Universität des Saarlandes

Master's Thesis

A Hierarchical Bayesian Model for Unsupervised Learning of Script Knowledge

submitted in partial fulfillment of the requirements for the degree of Master of Science in Language Science and Technology

Author:
Lea Frermann

Supervisors:
Dr. Ivan Titov
Prof. Dr. Manfred Pinkal

December 24, 2013

Abstract

We present a hierarchical Bayesian model for unsupervised induction of knowledge from scripts. Scripts are abstract representations of common everyday scenarios which consist of a temporally ordered, stereotypical sequence of events (Schank and Abelson, 1975; Regneri et al., 2010). In recent years, a body of work on inducing script knowledge automatically from data has emerged. We extend this work by presenting a new, unsupervised approach which *jointly* learns three the objectives of (1) equivalence classes of events, (2) constraints on the temporal order of events, and (3) equivalence classes of participants. We embed the three objectives in one unified framework, arguing that they provide strong cues for each other.

We formulate our model in the framework of Bayesian modeling. We provide the complete formulation of a hierarchical Bayesian model for our problem, and derive an inference algorithm. We incorporate a statistical model of permutations, the Generalized Mallows Model (GMM; (Fligner and Verducci, 1986)), for modeling ordering constraints. We further include prior knowledge of semantic similarity obtained from WordNet to guide the inference process and leverage the problem of relatively small training data sets we have available.

We present an evaluation for all three tasks, comparing our results to a system which learns the same three objectives using a pipeline-based architecture, and evaluating the benefit of different components in our model. We show that the GMM is a robust model of event orderings in scripts. While we do not achieve state-of-the-art performance on participant class learning, our model compares favourably on event clustering and temporal ordering constraint induction.

Declaration

Eidesstattliche Erklärung

Hiermit erkläre ich, dass ich die vorliegende Arbeit selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel verwendet habe.

Declaration

I hereby confirm that the thesis presented here is my own work, with all assistance acknowledged.

Saarbrücken, 13 March 2013
Signature

Contents

Chapter 1

Introduction

The goal of computational linguistics and natural language processing (NLP), as a research area is to enable communication of humans and computers using natural language. This problem has at least two dimensions: first, natural language needs to be formalized in a way that is understandable for machines. Secondly, NLP applications which require deep semantic knowledge, such as natural language understanding components, or question answering systems, need to be equipped with world knowledge, such that they are able to process and convey information (Ovchinnikova, 2012; Ponzetto and Strube, 2009).

In this work, we aim to contribute towards tackling the second problem. One bottleneck of any large-scale NLP system is common sense knowledge. Humans acquire a vast variety of implicit knowledge throughout their life time. However, it remains prohibitive to manually collect and explicitly represent broad-scale world knowledge for any system exceeding a restricted, domain-specific application (see, e.g. Lieberman et al. (2004)).

We present a system that learns a specific kind of world knowledge, namely *script knowledge* from data. Scripts capture the sequence of events involved in very common, every-day *scenarios*, such as `Eating in a restaurant`, `Doing laundry`, or `Taking a bus`. Scripts have first been introduced in the 1970s in the research area of Artificial Intelligence (AI) (Schank and Abelson, 1975). It has been shown that script knowledge improves performance of NLP applications, such as text understanding systems (Cullingford, 1978; Miikkulainen, 1995).

Through experience, humans acquire knowledge about the stereotypical event sequence involved in a scenario. There is general agreement, for example, that in the `Eating in a restaurant` scenario one necessarily has to "order the food", before he can "eat the food", and that "paying for the food" is another obligatory event, that tends to occur towards the end of the scenario. *Scripts* capture this information in an abstract way.

Our system will learn script knowledge, from explicit instantiations of scenario -specific scripts, *event-sequence descriptions* (ESDs), as introduced in Regneri et al. (2010). Table 1.1 displays three ESDs for the scenario `Eating in a restaurant`. Equivalent *event descriptions* are row-aligned, such that each row in the table corresponds to one event type. Event types are displayed in their chronological order. Equivalent *participant descriptions* are highlighted in the same color, each color corresponds to one type of participant. We define par-

ESD 1	**ESD 2**	**ESD 3**
	Enter	
Sit at table	Be seated	
	Wait	
Read menu		Check the menu
Listen to specials		
Give order to waiter	Order	Order the meal
	Wait	Wait for meal
		Talk to the friends
Consume meal	Enjoy food	Have meal
Pay check	Pay	Pay the bill
	Leave a tip	

Table 1.1: Three event-sequence descriptions (ESDs), each describing a typical chain of events that constitute the scenario of `Eating in a restaurant`. Descriptions of equivalent event types are row-aligned. Terms referring to equivalent types of participants are highlighted in the same color.

ticipants in a scenario as any object or person participating in any event in an ESD. While each individual ESD is a subjective, possibly incomplete or even erroneous description of the scenario, a corpus of such descriptions, each obtained from a different annotator, comprises a generic idea of the event sequence that makes up the scenario.

We follow Regneri et al. (2010) and Regneri et al. (2011), in defining our learning objective. Like the previous work, the learning process of our model is completely unsupervised. With the proposed model we aim to induce three kinds of characteristics of scripts from scenario-specific corpora of ESDs:

1. The *event types* involved in the scenario
2. Constraints on the *temporal ordering* of event types
3. The *participant types* in occurring the scenario

In previous work, the three objectives defined above were induced in separate steps, using a pipeline-based system architecture. Event types and ordering constraints were induced using graph-based methods (Regneri et al., 2010), and on this basis participant types were learnt using similarity-based methods (Regneri et al., 2011).

1.1 Contributions

We present a different view on the problem of script learning, by arguing that event types, event orderings and participant types involved in ESDs strongly correlate, and should thus provide useful cues for each other. To the best of our knowledge, we are the first to propose a model for learning event types, participant types, and ordering constraints *jointly* from ESDs.

We formalize our joint learning objective in the framework of Bayesian modeling. We provide a fully formalized, hierarchical Bayesian model for learning script knowledge, in the form of a generative story which includes the three learning objectives. An inference algorithm for the model is derived.

Two components of our model are worth pointing out. First, we incorporate a statistical model over orderings, the Generalized Mallows Model (GMM; Fligner and Verducci (1986)) for modeling ordering constraints. The GMM allows us to model event type-specific temporal flexibility, and we will be able to model event optionality, both characteristics of scripts that were not captured in previous approaches to the same task. We will show that the GMM is a robust model of event ordering constraints.

Secondly, we are faced with the problem of learning from small data sets, since the available scenario-specific ESD corpora are very limited in size. We will obtain prior knowledge about word similarities in a fully unsupervised way in order to leverage this problem and guide the learning process. We will use the covariance matrix of a Multivariate Gaussian distribution to encode this knowledge and guide inference of type-specific language models, by triggering correlation in realization probabilities for semantically similar words.

In short, our contributions can be summarized as follows:

1. We introduce a joint model for inducing event types, participant types, and ordering constraints in a fully unsupervised way.

2. We formalize the tasks using a hierarchical Bayesian model, and derive an inference algorithm.

3. We show that the GMM is an accurate model for constraints on event orderings.

4. We tackle the problem of learning from small data sets by incorporating prior knowledge about word correlation in a fully unsupervised way.

1.2 Thesis Outline

In Chapter 2, we present previous work in areas related to our project, namely script modeling, models of ordering and topic models. We continue in Chapter 3 by providing some technical background on the components and techniques we use in our model and in the inference procedure. We introduce our model in Chapter 4 by giving an informal overview, and subsequently describing the formal generative story. We derive the inference process for our model in Chapter 5. Chapter 6 explains how we extend our model with the prior knowledge component. We extend the generative process, and describe the modified inference algorithm. In chapters 7 and 8 we present our evaluation setup and a discussion of the results of our experiments, respectively. We finally conclude in Chapter 9, and present possibilities for future work.

Chapter 2

Related Work

In order to place the work presented here into the landscape of existing related research, we will discuss previous work which is relevant to our learning objective, and the methodology presented in this thesis; we will point out the extensions and contributions we aim to make.

Topic-wise, our work falls into the area of script acquisition and modeling, an area of research which has been pursued in fields such as Artificial Intelligence, and (computational) linguistics for decades. We give an overview over the concepts and approaches in this field in Section 2.1.

One central characteristic of our model is its ability to infer ordering constraints on event types. In Section 2.2, we provide a brief overview over models of ordering which have been proposed previously.

Methodologically, our work builds upon the area of topic modeling, which has become a popular approach towards a variety of problems in computational linguistics and NLP. We will provide a brief overview over topic models, and discuss some previously proposed extensions which are most related to our work in Section 2.3.

2.1 Scripts

A prominent opinion in fields broadly related to Artificial Intelligence is that the ultimate bottleneck for creating artificial intelligence is encoding world knowledge (Schank and Abelson, 1975). While humans acquire vast amounts of knowledge about rules of their physical and cultural environments over their whole lifetime, it is very hard to equip an artificial intelligence system with background knowledge that allows for flexible inference, reasoning and reaction towards the situations it encounters.

One possible encoding of a particular kind of common sense knowledge are *scripts*, which have been under active research since their introduction in the 1970s (Schank and Abelson, 1975; Barr and Feigenbaum, 1986). Scripts are abstract definitions of stereotypical sequences of actions involved in common every day situations, such as `Eating in a restaurant` or `Cooking pasta`. Scripts provide information about the temporal sequence of events involved in the situation, as well as about participants involved in the events[1]. An inherent script-knowledge base should enable an artificial agent to trigger appropriate scripts based on the input it receives, to draw inferences about such input and to react in an appropriate way. Concretely, script-like knowledge will be useful for standard NLP tasks such as automatic summarization or question answering (Miikkulainen, 1995).

A slightly different approach to encoding procedural knowledge has been made in the FrameNet project (Baker et al., 1998). Here the emphasis lies on atomic events, or *frames*, generally consisting of a verb, and constraints on the types of arguments – or semantic roles – the verb requires. For example the "selling" event requires a "seller", a "buyer" and "goods". Additionally, some relations between frames are captured, which allows for limited construction simple event chains from frames.

Both of the above approaches involve manual construction and are built in a top-down way: knowledge is manually defined, and systems using this knowledge can draw inferences based on exactly this available knowledge base. While hand-built knowledge is arguably very accurate and contains little or no noise, it is also very limited and inflexible. On the one hand it is prohibitive to manually encode large-scale knowledge databases, and on the other, the encoded rules and information tend to be intolerant to noise as it typically occurs in input data.

For flexible, broad-coverage applications it would thus be desirable to be able to learn scripts automatically from data. Chambers and Jurafsky conducted a series of research on the extraction of *narrative chains* from newswire text. Narrative chains are chains of events occurring in natural text (such as `Convicting a criminal`). Chambers and Jurafsky (2008) present-a three-stage approach for inducing event sets and a temporal ordering of events, given a parsed corpus with resolved co-referents: first, sub events of particular narrative chains are induced based on common protagonists between events, using pointwise mutual information (PMI)[2] based on how often a protagonist realization is shared by any pair of verbs in the text. Secondly, a classifier is trained on the TimeBank corpus (Pustejovsky et al., 2003) as well as manually defined linguistic features, and based on that classifier the events are sorted pairwise into a temporal order. Finally, events are clustered together into discrete sets, based on the PMI score described above, and the clusters are ordered using the temporal classifier. The resulting structure, containing temporally ordered equivalence classes of events, can be interpreted as a representation of a classical script. However, considering that the events are explicitly described in newswire text, narrative chains describe generally less fundamental knowledge than scripts.

Subsequently, Chambers and Jurafsky present an extension to their work, in which they not only consider the protagonist of events when building the narrative chain, but all involved participants (Chambers and Jurafsky, 2009). Their model learns sets of events which construct a narrative chain, as well as classes of entities that can fill the various argument positions of the verbs involved in the chain. Events and participants are thus learnt jointly, using

[1]According to the original definition of scripts, they additionally encode information about causal relations between events. This aspect, however, is beyond the scope of this work.

[2]Pointwise mutual information of two events a and b is a measure for the extent to which knowing event a reduces the uncertainty about event b, and vice versa. It expresses the shared information of a and b (cf. Manning and Schütze (1999)).

an extended version of the PMI score mentioned above, by conditioning on the presence of a particular argument. They show that considering all participants improves the quality of the induced narrative chains. Temporal ordering of events is not considered in this work.

