering relations between features. We will start by introducing the hand-annotated sample of MSR VDC which provided the input data for the experiment.

4.3.1 The Sample of MSR VDC

We selected and manually annotated 121 sentences from MSR VDC, describing 8 different clips. Some of the sentences were selected so that they contain interesting variation in description of the same clip, while other were chosen to be representative of the corpus. We introduce some interesting statistics of the sample here, and provide the whole collection of 121 sentences in Appendix A.

The statistics of the sample are shown in Table 4.2. We computed all the statistics after lemmatisation and stemming of the text.

[3] The estimates of $p(s)$ are based on the SemCor corpus.

Part I illustrates variability in the verb and noun types present in the 121-sentence sample. Seeing only 46 different verb types indicates that the descriptions mostly use common phrases, rather than being more inventive.

Lexical variability is investigated in more detail in Part II. The statistics are shown for each clip separately to illustrate that some clips tend to be described very uniformly, whereas others receive quite different descriptions. The clips are listed in the same order in which they come in Appendix A. The columns of the table show the average length of clip descriptions, and lexical entropy[4] in several positions in the sentence (the first matrix verb, all verbs, all nouns, and all verbs and nouns, respectively).

Note that there is high variability in the sentence length for all the clips, most notably for the last one. Remember though that this is influenced by generally short length of the descriptions, as well as small size of the samples for each clip.

Lexical entropy discriminates between the different clips much more clearly. Most notable is the low entropy for verbs in the "guitar" clip. In general, there seem to be many clips on the topic of performing with a musical instrument, and many other clips on the topic of cooking, in MSR VDC. Most of them have quite simple descriptions, uniform across different authors. From the entropy statistics in Table 4.2, we can see that it is the verbs that is so uniform, whereas the entropy of nouns is average or even higher.

In Part III, we look at differences on the syntactic level. These can be very influential for our results, especially because we represent sentences using the rigid feature vectors. For example, Table III.(a) shows that there are only 2 sentences having verb 1.1.1, and only one sentence having verb 2.1 and verb 3 in our annotated sample. That means that for features of those verbs, there will always be a single observed value only (respectively two values for verb 1.1.1).

Table III.(b) uses the term *key verb*. We take this to be the verb denoting the most salient event in the clip. We can see that the key verb comes first in the sentence in a large majority of cases. However, there are a few sentences where this is not the case. In fact, as shown in Table III.(c), there are similarly few sentences that have more than one verb at all. If we compare the 8 key verbs in another than the first position from (b) to the 16 descriptions with more than one clause, as shown in (c), this is a whole half of the cases where a different than the first verb can be the key verb.

Finally, Table III.(d) summarizes the frequencies of different instantiated valency frames of the verbs. We ignore all but the first three actants here, just to get a rough feeling for the character of the data. ACT stands for *actor*, PAT for *patient*, and EFF for *effect*. The table shows that most verbs (tokens) are transitive (corresponding to the ACT+PAT, and ACT+EFF frames), next are intransitive (single ACT), and there are just a few verbs with the actor implicit (single PAT or EFF) or ditransitive (ACT+PAT+EFF).

4.3.2 General Considerations

Our goal is to understand and characterize relations among the different features by way of finding patterns in the descriptions, or a small set of such patterns. There are in general two ways to tackle the problem. First, we could learn what relations tend to govern particular pairs

[4]Lexical entropy is the entropy in the random variable that generates the particular lexical unit whenever a word of the given class occurs in the text. For example, the random variable for nouns determines, whenever a noun occurs in a text, what particular noun it is.

I. General properties of the sample.

No. of types	
verbs	46
nouns	88

II. Length in words and lexical entropy, grouped by the clip.

clip name	**avg length ±sd [words]**	**lexical entropy [b]**			
		verb 1	any verb	any noun	any verb or noun
bird	9.87 ± 8.30	2.92	3.27	2.73	3.69
zebras	6.88 ± 6.99	2.09	2.02	1.65	2.75
bears	8.06 ± 10.70	2.02	2.14	3.22	3.57
soccer	10.56 ± 10.68	2.18	2.37	3.25	3.82
cucumber	10.85 ± 12.36	1.91	2.44	3.22	3.83
guitar	8.19 ± 6.50	**0.34**	**0.34**	2.72	3.08
mixing	8.87 ± 8.71	1.37	1.37	3.42	3.68
shooting	**10.69 ± 18.85**	1.85	2.32	2.33	3.24

III. Structural linguistic properties:

verb occurrences	
# occurrences	position
120	1
8	1.1
2	1.1.1
10	2
1	2.1
1	3

(a)

position of the key verb	
# of sentences	position
112	1
3	1.1
4	2
1	3

(b)

# of clauses in sentence	
# of sentences	# of clauses
0	1
104	1
12	2
3	3
5	1

(c)

frequency of verb frames	
frequency	frame
49	ACT
2	PAT
1	EFF
76	ACT, PAT
13	ACT, EFF
2	ACT, PAT, EFF

(d)

Table 4.2: Descriptive statistics of the sample of 121 sentences from MSR VDC.

of features, and then find larger patterns in terms of these basic relations. Alternatively, we could cluster all training observations by their full feature vectors. We would then proceed by analysing clusters within clusters, now considering perhaps only a part of the feature vector. The two ways are completely different in regard to how the problem is viewed, as well as in the method they require for implementation. They would probably also deliver different results – starting from basic pairwise relations will be too localised at first, with more complete patterns for sentences arising only in later stages; whereas the method of gradual refining clusters will yield complete sentence patterns after the first iteration, and successively learn more general pairwise relations.

We have chosen to implement the former approach, analysing pairwise relations first. Within the restricted time we could spend on this experiment, we only managed to perform the first step – extracting the pairwise relations. Apart from the extra time it would take to extract more complex relations, we would also need much more data than the manually annotated sample. For this, the features would need to be extracted automatically, which we worked towards, and elaborate on in the next chapter.

The problem we tackled and describe in this section is thus classifying pairwise relations of features for interdependence. We aim for soft classification with confidence scores rather than hard resolution of each case. For this, we need to be able to derive the probability distributions of feature values being independent, or dependent. Once we estimate these two distributions, we will be able to predict for any pair of feature values how likely it is to come from either of the distributions. That translates into being able to tell how likely it is for the two features to be dependent.

4.3.3 Statistical Dependencies

Let us first discuss how statistical dependencies can manifest themselves in feature values. Assume that, for example, whenever the first main verb is progressive, it denotes a process. In terms of our feature vector, this translates to the following assertion: whenever feature no. 5 is true, feature no. 7 is also true. Let us call the features A and B, respectively. Now, there can be many observations where the verb in question is not progressive or is absent from the sentence. Then the value for A is false; if the verb is present in the sentence, the value for B is governed by its other dependencies and otherwise random; if the whole verb is absent from the sentence, B must be false too. However, in observations where A is true, B *must* be true as well. Thus such a dependency will lead to a table of feature co-occurrences like Table 4.3.

	$\neg B$	B
$\neg A$	$\mathrm{P}(\neg V) + \mathrm{P}(\neg B \mid V)$	$\mathrm{P}(\neg A, B \mid V)$
A	0	$\mathrm{P}(A \mid V)$

Table 4.3: Example table of feature co-occurrence probabilities for features A and B governed by the implication $A \rightarrow B$. V is a random variable which is true iff the relevant verb is present in the sentence.

When we later classify pairs of features for interdependence, we do so based solely on values of the two features involved. Classifying for interdependence with three or more features would be the object of subsequent iterations. Yet there are three features in Table 4.3: A, B, and

the random variable V.[5] Hence, we shall approximate the table by abstracting from V, which in turn affects the estimates of $\mathrm{P}(A)$ and $\mathrm{P}(B)$. The estimate of unconditioned $\mathrm{P}(A)$, for example, will cover both $\mathrm{P}(A\,|V)$ and $\mathrm{P}(A\,|\neg V)$. Considering that $\mathrm{P}(A\,|\neg V) = 0$ (and the same for B), the estimated $\mathrm{P}(A)$ should have some positive probability mass in the point zero. The simplified co-occurrence table will then look like Table 4.4.

	$\neg B$	B
$\neg A$	$\mathrm{P}(\neg B)$	$\mathrm{P}(\neg A, B)$
A	0	$\mathrm{P}(A)$

Table 4.4: Example of a simplified table of feature co-occurrence probabilities for features A and B governed by the implication $A \rightarrow B$.

Generalizing from the above example, features can also be governed by other implications. Besides $A \rightarrow B$, we analogically obtain expected co-occurrence probabilities tables for $B \rightarrow A$, $\neg A \rightarrow B$, and $B \rightarrow \neg A$. If we recognize pairwise dependencies by looking at the co-occurrence table, the only other possible dependencies we can discover are equivalences: $A \leftrightarrow B$ and $A \leftrightarrow \neg B$. Those are recognized by having zeros in the opposite corners of the co-occurrence table, as in Table 4.5.

	$\neg B$	B
$\neg A$	$\mathrm{P}(\neg A)$	0
A	0	$\mathrm{P}(A)$

Table 4.5: Example table of feature co-occurrence probabilities for features A, B governed by the equivalence $A \leftrightarrow B$.

4.3.4 Handling Real-valued Features

Because of what we have identified as the possible dependencies between features, namely the relations of implication and equivalence, we need to fit all our features into the boolean-value paradigm. However, we use non-boolean features to express the IC of each constituent, in the terms of non-negative real numbers. We map them onto boolean values using a two-stage mapping. We first "squash" the domain from $[0, \infty)$ to $[0, 1]$ using a *squashing function* (see Figure 4.3), and then *discretise* the real number into the value range for the feature being "true", i.e. having the value 1, and the value range for the feature being "false". These discretised parts (call them *responsibilities*), one for true, one for false, are eventually regarded as fractions of an observation. One fraction represents the observation of the feature being clearly true, the other for it being clearly false. We will now look closer at the choice of the functions used for squashing and discretising.

We should start with the discretisation function, as it will help determine the right parameters for the squashing function. After some searching for a function with a smooth gradient

[5] Note that V is in fact captured by one of our extracted features, namely the IC of the verb. If the verb is present in the sentence, its IC is positive, else it is zero.

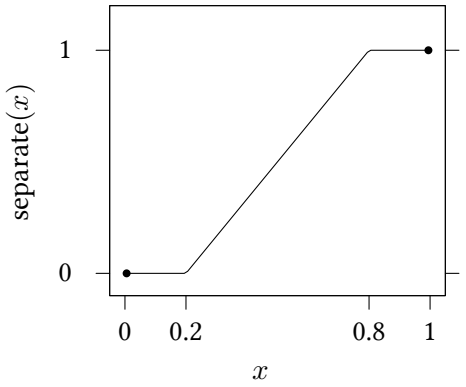

Figure 4.1: The discretisation function.

between $(0, 0)$ and $(1, 1)$, we concluded it would not be the optimal choice. We have little requirements on the discretisation function, so it should be as simple as possible (following Occam's razor). One condition we want the discretisation function to satisfy is for values close to zero to be considered a full responsibility of zero (i.e., false); the second condition is that it should be symmetric with respect to the qualities true vs. false. The simplest function satisfying these conditions is composed of three linear segments, a constant zero at the beginning, a constant one at the end, and a linear segment joining them. We set the constant parts to be 0.2 long. The resulting discretisation function is plotted in Figure 4.1. To put it explicitly, the function assigns to a real number from $[0, 1]$ the responsibility of an observed value being true. () The responsibility for false is one minus this value.

We will now be looking for an appropriate squashing function, knowing that whenever it maps an IC value onto a number 0.2 or less, it will equate to the IC of 0. Similarly, any IC mapped onto 0.8 or more will be interpreted as though the constituent is as explicit (or detailed) as possible. We obviously wanted these equivalences to be well aligned with our data. Having the 121 descriptions annotated, we created a histogram of IC for the actors of the first main verb[6] (see Figure 4.2).

We also looked at what kind of phrases have what value of IC to fit the squashing function to the data not only quantitatively, but also qualitatively. A few noun phrases from different parts of the range of IC, together with their estimated IC value, are shown in Table 4.6. The phrases *a person* and *someone* are the phrases with the minimal IC. Looking at the IC of *a man*, we note that it is surprisingly high. This is caused by a few uncommon senses of the word that have a very high IC according to WordNet InfoContent.[7] That illustrates why it would be helpful to do WSD, or at least know sense priors for our domain (whereas the priors we use are computed from SemCor).

[6]There were two obvious choices: the main verb itself, and its actor. These two are present in most descriptions, so we can have enough data for the histogram. As opposed to the actor, the IC for the verb is always computed from the IC of the single word – the verb. On the other hand, the IC of the actor is sometimes based on more words if the actor has any dependents, as in *a man wearing a black cape.* So we chose to use actors, as we can judge better when this phrase is as detailed as possible, whereas it would not be as clear for single verbs.

[7]The senses of *man* in WordNet 3.0 are: "a human being", "the mankind", "an adult male", "a husband", a "manly male person", "a human being (in a specific generic use)", "a male subordinate", "military personnel", "a valet", "a game piece", "Isle of Man".

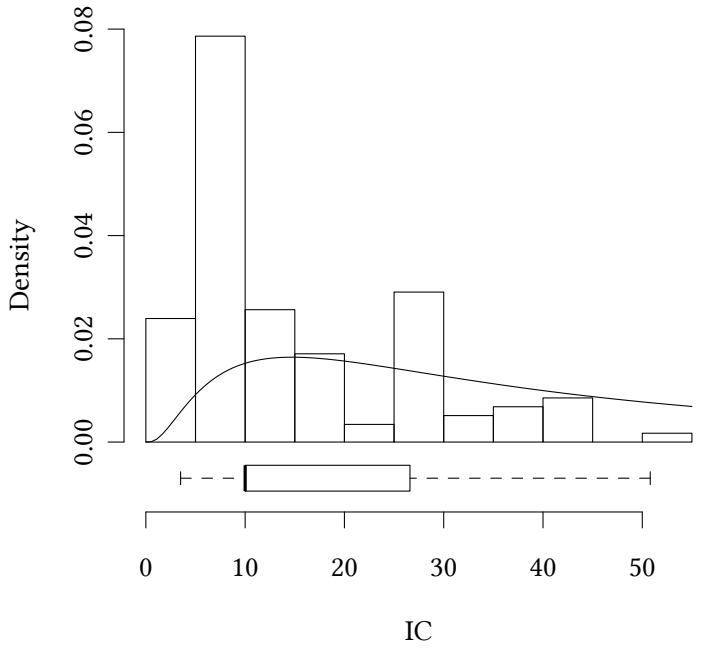

Figure 4.2: Histogram of IC values for the actors of the first main verb. The overlaid function is an approximation of the distribution, described below in Section 4.3.5.

noun phrase	**IC**	**squash**(**IC**)	**discr**(**squash**(**IC**))
a person	3.5	0.13	0.00
someone	3.5	0.13	0.00
an artist	8.5	0.29	0.15
a man	10.0	0.33	0.22
a lady	13.8	0.42	0.37
Jeff Beck	20.0	0.55	0.58
a man in a black cape	38.7	0.79	0.98
man wearing a black cape	50.8	0.87	1.00

Table 4.6: Selected noun phrases (or phrases with a record in the WordNet InfoContent for nouns), their estimated IC values, the corresponding squashed value, and the corresponding responsibility for "true". Decimal numbers are rounded.