In contrast to our work, Chambers and Jurafsky extract narrative chains from natural, newswire text. While the resources in this domain are vast, they may not be appropriate for script extraction since the fundamental feature of scripts is to explicitly define very basic event chains, which are usually left implicit in discourse. In natural texts parts of obvious causal chains are often omitted, since it is assumed that the readers share basic world knowledge. Chambers and Jurafsky (2009) provide a joint model for inducing event types and participant types, but do not incorporating event orderings into the setting. We argue that joint learning of all three factors will improve the quality of the learnt scripts. Finally, the temporal ordering inferred in Chambers and Jurafsky (2008) is based on local, pairwise decisions. We will induce one globally coherent order for each scenario.

O'Connor (2012) presents a latent variable model for unsupervised learning of frames from text. Their model builds on the Latent Dirichlet Allocation (LDA) topic model (Blei et al. (2003), cf. Section 2.3). Frames are incorporated as latent variables in the model, which link subject-verb-object triples. Furthermore, latent classes of words are induced which can realize any of the components in the triple. A document is assumed to be generated from a sparse multinomial distribution over frames. Each triple-component of a generated frame is then assumed to be drawn from a latent class of words. Realizing words are drawn from a class- and component-specific language model. Classes are shared across frames.

Titov and Klementiev (2011) propose a similar, but more complex, Bayesian model which, in addition to frame semantic information, induces syntactic relations, in the form of dependency tree fragments. Their model is non-parametric, meaning that they induce the number of distinct frames and roles, and is evaluated on the biomedical domain. Titov and Klementiev (2012) propose another unsupervised Bayesian model for semantic role labeling, which Modi et al. (2012) extend towards joint inference of semantic frames, and show that it successfully induces FrameNet-style annotations for a broader domain.

Like the model we propose, the models described above infer verb frames (types of events), and corresponding argument types (types of participants) in an unsupervised way from data, using an unsupervised Bayesian model. However, focus lies on independent frames, while we model events in the context of a whole scenario, and want to infer ordering constraints in addition to the targets mentioned above.

Our work is most closely related to the work conducted by Regneri et al (Regneri et al., 2010, 2011). In contrast to the previously described approaches, Regneri and colleagues collect a corpus of *explicit* descriptions of various scenarios, obtaining a variety of detailed scenario-specific *event-sequence descriptions* (ESDs). For each scenario they ask non-expert annotators to give an explicit, temporally ordered description of the sequence of events that typically happen in the scenario. Descriptions are given in "bullet-point style" language. The goal here is to learn paraphrase sets of descriptions of events, as well as temporal ordering constraints on these events. For each scenario, in a first step, event descriptions referring to the same type of event are aligned based on semantic similarity. Based on this alignment, a *temporal script graph* (TSG) is computed, using multiple sequence alignment. Edges encoding direct precedence of two events are added to the graph on the basis of semantic cues of the paraphrase class and structural cues of the TSG.

As an extension to this work, Regneri et al. (2011) present a system which induces participant classes on the basis of the TSG. The goal here is not to induce semantic roles, as in Chambers and Jurafsky's work, but to induce scenario-specific synonym sets describing the same participant, so-called *participant description sets* (PDSs). In a first step all noun phrases as identified by a dependency parser are marked as participants in the scenario. Subsequently PDSs are induced using semantic similarity information among the noun phrases, as well structural information obtained from the positions at which a noun phrase appears in the TSG. Methodologically, integer linear programming (ILP) is used to combine these cues and to decide for each pair of noun phrases in all descriptions of one scenario whether they belong to the same PDS or not.

In our work, we will use the dataset collected by Regneri and colleagues (cf. Section 7.1 for a more detailed description of the data), and we aim to learn the same objectives, namely identifying scenario-specific event equivalence classes, participant equivalence classes and constraints on event orderings. Unlike Regneri et al, who similar to previous approaches propose a pipeline-based architecture, we will learn all three objectives jointly. We will use one statistical model for inference, instead of relying on semantically or structurally based heuristics at various points. Furthermore, the graph induction algorithm used in Regneri et al. (2010), multiple sequence alignment, is not able to model some temporal characteristics of script events. In particular, event type-specific temporal flexibility and event type optionality cannot be encoded in the TSG. Taking the scenario of `cooking pasta` as an example, the event "grating cheese" can occur basically at any temporal position in the event sequence, while "boiling water" should clearly precede "adding pasta". As we describe in Section 4.2, our Bayesian model will encode these characteristics.

2.2 Modeling Ordering

One central aspect of our model is its ability to infer constraints on possible orderings of event types. As mentioned previously, event ordering has been modeled using a semantically trained classifier (Chambers and Jurafsky, 2008), or multiple sequence alignment (Regneri et al., 2010). Corpus-based methods for information ordering which extract features of adjacent sentences from corpora and use this for coherent summarization of multiple documents have been proposed (Barzilay et al., 2002; Lapata, 2003). Such methods rely on domain-specific corpora which need to be large enough to encode representative information about ordering possibilities, or hand-built for the particular objective.

It would be desirable, however, to model the probability of particular orderings in a fully unsupervised way, and with a statistical model that can be integrated in our general probabilistic framework. The Generalized Mallows model (GMM,Fligner and Verducci (1986)) is a statistical model over permutations, and lends itself particularly well to integration into our unsupervised, Bayesian setting. We explain the GMM in detail in Section 3.3.

The problem of combining rankings of multiple experts, such as the search results returned for a query by multiple search engines, has drawn much attention recently, and the GMM has been thoroughly tested on this problem (Lebanon and Lafferty, 2002; Klementiev et al., 2008; Meila et al., 2007).

From a slightly different perspective, the GMM can also be used to discover a "ground truth" from a set of observations, such as inferring the true sequence of events in an accident from a set of eyewitness reports. Steyvers et al. (2009) employed the Mallows model, a constrained version of the GMM, for this task, which is very similar to the problem we are tackling – inferring a canonical order of events involved in a scenario from a set of potentially inaccurate scenario descriptions obtained from non-expert annotators.

2.3 Topic Models

Methodologically, our approach falls into the area of topic modeling. In the past decade, topic models have gained increasing popularity as a method for modeling various aspects of natural language. Classic topic models such as *Latent Dirichlet Allocation* (LDA, Blei et al. (2003)) infer a latent structure underlying the text in an unsupervised way, by assigning each word a hidden label, or 'topic'. From the generative viewpoint, each document is assigned a distribution over topics, and for each topic, realizations (words) are drawn from a topic-specific word distribution. Crucially, each topic is drawn *independently* of all other topics from a topic distribution and, equivalently, words are drawn *independently* from topic-specific word distributions.

This fundamental idea has been extended in numerous ways in order relax the inherent independence assumption between topics and between words. Among many other extensions, hierarchical topic models (Blei et al., 2004) and correlated topic models (Blei and Lafferty, 2005; Hennig et al., 2012) have been proposed. Under the intuitive assumption that topics do not co-occur randomly, but subsets of topics are highly correlated[3], Blei and Lafferty (2005) induce topic correlations using the covariance matrix of the logistic-normal distribution (Aitchison, 1982). They show that their correlated topic model outperforms classic LDA.

We use basic concepts from classical LDA in the sense that we explain document structure as a mixture of event types and participant types, which can be seen as our set of 'topics'. We induce a specific language model for each event type and each participant type.

Due to limited amount of available training data, we will model word correlations as prior knowledge, in a similar way as Blei and Lafferty model topic correlations. While Blei and Lafferty (2005) *learn* the covariance matrix, we will construct it a priori and use it to encode correlations in the parameterization of our model. Raina et al. (2006) propose a similar idea in the framework of transfer learning. The matrix is induced based on learning problems similar to the actual task. Raina et al work in supervised learning framework, while we construct the covariance matrix in a completely unsupervised way.

We will incorporate the Generalized Mallows Model into our model of script knowledge, in order to model event ordering constraints. Chen et al. (2009)

[3]In a document about *economy* the topics *finances* and *trade* are very likely to occur, while the topic *interior design* is rather unlikely.

propose a topic model augmented with the GMM, and use it to infer global document structure. They tackle the problem of discourse segmentation in highly structured documents, such as Wikipedia articles. Topics in a document are drawn from a topic distribution and sorted with respect to an ordering which is drawn from the GMM. Topics are thus not only inferred based on the words occurring in them, but also based on the position at which they occur in the document. This extension can also be interpreted as relaxing the original topic-independence assumption of LDA by imposing coherent topical structures across documents. Chen et al successfully model document discourse structure using the GMM, which motivates us to develop a structurally similar model for inference on script data. We incorporate the GMM into our topic model of events and participants in order to encourage a coherent scenario-specific ordering across all scenario descriptions. In particular, the type-specific clusters of event descriptions induced by our model will not only be based on their particular word distribution, but also on the position at which the descriptions tend to occur within an ESD.

Chapter 3

Technical Background

We provide mathematical foundations on which the definition and learning procedure of our model, as described in the following sections, are based. We start by describing the general motivation behind using Bayesian models in Section 3.1, before we explain their mathematical characteristics relevant to this work in Section 3.2. We explain and formally define the kinds of distributions involved in our model in Sections 3.3 - 3.4, and the inference techniques which we will use for learning the model parameters in Section 3.5.

3.1 Generative Modeling

The general aim of statistical models is to explain observable real-world phenomena, such as language. In addition to such observable variables (y), auxiliary unobservable, or latent, variables (θ) are introduced. Such latent variables are believed to be the underlying mechanism which explains the behavior of the observable variables. Based on this, two ways of modeling have evolved (cf. e.g. Bishop (2006)):

1. Discriminative Modeling, which aims to infer a conditional probability distribution of the observable data y given the latent variables θ $P(y|\theta)$, and thus predicts y from θ.

2. Generative Modeling, which aims to infer a joint probability distribution $P(y, \theta)$, and thus explains both the observable data and the latent parameters.

In this work, we develop a *generative* model of script data. Generative models tend to be more robust with respect to incomplete or unlabeled data, because they allow for incorporation of *uncertainty*, as we will explain below. They thus are more suitable for our unsupervised inference problem from restricted datasets. Since the joint distribution $P(y, \theta)$ is learnt during inference, trained generative models are not only capable of labeling data, but they can also generate data themselves, given that the distributions of the hidden variables are known.

3.2 Bayesian Inference

In general, we want to learn a model with parameters θ which provides a good explanation for our observed data y. However, we need to select this model from all possible models $\theta_1, ...\theta_k$, and it is often the case that several of those explain our data well, potentially capturing different aspects. Deciding for one particular model might lead to over-confident predictions, because it ignores the uncertainty with which that particular model was selected. It would be desirable to be able to have a measure for this uncertainty.

Bayesian inference provides a way for dealing with uncertainty, by avoiding the choice of one particular model, but including all possible models into the final model to be learnt (Hoeting et al., 1999). Assume, there are K possible models which potentially explain the observed data. In Bayesian inference, the models are themselves considered to be random variables. Instead of deciding for one particular model, we sample model parameters from the probability space over all possible models $P(\boldsymbol{\theta})$.

In order to get a full probabilistic formulation of this setup, we need a prior measure for the probability of any particular model i, $P(\theta_i)$. This prior allows to inject some intuition into the inference process, about what kind of models we deem more likely. The probability of a model $P(\theta_i)$ is now a measure for uncertainty when picking this particular model.

Bayes' rule provides a way to relate the prior probability described above to the evidence for our parameter selection, as obtained from observed data, and thus allows us to update our uncertainty with this evidence. In the remainder of this section, we will explain how Bayes' rule is used to do inference in the Bayesian setting. Subsequently, we will explain how prior knowledge can be used to guide the inference process.

3.2.1 Bayes' Rule

The goal of an inference procedure is to learn a model with parameters θ which explains the observed data y as accurately as possible. Bayes rule formalizes a measure for the suitability of a model, by relating newly observed evidence from the data y to the prior probability of hypothesized parameters θ, before the data was observed:

$$P(\theta|y) = \frac{P(y|\theta)P(\theta)}{P(y)}. \tag{3.1}$$

We can directly relate Bayes' rule to the intuition, by naming the components as follows:

$$\text{posterior} = \frac{\text{likelihood} * \text{prior}}{\text{evidence}} \tag{3.2}$$

The *posterior* probability of the model is proportional to its *prior* probability and the *likelihood* of the observed data under the model. The denominator of Bayes' rule corresponds to the marginal likelihood of the data. It is constant w.r.t. the parameters θ and thus usually irrelevant when determining the best hypothesis.

In Bayesian estimation, however, we do not aim to infer the parameters that best explain our data, but rather view parameterizations as random variables which follow a distribution themselves. In our inference procedure, we take all parameterizations (all possible models) into account.

This leads to two modifications in the computation of Bayes' rule. First, the denominator of Equation 3.1 now is not the probability of the data given one particular parameterization anymore, but becomes the expected value given the distribution over all possible parameterizations $\boldsymbol{\theta}$. Given that the expected value of a variable is the weighted average over all possible values the variable can take, we now need to integrate over all possible parameterizations in the denominator of Bayes' rule.

Secondly, we need to base our probability for a particular parameterization on a probability distribution: a probability distribution over probability distributions. This distribution is itself parameterized with *hyperparameters* γ. Adding this additional layer makes our model a *hierarchical* Bayesian Model (see Section 3.2.2 for more information). Bayes' rule becomes:

$$P(\theta|y,\gamma) = \frac{P(y|\theta)P(\theta|\gamma)}{\int_\theta P(y|\theta,\gamma)P(\theta|\gamma)\mathrm{d}\theta} \tag{3.3}$$

The integral in the denominator often cannot be computed analytically, and it is in practice often approximated by variational or sampling methods (cf. Section 3.5).

3.2.2 Prior Knowledge

In Bayesian modeling, we can include a prior intuition about how much we believe in any particular model, or parameterization. This belief is represented as a distribution over all possible parameterizations θ, and it takes parameters itself. These parameters, γ, are called hyperparameters. Hyperparameters allow for injection of prior, subjective, knowledge: by assigning parameters θ with specific characteristics a high prior probability $P(\theta|\gamma)$, we can influence the shape of our posterior distributions given the parameters θ. Hyperparameters are thus defined manually, or empirically by optimization on a test data set[1]. Below, we introduce the notion of *conjugate* prior distributions. We describe the advantages of using conjugate prior distributions, and introduce the particular distributions used in our model.

Mathematically, it is particularly convenient to choose a prior distribution over parameterizations which is *conjugate* to the posterior distribution over parameterizations. This means that the posterior distribution, after updating the prior with the evidence, belongs to the same family of distributions as the prior distribution. Only the distribution's parameters changed.

One advantage of conjugate priors is the possibility to integrate out the actual parameters θ of the distribution we are interested in. By summing (or integrating) over all possible values the parameters θ can take, one can represent them implicitly in the model, which leads to a more efficient learning process.

Given a conjugate prior on a distribution, the hyperparameters can be interpreted as pseudo-counts: we assign n_i additional counts to the number of times event i has been observed in the data. In case there is no reliable prior intuition

[1]We manually define our hyperparameters in order to be able to specifically encode some properties of the data we work with. A common alternative is to set the hyperparameters using maximum-likelihood estimation over all possible values they can take.

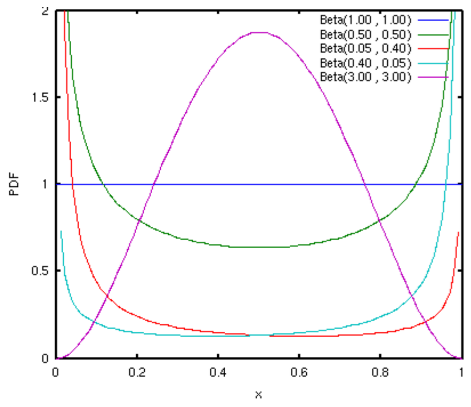

Figure 3.1: Shape of the Beta distribution with various parameter choices. Parameters > 1 lead to a unimodal shape. Parameters < 1 lead to a bimodal shape. Unequal parameters skew the distribution.

available, it is common to set the hyperparameters uniformly, thus providing an *uninformative*, or *uniform* prior. This amounts to assigning an equally valued pseudo count to every possible event.

The Beta-Binomial Distribution

The conjugate prior to the Binomial distribution is the Beta distribution (Bishop, 2006). The Beta distribution takes two parameters $\alpha > 0$ and $\beta > 0$ for the possible outcomes of the Binomial distribution. For a Binomial distribution with parameters π it is defined as

$$P(\pi|\alpha,\beta) = \frac{\Gamma(\alpha+\beta)}{\Gamma(\alpha)\Gamma(\beta)}\pi^{\alpha-1}(1-\pi)^{\beta-1} \tag{3.4}$$

$\Gamma(\cdot)$ stands for the Gamma function, an extension of the factorial to real numbers.

The Beta distribution can take various shapes depending on the choice of parameters. Figure 3.1 displays the Beta distribution with five different parameterizations. The value on the x-axis corresponds to the success probability π of the Binomial distribution. Beta(1,1) is a uniform distribution, which means that any success probability π for a Binomial distribution is equally likely. With uninformed parameters smaller than 1, the Beta distribution becomes symmetric and bimodal at the outer edges, which means that π is most likely either close to 1 or close to 0. This results in sparse posterior distributions, a feature we often want to achieve in our posterior distributions in Bayesian models of language. The smaller the parameters, the greater becomes the preference for parameterizations that lead to a sparse posterior distribution. Informed parameters smaller than zero result in a skewed distribution, favoring either a high success probability close to 1 (if $\alpha > \beta$) or a success probability close to 0 (if $\alpha < \beta$).

As mentioned above, given a conjugate prior it is possible to integrate over all possible parameterizations π. This allows us to interpret the parameters

α and β as pseudo counts. Imagine event a has been observed c_a times with probability π, and event b has been observed c_b times with probability $(1-\pi)$ in our data D. We combine this "likelihood" from the data with our "prior" belief in parameterization π, encoded as $Beta(\alpha, \beta)$.

$$\begin{aligned} P(\pi|D;\alpha,\beta) &\propto P(D|\pi)P(\pi|\alpha,\beta) && (3.5)\\ &\propto [\pi^{c_a}(1-\pi)^{c_b}][\pi^{\alpha-1}(1-\pi)^{\beta-1}] && (3.6)\\ &\propto \pi^{c_a+\alpha-1}(1-\pi)^{c_b+\beta-1} && (3.7)\end{aligned}$$

The equation in line 3.5 directly follows from Bayes' rule, and we substitute the definition for the respective distributions in line 3.6. The interpretation of α and β as pseudo counts becomes directly apparent in this notation, in line 3.7. The Beta parameters are simply added to the number of observations of the respective events. Note that the resulting term in 3.7 is again a Beta distribution, with updated parameters.

The Dirichlet-Multinomial Distribution

The Dirichlet distribution generalizes the Beta distribution to multiple possible outcomes[2], the same way the Binomial distribution is generalized by the Multinomial distribution (Bishop, 2006). The Dirichlet Distribution is parameterized by a vector of parameters $\boldsymbol{\gamma} = [\gamma_1, ...\gamma_I]$, one for each possible outcome i of the Multinomial. For a Multinomial distribution with parameters $\boldsymbol{\theta}$ the Dirichlet distribution is defined as

$$P(\boldsymbol{\theta}|\boldsymbol{\gamma}) = \frac{\Gamma(\sum_i \gamma_i)}{\prod_i \Gamma(\gamma_i)} \prod_i \theta_i^{\gamma_i - 1} \qquad (3.8)$$

The behavior of the Dirichlet distribution can be directly extrapolated from the Beta distribution towards the multidimensional case. Most relevant for our model, for parameters $1 > \gamma_i > 0$ the likely parameterizations $\boldsymbol{\theta}$ will result in sparser Multinomial distributions with decreasing values γ_i.

Like for the Beta-Binomial distribution, it is possible to integrate out the parameters $\boldsymbol{\theta}$ under the conjugate Dirichlet prior, and interpret the parameters $\boldsymbol{\gamma}$ as pseudo counts. Given data D containing observation counts c_i for each possible outcome i, the posterior distribution becomes

$$\begin{aligned} P(\boldsymbol{\theta}|D;\boldsymbol{\gamma}) &= P(D|\boldsymbol{\theta})P(\boldsymbol{\theta}|\boldsymbol{\gamma}) && (3.9)\\ &\propto \Big[\prod_i \theta_i^{c_i}\Big]\Big[\prod_i \theta_i^{\gamma_i-1}\Big] && (3.10)\\ &\propto \prod_i \theta_i^{c_i+\gamma_i-1}. && (3.11)\end{aligned}$$

Again, we can observe the influence of the Dirichlet parameters as pseudo counts directly in line 3.11, and we obtain a Dirichlet distribution with updated parameters.

[2]The Beta distribution is actually the Dirichlet distribution with exactly 2 dimensions.

We have explained the advantages and mathematical details of Bayesian inference, and how preference towards particular parameterization can be expressed through prior knowledge. Given that we infer the joint probability of all parameters $\boldsymbol{\theta}$ of the model, it is straightforward to incorporate many different components into a Bayesian model, as long as inference remains tractable. Below, we introduce the additional components we will include in our Bayesian script model.

3.3 The Generalized Mallows Model

We will use the Generalized Mallows Model (GMM, Fligner and Verducci (1986)) to model the ordering of events in ESDs. The GMM is a generalization of the Mallows Model (Mallows, 1957). We will first describe the Mallows Model and then introduce the GMM.

The Mallows Model is a probabilistic model over permutations of a set of elements. It defines a distribution over orderings, taking two parameters: a canonical ordering σ and a dispersion parameter $\rho \geq 0$. Generally, the canonical ordering of K elements can be defined as the identity ordering $\sigma = \{1, 2, ..., K\}$ without any loss in generality. We will assume the identity ordering as canonical ordering in the rest of this work. Finally, a distance function $d(\pi, \sigma)$ between an observed ordering π and the canonical ordering needs to be defined.

The probability of an ordering π decreases exponentially with increasing distance to the canonical ordering. The dispersion parameter ρ functions as a penalization factor, which encodes the degree to which dispersion from the canonical ordering is tolerated:

$$P(\pi) \propto e^{-\rho d(\pi, \sigma)}. \tag{3.12}$$

The single mode of the distribution is reached when $\pi = \sigma$, i.e. $d(\pi, \sigma) = 0$. An increasing value of ρ implies a stronger penalization of the distance, and thus results in a sharper distribution, centered around the canonical ordering.

Before we describe the *Generalized* Mallows Model (GMM) in detail, we explain how the distance between two orderings is represented in the GMM.

Distance in the Generalized Mallows Model

While in theory any distance metric can be used for measuring the distance between two permutations, the most common choice is Kendall's τ distance (Kendall, 1938). Kendall's τ distance defines the distance $d(\pi_1, \pi_2)$ between two orderings π_1 and π_2 as the number of pairwise flips necessary to turn π_1 into π_2. For the Generalized Mallows Model, we need to formulate Kendall's τ distance in a way that can be factorized into element-wise components.

Fligner and Verducci (1986) introduce the notion of *inversions* to specify the dispersion of each element i in the observed ordering from its canonical position. For an observed ordering π of a set containing K elements, the distance $d(\pi, \sigma)$ is represented through an inversion vector $\mathbf{v}$ of length $K - 1$. Each position i in $\mathbf{v}$ corresponds to element i in the canonical ordering σ, and its value is the number of elements j such that $j > i$ occurring before i in the observed ordering π. Given an observed ordering π with inversion count vector $\mathbf{v}$, each value v_i thus represents the distance of element i to its canonical position in the ordering, $d(\pi_i, \sigma_i) = v_i$.

As an example, taking the canonical ordering $\sigma = \{1, 2, 3\}$ and the observed ordering $\pi = \{3, 1, 2\}$, we obtain the inversion vector $v = \{1, 1\}$. The inversion count for element K is always 0 since there cannot be an element i such that $i > K$. Hence the length of $\mathbf{v}$ is $K - 1$.

The Generalized Mallows Model

Fligner and Verducci (1986) introduce the Generalized Mallows Model (GMM), generalizing the Mallows Model by introducing a *vector of dispersion parameters*, $\boldsymbol{\rho} = \rho_1, ...\rho_I$. Each element i in the permutation is assigned an individual dispersion parameter, defining an individual level of tolerance for distance from its canonical position. The probability of an ordering becomes:

$$GMM(\boldsymbol{\pi}; \boldsymbol{\rho}) = \frac{e^{-\sum_i -\rho_i d(\pi_i, \sigma_i)}}{\psi(\boldsymbol{\rho})} \tag{3.13}$$

$$= \prod_i \frac{e^{-\rho_i d(\pi_i, \sigma_i)}}{\psi_i(\rho_i)} \tag{3.14}$$

$$= \prod_i \frac{e^{-\rho_i v_i}}{\psi_i(\rho_i)} \tag{3.15}$$

with the normalizing constant $\psi_i(\rho_i) = \frac{1-e^{-(K-i+1)\rho_i}}{1-e^{-\rho_i}}$, K the total number of elements, and i the position of element i in the canonical ordering. Our inference algorithm works with the unnormalized GMM distribution, so we will omit the normalizing constant in the remaining discussion and change the equality to proportionality.

The definition in Equation 3.15 allows factorization of the distribution into individual components for each element i

$$GMM_i \propto e^{-\rho_i v_i}. \tag{3.16}$$

With parameters $\boldsymbol{\rho} > 0$ the probability of an ordering is maximized when $d(\pi_i, \sigma_i) = 0$ (or, equivalently, $v_i = 0$) for every i, i.e. when the observed ordering resembles the canonical ordering. A high value for parameter ρ_i indicates a strong tendency for element i to stay at its canonical position.