Lower in the table is *Jeff Beck* with IC 20.0. This is the IC which we decided to assign to any proper names for people. Although it is a rather arbitrary value, this IC is later interpreted as providing slightly more than half of the maximal detail, which seems reasonable. It can also be interpreted such that about $2^{20} \approx 1,000,000$ entities[8] could be mentioned with the same likelihood as the proper name. The last two rows of the table contain noun phrases with the highest IC within our sample. Note that *wearing* contributes to the IC of the actor of verb 1, even if its IC is counted also towards the IC of verb 1.1 (the word *wearing* itself). In fact, the IC of all the words *wearing*, *black*,[9] and *cape* contribute both to one of the verb 1.1 IC features, as well as to the IC of the actor.

The responsibilities derived from our final squashing function resulted are shown in Table 4.6 together with the phrases and their ICs. These are the kind of values we have been looking for – the phrases *a person* and *someone* get interpreted as entirely missing information. This is desirable, since they appear in the sentence mostly because it needs a subject, rather than to convey any information. On the other end of the spectrum, only a few of the noun phrases in our sample are considered to be maximally detailed. This is also intended, since most of the noun phrases are indeed quite terse.

Before we found the squashing function, we had several criteria it should satisfy. Firstly, as we discussed in the last paragraphs, we wanted it to be aligned to our sample, both quantitatively and qualitatively. The points 0.2 and 0.8 were fixed, as were the IC values of the observed phrases. Hence we aimed to map the ICs of *a person* and *someone* onto less than 0.2, and have the inverse of 0.8 somewhere around 40. Secondly, it must map from $[0, \infty)$, have the range $[0, 1]$, and it should be a non-decreasing function onto its range (or at least onto $[0.2, 0.8]$). () Put simply, higher IC should be interpreted as greater evidence that the constituent (or the corresponding aspect of the video) is described in good detail. Furthermore, it should be possible to have a constituent providing virtually no detail, and conversely, a constituent providing maximum detail.

Guided again by Occam's razor, we sought a smooth concave function. The one we settled on is defined as:

$$\text{squash}(x) = 1 - e^{-x/25}. \tag{4.2}$$

This function, plotted in Figure 4.3, satisfies all the above criteria perfectly. We found it by manually tweaking parameters of the Weibull distribution CDF (cumulative distribution function) to fit our criteria. The Weibull distribution CDF has the following generic definition:

$$F_{\text{Wei}}(x; k, \lambda) = \begin{cases} 1 - e^{-(x/\lambda)^k} & x \geq 0 \\ 0 & x < 0. \end{cases} \tag{Weibull CDF}$$

4.3.5 Generating Prototypes

As outlined above in Section 4.3.2, in order to be able to classify observed pairs of feature values, we need to estimate the distributions that govern

a) features with some kind of dependency between them

[8] Or, for example, $2^{20-8.5} \approx 3000$ different artists, since according to Table 4.6, the IC of *artist* is 8.5.

[9] Wherever an adjective had the same form as a noun which it is derived from and which was present in WordNet InfoContent, we counted its IC too.

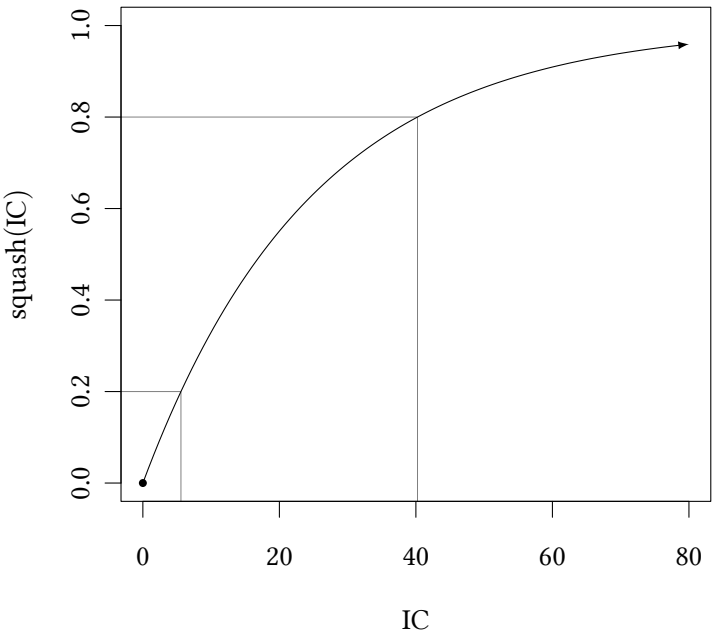

Figure 4.3: The squashing function. Its inverse for 0.2 is 5.58, and the inverse for 0.8 is 40.24 (both rounded).

b) independent features.

In our experiment, we only approximated these two distributions by generating a sample. The sample itself is drawn from an estimated pair of priors for the two features. The sample then directly serves to approximate the latter distribution (b). A sample of the former distribution (a) is obtained from this sample by discarding observations that counter the simulated dependency. Let us now describe how samples of the two distributions are obtained in more detail.

Modeling The Distributions

Assume we are trying to determine whether there seems to be any statistical dependency between given two features A and B. Assume further that these features were originally non-negative real-valued, but they have already been run through the squashing function, so that their domain is $[0, 1]$.

We proceed by approximating the distributions of A and B. We have little information about the nature of the true distribution (consider we need to estimate a distribution of an arbitrary feature from all the defined ones), but expect it will peak at a rather low value and gradually flatten out in a long tail. Therefore, we take as the basic component of the estimated distribution model the log-normal distribution. The log-normal distribution is meant to be on par with observed values of IC, hence it needs to be "squashed" to the right domain. Besides the log-normal component, we account for the common case of zero feature values by placing some probability mass in the point zero.

Thus the class of distributions among which we look for the best estimate of the true distributions have the following PDF (probability density function) formulation:

$$f_{\text{feat}}(x;\, p_0) = \begin{cases} 0 & x \leq p_0 \\ \text{squash}(f_{\text{lnorm}}(x;\, \mu_{\text{lnorm}}, \sigma_{\text{lnorm}})) & x > p_0, \end{cases} \tag{4.3}$$

where p_0 is the estimated probability mass in point 0. The parameters μ_{lnorm} and σ_{lnorm} were set to the fixed values 40 and 1, respectively, which seem to yield a PDF roughly corresponding to observed distribution of IC values. Figure 4.2 shows the log-normal component of the resulting PDF before squashing (which corresponds to f_{feat} for $p_0 = 0$). Even though it does not fit the data particularly well,[10] it is particularly simple. Furthermore, after squashing and sampling with noise, it works well for generating distinct prototypes with or without a dependency between features.

Having decided on the model for feature distributions, and having observations for the two features, we estimate the feature distributions by MLE. With μ_{lnorm} and σ_{lnorm} fixed, that amounts to setting p_0 to the ratio of number of observations of the feature with the value false to the total number of its observations.[11] This gives us formulas for f_A and f_B for the two features, A and B.

[10] A substantial part of the bad fit is caused by raggedness of the observed values. They peak first at the area of the IC of most English content words, then drop before peaking again at twice that value, corresponding to an NP (noun phrase) consisting of a pair of nouns, and so on. This will be partly remedied by factorizing the matrix of observations, described later in Section 4.3.6. Note however that the box plot below the histogram indicates that a distribution similar to our log-normal one should fit best. Possibly, the μ and σ parameters should not be fixed in future experiments.

[11] In the implementation, we quantize the estimates of p_0 to multiples of 0.1 for the sake of efficiency.

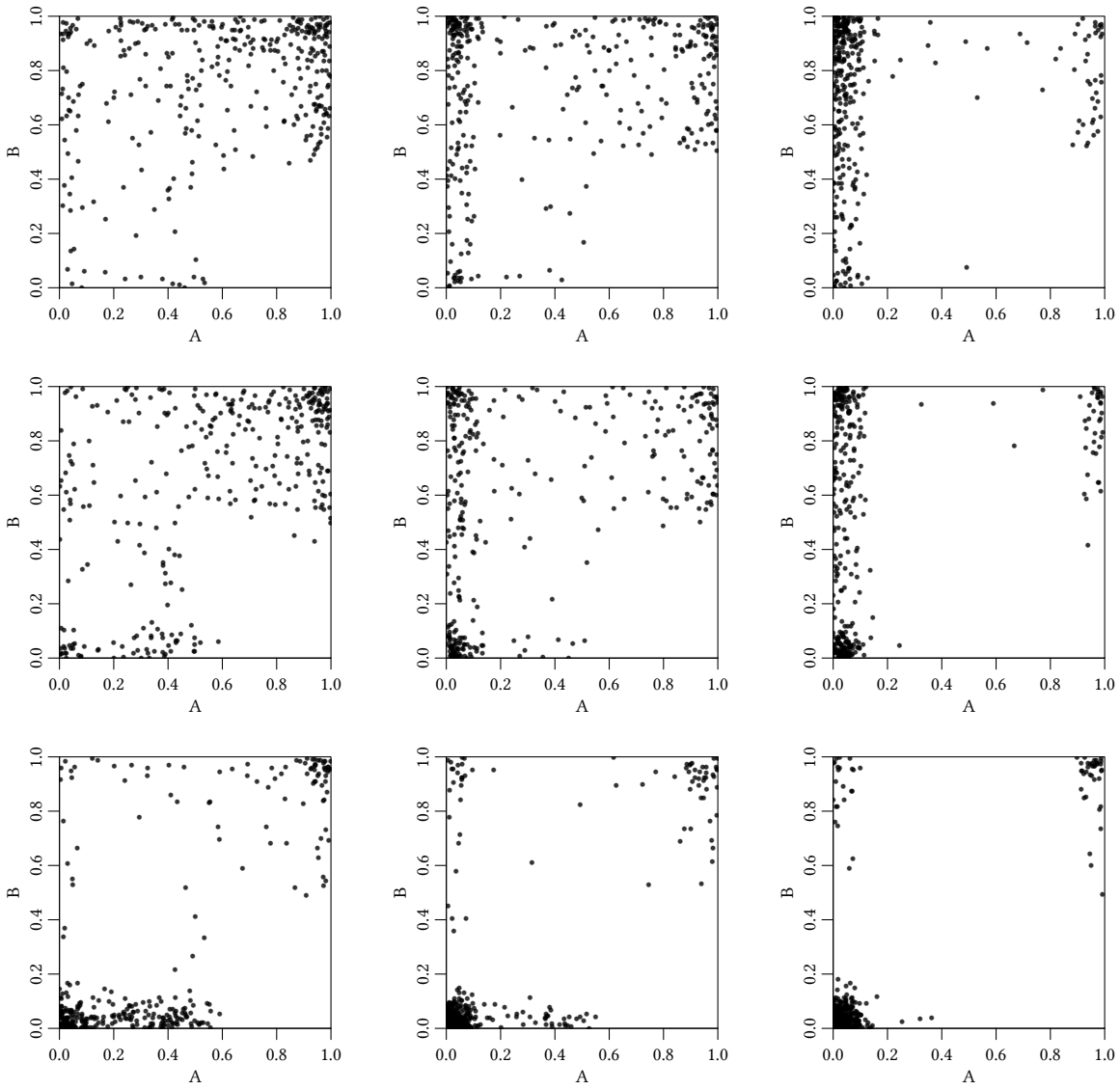

Figure 4.4: Random samples of 333 coordinates with differing P($A = 0$) (0.1, 0.4, and 0.9 for the three columns, respectively) and P($B = 0$) (0.1, 0.4, and 0.9 for the three rows, respectively). The amount of noise is low, and the features A and B are assumed to be governed by the implication $A \rightarrow B$.

Figure 4.4 shows the impact of different p_0 on the generated sample. You can note that as p_0 grows (from left to right, and from top to bottom), an always larger fraction of the sampled points has a zero value in either dimension (i.e., for one or both of the features). This corresponds to features being increasingly sparse. With very sparse features, the regions close to 1 become sparsely populated with sampled points, and thus much harder to distinguish from the regions that are actually blocked by the assumed dependency.

Sampling

We now sample the *joint distribution* of A and B, taken to be:

$$f_{A,B}(x;\, p_{0,A},\, p_{0,B}) = f_A(x;\, p_{0,A}) \cdot f_B(x;\, p_{0,B}) \tag{4.4}$$

assuming independence of the two features. To start simple, we can now say that we sampled 99 points for each pair of features. More precisely, we grouped pairs of features by their estimates of p_0, and for all pairs of features with the same $\langle p_{0,A}, p_{0,B}\rangle$, we sampled the joint distribution just once.

Pseudocode for sampling n points is given in Algorithm 3. It is slightly more complex than we have indicated so far, because we require each sample to contain a point with the maximal value (i.e., 1.0) for each of the two squashed features. This is to make different features comparable within the same $[0, 1] \times [0, 1]$ space. If, for example, only one sentence contains the verb 2.2,[12] we do not care anymore how much IC it has; it is notable by just being there. Thus we normalize both the testing observations (the annotated sample of the corpus), as well as the training observations (the prototypes sampled using Algorithm 3) to the maximum of 1.

If we did not enforce the above condition, the algorithm would shrink to random sampling points and discarding those points which are non-compliant with the given dependencies. However, when the points are normalized, some may slip to the region prohibited by the dependency. Consider the following example.

Example 7. Let us assume we initially sampled the following points: $\{(0, 0.4), (0.4, 0), (0.3, 0.3)\}$. Judging by comparing them to 0.5, all are instances of the $(0, 0)$ combination of boolean values. However, after normalization, they become $\{(0, 1), (1, 0), (0.75, 0.75)\}$. That suddenly classifies them into all other boolean combinations but the original $(0, 0)$. If the dependency we must enforce is the equivalence $A \leftrightarrow B$, we have to discard the first two points since they break the equivalence. That leaves us with a single point, $(0.75, 0.75)$. Here, the story starts anew, since our sample no longer has the values of 1.0 in both dimensions. Moreover, a repeated normalization could cause the other points, permissible so far, to slip again to the prohibited regions.

Algorithm 3, which we devised to cope with this problem, proceeds by trial and error. It starts by drawing a random sample from the two distributions (line 2). The sample is twice as big as required because part of it will most likely be discarded. The sample is then normalized on line 3 and prohibited points are filtered out (line 4). If we are left with less than the required amount of points, we start over, else we continue (line 5). If continuing, we try to normalize again the points after the filtering (lines 14 and 16). They still have to not break the imposed dependencies (lines 15 and 17), and we need to have enough of such suitable points (line 18). Because we may

[12] Recall the notation we use to refer to a verb in a sentence, introduced in Section 4.2.1.

Algorithm 3: Sampling points from $\mathrm{P}(A) \times \mathrm{P}(B)$.