In our script model we use the GMM in order to infer one specific dispersion parameter ρ_i for every event type i in our model. With the induced parameters, we will be able to model individual temporal flexibility for every event type.

The Conjugate Prior

Since the GMM belongs to the exponential family, a conjugate prior can be defined. From this prior, the parameterization of the GMM is sampled, in particular the dispersion parameter vector $\boldsymbol{\rho}$. Like the GMM, the conjugate prior can be factorized into element-wise components for each ρ_i (Fligner and Verducci, 1990):

$$GMM_0(\rho_i | v_{i,0}, \nu_0) \propto e^{-\rho_i v_{i,0} - log(\psi_i(\rho_i))\nu_0}, \tag{3.17}$$

where $v_{i,0}$ and ν_0 are hyperparameters which need to be defined manually. The hyperparameter $v_{i,0}$ encodes the distance of element i from its canonical position observed in prior 'pseudo trials', and ν_0 is the number of such pseudo trials, i.e. the weight of the prior information.

It is difficult to manually set $v_{i,0}$ for every element i. Remember that v_i encodes the number of elements j such that $j > i$ in the observed ordering. This definition implies that the range of possible values for v_i is individual for each element i, namely $K - i$, where K is the total number of elements and i is the canonical position of element i. We follow Chen et al. (2009), in defining the vector of prior inversion counts by specifying another parameter, common to all i, which we call $\rho 0$. Now each $v_{i,0}$ is computed such that the maximum likelihood estimate of ρ_i is $\rho 0$. This amounts to evaluating the equation

$$v_{i,0} = \frac{1}{e^{\rho 0} - 1} - \frac{K - i + 1}{d^{(K-i+1)\rho 0} - 1}. \tag{3.18}$$

3.4 The Logistic Normal Distribution

The Logistic Normal distribution (Aitchison, 1982; Blei and Lafferty, 2006) models correlations among components of a vector defined on the simplex[3]. A normally distributed parameter vector $\mathbf{x}$ is sampled from a Multivariate Gaussian $N(\Sigma, \boldsymbol{\mu})$ distribution in logarithmic space

$$\log p(\mathbf{x}|\Sigma, \boldsymbol{\mu}) \propto \psi - \frac{1}{2}(\mathbf{x} - \boldsymbol{\mu})^T \Sigma^{-1} (\mathbf{x} - \boldsymbol{\mu}), \tag{3.19}$$

with the normalizing constant $\psi = \frac{d}{2}\log(2\pi) - \frac{1}{2}\log(|\Sigma|)$, and d the number of components in $\mathbf{x} = [x_1, ..., x_d]$. The Multivariate Gaussian $N(\Sigma, \boldsymbol{\mu})$ is the generalization of the Gaussian distribution to the multivariate case. It is used for modeling the correlations among a number of normally distributed random variables. Assuming d random variables, the Multivariate Gaussian has parameters Σ, a $d \times d$ covariance matrix and $\boldsymbol{\mu}$, a mean vector with d elements. The parameter μ_i encodes the mean value for every random variable i, and Σ encodes the variance of every random variable (along the diagonal), and the co-variances between any two components i and j in cell $\Sigma_{i,j}$.

The sampled parameter vector $\mathbf{x}$ is subsequently normalized, in order to fulfill the simplex-constraint. The logistic transform is used for normalization, yielding a vector $\boldsymbol{\eta}$, such that for each component i:

$$\eta_i = \frac{exp(x_i)}{\sum_{i'} exp(x_{i'})}. \tag{3.20}$$

In our model we will use the Logistic Normal distribution to induce type-specific hyperparameters for our language models. We will specify the covariance matrix Σ such that it encodes word similarities between all terms in our vocabulary. Pseudo counts for semantically related words will then correlate, which results in correlated term probabilities for each induced cluster. Please refer to Section 6.1 for a more detailed explanation of how we use the Logistic Normal distribution to encode prior knowledge in our model.

[3]This means that the components of the vector must some to one; it is effectively the space of all possible parameterizations of the Multinomial distribution.

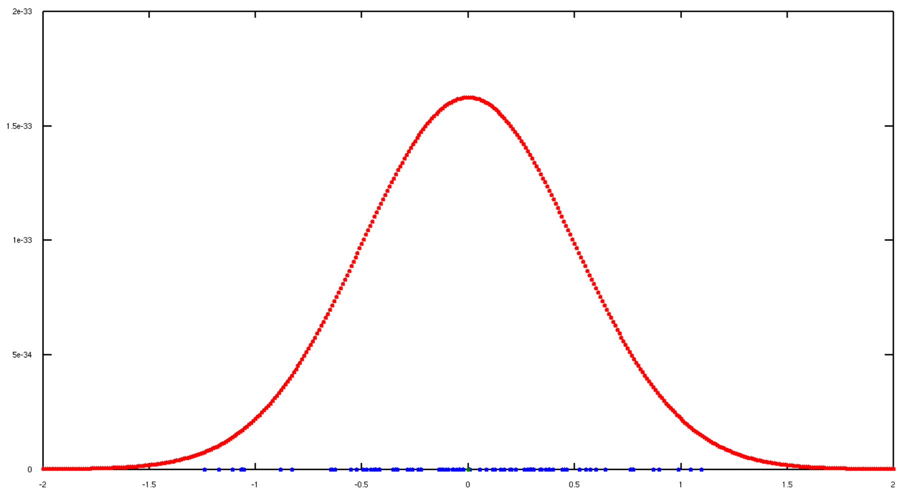

Figure 3.2: Illustration of approximate samples our slices sampler returns for a normally distributed random variable. In red, we show the true distribution (a Normal distribution), and in blue we show the data points returned in 100 runs of the slice sampler.

3.5 Markov Chain Monte Carlo

Markov Chain Monte Carlo (MCMC) methods are a family of techniques for approximate sampling from probability distributions (Neal, 1993; Andrieu et al., 2003). As indicated before, the denominator of Equation 3.3 can generally not be computed analytically for any reasonably complex model. MCMC methods are one possible way of approximating a distribution as similar as possible to the target posterior distribution of parameterizations given the data $P(\theta|y)$. They are used when direct sampling from the target distribution is intractable, but it can be evaluated up to the normalizing constant.

Figure 3.2 illustrates the process behind approximate sampling on a simple example. We show the true distribution, a Normal distribution, in red. Below, in blue, we show the data points returned by one particular MCMC algorithm, a slice sampler. The slice sampler computes data points based on the unnormalized Normal distribution. It can be seen that the data points are distributed proportionally to the probability mass of the true distribution.

In MCMC sampling, observations from the distribution to be approximated are simulated by collecting sample points through a simulated *random walk* over the probability space. Based on the observations, the characteristics of the distribution are estimated, since the sampler will spend time in the regions of the probability space proportional to the regions' probability. The process must be a valid Markov chain: the probability of being in a certain state at time t must depend only on the previous state at time $(t-1)$.

This method is only valid if the Markov chain has our desired posterior distribution $P(\theta|\mathbf{y})$ as its *stationary distribution* (MacKay, 2002). The randomly initialized Markov chain will then converge to this distribution, which basically means that the probabilities of the possible state transitions do not change

```
*Initialize all variables* $\mathbf{x}^0$ / *initialize the markov chain*
**for** *Sampling iterations* $t = \{1...N\}$ **do**
  **for** *variables* $x_i^t \in \mathbf{x}$ **do**
    $x_i^{t+1} \sim P(x_i^{t+1}|x_1^{t+1}, ..., x_{i-1}^{t+1}, x_{i+1}^t, x_{i+2}^t, ...)$
```

Figure 3.3: The basic Gibbs sampling algorithm.

anymore. The chain can then be described by a single transition matrix. Or, equivalently, the probability distribution over $x^{(t+1)}$ given x^t does not depend on the particular t.

It is difficult in practice to check whether the Markov chain has converged to its stationary distribution. It is thus common practice to:

1. Run the sampler for a 'burn-in' period during which no samples are collected. This should eliminate the influence of the initial values.
2. Only take every nth sample for each variable to avoid auto-correlations between the samples.
3. Run several samplers in parallel and average the results in the end.

Below, we explain the two MCMC algorithms we will use in the inference procedure of our script model.

3.5.1 The Gibbs Sampler

The Gibbs sampler is one particular MCMC method, which we will use for inference in our model (Geman and Geman, 1984; MacKay, 2002). It is particularly useful when samples from a high-dimensional probability distribution need to be obtained. Since we need to obtain the joint probability of all parameters in the model given the data, the Slice sampler is a natural choice. Obtaining samples from the joint distribution of all components $\mathbf{x}$ of a Multivariate probability distribution is often hard. Gibbs sampling allows to instead sample in turn from the conditional distribution of each variable x_i conditioned on all other variables except x_i itself in order to approximate the joint distribution. This is a valid approximation since the conditional distribution of a variable given all other variables is proportional to the joint distribution:

$$P(x_i = v|x_1, ..., x_{i-1}, x_{i+1}, ..., x_n) = \frac{P(x_1, ..., x_n)}{P(x_1, ..., x_{i-1}, x_{i+1}, ..., x_n)}. \quad (3.21)$$

The normalization constant can generally be ignored since it does not depend on x_i.

A probability distribution over all possible values v for variable x_i is computed, conditioned on the current values of all other variables. The previous value of x_i is ignored in the computation. The value of variable x_i is finally updated by sampling the new value from the computed distribution over all possible v. An overview over the sampling process is given in Figure 3.3.

We will used *collapsed* Gibbs sampling for inference (Griffiths and Steyvers, 2004). This means that we integrate over, or marginalize out, some of our

```
Draw vertical corrdinate $u \sim Uniform(0, P^*(x^t))$        [ $P^*(x) \propto P(x)$ ]
Create horizontal interval $(x_l, x_r)$ enclosing $x^t$
while $True$ do
  Draw horizontal coordinate $x^{t+1} \sim Uniform(x_l, x_r)$
  Evaluate $P^*(x^{t+1})$
  if $P(x^{t+1}) > u$ then
    Break                                  [ accept $x^{t+1}$ as new sample ]
  else
    Shrink the interval $(x_l, x_r)$       [ if $(x^{t+1}, u)$ lies outside curve ]
```

Figure 3.4: The basic Slice sampling process for a single transition of variable x at $t \to t+1$. Adapted from MacKay (2002), page 375.

parameter distributions to reduce the state space of the Gibbs sampler, and make inference more efficient. This is possible, because we decide to use conjugate prior distributions for our parameter distributions (see Section 3.2.2). The mathematical details for our model are given in Section 5.2.

3.5.2 The Slice Sampler

While the posterior distribution of most parameters in our model is discrete, and posterior samples can thus be gained in a straightforward way, some parameters follow a continuous distribution for which the normalization factor cannot be computed. We use Slice sampling (Neal, 2003) to resample those parameters from their continuous posterior.

Slice sampling is an efficient MCMC method which is particularly robust to parameter choices. Like Gibbs sampling, it provides samples from unnormalized probability distributions but requires the distribution to be evaluate-able at every point (MacKay, 2002).

The basic idea is to randomly sample a value y $\sim [0..P(x_i^t)]$, where x_i^t is the previous value of the variable x_i we want to resample. A horizontal line is drawn at y through the curve of $P(\mathbf{x})$. Then, sample x_i^{t+1} is drawn uniformly from the area above the line, but inside the curve of $P(\mathbf{x})$. Intuitively, the x-values in the respective area are distributed proportional to their probability under the curve. With Multivariate distributions this procedure can be repeated individually for each component. Figure 3.4 summarizes the basic Slice sampling procedure.

Slice sampling might cause problems with multimodal densities, since the horizontal line may be cut into disconnected pieces by the distribution. While there are ways to optimize Slice sampling for multimodal distributions, the distributions we apply Slice sampling to are unimodal, so we do not expect any problems here.

Chapter 4

A Hierarchical Bayesian Model for Scripts

After having established the necessary technical background, we will now describe our Bayesian model for inducing script knowledge. We start in Section 4.1 by providing a high-level description of the problem we want to solve, and continue by giving an overview over the modeling assumptions we make in Section 4.2. In Section 4.3, we will explain the formalized generative story of the script model. For a key to the notation we use in this and the following chapters, please refer to Table 4.1.

4.1 Problem Formulation

Our input data consists of scenario-specific corpora of event-sequence descriptions. For each corpus, we want to cluster together equivalent event descriptions referring to the same event type, and equivalent participant descriptions, referring to the same participant type, as illustrated in our introductory example in Table 1.1. Practically, we label each event- and each participant description in our corpus, and assign descriptions with identical labels to the same cluster. We want to capture the canonical ordering of event types in our clusters. Concretely this means that event types which tend to occur early in the scenario are preferably labeled with a lower ID by our model than event types which occur later[1].

We specify the maximum number of possible event types E and the maximum number of possible participant types P that our model can use during inference a priori. We set these numbers much higher than the expected actual number of types of events and participants in the data. The model will use a subset of all possible types.

Formally, assume a scenario-specific corpus c consisting of D ESDs $c = \{d_1, d_2, ..., d_D\}$. Each ESD d_i consists of N_d event descriptions $d_i = \{d_{i,1}, ..., d_{i,N_i}\}$. Boundaries between descriptions of single events are marked in the data.

Each event description $d_{i,n}$ is split into two parts, by extracting all participant descriptions from the original event description. Each participant descrip-

[1]This follows from the fact that we define the identity ordering for the GMM as the canonical ordering $\sigma = [1, 2, 3, ..., n]$.

Symbol	**Explanation**
Iterators	
$d = 1...D$	Iterator over documents (ESDs in our corpus)
$e = 1...E$	Iterator over event types
$p = 1...P$	Iterator over participant types
$i = 1...I$	Iterator over event descriptions in an ESD
$j = 1...J$	Iterator over participant descriptions in an event description in an ESD
$v = 1...\|V\|$	Iterator over terms in event- / participant vocabulary (clear from context)
Hyperparameters	
$\alpha^+/\ \alpha^-$	hyperparameters for eventtype realization
$\beta^+/\ \beta^-$	hyperparameters for participanttype realization
$\boldsymbol{\gamma}_e = [\gamma_1, ...\gamma_{\|V_\eta\|}]$	hyperparameters for language model of eventtype e
$\boldsymbol{\delta}_p = [\delta_1, ...\delta_{\|V_\xi\|}]$	hyperparameters for language model of participanttype p
ρ_0, ν_0	hyperparameters of the GMM
Σ_η/Σ_ξ	covariance matrix encoding event term similarities / participant term similarities
Distributions over Latent Variables	
$\boldsymbol{\theta} = [\theta_1...\theta_E]$	Binomial parameters modeling event type realization probabilities
$\boldsymbol{\varphi}_e = [\varphi^e_1...\varphi^e_P]$	Binomial parameters modeling participant type realization probabilities under eventtype e
$\boldsymbol{\rho} = [\rho^1...\rho^E]$	dispersion parameters of the Generalized Mallows Model
$\boldsymbol{\vartheta}_e = [\vartheta^1_e...\vartheta^{\|V_\eta\|}_e]$	Multinomial parameters of language model for eventtype e
$\boldsymbol{\varpi}_p = [\varpi^1_p...\varpi^{\|V_\xi\|}_p]$	Multinomial parameters of the language model for participanttype p
Latent Variables	
$\boldsymbol{\tau}_d = [\tau^1_d, ..., \tau^E_d]$	realized eventtypes in ESD d
$\boldsymbol{\kappa}^{d,i} = [\kappa^{d,i}_1, ..., \kappa^{d,i}_P]$	realized participanttypes in event i such that $\boldsymbol{\tau}_{d,i} = 1$ in ESD d
$\boldsymbol{\pi}_d$	event ordering of ESD d
$\mathbf{v_d} = [v^1_d, ...v^{E-1}_d]$	inversion count representation of event ordering of ESD d
$\boldsymbol{\eta}_e = [\eta^1_e, ..., \eta^{\|V_\eta\|}_e]$	parameter vector sampled from multivariate gaussian $N(\Sigma_\eta, 0)$ for event type e
$\boldsymbol{\xi}_p = [\xi^1_p, ..., \xi^{\|V_\xi\|}_p]$	parameter vector sampled from multivariate gaussian $N(\Sigma_\xi, 0)$ for event type p
Observable Variables	
$\mathbf{w_{d,i}}$	realization of event i in ESD d
$\mathbf{w^j_{d,i}}$	realization of participant j in event i in ESD d
V_η	event vocabulary
V_ξ	participant vocabulary

Table 4.1: Overview over the notation we use in the description of the model and inference algorithm.

tion corresponds to one noun phrase as identified automatically by a dependency parser (cf. Regneri et al. (2011)). The event description now consists of two parts: (1) a bag of participant descriptions $\boldsymbol{\kappa}^{i,n}$, where each participant description corresponds to one noun phrase, and (2) the remainder of the original phrase $\tau_{i,n}$, which corresponds to a verb phrase lacking its arguments. For illustration, please consider the following example:

$d_{i,n}$ = "put pasta into boiling water"
$\tau_{i,n}$ = "put into"
$\boldsymbol{\kappa}^{i,n}$ = {"pasta", "boiling water"}

Given a corpus of ESDs in the format defined above, our model labels each ESD d_i by assigning each event description $\tau_{i,n}$ exactly one event type $\tau_{i,n} = e$, where $e \in \{1, ..., E\}$. Each participant description j in the bag of participant descriptions is assigned exactly one participant type $\kappa_j^{i,n} = p$, where $p \in \{1, ..., P\}$.

4.2 Model Overview

We define our generative model of script knowledge by deciding, for each hidden variable in our model, for the kind of distribution the hidden variable is drawn from. We describe and justify our choice of distributions in this section, before we formally describe the generative story of how an ESD is be generated on the basis of these distributions in the following section.

Our model includes three types of latent variables (1) event types, (2) participant types, and (3) orderings of event types. We assume that the underlying structure of an observed corpus of scenario-specific ESDs can be explained through these three types of latent labels. In order to be able to relate latent labels to observable data, namely words, we need to define one language model for each event type e and each participant type p.

Event Types We model event types using Binomial distributions. For each event type e in our model, we define a Binomial distribution $Binomial(\theta_e)$, where θ_e corresponds to the probability with which an event type is realized in an ESD. One advantage of this modeling decision is that it naturally explains optional events. Taking the example of the `Eating in a restaurant` scenario, we could model the optional event type "complaining about the food" with a Binomial distribution with a low realization probability as parameter. This would account for the observation that some ESDs for this scenario contain the mentioned event type, while most ESDs do not.

Participant Types We model participant types through a set of Binomial distributions $Binomial(\varphi_p^e)$, each with a realization probability φ_p^e specifying how likely it is to observe participant type p in event type e. Note that we thus model participant type probabilities specifically for each event type. As an example, we can model that the participant type "food" is likely to occur within the event type "eat", by assigning a high realization probability, while we can assign a low realization probability to the participant type "bill" under the same event type. It is also possible to model event types in which no participants occur at all, through a sufficiently low realization probability for each participant type in the model under the event types.

Event Orderings We model event orderings using the Generalized Mallows Model. Specifying the identity ordering $\sigma = [1, 2, 3..., N]$ as a parameter of the GMM, our model will have a preference towards labeling event types in an ESD incrementally. However, we can specify a dispersion parameter ρ_e for each event type e, which encodes the temporal flexibility of this specific event. Through the GMM we are able to model event type-specific temporal flexibility. Taking the `Cooking pasta` scenario as an example, we could specify that the event type "boiling water" is strongly preferred to occur at its canonical position σ_e, in the beginning of the ESD, by assigning a high value to its corresponding dispersion parameter. The event type "graining the cheese", which can occur at any temporal position in the scenario, would be assigned a lower-valued dispersion parameter, and divergence from its canonical position is thus only moderately penalized.

Language Models We need to relate hidden variables, namely event types and participant types, to the observable words. Given our corpus format as described in Section 4.1, we can separate our vocabulary into an event vocabulary V_η and a participant vocabulary V_ξ. We then define one language model for each event type e and each participant type p as a Multinomial distribution over the respective vocabulary: $Mult(\boldsymbol{\vartheta}_e)$ for event types e and $Mult(\boldsymbol{\varpi}_p)$ for participant types p. Concretely, for each term in the event vocabulary $v \in V_\eta$, the parameter ϑ_e^v specifies the probability that term v occurs in an event description of event type e. Participant language model parameters are interpreted equivalently.

4.3 The Generative Process

Given the specification of our set of latent variables, and our respective choice of distributions, we will now describe the formal generative process for generating a document (ESD) d. The generative process is formalized in Figure 4.1. Our model is displayed as a plate diagram for a graphical overview in Figure 4.2.

Events We start by generating event types for ESD d. For this, we independently draw for each event type e from $Binomial(\theta_e)$, with θ_e being the globally defined realization probability of the respective event type. We thus obtain a binary event vector $\boldsymbol{\tau}_d$ of length E, where $\tau_{d,e} = 1$ if event type e is realized in ESD d, and 0 otherwise.

Ordering We continue by generating an ordering in which the generated event types will be realized in d. We draw an event ordering π_d from $GMM(\boldsymbol{\rho})$. We represent our ordering as a vector or inversion counts $\mathbf{v}_d$ (see Section 3.3).

Participants Given an ordered set of realized event types, we generate participant types for each generated event i, such that $\tau_{d,i} = 1$. We successively

Generation of parameters

```
for event type $e = 1, \dots, E$ do
  $\theta_e \sim Beta(\alpha^+, \alpha^-)$                [ freq of event ]
  $\vartheta_e \sim Dirichlet(\boldsymbol{\gamma})$              [event lang mod]
  for participant type $p = 1, \dots, P$ do
    $\varphi_p^e \sim Beta(\beta^+, \beta^-)$              [ freq of ptcpt ]
for participant type $p = 1, \dots, P$ do
  $\varpi_p \sim Dirichlet(\boldsymbol{\delta})$                 [ ptcpt lang mod ]
for event type $e = 1, \dots, E - 1$ do
  $\rho_e \sim GMM_0(\boldsymbol{\rho}_0, \boldsymbol{\nu}_0)$              [ ordering params]
```

Generation of ESD d

```
for event type $e = 1, \dots, E$ do
  $\tau_e \sim Binomial(\theta^e)$                [ realized events ]
$\mathbf{v} \sim GMM(\boldsymbol{\rho}, \boldsymbol{\nu})$                    [ inversion counts ]
$\boldsymbol{\pi} \leftarrow sort(\mathbf{v}, \boldsymbol{\tau})$                [ sort $\boldsymbol{\tau}$ wrt $\mathbf{v}$ (deterministic) ]
for event $i$ from $\boldsymbol{\pi}$ s.th. $\tau_i$=1 do
  $w_i \sim Mult(\vartheta_i)$                  [ event lexical unit ]
  for participant type $p = 1, \dots, P$ do
    $\kappa_p^{d,i} \sim Binomial(\varphi_p^i)$         [ realized ptcpts ]
    if $\kappa_p^{d,i} = 1$ then
      $w_p \sim Mult(\varpi_p)$              [ ptcpt lexical unit]
```

Figure 4.1: The generative story of the script model. Top: The generative story for parameters. Bottom: The generative story for data, given the parameters.

consider each realized event type in the order specified in $\boldsymbol{\pi}_d$. For each event type i, we independently decide for each participant type p whether to generate it or not, by drawing from $Binomial(\varphi_i^p)$. We thus obtain a binary participant vector of length P for each realized event type i in d, $\boldsymbol{\kappa}^{d,i}$ where $\kappa_p^{d,i} = 1$ if participant type p is realized, and 0 otherwise.

Words Finally, we generate words for each realized event type and each realized participant type. For each realized event type i, we decide on the number of realizing words n_i, and generate n_i words by drawing n_i times from $Mult(\boldsymbol{\vartheta}_i)$. Similarly, we generate words for each realized participant type j. In this case we draw only once from the the language model $Mult(\boldsymbol{\varpi}_j)$, for each realized participant type j, because each participant description consists of one head word only.