Input: number of points to generate: n; critical probabilities: $p_{0,A}$, $p_{0,B}$; prohibited combinations: ProhCombs

Output: n points from $[0, 1] \times [0, 1]$ not breaking prohibited combinations

```
repeat
    // Sample from A and B double the required amount.
    CoordsA ← sample( 2n, P_feat(p_0,A) ); CoordsB ← sample( 2n, P_feat(p_0,B) )
    // Normalize in each dimension to the maximum of 1.
    NCoordsA ← maxNorm( CoordsA); NCoordsB ← maxNorm( CoordsB)
    // Evaluate which of the points satisfy the dependencies.
    Good ← filter( breaksNot( ProhCombs), NCoordsA, NCoordsB)
    if length( Good) ≥ n then
        for Try ← 1 to 3 do
            // Sample to obtain future maxima in both dimensions.
            InitSample ← sample n points from Good
            // Find the maximum values and points having them.
            MaxA ← max( first coordinates of InitSample)
            MaxAPoint ← a point with MaxA as the first coordinate
            MaxB ← max( second coordinates of InitSample)
            MaxBPoint ← a point with MaxB as the second coordinate
            MaxPoints ← [ MaxAPoint, MaxBPoint ]
            // Select all points smaller than the maxima.
            Smaller ← filter( lessThan( Point( MaxA, MaxB ) ), Good)
            // Normalize the maxima to 1 in both dimensions.
            NMaxPoints ← map ( Point( a, b ) ↦ Point( a/MaxA, b/MaxB ),
                               MaxPoints )
            // Check again that the maxima satisfy all
               dependencies.
            if any( x ↦ breaks( x, ProhCombs), NMaxPoints) then continue
            // Normalize the smaller points to the maximum of 1.
            NSmaller ← map( Point( a, b ) ↦ Point( a/MaxA, b/MaxB ), Smaller)
            // Check again which points satisfy all dependencies.
            Suitable ← filter( breaksNot( ProhCombs), NSmaller)
            if length( Suitable) + 2 ≥ n then
                SmallerSample ← sample (n − 2) points from Suitable
                return concat( SmallerSample, NMaxPoints)
```

have up to $2n$ points when entering the inner cycle (on line 6), if we did not succeed with the second normalization and filtering, we can try other n of all the points and succeed with those. That is the rationale of the inner cycle. In the end, having found a big enough Suitable sample, we only need to make sure the returned sample contains points with the maximal values in both dimensions (line 20). Note that the pseudocode neglects some boundary cases, aiming only to describe the overall structure of the algorithm.

Obtaining Training Examples from Sampled Points

Having spent a substantial amount of CPU time on generating the 99-point sample for our two features, A and B, we want to obtain many training observations from each sampled point, as cheaply as possible. We will now describe how we add multiple samples of noise and break the logical tables into four observations, thus creating larger (and balanced) training data.

We use the customary additive noise model with the noise normally distributed. However, we want to account for differing reliability of the observed feature values. For instance, samples of features with many non-zero observations should be considered more reliable than those with only few non-zero values. Higher reliability of a feature translates into lower variance in the estimate of its values. This in turn means we need to reflect the different variance of the observations in our generated training data. Hence, we set the variance of the generated noise to be inversely proportional to a measure of feature reliability:

$$\mathrm{Var}(r) = \nu/(r + \varepsilon). \tag{4.5}$$

Here, ν is the noise scale, and ε a small number to prevent division by zero. This has the effect that it is harder for less reliable features to be classified as governed by some kind of dependency: we do not trust features which show us only a few non-zero observations.

For the purpose of choosing the right variance in the generated noise, we measure the reliability of a feature by its entropy. More precisely, we discretise the value of each feature observation into its responsibilities. This is done using the discretisation function discussed above, after optional squashing in case of non-boolean features. We then estimate the entropy of the resulting true/false observations, as though they were generated from an alternative distribution:

$$\mathrm{H}(F) = -\mathrm{P}(F = 0) \log \mathrm{P}(F = 0) - \mathrm{P}(F = 1) \log \mathrm{P}(F = 1). \tag{4.6}$$

We estimate these probabilities using MLE, i.e. by the ratio of true (or false) observations to all observations. Similarly to the estimates of p_0, the reliability estimate is also quantized. We use this measure because it penalizes features with few true observations, as well as features with few false observations. We do not want to trust such features because these few observations might have been only random.

The effect of different noise variance is illustrated in Figure 4.5. You can note that training data for features with low entropy are more or less uniformly distributed over the $[0, 1]$ interval, thus effectively prohibiting successful classification. Only when both the features have high enough entropy (which means that each of their observations provides decent amount of information), a confident classifier can be trained from the training data.

While we mentioned sampling 99 points from each distribution of A, B in the previous section, we actually generate many more training examples per distribution. Firstly, we sample

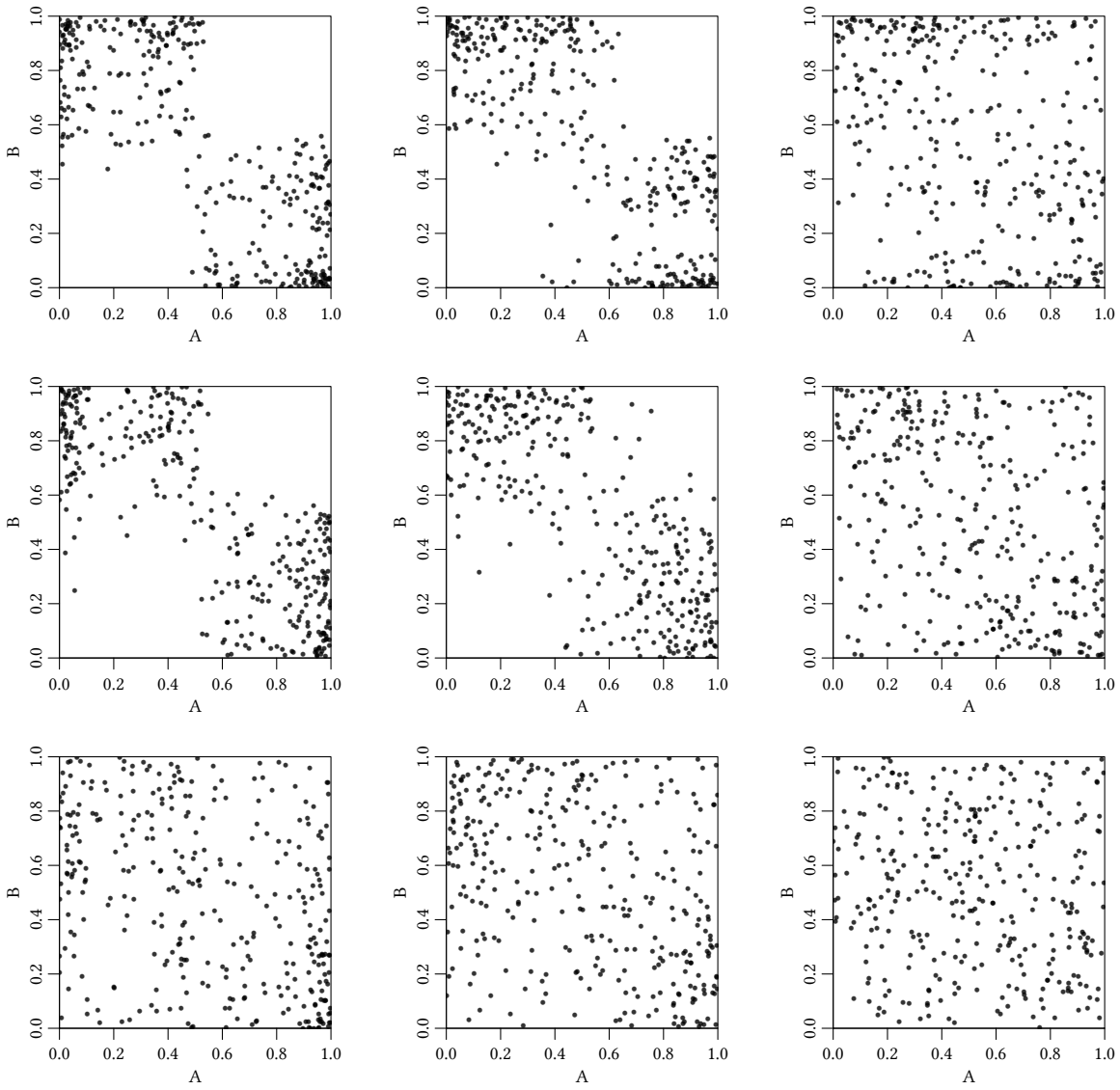

Figure 4.5: Random samples of 333 coordinates with differing entropy of A (0.9, 0.4, and 0.1 for the three columns, respectively) and entropy of B (0.9, 0.4, and 0.1 for the three rows, respectively). The probabilities $\mathrm{P}(A = 0)$ and $\mathrm{P}(B = 0)$ are low, the noise scale $\nu = 0.05$, and the features A and B are assumed to be governed by the equivalence $\neg A \leftrightarrow B$.

99 points for *each of the 7 possible relations* of the features: 4 implications, 2 equivalences, and the case of no dependency. Secondly, we generate 12 samples of noise for each of the points with no dependency, 12 samples for each implication, 12 other samples for each implication, which are treated specially, and 12 samples for each equivalence. Let us term the special noise samples for implications *one corner samples*; we will explain them shortly. The remainder of sampled points serves as four training examples each, and we obtain the four examples in the following way.

The reader will recall that the sampled points come from the joint distribution $\mathrm{P}(A, B)$, from which it is clear that they are pairs of real coordinates in the range $[0, 1]^2$. That corresponds to real-valued features after squashing, before discretisation. Similarly to the feature values, these sampled coordinates get discretised, in both their dimensions. Discretising in the dimension for feature A gives us responsibilities for $(A = 0)$ and $(A = 1)$; the other dimension works analogically for B. This all results in generating the four values for the table of boolean combinations of A and B. Depending on what dependencies we enforced when sampling this point, zero, one, or two corners of the table are (approximately) zero.

Note that all the dependencies we consider can be expressed in terms of the four implications. For the only non-trivial case, each equivalence consists of two directions of implication. If we classify only among the four implications, that already has enough power to distinguish all the dependencies. Furthermore, to reduce the running times and alleviate sparseness in the training data, we decided to throw away one of the four fields of the boolean table; if the values in the table are properly normalized, the fourth value is implied anyway. Hence we a) lose no information; b) reduce the dimension of the space of observations from 4 to 3; and c) need to classify only two ways: "displays the pattern of an implication" vs. "does not display the pattern". All that needs to be done is adding all 4 triples of the table's fields after discarding one to the pool of training examples. Those for which the dependency dictated the middle field to be zero are positive examples, all others are negative.

We admit that in doing so, we mix snapshots of different parts of the distribution (different parts of the domain $[0, 1]^2$) without remembering which came from where. However, the property of positive examples – that they capture the (noisy) combination $\langle$non-zero, zero, non-zero$\rangle$ – remains. And if a testing example happens to be more similar to other parts of the distribution than it was supposed to, it does not matter. There are always enough positive examples, as well as negative examples, for all the four corners of the distribution.

In fact, we manage to have the same number of positive and negative examples in the training data. We believe that that is desirable, because it sets the non-informative prior on the dependency classification – there might be a dependency between the given two features, or there might be not. The training set does not bias the decision. To balance the training set, we had to provide additional positive examples.[13] That is the purpose of the aforementioned one corner samples. These were implications for which only the positive example (the corner with the zero) was added to the training data. Having done so, we multiplied the original sample size by $12 \cdot 4$ (no-dependency) plus $4 \cdot 12 \cdot 4$ (4 implications) plus $4 \cdot 36$ (4 implications – positive samples) plus $2 \cdot 12 \cdot 4$ (2 equivalences), i.e. 480 times. 240 of those examples are positive.

[13] Each equivalence table has two zeros and two non-zeros, yielding two positive and two negative 3-field training examples. However, implications and no-dependency tables give us more negative examples (no zero) than positive ones (zero in the middle).

4.3.6 Classifying with A Linear Model

The previous section described how we generated prototype examples of the two classes – with a dependency, and without it. It would thus be natural to classify new observations by a prototype classification method, such as k nearest neighbours. We did that in our earlier experiments but we have found it very inefficient. Therefore, we reimplemented the classification to learn a linear model. The following are reasons that speak for a model-based approach, against k nearest neighbours classification.

1. With k nearest neighbours, extending the training data by sampling noise is not effective. It is not improbable for samples of the distribution to contain outliers. By drawing multiple samples of the outlier point with added noise, its importance is reinforced, and it effectively causes its neighbourhood to be wrongly classified. Therefore, prototypes have to be generated using the more expensive sampling of the joint distribution more heavily than by the simple sampling of the noise.

2. As the number of training examples grows, so too does the complexity of classifying new observations. There might be a better implementation of the algorithm for searching k nearest points in a 3-dimensional space than the one we had. But with the one we used, the time to run the experiment grew substantially when we increased the number of training samples. However, with fewer training samples, the quality of the classification was low.

In this section, we will give an overview of the remaining part of the machine learning process. Only the training data has been described so far. We thus have yet to address further processing (page 68), learning and selecting a linear model (page 69), but first of all, we should start by describing how the input observations were smoothed out.

Smoothing The Matrix of Observations

It is the case with high-dimensional feature vectors that most of them are outliers (as nicely explained in Hastie, Tibshirani and Friedman (2009, pp. 22–27)). Our feature space has 102 dimensions, out of which 57 contain a non-zero value (the other 45 we ignored). For all those dimensions, we have only 121 observations. When our goal is to discover whether two features are random or not, for all the pairs of features (there are $\binom{57}{2} = 1,596$ of them), it is most likely that a huge number of "discovered" dependencies would be just due to noise. Even though the chances of accidental "dependencies" between features with low support[14] will be countered by a larger amount of noise added to their training examples, it is likely for accidental "dependencies" to come up even among features with higher support. Therefore, we need to smooth out the matrix of observations. The way we did it was *matrix factorization* (MF), i.e., for our matrix of observations $V \in \mathbb{R}^{m \times n}$, finding two matrices, $W \in \mathbb{R}^{m \times k}$ and $H \in \mathbb{R}^{k \times n}$, such that their product approximates V: $W \cdot H \approx V$. The matrices W and H are called *factor matrices*.

Among the methods for matrix factorization, we opted for *non-negative matrix factorization* (NMF). There are two reasons for that:

[14] By the support of a feature, we mean the number of its non-zero observations.

1. By decomposing our matrix of non-negative observation values into *non-negative* factor matrices, we alleviate complications with mapping between the unrestricted and the non-negative domain. We already have a method for mapping arbitrary non-negative observation values onto boolean values; it is not clear how negative values would have to be interpreted.

2. As shown by D. D. Lee and Seung (1999), NMF tends to decompose the whole observation vectors into meaningful *parts*. That contrasts with other methods for MF, which tend to rather decompose into one or two dimensions accounting for most part of the matrix, and the rest of dimensions, covering small remaining discrepancies. The latter do not yield themselves to interpretation; they do not correspond to natural *parts* of the observations.

   We believe that having our observations factored into their natural parts will smooth out the rare, random co-incidences in the data, which is exactly the goal.

We did some initial experiments, comparing visualized results of NMF run with the matrix of observations and with a small artificial matrix. Based on them, we set the dimension for the factor space to 6. We believe that this setting best balances the variance of the result of NMF with its bias.