Note that we assume *event type-specific* realization probabilities for each participant type p, φ_p^e. However, the model includes only one *global* language model for each participant type, $Mult(\boldsymbol{\varpi}_p)$. This allows us to model participants in a globally coherent way, while still allowing for flexibility on the event type level, basically learning sets of possible realizations for all argument slots of

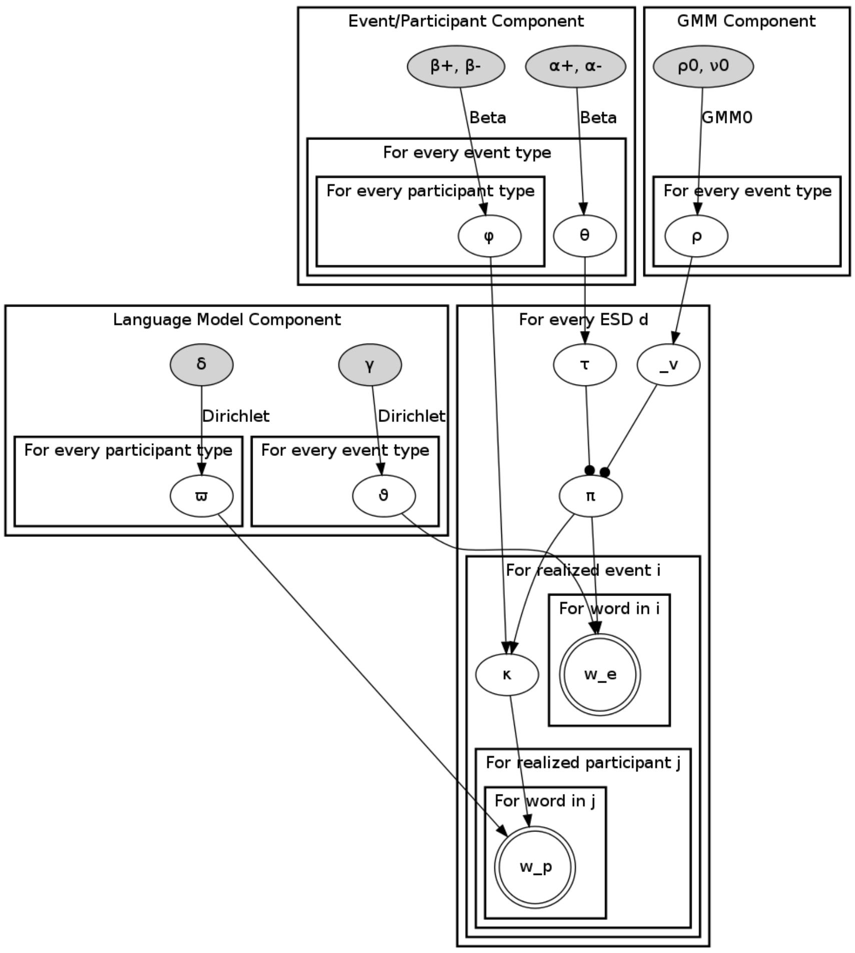

Figure 4.2: A plate diagram of the script model, illustrating the generative process, and dependencies between the variables. An arrow from node a to node b means that b depends on a. Arrows with a round head indicate deterministic computations. Shaded variables indicate manually specified hyperpriors. Observed variables (i.e. words) are indicated through double circles.

each particular event type.

4.3.1 Generating Parameters from Prior Knowledge

We have not yet explained how we obtain the parameters of all Binomial distributions, Multinomial distributions, and the GMM. Proposing a *Bayesian* model, we draw those parameters themselves from a distribution over parameters, thus accounting for the uncertainty involved when we decide for a particular parameterization. All parameterizations are *global* within one scenario type, and are shared among all ESDs. The generative process for generation of parameters is formalized in the top half of Figure 4.1, and can be described as follows.

Event realization parameters First, we generate the realization parameter for each event type-specific Binomial distribution. In particular, for each event type e we draw the parameter θ_e from $Beta(\alpha^+, \alpha^-)$, the conjugate prior distribution to the Binomial distribution, with hyperparameters α^+ and α^- (see Section 3.2.2). Through the choice of hyperparameters we can prefer sparsity in our posterior event type distribution, which means that we prefer all realization parameters to be either close to 0 or close to 1. This results in using only few distinct event types in our posterior corpus labeling, thus encouraging clustering similar descriptions together. We can achieve this by setting defining $\alpha^+ << 1$ and $\alpha^- << 1$ (cf. Figure 3.1).

Participant realization parameters Similarly, we generate realization parameters for event type-specific participant Binomial distributions, by drawing the parameters φ_p^e from $Beta(\beta^+, \beta^-)$. Again, we specify the hyperparameters in a way that encourages sparsity in the posterior participant labelings, using the same reasoning as described above.

Language Model Parameters We draw the language model parameters for each event type e, $\boldsymbol{\vartheta}_e$, from $Dirichlet(\boldsymbol{\gamma})$. The Dirichlet distribution is the conjugate prior distribution to the Multinomial (see Section 3.2.2). The hyperparameters $\boldsymbol{\gamma}$ in this case is a vector with dimensions corresponding to the possible outcomes of the Multinomial, i.e. the terms in the event vocabulary. By choosing a value $\gamma_t << 1$ for each term t in the vocabulary, we again can encourage sparsity across our language models[2]. This means concretely, that only few distinct terms are a likely realization for each event type e.

Equivalently, we define the hyperparameters $\boldsymbol{\delta}$ for our participant type language models, and draw language model parameters $\boldsymbol{\varpi}_p$ for each participant type p from $Dirichlet(\boldsymbol{\delta})$.

Ordering Parameters We draw our ordering parameters $\boldsymbol{\rho}$ from the conjugate prior distribution of the Generalized Mallows Model $GMM_0(\rho_0, \nu_0)$. Using hyperparameter ρ_0, we compute a prior inversion count $v_{e,0}$ for each event type e, encoding a prior intuition about how much dispersion the model should tolerate for each event type e. The hyperparameter ν_0 encodes the strength of our

[2]Assume, for simplicity, an uninformed prior for the moment, where each γ_v has the same value. We will describe an augmentation of our model for language model prior determination in Chapter 6.

belief in this prior intuition (cf. Section 3.3). Practically, we want our model to prefer event orderings similar to the identity ordering, such that we specify very low prior inversion counts for all event types.

4.4 Summary

We have provided a description of the Bayesian script model, which takes a corpus of scenario-specific event-sequence descriptions as input, and induces clusters of equivalent event descriptions and equivalent participant descriptions. The event clusters furthermore capture the underlying ordering of events in the scenario. We described the formalized generative story to illustrate how observable data relate to the hidden variables in our model.

Note that some fundamental simplifying assumptions are encoded in our model. By realizing event types through *one draw* from each event type-specific Binomial distribution, we assume that each event type can occur only once per ESD. Similarly, by using event type-specific Binomials for modeling participant type realization and drawing *once* from each Binomial, we assume that each participant type can occur only once per realized event.

Furthermore, by drawing *independently and successively* from each event type-specific Binomial $Binomial(\theta_e)$ and each event type-specific participant Binomial $Binomial(\varphi_p^e)$, we assume independence between event types in an ESD, and between participant types in a realized event type. We thus generate a bag of participants for a realized event, which implies that we do not include any notion of the syntactic position - or the semantic role - of the individual participants.

Finally, we draw realizing words independently from the respective event language models ϑ_e. This means that we assume independence between words within each particular realization of an event type, encoding the bag-of-words assumption, which is a common choice in topic models. Since each participant is realized by only one head word, this is not an issue in this component.

Chapter 5

Inference

In the previous chapter, we have described the architecture of the script model, using the generative process for illustration. The generative process assumes that we know all hidden parameters, and can generate data (ESDs) on this basis. However, the actual problem setting we are faced with is that we observe the data, and want to learn the parameters of the distributions of the hidden variables from the data. In order to do this, we need to revert the generative story and define an inference procedure. We use Bayesian inference for formulating the posterior distributions of all our hidden variables, and use Gibbs sampling for obtaining approximate samples for those variables.

We will start by describing the full, joint posterior probability of all hidden parameters given the data. We continue simplifying the inference process by (1) utilizing the independence assumptions inherent in our model to make inference more efficient in Section 5.1, and (2) integrating over some hidden distributions to reduce the state space from which we learn our hidden parameters in Section 5.2. We finally define the conditional posterior distributions of all remaining parameters in the Gibbs sampling framework in Section 5.3.

5.1 The Full Posterior

In Bayesian modeling, we want to learn the joint probability of all variables in the model, given the observable data. For our model, this amounts to (please refer to Table 4.1 for the notation we use):

$$P(\boldsymbol{\tau}, \boldsymbol{\kappa}, \mathbf{v}, \boldsymbol{\theta}, \boldsymbol{\varphi}, \boldsymbol{\vartheta}, \boldsymbol{\varpi}, \boldsymbol{\rho}; \alpha^+, \alpha^-, \beta^+, \beta^-, \boldsymbol{\gamma}, \boldsymbol{\delta}, \rho_0, \nu_0 | \mathbf{D}), \tag{5.1}$$

where $\mathbf{D}$ stands for all our data, all variables left of the semicolon are our hidden parameters, and everything right of the semicolon are the manually optimized hyperparameters of the model, or the prior knowledge we define.

According to the independence assumptions made in generative story, and graphically shown in the plate diagram of our model in Figure 4.2, we can simplify the joint distribution by factorizing it into independent components:

$$P(\boldsymbol{\tau}, \boldsymbol{\kappa}, \mathbf{v}, \boldsymbol{\theta}, \boldsymbol{\varphi}, \boldsymbol{\vartheta}, \boldsymbol{\varpi}, \boldsymbol{\rho}; \alpha^+, \alpha^-, \beta^+, \beta^-, \boldsymbol{\gamma}, \boldsymbol{\delta}, \rho_0, \nu_0 | \mathbf{D}) \tag{5.2}$$

$$= \prod_{e=1}^{E} \Big[P(\theta_e|\alpha^+, \alpha^-) P(\rho_e|\rho_0, \nu_0) \prod_{p=1}^{P} \Big[P(\varphi_e^p|\beta^+, \beta^-) \Big] \Big] \tag{5.3}$$

$$\prod_{e=1}^{E} \Big[P(\boldsymbol{\vartheta}_e|\boldsymbol{\gamma}_e) \Big] \prod_{p=1}^{P} \Big[P(\boldsymbol{\varpi}_p|\boldsymbol{\delta}_p) \Big] \tag{5.4}$$

$$\prod_{d=1}^{D} \Big[P(\mathbf{v_d}|\boldsymbol{\rho}, \boldsymbol{\kappa}^d) P(\boldsymbol{\tau}_d|\boldsymbol{\theta}, \mathbf{v_d}, \boldsymbol{\kappa}^d) \prod_{i:\boldsymbol{\tau}_{d,i}=1} \Big[P(\mathbf{w_{d,i}}|\boldsymbol{\vartheta}_i) P(\boldsymbol{\kappa}^{d,i}|\boldsymbol{\varphi}_i) \tag{5.5}$$

$$\prod_{j:\kappa_j^{d,i}=1} \Big[P(\mathbf{w^j_{d,i}}|\boldsymbol{\varpi}_j) \Big] \Big] \Big] \tag{5.6}$$

The term 5.3 in the above equation corresponds to the event type/participant type component and the GMM component in the plate diagram in Figure 4.2. Term 5.4 corresponds to the language model component, terms 5.5 – 5.6 correspond to the ESD generation component.

We use conjugate prior distributions over the distributions of our hidden variables. This allows us to integrate out the Binomial parameters, specifying event realization probability and participant realization probability for each type, $\boldsymbol{\theta}$ and $\boldsymbol{\varphi}$, and the Multinomial parameters, specifying the type specific language model parameters $\boldsymbol{\vartheta}$ and $\boldsymbol{\varphi}$. In the following section, we will describe the integration process of the mentioned parameter distributions. Subsequently, we define the simplified posterior distribution for each hidden variable in the reduced sampling space.

5.2 Integrating out Parameter Distributions

In this section, we describe the process of integrating out the Binomial event type realization parameters $\boldsymbol{\theta}$ and participant type realization parameters $\boldsymbol{\varphi}$, and the Multinomial language model parameters $\boldsymbol{\vartheta}$ and $\boldsymbol{\varpi}$. For each type of distribution, we go through one derivation in detail, taking $\boldsymbol{\theta}$ and $\boldsymbol{\vartheta}$ as respective examples.

Intuitively, by summing (or integrating) over the parameterizations of the Binomial and Multinomial distributions, we incorporate all possible values the respective parameters can take in our model, without representing them explicitly. In fact, the probability of a particular parameterization can be represented purely on the basis of the respective hyperparameters, and the *sufficient statistics* in the data (Bishop, 2006). The sufficient statistics in our case are simply the counts of particular observations (words occuring with particular event type labels and participant type labels) in our corpus, as labeled by the model. Please refer to Table 5.2 for an overview over the counts relevant to our posterior distributions, and an explanation for the notation we use to encode the sufficient statistics. Just like we sampled the Binomial and Multinomial parameters from their conjugate Prior distribution shaped by our choice of hyperparameters before, after the parameters are integrated out the sufficient statistics will be modified, or shaped, by the hyperparameters encoding our prior intuitions.