We did the matrix factorization using the R package 'NMF' (Gaujoux & Seoighe, 2010). This package implements a few different algorithms for NMF, from which we tried the following ones:[15]

"**brunet**" The standard NMF algorithm based on Kullback-Leibler divergence (D. D. Lee & Seung, 2000; Brunet, Tamayo, Golub & Mesirov, 2004).

"**nsNMF**" Non-smooth NMF – "brunet" modified so as to yield sparser results (Pascual-Montano, Carazo, Kochi, Lehmann & Pascual-Marqui, 2006).

"**pe-nmf**" Pattern-Expression NMF. Based on Euclidean distance and regularized for effective expression of patterns using factor vectors (Zhang, Wei, Feng, Ma & Wang, 2008).

We have chosen these algorithms for the following reasons: The "brunet" algorithm is considered the standard NMF algorithm, according to the package documentation, thus it should not be omitted. Sparse output of "nsNMF" should approximate our data well since they are also sparse. Lastly, "pe-nmf" was originally designed for blind separation of sources, which is aligned with our dispositions – we do not want to inform the NMF process in any way, it has to separate the source factors blindly.

The last mentioned algorithm, "pe-nmf", regularizes the search for the factor matrices with two parameters, α and β. These parameters are motivated by criteria on the ideal *basis vectors* (rows of W) and *coefficient vectors* (columns of H). The criteria demand that the basis vectors be spread, having wide enough angles between them; and that the basis vectors describe the column space of V as effectively as possible. These two criteria are reflected in α and β, respectively. The higher the value of a parameter, the more importance is put on that criterion.

We prepared the matrix of observations for matrix factorization by applying the squashing function (see Section 4.3.4) to the real-valued features. The boolean features we left alone.

[15] The descriptions are based on the documentation of the R package 'NMF'.

Let us call $\widetilde{V}(= W \cdot H)$ the matrix approximating V. The new observation values, fields of $\widetilde{V}$, are all guaranteed to be non-negative. However, none of them are guaranteed to be boolean (i.e., either 0 or 1) anymore. We have to treat all features as real-valued from this point on.

After the NMF is completed, we normalized values for each feature to the maximum of 1, the same way we normalize training observations.

Basis Expansions

We can now read from the (smoothed) matrix of observations for each pair of features, how many times they occurred in each combination (one of $\langle$0,0$\rangle$, $\langle$0,1$\rangle$, $\langle$1,0$\rangle$, or $\langle$1,1$\rangle$). Because each observation is split between true and false using the "discretisation" function, number of occurrences of feature combinations is again a non-negative real. The four numbers get aligned into a table, and the four corners of the table can be classified by their comparison to the training examples.

In an ideal world, if there was no noise in observed feature values, we could classify each corner positive (as indicating a feature dependency) iff it looked like this:

$n(\neg A, \neg B) > \tau$	
0	$n(A, B) > \tau$

,

for some defined threshold τ. (The above example would classify whether there is the $A \rightarrow B$ dependency.) In an ideal world, τ would have the limit of 0 as the number of observations grows.

If we accept the fact, however, that there is a noise inherently present in our observations, it adds a margin of uncertainty to the above numbers. The noise can be caused by, for example, inadequate video descriptions, softness of the kind of dependencies we are looking for, or non-precise feature extraction – recall the crude method for measuring IC. Allowing for noise, we probably want to classify the example on the left as positive – unless τ is low, as at the right hand scheme ($\tau = 0.01$), in Figure 4.6.

Let us call the fields of each observation triple $\langle x, y, z\rangle$ in the order top left, bottom left, bottom right, as drawn above. We are always looking for x and z being non-zero, and y being zero. To prevent the example on the right in Figure 4.6 being classified as positive, we should probably look at the ratio of y to the other counts (x and z). If these ratios are high enough, we will be confident that the example displays the pattern of a dependency. What "high enough" means precisely, will be determined by machine learning, depending on the set of training examples. What we are now more concerned about is whether it is exactly these two ratios that the classification can be based on. There are several statistics that can help classify an observation. Those we chose and used in our experiment, are enumerated in Table 4.7.

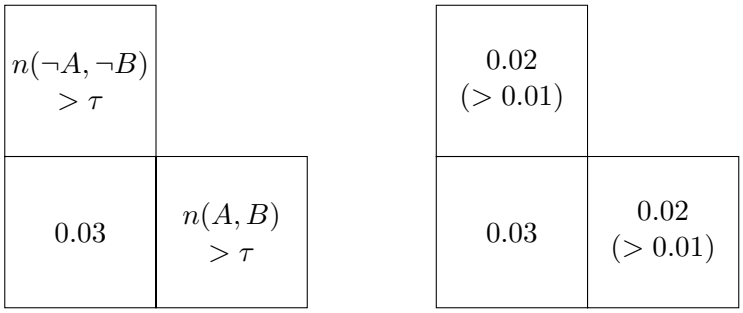

Figure 4.6: Example of observations with fuzzy zeros.

x	$\log(1/(x+y+z+\varepsilon))$	$1/(x+y+z+\varepsilon)$
y	$\log((y+\varepsilon)/(x+y+\varepsilon))$	$y/(x+y+\varepsilon)$
z	$\log((y+\varepsilon)/(y+z+\varepsilon))$	$y/(y+z+\varepsilon)$
y^2	$y/(x+\varepsilon)$	$y/(z+\varepsilon)$

Table 4.7: The expanded basis used. We assume the observation triplet to be of the form $\langle x, y, z\rangle$ for the notation here. The ε parameter (fixed to 0.01) serves solely to avoid division by zero.

Lasso-regularized Linear Models

We used the Lasso (Tibshirani, 1996) regression shrinkage and selection method as implemented in the R package lars v1.1. We generated a set of training data using the procedure described in Section 4.3.5, and expanded the basis, as noted in the preceding section. From the sequence of models found by Lasso, we selected one with the lowest degree of freedom within one standard deviation of performance from the best performing one.

Performance was measured on an evaluation set using residual sum of squares (RSS). RSS was evaluated for the true class labels being either -1 or 1, and the predicted labels likewise – if the model predicted a negative value, the prediction was "labeled" (substituted by) -1, otherwise by 1:

$$\text{RSS}_{\text{eval}} = \sum_{x \in E} (\text{sign}_1(\text{prediction}(x)) - \text{class}(x))^2, \tag{4.7}$$

where E are the evaluation data, and the function sign_1 is defined in the following way:

$$\text{sign}_1(x) = \begin{cases} -1 & x < 0 \\ 1 & x \geq 0. \end{cases} \tag{4.8}$$

To be able to compute the standard deviation in performance (i.e. in RSS), we generated 14 evaluation sets, each in the same way as training data were generated, but sampling the $\text{P}(A, B)$ distribution only 19 times for each of the sets (as opposed to 99 for the training set).

Having modeled the training data with the simplest good model, we needed to assess its trustworthiness over the range of values it produces. Without evaluating it, we could only classify positive vs. negative examples, taking 0 for the decision boundary. This would not suffice to establish confidence in different dependencies output by the algorithm.

We can expect the selected model to assign values around 1 to positive examples, and values around -1 to negative examples. These were the values it was trained to assign to the respective classes. Having little counterevidence, we assumed the predicted values for positive, as well as negative examples, to be normally distributed. Let us call these distributions *score distributions*. We estimated the parameters of score distributions from predicted values for all the evaluation data. If the model fits the data well, the score distribution for positive examples will have its mean in 1, while the negative examples will have it in -1. Thanks to the generated data being balanced, priors for the two classes are equal. Thus, it is easy to compute the optimal decision boundary for the two normal distributions. Regarding confidence into the classification decision, we used two formulae for it in our experiments.

In an earlier version of our program (version 2),[16] we adopted a simple formula for evaluating the classification confidence:

$$\text{confidence}_{\text{v2}}(x) = \frac{\text{P}(x = X)}{\text{P}(x = X) + \text{P}(x = Y)}, \tag{4.9}$$

with X being the class assigned to the observation based on the decision boundary (either "positive", or "negative") and Y being the other class. By x we denote the response of the model for the given observation (and we shall adhere to this notation until the end of the section).

A typical profile of confidence is displayed in Figure 4.7. Note that the confidence is the higher the further the predicted value from the middle ground. The decision boundary is not in zero anymore, it is computed from the estimated score distributions. This does not yield the optimal classification performance (at least for the data that the model was trained on), but it is a small price for having a confidence function. In any case, the confidence around the decision boundary is rather small, so we will not be confused by misclassified observations close to the decision boundary.

When evaluating the results of the version 2 of our program, we found that the above confidence function is not good in all cases. When the amount of noise in the training data is so high that even the best model performs rather bad, there is in fact little confidence that any future classification will be reliable. Yet the confidence function assigns values close to 1 to any observations further from the decision boundary, cf. Figure 4.8. This confidence profile arose for features that had only a few non-zero observations, whose training data are thus most impacted by added noise.

In order to lower the confidence in models that do not deserve it, we updated the confidence formula to:

$$\text{confidence}_{\text{v3}}(x) = \text{confidence}_{\text{v2}}(x) \cdot \text{prototypicality}(x; \mu_X, \sigma_X^2), \tag{4.10}$$

where *prototypicality* measures how close a point is to the center of a given normal distribution, X is the class assigned to the observation (as in equation (4.9)), and μ_X and σ_X^2 the estimated parameters of the score distribution for X. We will now turn our focus to the measure of prototypicality.

From illustrations of the problem with $\text{confidence}_{\text{v2}}$ (cf. Figure 4.8), we saw that it was points that were far from means of both the score distributions, that got assigned an inappropriately high confidence. They were not necessarily far in absolute terms, but they were far relative

[16] In this version, we did not include the $y/(x + \varepsilon)$ and $y/(z + \varepsilon)$ statistics into basis expansion.

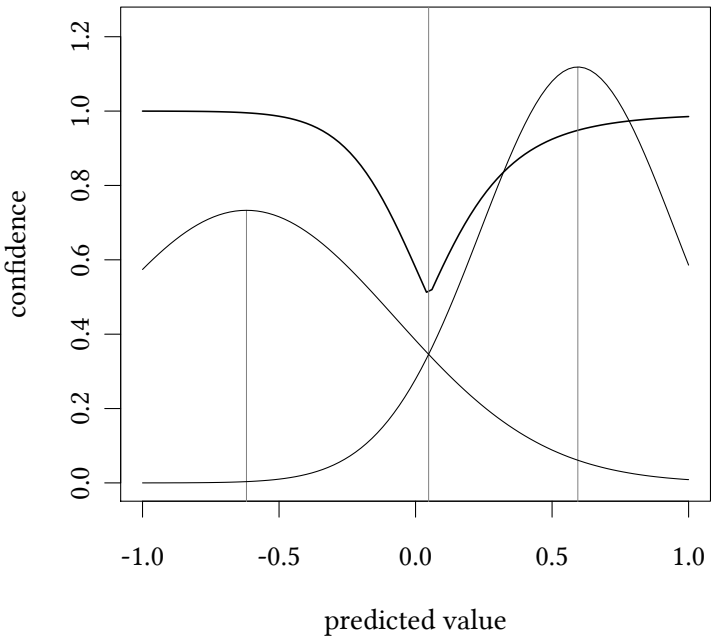

Figure 4.7: Confidence in classification for values predicted by a linear model. Here, it is computed using equation (4.9) for classes having their means far apart. The class distributions are drawn with thinner lines, the confidence curve is the top one. The middle vertical line demarcates the decision boundary.

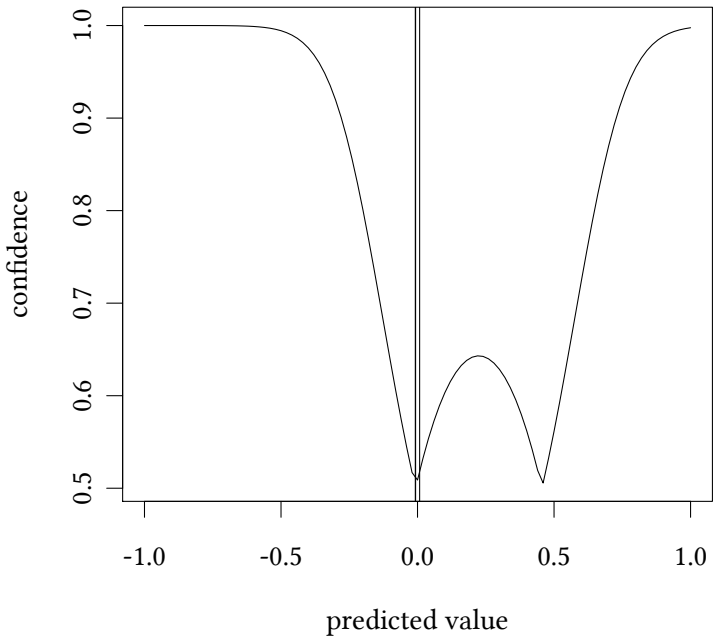

Figure 4.8: Confidence in classification for values predicted by a linear model. Here, it is computed using equation (4.9) for classes having their means close to each other.

to the score distribution parameters – they had a high *z-score*. That motivated us to defining prototypicality as a measure of centrality within the normal distribution. For a given distribution $X \sim N(\mu, \sigma^2)$, it maps a point x to the percentage of the probability mass that is further from μ than x. The mean thus has prototypicality of 1, and for instance the points $\mu \pm 1.96\sigma$ receive prototypicality of about 5%. The formula is the following:

$$\text{prototypicality}(x; \mu, \sigma^2) = 1 - 2 \cdot \left(F\left(|x - \mu|; 0, \sigma^2\right) - 0.5\right), \tag{4.11}$$

where F is the CDF of the normal distribution.

We also noticed that the problems with $\text{confidence}_{\text{v2}}$ are most severe when means of the score distributions are far from the (a priori) expected -1 for negative examples and 1 for positive examples. Therefore, we also tried to relativise the confidence with respect to this prior. For convenience, we chose the priors to be normal and centred in ± 1. We can then update the confidence formula simply by factoring in the prototypicality w.r.t. the prior:

$$\text{confidence}'_{\text{v3}}(x) = \text{confidence}_{\text{v3}}(x) \cdot \text{prototypicality}(x; \mu_{P_X}, \sigma^2_{P_X}). \tag{4.12}$$

Here, P_X is the prior for X – either centred in -1 (for $X = -1$), or else in 1. Only if x is close both to the mean of X and P_X, i.e. when also the two means are close to each other, does $\text{confidence}'_{\text{v3}}$ get high.

We compared the confidence measures with, and without the coefficient for the prior. The outcome is illustrated in Figures 4.9 and 4.10, which contain plots of both the measures together. As can be seen from the plots, the two measures are highly correlated. On the evaluation data, their Pearson correlation coefficient exceeds 0.9999. By that token, we abandoned the term for prototypicality w.r.t. the prior, and used the simpler formula (equation (4.10)) for confidence computations. We always determined confidence using the same data that had been used for model selection.

4.4 Method of Evaluation

In this section, we discuss against what data we evaluate result sets from the algorithm, and then consider different ways of measuring agreement between the reference result set and the result set originating from the algorithm.

4.4.1 Creating Reference Result Set

In the present problem, the one of classifying relations, we do not know the true answer even for the small sample data. Therefore, it gets slightly complicated when we need to compare answers of different algorithms aiming to solve the problem. What we chose to do was restricting the evaluation only to a small set of feature pairs, for which we know the true answer (from their definition).

In the feature space, described in Section 4.2.1, there are several mutually exclusive features – those capturing the verb tense, and those capturing its aspectual class. Furthermore, we can be very confident that one of the tense features will be true for verb 1 (the first matrix verb). This does not hold true only in the exceptional case when the description lacks a verb.

We built our reference result set based on the three assertions just described. For each verb position in sentence, we obtained positive examples for dependencies, such as:

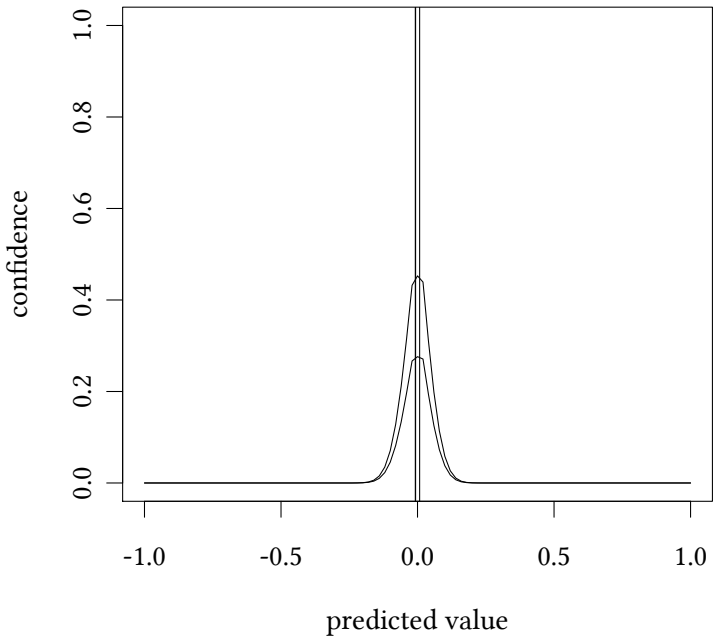

Figure 4.9: Confidence in decided class for values predicted by a linear model. Here, it involves the likelihood, and the prior too. These two features have high p_0, hence the amount of noise is relatively high and classification is virtually random.

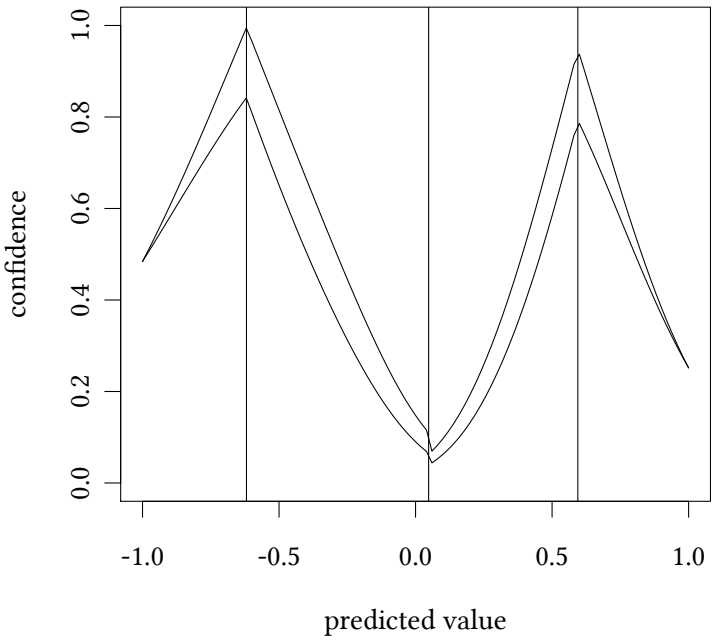

Figure 4.10: Confidence in decided class for values predicted by a linear model. Here, it involves the likelihood, and the prior too. These two features have low p_0, hence the amount of noise is relatively smaller and class centroids well separated.

Verb V is in the present tense. $\rightarrow$ Verb V is *not* in the past tense.

We also obtained negative examples, such as:

Verb V is in the present tense. $\rightarrow$ Verb V is in the past tense.

The former dependency holds true in all cases, and therefore it is assigned the value 1 in the reference result set; whereas the latter dependency can never be true, hence it is assigned the value -1. Finally, there is the almost certain dependency for verb 1:

Verb 1 is *not* in the past tense. $\rightarrow$ Verb 1 is in the present tense.

We assign this dependency the score of 0.9 in the reference result set.

The resulting reference set comprises 121 asserted dependencies, out of which 25 are positive examples, 95 negative examples, and the remaining 1 being the special positive example with confidence 0.9.

4.4.2 Measuring Agreement

We now turn to evaluation measures for scoring the result set output by the algorithm relative to the reference result set. Each result set is a sequence of confidence scores between -1 and 1, and we need the evaluation measure to assign a better score to result sets more similar to the reference result set, and a lower score to the less similar ones.

An obvious measure for similarity of two result sets is the *correlation coefficient*. The correlation coefficient reflects the linear relationship between two sequences. If there is a strong linear relationship, correlation is high (close to 1) in the absolute value. The sign of the correlation coefficient is the same as the sign of the linear coefficient of the relationship.

While correlation captures sharpness of the linear relationship (or absence thereof), it ignores the absolute value of the linear coefficient of the relationship. Therefore, we estimated the *linear coefficient* separately. We trained a linear model predicting the confidence value output by the algorithm from the corresponding value in the reference result set.[17] The only coefficient of the linear model is then the *linear coefficient* which we use as the measure of agreement.

Another view on the agreement is in terms of the *error* introduced by approximating the reference result set by the results output by the algorithm. We found the *RMS (root-mean-square) error* to be more suitable a measure than other error measures. The RMS error is defined as:

$$\mathrm{RMS}(\boldsymbol{a}, \boldsymbol{b}) = \sqrt{\frac{\sum_{i=1}^{n} (a_i - b_i)^2}{n}}, \tag{4.13}$$

where $\boldsymbol{a}$ and $\boldsymbol{b}$ are vectors of the length n (the reference result set, and the algorithm output, in our case).

[17] The linear model has this single input. Specially, it does not use the constant input (intercept), which is a common practice.

4.5 Results

We ran the experiment varying settings w.r.t. the matrix factorization (MF)method, and varying the level of noise (or, the noise scale ν, as introduced in equation (4.5)). Apart from MF methods enumerated in Section 4.3.6 (page 67), we included the trivial baseline of not using approximation with MF at all. In this section, we provide, compare and comment on results from the different runs.

4.5.1 General Evaluation of All Experiment Runs

The resulting scores for all runs of the experiment are shown in Figure 4.11. Let us first describe the upper plot. On the x-axis are individual runs of the experiment, each with a different setting. They are divided into four parts, corresponding to the three NMF algorithms plus the baseline of performing no NMF. Within each part, the runs are ordered w.r.t. the RMS error. The marks below the plot indicate for each run what noise scale ν was used. We experimented with three values for ν: 0, 0.05, and 0.1.

The horisontal lines highlight important values: Zero is the minimal required value of the linear coefficient for the results to be trustworthy. Similarly, zero is a lower bound for acceptable values of correlation. Value 1 is the theoretical maximum for the correlation coefficient. With confidence scores taking the value from $[-1, 1]$, the value of 1 is also the maximal desirable value for the RMS error, as the RMS error of 1 can be trivially achieved by outputting 0 for every observation. The two horisontal lines between 0 and 1 demarcate the maximum achieved correlation coefficient (0.885), and the minimum achieved RMS error (0.530).

Now, having explicated the basic criteria for considering a result good, we note that most of the experiment runs clearly fail the criteria. It is only the best scoring run for the "none" and "brunet" algorithms, and the five best scoring runs for "pe-NMF", which meet the criteria.

Surprisingly enough, the "nsNMF" algorithm, supposed to yield sparse results, and thus capture well our input data, did not show good results in any of the runs. This might be caused by the small size of our training sample, though – the algorithm approximated the sparse, ragged matrix by another sparse and ragged matrix, but with too few observations to deliver a reliable result.

On the other side of the spectrum is "pe-NMF". For the five runs that meet the above criteria, the scores are very promising even in absolute terms. The correlation coefficient exceeds 0.7 for the four best-scoring settings, getting as high as 0.885 for the best setting. The RMS error gets down to 0.53, corroborating that the extracted dependencies are fairly well aligned with the reference result set.

The careful reader will have noticed that the best experiment setting within each NMF method was with $\nu = 0$, i.e. with no added noise. We included this setting to have a baseline w.r.t. the noise parameter. However, it is not meant to reflect performance of the algorithm on real inputs, because those are going to be always encumbered with error, especially should the features of descriptions be extracted automatically.

For the above reason, we are more interested in the results for $\nu > 0$. These are shown in the lower plot of Figure 4.11. This plot clearly justifies the contribution of NMF, its potential benefit over the "none" baseline. Whereas the baseline was equally good as the best-scoring run of "pe-NMF" on results including the case of $\nu = 0$, it is uncompetitive within runs having $\nu > 0$.

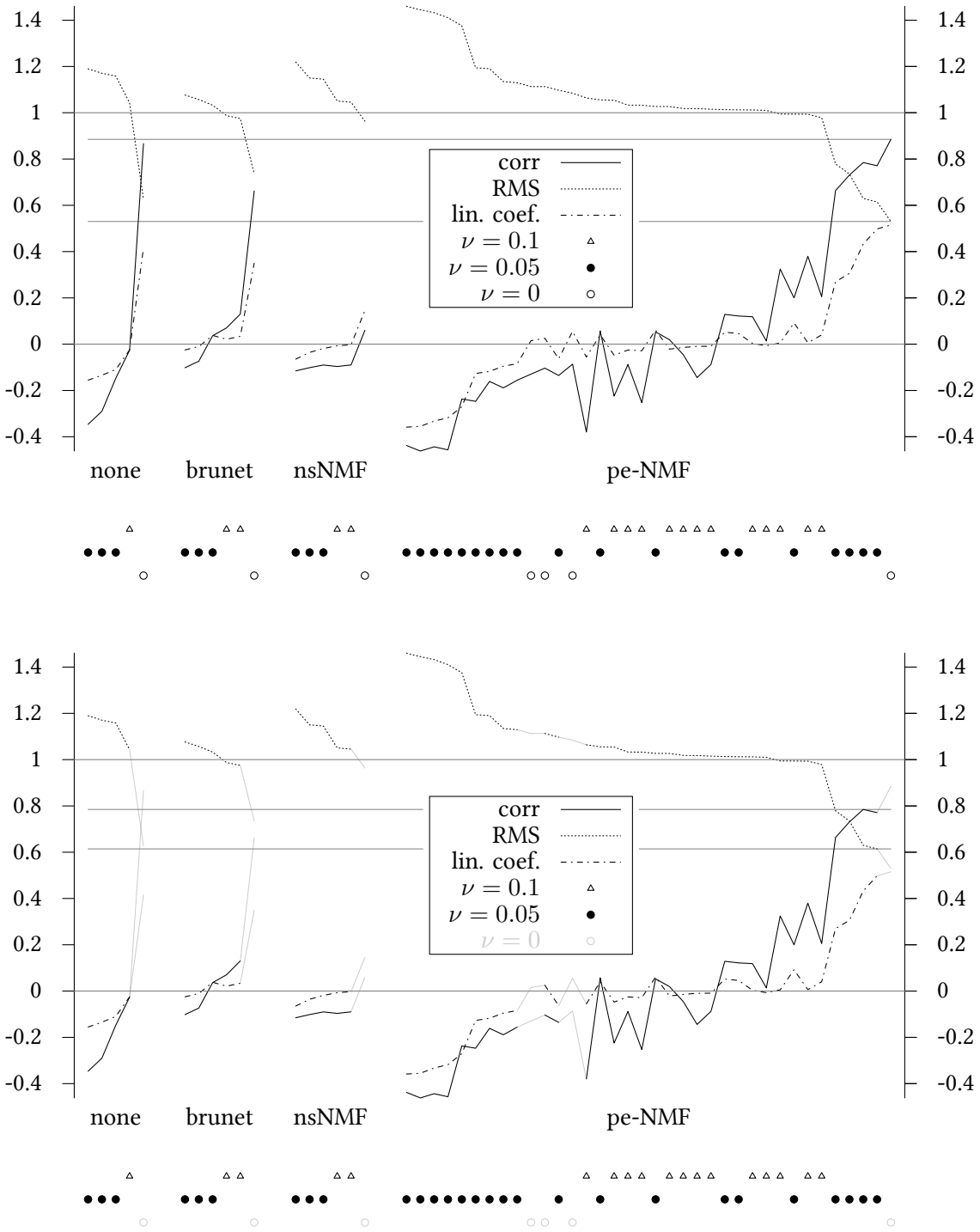

Figure 4.11: The three evaluation scores (correlation, RMS error, and linear coefficient) for all the experiment settings. Each plot is divided into four regions by the NMF method, and within each method, different experiment runs are ordered by the RMS error in the descending order.

FAST

	$\nu = 0$			$\nu = 0.05$			$\nu = 0.1$		
			β						
			1	4	16		1	4	16
		ε	0.66	0.13	-0.46	ε	-0.09	0.32	-0.14
		1	-0.46			1	-0.38		
		4	-0.19			4	-0.25		

THOROUGH

α											
		1	16		ε	1	4		ε	1	4
	ε	0.89	-0.09	ε		0.78	0.20	ε		0.02	0.38
	1	-0.13		1	-0.16	-0.24	0.06	1	0.12	-0.22	0.01
	4	-0.10		4	-0.25	-0.16	0.05	4	0.21	-0.09	-0.05

Table 4.8: Correlation scores for different parameters of pe-NMF.

Similarly, also "brunet" drops dramatically when we ignore the no-noise runs. On the other hand, there are settings of "pe-NMF" under which it performed almost as well as in the no-noise case. This can be seen at the maximum-correlation and minimum-RMS-error horisontal lines, which shift only slightly from their position in the upper plot. The good performance of the method on noisy data, and particularly its relative effectiveness on the noisy data as opposed to the no-noise situation, are the main results of our experiment.

Most of the experiment settings ended unsuccessful in the evaluation, but that does not mean we should ignore them. They can show us a few interesting facts.

Firstly, although these unsuccessful experiment settings generally achieve negative or close-to-zero correlation score, there are four runs of "pe-NMF" with the correlation coefficient higher (between 0.2 and 0.4), but with a very low linear coefficient score. This proves the usefulness of the linear coefficient score, because it is this score that clearly says those runs should be discarded.

Secondly, note that among the less successful experiment runs in the lower plot of Figure 4.11, the settings with more noise consistently lead to better results. This suggests that the algorithm can actually extract more information from runs with lower level of noise, but it uses the information incorrectly. This could be partly an artifact of the reference result set, which is heavily biased towards negative examples (as would, presumably, be the complete set of all true relations). Only "pe-NMF" can overcome this obstacle and emphasize the "true" observations, damping those due to noise.

4.5.2 Comparison of Parameter Settings for "pe-NMF"

Tables 4.8, 4.9, and 4.10 provide an overview of the correlation coefficient, RMS error, and linear coefficient evaluation scores, respectively, for different settings of the parameters α, β, and ν for "pe-NMF". We ran the experiment in some of the settings more than once, but the tables contain only the evaluation score for the first run of the experiment for each parameter combination.