Symbol	**Explanation**
N	Number of ESDs in the corpus
$N_{\cdot}^{\cdot\prime}$	Any count *excluding* the values of the currently resampled variable
Relevant Counts for Event Binomials	
$N_{\cdot}^{e}$	Number of times event type e is realized
$N_{\cdot}^{\bar{e}}$	Number of times event type e is not realized
Relevant Counts for Participant Binomials	
$N_{\cdot}^{e,p}$	Number of times participant type p is realized under event type e
$N_{\cdot}^{e,\bar{p}}$	Number of times event type e is realized without participant type p
Relevant Counts for Word Multinomials (language models)	
N_{v}^{e}	Number of times term $v = 1..\lvert V_\eta\rvert$ in the event vocabulary occurs in event type e
N_{v}^{p}	Number of times term $v = 1..\lvert V_\xi\rvert$ in the participant vocabulary occurs in participant type p

Table 5.1: Notation for the sufficient statistics, relevant for the different kind of parameter distributions we model.

On a high level, the basic integration procedure can be described as follows:

1. Identify all terms in Equation 5.2 that are affected by the parameters to be integrated over
2. Remove potential conditioning factors that are not directly influenced by the parameters
3. Go through the integration process
4. Obtain a simplified equation which is parameterized only by sufficient statistics and hyperparameters

Integrating out Binomial parameters $\boldsymbol{\theta}$ and $\boldsymbol{\varphi}$

We will describe the process of integrating over the parameters of a Binomial distribution, which are in turn drawn from its conjugate prior distribution, the Beta distribution. The resulting term is called a Beta-Binomial distribution (BBM), which is parameterized by counts of observations in the data and the parameters of the Beta distribution. The parameters of the Binomial distribution will not be mentioned in the final definition.

We exemplify the process taking the Binomial event type realization parameters $\boldsymbol{\theta}$. The relevant terms from Equation 5.2 which are affected by $\boldsymbol{\theta}$ are $P(\boldsymbol{\theta}|\alpha^{+}, \alpha^{-})$ and $\prod_d P(\boldsymbol{\tau}_d|\boldsymbol{\theta}, \mathbf{v}_d, \boldsymbol{\kappa}^d)$. The conditioning factors $\mathbf{v}_d$ and $\boldsymbol{\kappa}^d$ can be ignored in the derivation below, since they do not depend on $\boldsymbol{\theta}$. Below, we focus on one document only, which allows us to temporarily drop the index d.

$$
\begin{aligned}
&\int_{\boldsymbol{\theta}} \prod_{e=1}^{E} P(\theta_e|\alpha^+,\alpha^-)P(\tau_e|\theta_e)\mathrm{d}\boldsymbol{\theta} && (5.7)\\
&= \prod_{e=1}^{E} \int_{\theta_e} \frac{\Gamma(\alpha^+ + \alpha^-)}{\Gamma(\alpha^+)\Gamma(\alpha^-)}\theta_e^{\alpha^+-1}(1-\theta_e)^{\alpha^--1}\theta_e^{N^e_{\cdot}}(1-\theta_e)^{(N^{\bar{e}}_{\cdot})}\mathrm{d}\theta_e && (5.8)\\
&= \prod_{e=1}^{E} \int_{\theta_e} \frac{\Gamma(\alpha^+ + \alpha^-)}{\Gamma(\alpha^+)\Gamma(\alpha^-)}\theta_e^{N^e_{\cdot}+\alpha^+-1}(1-\theta_e)^{(N^{\bar{e}}_{\cdot})+\alpha^--1}\mathrm{d}\theta_e && (5.9)\\
&= \prod_{e=1}^{E} \frac{\Gamma(\alpha^+ + \alpha^-)}{\Gamma(\alpha^+)\Gamma(\alpha^-)}\frac{\Gamma(N^e_{\cdot}+\alpha^+)\Gamma(N^{\bar{e}}_{\cdot}+\alpha^-)}{\Gamma(N+\alpha^+ +\alpha^-)}\times && (5.10)\\
&\qquad \int_{\theta_e} \frac{\Gamma(N+\alpha^+ +\alpha^-)}{\Gamma(N^e_{\cdot}+\alpha^+)\Gamma(N^{\bar{e}}_{\cdot}+\alpha^-)}\theta_e^{N^e_{\cdot}+\alpha^+-1}(1-\theta_e)^{(N^{\bar{e}}_{\cdot})+\alpha^--1}\mathrm{d}\theta_e\\
&= \prod_{e=1}^{E} \frac{\Gamma(\alpha^+ + \alpha^-)}{\Gamma(\alpha^+)\Gamma(\alpha^-)}\frac{\Gamma(N^e_{\cdot}+\alpha^+)\Gamma(N^{\bar{e}}_{\cdot}+\alpha^-)}{\Gamma(N+\alpha^+ +\alpha^-)} && (5.11)\\
&\propto \prod_{e=1}^{E} \frac{\Gamma(N^e_{\cdot}+\alpha^+)\Gamma(N^{\bar{e}}_{\cdot}+\alpha^-)}{\Gamma(N+\alpha^+ +\alpha^-)} && (5.12)
\end{aligned}
$$

We start by factorizing the distribution into independent components, specific to each event type e in term 5.7, and integrate over possible values for each θ_e independently. We continue by substituting the definitions of the Binomial distribution and the Beta distribution in term 5.8[1], and then proceed with the integration procedure[2]. The realization probability of a particular event type e is finally represented by the Beta-Binomial distribution, as defined in term 5.11. During learning, we will sample from the non-normalized Beta-Binomial, as given in term 5.12, which we will call BBM_e

$$\frac{\Gamma(N^e_{\cdot}+\alpha^+)\Gamma(N^{\bar{e}}_{\cdot}+\alpha^-)}{\Gamma(N+\alpha^+ +\alpha^-)} \qquad BBM_e. \tag{5.13}$$

Intuitively, the posterior probability generating an event type e given the data and the hyperparameters is proportional to the number of times e was observed/not observed as a label in the data ($N^e_{\cdot}/N^{\bar{e}}_{\cdot}$) plus our prior intuition about how often event types should be realized (α^+) and not realized (α^-), as encoded in the hyperparameters.

Analogously, we can integrate out the parameters of the event type-specific participant distributions φ^p_e. By considering only the relevant terms from Equation 5.2, the Binomial parameters $\boldsymbol{\varphi}$ can be integrated out using the same procedure as above.

[1]The Gamma function $\Gamma(\cdot)$ is part of the definition of the Beta distribution, and is an extension of the factorial to real numbers.

[2]

- In term 5.10 we add the term $\left[\frac{\Gamma(N^e_{\cdot}+\alpha^+)\Gamma(N^{\bar{e}}_{\cdot}+\alpha^-)}{\Gamma(N+\alpha^+ +\alpha^-)}\frac{\Gamma(N+\alpha^+ +\alpha^-)}{\Gamma(N^e_{\cdot}+\alpha^+)\Gamma(N^{\bar{e}}_{\cdot}+\alpha^-)}\right] = 1$.
- To get from term 5.10 to term 5.11 we use the fact that $\left[\int_{\theta_e} \frac{\Gamma(N+\alpha^+ +\alpha^-)}{\Gamma(N^e_{\cdot}+\alpha^+)\Gamma(N^{\bar{e}}_{\cdot}+\alpha^-)}\theta_e^{N^e_{\cdot}+\alpha^+-1}(1-\theta_e)^{(N^{\bar{e}}_{\cdot})+\alpha^--1}\mathrm{d}\theta_e\right] = 1$

$$\int_{\boldsymbol{\varphi}} \prod_{e=1}^{E} \prod_{p=1}^{P} P(\varphi_e^p|\beta^+, \beta^-) P(\kappa_p^e|\varphi_e^p) \mathrm{d}\boldsymbol{\varphi} \tag{5.14}$$

$$\propto \prod_{e=1}^{E} \prod_{p=1}^{P} \frac{\Gamma(N_{\cdot}^{e,p} + \beta^+)\Gamma(N_{\cdot}^{e,\bar{p}} + \beta^-)}{\Gamma(N_{\cdot}^{e} + \beta^+ + \beta^-)} \tag{5.15}$$

The realization probability of a participant type p in an event type e is now represented using the Beta-Binomial distribution, which we will call BBM_p^e

$$\frac{\Gamma(N_{\cdot}^{e,p} + \beta^+)\Gamma(N_{\cdot}^{e,\bar{p}} + \beta^-)}{\Gamma(N_{\cdot}^{e} + \beta^+ + \beta^-)} \qquad BBM_p^e \tag{5.16}$$

Integrating out Multinomial Parameters $\boldsymbol{\vartheta}$ and $\boldsymbol{\varpi}$

The process described above can be generalized straightforwardly to the Multinomial case. We will now integrate over Multinomial parameters which are drawn from their conjugate prior distribution, the Dirichlet distribution. The resulting term is called a Dirichlet-Compound-Multinomial distribution (DCM), which is the multinomial generalization of the Beta-Binomial distribution, and is analogously characterized solely by the sufficient statistics in the data and the Dirichlet parameters. We go through an example, integrating over the event type language model parameters $\boldsymbol{\vartheta}$. The relevant terms from Equation 5.2 are $\prod_e P(\boldsymbol{\vartheta}_e|\boldsymbol{\gamma}_e)$ and $\prod_d \prod_e P(\mathbf{w_{d,e}}|\boldsymbol{\vartheta}_i)$. We will further need to iterate over all terms v in our event vocabulary V_η.

$$\int_{\boldsymbol{\vartheta}} \prod_{e=1}^{E} P(\boldsymbol{\vartheta}_e|\boldsymbol{\gamma}_e) \prod_{d=1}^{D} P(\mathbf{w}_{d,e}|\boldsymbol{\vartheta}_e) \mathrm{d}\boldsymbol{\vartheta} \tag{5.17}$$

$$= \prod_{e=1}^{E} \int_{\boldsymbol{\vartheta}_e} P(\boldsymbol{\vartheta}_e|\boldsymbol{\gamma}_e) \prod_{d=1}^{D} P(\mathbf{w}_{d,e}|\boldsymbol{\vartheta}_e) \mathrm{d}\boldsymbol{\vartheta}_e \tag{5.18}$$

$$= \prod_{e=1}^{E} \int_{\boldsymbol{\vartheta}_e} \frac{\Gamma(\sum_v \gamma_e^v)}{\prod_v \Gamma(\gamma_e^v)} \prod_v \boldsymbol{\vartheta}_{e,v}^{N_v^e} \boldsymbol{\vartheta}_{e,v}^{\gamma_e^v - 1} \mathrm{d}\boldsymbol{\vartheta}_e \tag{5.19}$$

$$= \prod_{e=1}^{E} \int_{\boldsymbol{\vartheta}_e} \frac{\Gamma(\sum_v \gamma_e^v)}{\prod_v \Gamma(\gamma_e^v)} \prod_v \boldsymbol{\vartheta}_{e,v}^{N_v^e + \gamma_e^v - 1} \mathrm{d}\boldsymbol{\vartheta}_e \tag{5.20}$$

$$= \prod_{e=1}^{E} \frac{\Gamma(\sum_v \gamma_e^v)}{\prod_v \Gamma(\gamma_e^v)} \frac{\prod_v \Gamma(N_v^e + \gamma_e^v)}{\Gamma(\sum_v N_v^e + \gamma_e^v)} \times \tag{5.21}$$

$$\int_{\boldsymbol{\vartheta}_e} \frac{\Gamma(\sum_v N_v^e + \gamma_e^v)}{\prod_v \Gamma(N_v^e + \gamma_e^v)} \prod_v \boldsymbol{\vartheta}_{e,v}^{N_v^e + \gamma_e^v - 1} \mathrm{d}\boldsymbol{\vartheta}_e$$

$$= \prod_{e=1}^{E} \frac{\Gamma(\sum_v \gamma_e^v)}{\prod_v \Gamma(\gamma_e^v)} \frac{\prod_v \Gamma(N_v^e + \gamma_e^v)}{\Gamma(\sum_v N_v^e + \gamma_e^v)} \tag{5.22}$$

$$\propto \prod_{e=1}^{E} \frac{\prod_v \Gamma(N_v^e + \gamma_e^v)}{\Gamma(\sum_v N_v^e + \gamma_e^v)} \tag{5.23}$$

The likelihood of observed words $\mathbf{w}$ under the current event labeling e is now computed using the Dirichlet-Compound-Multinomial, which we will call $DCM(e;\boldsymbol{\gamma}_e)$

$$\frac{\prod_{v=1}^{|V_\eta|}\Gamma(N_v^e+\gamma_e^v)}{\Gamma(\sum_{v=1}^{|V_\eta|}N_v^e+\gamma_e^v)} \qquad DCM(e;\boldsymbol{\gamma}_e) \tag{5.24}$$

Exactly analogously, we define the probability of an observed participant description consisting of words from the participant vocabulary V_ξ under the participant label p as $DCM(p;\boldsymbol{\delta}_p)$

$$\frac{\prod_{v=1}^{|V_\xi|}\Gamma(N_v^p+\delta_p^v)}{\Gamma(\sum_{v=1}^{|V_\xi|}N_v^p+\delta_p^v)} \qquad DCM(p;\boldsymbol{\delta}_p) \tag{5.25}$$

Again, our final formula have very intuitive interpretations: The probability of all our observed event descriptions $\mathbf{w}$ given their current labeling is proportional to the number of times each term $v \in V_\eta$ was labeled as each event type e, plus our prior intuition about how probable term v should be under label e, as encoded in our hyperparameters. The same holds for observed participant descriptions from $v \in V_\xi$ under all participant labels p.