Each of the tables is organized in the following way. It is divided into three columns,

		$\nu = 0$			$\nu = 0.05$			$\nu = 0.1$		
					β					
					1	4	16	1	4	16
FAST				ε	0.78	1.01	1.41	1.03	0.99	1.02
				1	1.45			1.06		
				4	1.13			1.03		
	α									
		1	16		ε	1	4	ε	1	4
	ε	0.53	1.08	ε		0.63	0.99	ε	1.03	0.99
THOROUGH	1	1.11		1	1.19	1.38	1.06	1.01	1.05	1.01
	4	1.11		4	1.19	1.13	1.03	0.98	1.02	1.02

Table 4.9: RMS-error scores for different parameters of pe-NMF.

		$\nu = 0$			$\nu = 0.05$			$\nu = 0.1$		
					β					
					1	4	16	1	4	16
FAST				ε	0.27	0.05	-0.32	-0.03	0.01	-0.01
				1	-0.36			-0.06		
				4	-0.10			-0.03		
	α									
		1	16		ε	1	4	ε	1	4
	ε	0.52	0.06	ε		0.43	0.09	ε	-0.02	0.01
THOROUGH	1	0.02		1	-0.12	-0.27	0.04	1	0.00	-0.05	-0.01
	4	0.03		4	-0.13	-0.08	0.06	4	0.04	-0.01	-0.01

Table 4.10: Linear coefficient scores for different parameters of pe-NMF.

corresponding to different values for the noise scale ν. Similarly, it is divided into two rows, labeled FAST and THOROUGH. Difference between these will be explained shortly. Each of the cells contains a smaller table, rows of which differ w.r.t. the parameter α, and the columns differ w.r.t. β. (ε conventionally stands for a negligibly small value.) The smallest table cells thus each uniquely correspond to a combination of ν, α, β, and one of FAST or THOROUGH. These cells are either gray, meaning that we did not run an experiment with those parameters,[18] or colour-coded and showing the corresponding evaluation score. For the colours, pure green or blue would be an ideal, pure red would be the opposite from ideal.

FAST is the default experiment setup, as described in Section 4.3, whereas in the case of THOROUGH, we generated 12 times more samples of the joint distribution and only 1 sample of noise for each of them. This way, we traded speed for accuracy (cf. the motivation for not doing this in Section 20, page 63). We decided to run the THOROUGH experiments after obtaining the initial results, seeing the relatively good performance of "pe-NMF", and having optimized the script enough to make the THOROUGH version feasible timewise.

All the tables illustrate that the setting $\alpha := \varepsilon,\ \beta := 1$ performs the best for the low ($\nu = 0$) and moderate ($\nu = 0.05$) level of noise. In the THOROUGH variant with $\nu = 0.1$, we can see a rather weak trend for the correlation coefficient to improve with either $\alpha = \varepsilon$ or $\beta = \varepsilon$ and the other parameter set to 1, or better, to 4. However, the other two tables clearly show that this is irrelevant – especially because the linear coefficient is practically equal to 0. Hence, we have no well performing algorithms for the setting with high noise level. That is to be interpreted as that there is virtually no hope for extracting any dependencies from so small data with this much noise, rather than being unable to find a good algorithm.

4.5.3 Top Extracted Relations

In this section, we present the top extracted results by the algorithm under the best-scoring settings for the low noise level ($\nu = 0$), and for the moderate noise level ($\nu = 0.05$). These best-scoring settings were, according to Section 4.5.1, "pe-NMF" with $\alpha = \varepsilon$ and $\beta = 1$, run in the THOROUGH setting (i.e., sampling more points of the joint distribution of each two features, and getting the less noise samples), for both $\nu = 0$ and $\nu = 0.05$.

We filter the results after they have been output by the algorithm, to select only pairs of features which both have high enough support. To be more precise, we apply a threshold θ for the x and z value of each observation, where x and z are the two values of the $\langle x, y, z\rangle$ corner that are supposed to be non-zero (this notation was introduced in Section 4.3.6 on page 68).

Tables 4.11–4.13 list the top extracted dependencies for $\nu = 0$, with the threshold θ set to 0, 0.05, and 0.1, respectively. Tables 4.14–4.16 list the top extracted dependencies for $\nu = 0.5$, for the same sequence of thresholds θ.

Unfortunately, it is hard to directly interpret the extracted elementary dependencies. They do not appear to be nonsensical or contradictory, which is a positive sign; however, many seem to persist to random properties of the training data. This effect of overtraining is countered by the threshold θ, at the price of considerably reducing size of the result set. For our experiment, $\theta = 0.05$ seems to be a reasonable compromise.

[18] Note that we never set $\alpha = \beta = \varepsilon$, since that would put negligible weight on the two defining criteria of "pe-NMF", reducing it to the standard NMF.

dependency	confidence
verb 2: prog $\rightarrow \neg$(verb 1: PAT IC)	0.98
verb 2: arg2 IC $\rightarrow \neg$(verb 1: PAT IC)	0.98
verb 2: argM IC $\rightarrow \neg$(verb 1: PAT IC)	0.98
verb 2: loc IC $\rightarrow \neg$(verb 1: PAT IC)	0.98
verb 1: loc IC $\rightarrow \neg$(verb 1: past)	0.97
verb 1: loc IC $\rightarrow \neg$(verb 1: culproc)	0.97
verb 1: loc IC $\rightarrow \neg$(verb 1: point)	0.96
verb 1: loc IC $\rightarrow \neg$(verb 1: iter)	0.96
verb 3: ACT IC $\rightarrow \neg$(verb 1: prog)	0.96
verb 3: prog $\rightarrow \neg$(verb 1: prog)	0.96
verb 2: PAT IC $\rightarrow \neg$(verb 1: prog)	0.95

Table 4.11: Top extracted dependencies for $\nu = 0$; $\theta = 0$.

dependency	confidence
verb 3: ACT IC $\rightarrow \neg$(verb 1: prog)	0.96
verb 3: prog $\rightarrow \neg$(verb 1: prog)	0.96
verb 3: verb IC $\rightarrow \neg$(verb 1: prog)	0.95
verb 1: adv IC $\rightarrow \neg$(verb 1: prog)	0.94
verb 1: verb IC $\rightarrow$ verb 1: prog	0.85
verb 2: adv IC $\rightarrow \neg$(verb 1: loc IC)	0.81
verb 1: ACT IC $\rightarrow$ verb 1: prog	0.81
verb 1: verb IC $\rightarrow$ verb 1: process	0.79
verb 1: loc IC $\rightarrow$ verb 1: verb IC	0.78
$\neg$(verb 1: prog) $\rightarrow$ verb 2: adv IC	0.77
$\neg$(verb 1: prog) $\rightarrow$ verb 3: verb IC	0.76

Table 4.12: Top extracted dependencies for $\nu = 0$; $\theta = 0.05$.

dependency	confidence
verb 1: loc IC $\rightarrow$ verb 1: verb IC	0.78
verb 1: loc IC $\rightarrow$ verb 1: ACT IC	0.74
verb 1: loc IC $\rightarrow$ verb 1: pres	0.68
verb 1: loc IC $\rightarrow$ verb 1: process	0.41

Table 4.13: Top extracted dependencies for $\nu = 0$; $\theta = 0.1$.

dependency	confidence
¬(verb 1: prog) → verb 2: ACT IC	0.95
verb 1: ACT IC → verb 1: prog	0.92
verb 2: prog → ¬(verb 1: PAT IC)	0.91
verb 2: arg2 IC → ¬(verb 1: PAT IC)	0.91
verb 2: argM IC → ¬(verb 1: PAT IC)	0.91
verb 2: loc IC → ¬(verb 1: PAT IC)	0.91
verb 1: pass → verb 2: verb IC	0.91
verb 1: verb IC → verb 1: prog	0.91
verb 3: ACT IC → verb 3: prog	0.91
verb 1: arg2 IC → ¬(verb 1: process)	0.90

Table 4.14: Top extracted dependencies for $\nu = 0.05$; $\theta = 0$.

If we compare Tables 4.11–4.13, we notice an evident difference in the top extracted dependencies for different values of θ. Table 4.11 consists only of dependencies of the type $B \rightarrow \neg A$, where, furthermore, particular features seem to be especially prone to being classified positively – either as the antecedent (as the feature "verb 1: loc IC" here), or the consequent (here, "verb 1: PAT IC"). This can be a sign of overtraining, which causes too high sensitivity to random co-incidences. However, we can find this property also in all the other tables, so, probably, the algorithm was vulnerable to overtraining under all settings. This does not mean the algorithm was bad – recall that it performed very well in comparison to the reference result set. It rather implies that more training data would be beneficial.

Regarding the type of extracted dependencies, it seems that while Table 4.11 is clearly dominated by the type $B \rightarrow \neg A$, Table 4.13 contains only the type $B \rightarrow A$[19] and Table 4.12 is mixed. This indicates that $B \rightarrow A$ be the type of dependency best represented in the data, whereas $B \rightarrow \neg A$ the worst represented, yet the strongest (resulting in the highest confidence values). Without further investigation, though, we cannot explain what causes this happening.

The results for $\nu = 0.05$ (Tables 4.14–4.16) look very similar to those for $\nu = 0$, at least on the qualitative level – our comments from above paragraphs can be transfered here almost verbatim, only noting a difference in some other implication types chiseling in to the top dependencies where $\theta = 0$. The quantitative comparison of result sets for $\nu = 0.05$ and $\nu = 0$, truncated only to dependencies with a confidence higher than 0.75, is shown in Table 4.17. It is somewhat interesting that the more noisy setting always generates more dependencies with high confidence than the no-noise setting. This could be an effect of the push to high confidences, created by the reference result set used for evaluation. It is more interesting to see the MAP (*mean average precision*) and other scores corroborating that the robustness of extracted dependencies grows with growing θ, negating the effects of ν.

Having compared the effects of different values of θ, and considering that noise needs to be taken into account in later applications (i.e., we aim for $\nu > 0$), the experiment described in this chapter has determined which settings are the most promising. The best settings are the

[19] It is indeed $B \rightarrow A$, and not $A \rightarrow B$, because we always denote A the feature that comes first in the entire feature vector, the one at a lower index.

dependency	confidence
verb 1: ACT IC → verb 1: prog	0.92
verb 1: verb IC → verb 1: prog	0.91
verb 1: ACT IC → verb 1: process	0.89
verb 1: loc IC → ¬(verb 1: arg2 IC)	0.89
verb 2: verb IC → verb 2: pres	0.86
verb 2: adv IC → ¬(verb 1: loc IC)	0.85
verb 1: verb IC → verb 1: process	0.83
verb 1: argM IC → ¬(verb 1: PAT IC)	0.81
verb 1: loc IC → verb 1: verb IC	0.81
verb 1: loc IC → verb 1: ACT IC	0.77
verb 1.1: PAT IC → ¬(verb 1: PAT IC)	0.77
verb 2: ACT IC → verb 1: ACT IC	0.75

Table 4.15: Top extracted dependencies for $\nu = 0.05$; $\theta = 0.05$.

dependency	confidence
verb 1: loc IC → verb 1: verb IC	0.81
verb 1: loc IC → verb 1: ACT IC	0.77
verb 1: loc IC → verb 1: pres	0.71
verb 1: loc IC → verb 1: process	0.40

Table 4.16: Top extracted dependencies for $\nu = 0.05$; $\theta = 0.1$.

θ	ν	# of results	precision	recall	MAP
0	0	127	(100%)	(100%)	(100%)
	0.05	648	4%	22%	5%
0.05	0	11	(100%)	(100%)	(100%)
	0.05	12	42%	45%	33%
0.1	0	1	(100%)	(100%)	(100%)
	0.05	2	50%	100%	100%

Table 4.17: Quantitative comparison of the results for $\nu = 0$ and $\nu = 0.05$, restricted only to those with confidence greater than 0.75. The evaluation measures are evaluated with the $\nu = 0$ setting as the ground truth, and $\nu = 0.05$ as its approximation.

following:

training strategy	THOROUGH
$\boldsymbol{\nu}$	0.05
NMF algorithm	pe-NMF
$\boldsymbol{\alpha}$	ε
$\boldsymbol{\beta}$	1
$\boldsymbol{\theta}$	0.05.

Table 4.18: The best settings for dependency extraction, as determined in the present experiment.

4.6 Discussion And Future Work

In the previous sections of this chapter, we have described the initial experiment with extracting feature dependencies, and reported on its outcomes. This opens the door to automating the whole process, especially the feature extraction, to be able to scale up to much larger input data. That will be the subject of the next chapter. However, there are a few potential improvements of the process other than automation, and these are outlined below.

We should first discuss the fundamental decision, why we developed a completely new procedure for relation extraction when there is a well studied problem with a very similar formulation, known as *association rule mining*. The reasons are the following:

1. Our data have properties we wanted to account for in the relation extraction procedure. Namely, we have a basic idea about the shape of the distribution of values of the features we work with. As discussed in Section 4.3.3 and illustrated in Figure 4.2, the distribution shape is expected to fit the parameterized PDF given by equation (4.3). The properties of the distribution that we consider important are the point probability mass in 0, and the log-normal tail.

   While incorporating these distribution properties into the paradigm of prototype generation is simple, and even required, we are not aware of a way of injecting such assumptions into the paradigm of association rule mining.

2. The data we had available for this experiment were way too small for application of association rule mining methods, as far as we know. Where spurious dependencies are practically ignored in our approach due to noise added to the sampling procedure, association rule mining would be likely to report them as true dependencies, leading to gross overfitting to the training dataset.

On the other hand, association rules can have more antecedents in them, whereas we only considered simple implications with only one antecedent. However, be the approach whatever, much larger data would be needed for extracting rules involving more variables, in order to keep the number of purely random relations discovered reasonably low. As an alternative way to including more variables, we suggest combining the simple implications into one whole in the following paragraph.

The reader will agree that the list of top n extracted dependencies, as shown for example in Table 4.15, is hard to interpret. It would thus be useful to combine them into a larger structure and discover repetitive patterns within that structure. One such basic pattern would be, for example, the equivalence of features. The equivalence $\neg A \leftrightarrow B$, for instance, would then be identified by the edges $B \rightarrow \neg A$ and $\neg A \rightarrow B$ both having a high confidence assigned, possibly requiring also the confidence of the other two implications, $A \rightarrow B$ and $B \rightarrow A$, to be very low. The results from our experiment actually feature at least one such equivalence, captured by the third and the last dependency in Table 4.12. However, without the additional computational processing of our results, this implied equivalence would be easy to overlook.

Eventually, we could think of reconciliating all the elementary dependencies into one whole, either the strict structure of a Heyting algebra, or into a fuzzy variant thereof, weighted with scores obtained from what was originally the confidence scores. This would provide one *consistent* view of all dependencies. As noted in the previous paragraph, this structure could serve for finding larger patterns of how information tends to be distributed over sentence, another way to this target besides the one outlined in the introduction of Section 4.3.2.

Let us now turn our attention to the one goal we have missed so far. In Section 4.1, we set the three main objects for our analysis to be: 1. level of detail; 2. focus shift; and 3. verb form. The first and the third are well represented in our current analysis, however the second is not. To properly account for focus shift, we would need to word-align different descriptions of the same clip, and then determine the *standard sentence template* for descriptions of the clip. We motivated this task earlier in Section 4.2.2, leaving the suggested method for now.

The standard sentence template for a clip could be found in the following way. We would first need to find the *key verb* of the descriptions. *Key verb* was defined on page 51 as the verb denoting the most salient event in the clip. At the same place, we noted that the key verb appeared as the verb 1 in a large majority of descriptions in the annotated sample of MSR VDC. Hence we suggest to use a heuristic that counts which verb appears most often in the first position in descriptions for the clip, and asserts that to be the key verb. Then, it collects arguments that this verb takes in different descriptions, and constructs the standard sentence template by enumerating the verb and its arguments in the natural order (actor, verb, patient, effect, adverbial arguments). Finally, a feature would be included that reflects the sentence focus, marking constituents that appear earlier in the sentence than they should according to the standard sentence template.

Finally, let us revisit the estimation of IC. In our experiment, we approximated the IC of a word w using ICs of its known senses, linearly combined with each coefficient being the prior for the sense derived from SemCor (see equation (4.1) on page 50).

An obvious room for improvement here is deriving the sense priors from the corpus we work with, or at least from texts from the same domain. This would require either extensive manual annotations, or application of WSD to at least a part of the corpus.

If WSD was successfully applied to the corpus, it would actually assign each word a single sense s and we would not need the formula involving sense priors anymore; the definition of IC (equation (IC) on page 49) would be directly applicable. However, we could estimate the IC even better with WSD output in the form of the posterior estimate of $\mathrm{P}(s|w, C)$, with C denoting

the context. In that case, the MLE estimate of the IC would be the following:

$$\mathrm{IC}_{\text{optimal}}(w) = -\sum_{s \in S(w)} \frac{p(s \mid w, C)}{\sum_{s \in S(w)} p(s \mid w, C)} \cdot \log_2 p(s). \tag{4.14}$$

These formulae (equations (4.1) or (IC) with the adapted sense priors, or equation (4.14)) need to be tested on real data before we can conclude which one performs better. Although we have discussed their theoretical merits, their practical performance is probably going to be impacted heavily by the performance of WSD.

Another idea for improving our IC estimates would allow us to include a majority of adjectives and adverbs, even if using the same resource (WordNet InfoContent). We would assume that adjectives and adverbs derived from nouns or verbs express a content similar to the noun or verb's content, differing merely in the syntagmatic properties. Under this assumption, we could take their IC to be the same as the IC of the noun or verb they are derived from. However, in the current version of this experiment, we simply ignored these parts of speech unless they happened to be covered by WordNet InfoContent.

Chapter 5

Scaling Up

In this section, we shall discuss the next steps that would need to be taken to allow for running the dependency extraction on larger input data without human intervention. The main task that remains to be tackled is automation of the feature extraction. We will document what we have done on the way to solving the task in the following sections, and outline what should be done next. By the "larger input data" we want to be able to process, we always mean the whole set of English descriptions from the MSR VDC corpus.

5.1 Word Alignment

As mentioned in Section 4.6 in the previous chapter, we need to word-align MSR VDC in order to be able to capture the phenomenon of focus shift. However, word alignment also provides an invaluable source of information for determining semantic roles, at least in our case of a parallel corpus like MSR VDC. Semantic roles form the base for extracting the features as we have defined them.

Let us now demonstrate how word alignment can be used to complement, or even supersede a standalone *semantic role labeling* (SRL) system. We will reuse the examples from Chapter 4:

Example 8.

1. A man is watching two bear cubs digging.
2. The baby bears dug in the dirt for insects.
3. It is under the roots of a big tree that the two baby bears are digging.

In order to extract the features of these sentences, we need to determine that *two bear cubs*, *the baby bears*, and *the two baby bears* all fill the role of the actor of the verb *to dig* (and similarly for all other semantic roles and other verbs). If we apply a standard SRL system, it processes the three sentences separately, using syntactic and semantic hints to find the actor. It does not take into account the fact that the words *bear* or *bears* denote the same thing in all the descriptions.

However, the sentences of MSR VDC are simple enough that we can assume that the same word (other than a stopword) occurring in multiple descriptions of one clip denotes the

same object. If, furthermore, it belongs *syntactically* to the same verb in those descriptions, it is almost certain that it fills the same *semantic* role.

We can take this line of thought a bit further. If we substitute the requirement for the words in parallel descriptions to be the same by the requirement for them to correspond to each other, the argument is still valid. Here, the motivation for word alignment arises – we need to determine which words correspond across different descriptions.

There are two possibilities how to incorporate word alignment. It can be used side by side with a standalone SRL system, and their independent predictions combined to improve the accuracy of SRL. Alternatively, and this might be the first thing to try, word alignment can be used alone to approximate SRL. We think this is a promising approach, given the high number of parallel descriptions for most clips in MSR VDC.

In the latter approach, word alignment itself would only determine what are the role fillers and their correspondence across descriptions. However, it would not tell us *what role* they fill. Therefore, it would need to be helped by a heuristic to determine which role is which. Such a heuristic could be based on simple observations, such as:

- If a verb is in the active voice, its subject is most likely its actor; otherwise, it is most likely the patient, or maybe the effect.
- If there are two candidates for the two roles of the patient and the effect of the same verb among descriptions of one clip, the word that appears more often as the direct object of its governing verb is more likely to be the patient.
- An adverb of place, which is syntactically a child of the verb, most likely expresses the locus for the verb, and so does its whole syntactic subtree.

5.1.1 Setup of Word Aligning

We word-aligned the MSR VDC corpus using the Berkeley aligner.The training data for the aligner comprised pairs of corresponding sentences from the training portion of MSR VDC (700,990 pairs), and from MTA and MTC (283,044 pairs). 9 from all the training pairs (all 9 from MTA or MTC) were not considered for training of the aligner due to a maximum sentence length limit.

We additionally compiled lists of all words present in the input corpora and in the "test – known" portion of MSR VDC, and fed the identity mapping of these words onto themselves into the aligner as an a priori translation dictionary. Each of these identity mappings was assigned the weight of 4, i.e. equal to the weight of 4 ordinary training observations. We did not use the "test – unknown" portion of MSR VDC for training in any way. The identity mappings comprise 20,070 pairs for MTA and MTC, 4,687 pairs for MSR VDC, and they were also part of the training data.

After a cursory evaluation of smaller setups, we decided to use the following cascade of models in the training: IBM Model 1 (8 iterations), IBM Model 2 (12 iterations), HMM (18 iterations, taking syntax into account in the forward model). In each iteration, the forward and backward models were trained jointly. The training consumed 16 GB of memory for 34½ days, running in 7 threads on a 64bit Intel® Xeon™ 3.16GHz machine with 8 cores.

We also trained the Berkeley aligner in the same setup only on the MSR VDC portion of the training data, for comparison. This took an order-of-magnitude shorter time – mere 6 hours

–, which corroborates our earlier theoretical reasoning in Section 2.2.5 (claiming that short length of parallel sentences is an important factor for speed of word alignment), and provides strong evidence for usefulness of the alignment refining step in preprocessing of MTA and MTC.

We postprocessed the output alignments by re-substituting the pronominal coreferring expressions back in place of their coreferents, which had been substituted there prior to the word alignment (see Sections 3.1.3 and 3.2.4). This required us to also restore the alignment for the original wording (with the pronoun) from the alignment for the expanded version (with the coreferent). We solved this by simply gathering all correspondence links found for the expanded coreferent, and re-assigning them to the one pronoun.[1]

Although we have trained a word aligner, we have not yet evaluated its performance. We are planning to build a gold-standard set of alignments for the "test – known" portion of MSR VDC and evaluate both the training on MSR VDC only, and with MTA+MTC added, against this gold standard. We expect to see a significant improvement for alignments when trained with MTA+MTC, as opposed to not using MTA+MTC for training.

5.2 Extracting Verb Aspectual Classes

The set of features we designed presents one more challenge for automating feature extraction, apart from semantic role labeling, which was discussed in the previous section. This second challenge is determining the aspectual class of verbs, and we will present work we have done to tackle the challenge in the present section.

We base this piece of research on results reported in Siegel, 1998. This work uses a set of 14 linguistic features to train classifiers of the aspectual classes. It evaluates several machine learning approaches, with decision trees performing the best in most of the classification subproblems. Despite that the author provides the training data in form of the extracted features, we decided to create a new gold standard training set for our purposes. The reason is, it seems the provided data classify verb types rather than tokens, and we think that this introduces an inherent error.

After peeking into a few books from Project Gutenberg,[2] we selected Alice's Adventures in Wonderland (Carroll, 1865) as the text to create the gold standard from. It seems as an ideal text for learning aspectual classes, since it has very diverse verbs in this regard right from the beginning, with the verbs occurring in interesting syntactic structures. The reader will best appreciate it from a citation:

> Alice *was beginning to get* very *tired* of *sitting* by her sister on the bank, and of *having* nothing *to do*: once or twice she *had peeped* into the book her sister *was reading*, but it *had* no pictures or conversations in it, 'and what *is* the use of a book,' *thought* Alice 'without pictures or conversation?'

The 14 features for classifying verb aspectual classes, according to Siegel, 1998, are enumerated in Table 5.1. Because we were not sure whether to interpret the "participle" feature as referring to the syntactic category of the verb (as in *She came, crying.*), or just to its morphology (e.g., *crying*, also used in a context like *She was crying.*), we used two features for "participle" with

[1] Put in terms of graph theory, what we did is collapsing the nodes of every expanded coreferent into one.

[2] http://www.gutenberg.org

feature	**example clause**
frequency	(not applicable)
not or *never*	She can *not* explain why.
temporal adverb	I saw to it *then*.
no subject	He was admitted to the hospital.
past/pres participle	...blood pressure going up.
duration *in*-PP	She built it *in an hour*.
perfect	They have landed.
present tense	I am happy.
progressive	I am behaving myself.
manner adverb	She studied *diligently*.
evaluation adverb	They performed *horribly*.
past tense	I was *happy*.
duration *for*-PP	I sang *for ten minutes*.
continuous adverb	She will live *indefinitely*.

Table 5.1: Linguistic indicators of verb aspectual class, according to Siegel, 1998.

these two interpretations. We extracted one more linguistic feature, the verb depth within the syntactic tree, relative to other full verbs.[3] We also extracted a few features of a more technical nature, such as the number of roots of the syntactic tree of the sentence, mainly to be able to discover defective parses.

As is clear from the definition of the features, extracting them presupposes knowing a parse of the sentence. We obtained these using the RASP parser (Briscoe, Carroll & Watson, 2006), asking it to output top 4 parses for each sentence. The features were then extracted from the parses using an AWK script, after which we manually annotated the verbs with their aspectual class.

We managed to annotate first 314 verbs, where we had to leave this task for more critical tasks in the project. We found the annotation task notably hindered by the necessary searching for the context of each verb being annotated. We therefore started developing an annotation interface and believe this would be a necessity for creating a large enough dataset.

5.3 Conclusions

The research conducted within this Masters project presents a way for extraction of statistical dependencies between arbitrary features. We specifically show how the procedure can be successfully applied to linguistic features of one-sentence descriptions a) that capture the distribution of information (viz *information content* features for various constituents), and b) a few other linguistic properties that need to be instantiated in any sentence (verb tense, aspect, aspectual class). We planned to also include features reflecting the sentential focus and prepared the ground for that, but have not completed this task.

[3] This is a similar concept to the verb positions 1, 1.1, 2.1 etc., used throughout Chapter 4. Here, though, we only counted the level of nesting, without numbering the verbs w.r.t. the surface order.

We have cleaned and processed the English portion of the Microsoft Research Video Description Corpus, and Multiple Translation Arabic and Multiple Transation Chinese corpora. These resources were used as English bitext data, carefully optimised for the task of training word aligners, and possibly also for extracting a synchronous grammar for learning to paraphrase.

As part of resource processing, we devised and implemented two sentence alignment algorithms tailored for aligning of very small bitexts. We have shown their applicability to the problem of refining sentence alignment of the MTA and MTC corpora, and report on their accuracies, which reach 97% on our hand-annotated gold standard set.

As the central piece of our present project, we implemented a system for discovering apparent dependencies (*material implications*) in the setting of the statistical learning paradigm. The whole dependency extraction procedure consists of five steps:

1. feature extraction
2. smoothing and other transformations of the matrix of observations
3. generating artificial observations for training and evaluation purposes
4. training and selecting linear models
5. evaluation of the extracted dependencies using a gold standard reference set.

In the present work, we substitute the automated *feature extraction* by manual feature extraction on a small sample of MSR VDC. This is the step that is the most urgent one to further work on. We *smoothed matrices* of observations using NMF (non-negative matrix factorisation), and transformed them so that each pair of features gets described in terms of a boolean table. These transformations were needed to reduce the observation space dimension, as well as to facilitate analysing the features in terms of implications between them. For *generation of artificial feature observations*, we devised a parameterisable PDF (probability distribution function) to better reflect different nature of different features. The PDF is designed such as to be in accordance to the observed feature values in real data. Different nature of features is also accounted for by a parameterised noise model that effectively prevents extracting dependencies that have dubious support in the data. We *trained linear models* using the Lasso method, selecting from them one best model for each pair of features based on additional generated observations. For each model, we also compute the confidence in its predictions, so that the dependencies it discovers can be assigned their appropriate weight. Finally, we *evaluate extracted dependencies* under various settings of the experiment against a reference set of dependencies. We analyse the evaluation results and report on the best performing settings for the experiment run on the small data sample.

We suggest a number of ways how this work can be extended: we worked on automating the feature extraction and outline what remains to be done; possible ways of improving the current handling of the features are presented too; and finally, we describe how the extracted pairwise dependencies can be postprocessed by combining them into a single dependency graph.

Once the system presented in this thesis is run on larger data (preferably on the whole cleaned English portion of MSR VDC), it will extract information patterns present in the human-generated video descriptions. These will in turn be advantageous to use in generating video descriptions by machine, so that content is transformed into text that sounds naturally. This state is still a long way to go from where the present work finished, however, we provided a successful proof of concept and built the foundations for implementing the system as fully automated.

Bibliography

Aggarwal, J. & Ryoo, M. (2011, April). Human activity analysis: a review. *ACM Computing Surveys*, *43*(3), 1–43. doi:10.1145/1922649.1922653

Bird, S., Klein, E. & Loper, E. (2009). *Natural language processing with Python*. Beijing; Cambridge [Mass.]: O'Reilly.

Briscoe, T., Carroll, J. & Watson, R. (2006). The second release of the RASP system. (pp. 77–80). Association for Computational Linguistics. doi:10.3115/1225403.1225423

Brown, P. F., Cocke, J., Pietra, S. D., Pietra, V. D., Jelinek, F., Mercer, R. & Roossin, P. (1988). A statistical approach to language translation. (Vol. 1, pp. 71–76). Association for Computational Linguistics. doi:10.3115/991635.991651

Brown, P. F., Lai, J. C. & Mercer, R. L. (1991). Aligning sentences in parallel corpora. (pp. 169–176). Association for Computational Linguistics. doi:10.3115/981344.981366

Brown, P. F., Pietra, V. J. D., Pietra, S. A. D. & Mercer, R. L. (1993, June). The mathematics of statistical machine translation: parameter estimation. *Comput. Linguist. 19*(2), 263–311.

Brunet, J., Tamayo, P., Golub, T. R. & Mesirov, J. P. (2004, March). Metagenes and molecular pattern discovery using matrix factorization. *Proceedings of the National Academy of Sciences of the United States of America*, *101*(12), 4164–4169. PMID: 15016911. doi:10.1073/pnas.0308531101

Carroll, L. (1865). *Alice's adventures in Wonderland*. Project Gutenberg.

Chambers, N. & Jurafsky, D. (2009, August). Unsupervised learning of narrative schemas and their participants. In *Proceedings of the joint conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP* (602–610). Suntec, Singapore: Association for Computational Linguistics.

Chen, D. L. & Dolan, W. B. (2011, June). Collecting highly parallel data for paraphrase evaluation. In *Proceedings of the 49th annual meeting of the Association for Computational Linguistics (ACL-2011)*. Portland, OR.

Dempster, A. P., Laird, N. M. & Rubin, D. B. (1977). Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society. Series B (Methodological)*, *39*(1), 1–38. doi:10.2307/2984875

Farhadi, A., Hejrati, M., Sadeghi, M. A., Young, P., Rashtchian, C., Hockenmaier, J. & Forsyth, D. (2010). Every picture tells a story: generating sentences from images. In K. Daniilidis, P. Maragos & N. Paragios (Eds.), *Computer vision – ECCV 2010* (Vol. 6314, pp. 15–29). Berlin, Heidelberg: Springer Berlin Heidelberg.

Fellbaum, C. (1998). *WordNet: an electronic lexical database*. Cambridge Mass: MIT Press.

Feng, Y. & Lapata, M. (2010). Topic models for image annotation and text illustration. In *Human language technologies: the 2010 Annual Conference of the North American Chapter of the Asso-*

ciation for Computational Linguistics (831–839). HLT '10. Stroudsburg, PA, USA: Association for Computational Linguistics.

Finkel, J. R., Grenager, T. & Manning, C. (2005). Incorporating non-local information into information extraction systems by Gibbssampling. (pp. 363–370). Association for Computational Linguistics. doi:10.3115/1219840.1219885

Gale, W. A. & Church, K. W. (1991). A program for aligning sentences in bilingual corpora. (pp. 177–184). Association for Computational Linguistics. doi:10.3115/981344.981367

Gaujoux, R. & Seoighe, C. (2010). A flexible R package for nonnegative matrix factorization. *BMC Bioinformatics*, *11*(1), 367. R package version 0.5.02. doi:10.1186/1471-2105-11-367

Gupta, A., Srinivasan, P., Shi, J. & Davis, L. (2009, June). Understanding videos, constructing plots learning a visually grounded storyline model from annotated videos. In *2009 IEEE Conference on Computer Vision and Pattern Recognition* (pp. 2012–2019). Miami, FL. doi:10.1109/CVPR.2009.5206492

Hastie, T., Tibshirani, R. & Friedman, J. (2009). *The elements of statistical learning: data mining, inference, and predication* (2nd ed.). Berlin, New York: Springer.

Huang, S., Graff, D. & Doddington, G. (2002, February). *Multiple-translation Chinese corpus, part 1.* LDC2002T01. Philadelphia: Linguistic Data Consortium.

Huang, S., Graff, D., Walker, K., Miller, D., Ma, X., Cieri, C. & Doddington, G. (2003, October). *Multiple-translation Chinese corpus, part 2.* LDC2003T17. Philadelphia: Linguistic Data Consortium.

Klein, D. & Manning, C. D. (2002). Fast exact inference with a factored model for natural language parsing. In *NIPS'02* (pp. 3–10).

Koehn, P. (2007). *Statistical machine translation.* Cambridge: Univ. Pr.

Lee, D. D. & Seung, H. S. (1999). Learning the parts of objects by non-negative matrix factorization. *Nature*, *401*(6755), 788–91. doi:10.1038/44565

Lee, D. D. & Seung, H. S. (2000). Algorithms for non-negative matrix factorization. In *In NIPS* (556–562). MIT Press.

Lee, H., Peirsman, Y., Chang, A., Chambers, N., Surdeanu, M. & Jurafsky, D. (2011). Stanford's multi-pass sieve coreference resolution system at the CoNLL-2011 shared task. In *Proceedings of the fifteenth Conference on Computational Natural Language Learning: shared task* (28–34). CONLL Shared Task '11. Portland, Oregon: Association for Computational Linguistics.

Liang, P., Taskar, B. & Klein, D. (2006). Alignment by agreement. (pp. 104–111). Association for Computational Linguistics. doi:10.3115/1220835.1220849

Lowe, D. (1999). Object recognition from local scale-invariant features. In *Proceedings of the seventh IEEE International Conference on Computer Vision* (pp. 1150–1157 vol.2). Kerkyra, Greece. doi:10.1109/ICCV.1999.790410

Ma, X. (2004, July). *Multiple-translation Chinese corpus, part 3.* LDC2004T07. Philadelphia: Linguistic Data Consortium.

Ma, X. (2005, February). *Multiple-translation Arabic corpus, part 2.* LDC2005T05. Philadelphia: Linguistic Data Consortium.

Ma, X. (2006, January). *Multiple-translation Chinese corpus, part 4.* LDC2006T04. Philadelphia: Linguistic Data Consortium.

Melamed, I. D. (1999, March). Bitext maps and alignment via pattern recognition. *Comput. Linguist.* *25*(1), 107–130.

Pascual-Montano, A., Carazo, J., Kochi, K., Lehmann, D. & Pascual-Marqui, R. (2006, March). Nonsmooth nonnegative matrix factorization (nsNMF). *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *28*(3), 403–415. doi:10.1109/TPAMI.2006.60

Pedersen, T., Patwardhan, S. & Michelizzi, J. (2004). WordNet::Similarity: measuring the relatedness of concepts. Boston, Massachusetts: Association for Computational Linguistics.

Porter, M. F. (1997). An algorithm for suffix stripping. In K. Sparck Jones & P. Willett (Eds.), *Readings in information retrieval* (313–316). San Francisco, CA, USA: Morgan Kaufmann Publishers Inc.

Raghunathan, K., Lee, H., Rangarajan, S., Chambers, N., Surdeanu, M., Jurafsky, D. & Manning, C. (2010). A multi-pass sieve for coreference resolution. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing* (492–501). EMNLP '10. Cambridge, Massachusetts: Association for Computational Linguistics.

Regneri, M., Koller, A. & Pinkal, M. (2010). Learning script knowledge with web experiments. In *ANNUAL MEETING - ASSOCIATION FOR COMPUTATIONAL LINGUISTICS* (pp. 979–988). Association for Commputational Linguistics.

Regneri, M., Koller, A., Ruppenhofer, J. & Pinkal, M. (2011). Learning script participants from unlabeled data. In *Proceedings of RANLP 2011*. Hissar, Bulgaria.

Resnik, P. (1995). Using information content to evaluate semantic similarity in a taxonomy. In *Proceedings of the 14th international joint conference on artificial intelligence* (Vol. 1, 448–453). Montreal, Quebec, Canada: Morgan Kaufmann Publishers Inc.

Rohrbach, M., Amin, S., Andriluka, M. & Schiele, B. (2012, June). A database for fine grained activity detection of cooking activities. In *2012 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. The dataset and relevant code is available at http://www.d2.mpi-inf.mpg.de/mpii-cooking. IEEE. Providence, United States: IEEE.

Rohrbach, M., Regneri, M., Andriluka, M., Amin, S., Pinkal, M. & Schiele, B. (2012, October). Script data for attribute-based recognition of composite activities. In *Computer Vision - ECCV 2012 : 12th European Conference on Computer Vision* (Vol. 2012). Lecture Notes in Computer Science. Springer. Firenze, Italy: Springer.

Schank, R. C. & Abelson, R. P. (1977). *Scripts, plans, goals, and understanding : an inquiry into human knowledge structures*. The Artificial intelligence series. Hillsdale, N.J. : L. Erlbaum Associates ; New York : distributed by the Halsted Press Division of John Wiley and Sons, 1977.

Shoham, S., Yahav, E., Fink, S. J. & Pistoia, M. (2008, September). Static specification mining using Automata-Based abstractions. *IEEE Transactions on Software Engineering*, *34*(5), 651–666. doi:Article

Siegel, E. V. (1998). *Linguistic indicators for language understanding: using machine learning methods to combine corpus-based indicators for aspectual classification of clauses* (Doctoral dissertation, Columbia University, New York, NY, USA). AAI9834376.

Tibshirani, R. (1996). Regression shrinkage and selection via the Lasso. *Journal of the Royal Statistical Society. Series B (Methodological)*, *58*(1), 267–288.

Toutanova, K., Klein, D., Manning, C. D. & Singer, Y. (2003). Feature-rich part-of-speech tagging with a cyclic dependency network. (Vol. 1, pp. 173–180). Association for Computational Linguistics. doi:10.3115/1073445.1073478

Toutanova, K. & Manning, C. D. (2000). Enriching the knowledge sources used in a maximum entropy part-of-speech tagger. (Vol. 13, pp. 63–70). Association for Computational Linguistics. doi:10.3115/1117794.1117802

Vogel, S., Ney, H. & Tillmann, C. (1996). HMM-based word alignment in statistical translation. (Vol. 2, p. 836). Association for Computational Linguistics. doi:10.3115/993268.993313

Walker, K., Bamba, M., Miller, D., Ma, X., Cieri, C. & Doddington, G. (2003, October). *Multiple-translation Arabic corpus, part 1*. LDC2003T18. Philadelphia: Linguistic Data Consortium.

Zhang, J., Wei, L., Feng, X., Ma, Z. & Wang, Y. (2008). Pattern expression nonnegative matrix factorization: algorithm and applications to blind source separation. *Computational Intelligence and Neuroscience*, *2008*, 1–10. doi:10.1155/2008/168769

Appendix A

Sample from VDC Used in Chapter 4

1 A bird in a sink keeps getting under the running water from a faucet.
2 A bird is bathing in a sink.
3 A bird is splashing around under a running faucet.
4 A bird is bathing in a sink.
5 A bird is standing in a sink drinking water that is pouring out of the facet.
6 A faucet is running while a bird stands in the sink below.
7 A bird is playing in a sink with running water.
8 A bird is playing in tap water.
9 A bird is bathing in the sink.
10 A bird is taking a bath.
11 A bird is taking a shower in a sink.
12 A bird is showering in the sink.
13 The bird is taking a bath under the faucet.
14 A parakeet is taking a shower in a sink.
15 A bird gets washed.

---

16 The zebras are playing.
17 Two zebras are running and playing with each other in the grass.
18 Two zebras are fighting.
19 Two zebras playing with each other.
20 A pair of zebras interact on the savannah.
21 Two zebras are playing.
22 Two zebras are playing.
23 Two zebras are playing in a field.
24 Zebras are playing.
25 Two zebras fight with each other.
26 Zebras are socializing.
27 Two zebras are playing.
28 Two zebras play in an open field.

29 Two zebras are playing around with each other on grassland.
30 Two zebras playfully nuzzled each other.
31 The two zebras are playing.

---

32 Two bear cubs are digging into dirt and plant matter at the base of a tree.
33 Two baby bears are digging.
34 Baby bears are digging.
35 A man is watching two bear cubs digging.
36 Black bear cubs are digging holes.
37 Two baby bears are walking around.
38 Bears.
39 A man is watching two baby bears.
40 Two bears dig around in a dirt pile.
41 Two bear cubs are digging.
42 Bear cubs are digging in the dirt.
43 Baby bears are scratching dirt.
44 Two bear cubs are digging in the dirt.
45 Two baby bears are standing beside each other in mud as one of them walks ahead.
46 The bears dug in the dirt.
47 The bear cubs are digging the ground.
48 Two bear cubs are digging.

---

49 A soccer player kicking a soccer ball to himself.
50 A soccer player is kicking a ball back and forth between his legs.
51 A man is kicking a ball around.
52 A man is kicking a soccer ball back and forth.
53 A man is passing a football to and from one leg to the other standing on a football field.
54 A man is practicing football.
55 A man kicks a soccer ball from side to side.
56 A man passing a soccer ball between his feet.
57 A person dribbles a soccer ball.
58 A person in blue is dribbling a ball.
59 A person is juggling a soccer ball with his feet.
60 A person is kicking a soccer ball back and forth between his feet.
61 A person is passing a soccer ball from foot to foot.
62 A person is playing football.
63 Someone is dribbling a soccer ball.
64 Someone is kicking a soccer ball.

---

65 A man cuts a long round green vegetable in half, then lengthwise, and scoops out the inside with a spoon.

66 A person is slicing a cucumber.
67 A person is slicing a cucumber.
68 A man cuts a cucumber in half and then scoops out the middle.
69 A person is cutting a central of a cucumber.
70 A man is slicing some vegetables.
71 A man is seeding cucumbers.
72 A man hollows out the inside of four pickles.
73 A man cuts a cucumber, and removes the seeds.
74 A man is slicing a cucumber lengthwise and scooping out the seeds.
75 A man is cutting up a cucumber.
76 The man cut the pickle in half and sliced it lengthwise before scooping the insides out.
77 The man is slicing a vegetable.

---

78 A man is performing on the electric guitar.
79 A man is playing a guitar.
80 A man is playing a guitar.
81 A man is playing an electric guitar.
82 A man is playing an electric guitar.
83 A man is playing electric guitar.
84 A man is playing guitar at a concert.
85 A man is playing guitar on a stage.
86 A man is playing guitar.
87 A man is playing the guitar on stage in front of an audience.
88 A man playing a guitar in front of a large crowd.
89 A man plays guitar.
90 An artist is playing guitar.
91 Jeff Beck is playing the guitar onstage.
92 Jeff Beck is playing the guitar.
93 Jeff Beck plays his guitar onstage.

---

94 Ingredients are being mixed in a mixing bowl.
95 A lady mixed up a batter.
96 A man is mixing batter.
97 A man mixes a baking mixture.
98 A person is mixing a bowl of ingredients.
99 A person is mixing ingredients in a bowl.
100 A person is mixing ingredients in a bowl.
101 A person mixes ingredients in a bowl.
102 A woman is mixing ingredients in a bowl.
103 A woman is mixing ingredients in a bowl.

104 A woman is mixing ingredients in a bowl.

105 A woman is stirring a cookie dough mixture.

106 A woman is whisking a mixture of sugar, butter and molasses in a bowl using a wooden spatula.

107 A woman preparing oatmeal cookies.

108 Cookie batter is being mixed.

---

109 A man in a black cape causes a policeman to disappear and continues to walk after being shot several times by another man.

110 A man shoots a man.

111 A man is being shot.

112 A man wearing a black cape is walking toward a group of people and a man in the group is shooting at him with a pistol.

113 A man is shooting another man.

114 A man sprays a dust at another man.

115 A man is firing at another man.

116 A man with a pistol shoots another man.

117 Two men are shooting a cloaked man.

118 A man is shooting another man with a pistol.

119 People are shooting a man.

120 Two men are shooting at a vampire.

121 The man tried to shoot the vampire.