The Simplified Posterior

After the integration process, all Binomial parameters and Multinomial parameters disappear from the full posterior as defined in Equation 5.2. While we can integrate out the parameters of the Multinomials and Binomials, this is not possible for the parameters of the Generalized Mallows Model $\boldsymbol{\rho}$. Consequently, we will explicitly need to sample event type-specific GMM parameters from the conjugate prior distribution of the GMM $\prod_e^E \rho_e \sim GMM_0(\rho_0,\nu_0)$, and we need to sample an event ordering for each document $\prod_d \mathbf{v_d} \sim GMM(\boldsymbol{\rho})$. The full posterior from Equation 5.2 can now be simplified to

$$\begin{aligned}&P(\boldsymbol{\tau},\boldsymbol{\kappa},\mathbf{v},\boldsymbol{\rho};\alpha^+,\alpha^-,\beta^+,\beta^-,\boldsymbol{\gamma},\boldsymbol{\delta},\rho_0,\nu_0|\mathbf{D})\\ &\propto\prod_e\frac{\Gamma(N_\cdot^e+\alpha^+)\Gamma(N_\cdot^{\bar{e}}+\alpha^-)}{\Gamma(N+\alpha^++\alpha^-)}\prod_p\frac{\Gamma(N_\cdot^{e,p}+\beta^+)\Gamma(N_\cdot^{e,\bar{p}}+\beta^-)}{\Gamma(N_\cdot^e+\beta^++\beta^-)}\\ &\prod_e\frac{\prod_{w=1}^{|V_\eta|}\Gamma(N_w^e+\gamma_e^w)}{\Gamma(\sum_{w=1}^{|V_\eta|}N_w^e+\gamma_e^w)}\prod_p\frac{\prod_{v=1}^{|V_\xi|}\Gamma(N_v^p+\delta_p^v)}{\Gamma(\sum_{v=1}^{|V_\xi|}N_v^p+\delta_p^v)}\\ &P(\boldsymbol{\rho}|\rho_0,\nu_0)P(\mathbf{v}|\boldsymbol{\rho}).\end{aligned} \tag{5.26}$$

Replacing the explicit formula with the notation introduced above, we get

$$\begin{aligned}&P(\boldsymbol{\tau},\boldsymbol{\kappa},\mathbf{v},\boldsymbol{\rho};\alpha^+,\alpha^-,\beta^+,\beta^-,\boldsymbol{\gamma},\boldsymbol{\delta},\rho_0,\nu_0|\mathbf{D})\\ &\propto\prod_e\Big[BBM_e\prod_p\Big[BBM_p^e\Big]\Big]\\ &\prod_e\Big[DCM(e;\boldsymbol{\gamma}_e)\Big]\prod_p\Big[DCM(p;\boldsymbol{\delta}_p)\Big]\\ &GMM_0\ \ GMM.\end{aligned} \tag{5.27}$$

5.3 Defining the Resampling Steps

In the previous section we defined and simplified the joint posterior distribution of all hidden variables in our model, given the data. Since it is intractable to sample from this distribution directly, we will use Gibbs sampling for approximate inference.

Gibbs sampling breaks the joint distribution down into independent factors for each hidden variable. Each hidden variable is then resampled in turn conditioned on the current valued of all other hidden variables, but *excluding* the previous value of the variable which is currently resampled.

In this section, we will define a posterior distribution for each of the hidden variables $\boldsymbol{\tau}, \boldsymbol{\kappa}, \mathbf{v}, \boldsymbol{\rho}$ in our model. Note that whenever a value of one particular hidden variable is updated, this will affect only a subset of the other types hidden variables. We will consider only those terms which directly affect the posterior distribution over possible values for each hidden variable.

For each type of hidden variable we will compute a posterior probability distribution over the possible values the variable can take. We compute the posterior probability under the hypothesis that the variable takes a particular value, by temporarily assigning the variable this value. The sufficient statistics of our data will thus change with each hypothesis. Table 5.2 summarizes our notation of the sufficient statistics. N_{var}^{var} stands for the counts of a particular observation in our data. A horizontal bar over any superscript means the number of times that observation was *not* present in a document (the failure case for the Binomial distributions). A dot in any variable position means the sum over all possible values is taken. The apostrophe means that the label of the currently resampled variable is excluded from the counts.

5.3.1 Computing the Document Likelihood

We start by defining the document likelihood, which is the probability of the observed words in a document d given the current labeling of the data as assigned by the model. We need the document likelihood as a function of our parameters as a factor of our posterior distributions for most hidden variables.

Given the definition of Gibbs resampling steps, the probability of a particular value for a hidden variable (words $\mathbf{w_d}$ in document d in the definition below) is conditioned on the current values of all other hidden variables except the label of the variable we are currently resampling (indicated by $\mathbf{w^{-d}}$)

$$\begin{aligned}
&P(\mathbf{w}_d|\mathbf{w}^{-d}, \boldsymbol{\tau}, \boldsymbol{\kappa}, \mathbf{v}, \boldsymbol{\gamma}, \boldsymbol{\delta}) \\
&\propto \prod_e^E \frac{\prod_v^{|V_\eta|} \Gamma(N_v^{e'} + \gamma_v^e)}{\Gamma(\sum_v N_v^{e'} + \gamma_v^e)} \prod_p^P \frac{\prod_v^{|V_\xi|} \Gamma(N_v^{p'} + \delta_v^p)}{\Gamma(\sum_v N_v^{p'} + \delta_v^p)} \\
&= \prod_e DCM(e; \boldsymbol{\gamma}_e) \prod_p DCM(p; \boldsymbol{\delta}_p). \qquad (5.28)
\end{aligned}$$

Note that the document likelihood consists of two independent factors, one for event word likelihood, and one for participant word likelihood. Depending on which type of hidden variable will be resampled, we will only need to consider one of these factors in the following definitions, since the other one will not be affected by the change.

5.3.2 Resampling the Hidden Variables

We continue by defining the Gibbs update steps for each hidden variable in our model. Most posteriors will consist of two parts: (1) the probability of the parameters, and (2) the likelihood of the observed data given the parameters as defined in 5.3.1.

Resampling Participant Types

We resample the label of the participant j within event i, in ESD d. We create a distribution over all possible labelings l, by temporarily assigning j to the hypothesized label l and computing the following posterior under this assumption[3]:

$$\begin{aligned}
&P(\kappa^j_{d,i} = l|\boldsymbol{\kappa}^{-d,i,j}, \boldsymbol{\tau}, \mathbf{w}^{-d,i,j}, \beta^+, \beta^-, \boldsymbol{\delta}) \\
=&P(\kappa^{d,i}_j = l|...)P(\mathbf{w}^j_{d,i}|\kappa^{d,i}_j = l, ...) \\
\propto&\prod_e^E \prod_p^P \frac{\Gamma(N^{e,p'}_{\cdot} + \beta^+)\Gamma(N^{e,\bar{p}'}_{\cdot} + \beta^-)}{\Gamma(N^e_{\cdot} + \beta^+ + \beta^-)} \times DCM(p; \boldsymbol{\delta}_p) \\
\propto&\prod_e^E \prod_p^P BBM^p_e \times DCM(p; \boldsymbol{\delta}_p).
\end{aligned} \tag{5.29}$$

The first factor of the equation stands for the probability of our parameters given the proposed label l, and it favors participant types l which have been observed frequently with event type i in the whole corpus. The second factor is the document likelihood given the proposed label l, and favors participant types which have occurred often with the observed words $\mathbf{w^j_{d,i}}$ over the whole corpus.

We resample participant j from the resulting multinomial distribution over all possible labels, and update j's label, and all relevant sufficient statistics, accordingly.

Resampling Event Types

We resample the event type of event i in ESD d. Resampling an event type affects the event BBM and the participant BBM, since event-participant co-occurrence will change. We create a probability distribution over all possible labelings k, by temporarily assigning the hypothesized label k to event i and

[3]This means that the counts $N^{e,p'}$ and $N^{e,\bar{p}'}$ change whenever $l == p$ and $i == e$. Equivalently, the DCM components for hypothesized label l and j's old label change because the word-participant type co-occurrence counts change with each hypothesis. The same reasoning applies to all BBMs and DCMs whenever a hypothesis for an update of a hidden variable affects the current corpus labeling.

computing the following posterior under this assumption:

$$\begin{aligned}
&P(\tau_{d,i} = k|\boldsymbol{\tau}^{-d,i}, \boldsymbol{\kappa}^{-d,i,\cdot}, \mathbf{w}^{-d,i}\alpha^{+}, \alpha^{-}, \beta^{+}, \beta^{-}, \boldsymbol{\gamma}) \\
=&P(\tau_{d,i} = k|...)P(\boldsymbol{\kappa}^{d,i}|\tau_{d,i} = k, ...)P(\mathbf{w}_{d,i}|\tau_{d,i} = k, ...) \\
\propto&\prod_{e}^{E} \frac{\Gamma(N^{k'} + \alpha^{+})\Gamma(N^{\bar{k}'} + \alpha^{-})}{\Gamma(N + \alpha^{+} + \alpha^{-})} \times \prod_{p}^{P} BBM_e^p \times DCM(e, \boldsymbol{\gamma}_e) \\
\propto&\prod_{e}^{E} BBM_e \times \prod_{p}^{P} BBM_e^p \times DCM(e, \boldsymbol{\gamma}_e) \qquad (5.30)
\end{aligned}$$

We can interpret the three factors as, given our current corpus labeling, favoring generally frequently observed event types, favoring event types which co-occurred often with the participant types realized in $\boldsymbol{\kappa}^{d,i}$, and favoring event types with which the observed words $\mathbf{w}_{d,i}$ have occurred often over the whole corpus, respectively.

We sample the new label for event i from the resulting multinomial over all possible labels k, and update i's label and the relevant sufficient statistics accordingly.

Resampling Ordering

Since ordering in the GMM is parameterized individually for each element by providing element-specific inversion counts, we can resample the position of each event type in a document d individually. We successively and independently resample the inversion count of each event type $e \in \{1..E-1\}$[4] for document d:

$$\begin{aligned}
&P(v_{d,e} = v|\mathbf{v}^{-d,e}, \boldsymbol{\tau}, \boldsymbol{\kappa}^{-d}, \mathbf{w}^{-d}, \rho_e, \boldsymbol{\gamma}) \\
=&P(v_{d,e} = v|\rho_e) \times P(\boldsymbol{\kappa}^{d,\cdot}|v_{d,e} = v, ...)P(\mathbf{w}_{d,\cdot}|v_{d,e} = v, ...) \\
\propto&GMM_e(v; \rho_e) \times \prod_{e}\prod_{p} BBM_e^p \times DCM(e; \boldsymbol{\gamma}_e) \qquad (5.31)
\end{aligned}$$

where GMM_e is evaluated using Equation 3.16. We compute a probability distribution over all possible values $v = \{0..E-e\}$[5] and resample the new inversion count for event type e from the resulting distribution.

Note that a full ordering over all event types is sampled for every document. It is irrelevant at this point which event types are realized in the document and which are not.

When resampling an event type ordering, the event realizing words in the document might be assigned different labels, since the order of the realized event types might change, and the event-participant co-occurrences might change for the same reason. Hence we consider three factors in the posterior distribution, as defined above. While the GMM factor generally favors low event inversion counts (attenuated by the dispersion parameter ρ_e), the Beta-Binomial factor BBM_e^p favors orderings which lead to globally frequently observed event type-

[4]Recall that, given the identity ordering as canonical ordering, the inversion count for event type E is always 0.

[5]Given the identity ordering as canonical ordering of E elements, at most $E-e$ elements i such that $i > e$ can possibly occur before each element e.

participant type co-occurrences in d. The document likelihood, as before, is maximized when the proposed event labeling for each realized event type i in d has been observed frequently with the observed words $\mathbf{w}_{i,d}$ over the whole corpus.

Resampling GMM parameters

We resample the GMM parameter vector $\boldsymbol{\rho}$ element-wise from the GMM prior distribution $GMM_0(v_{e,0}, \nu_0)$. The parameter ν_0, the number of pseudo trials is computed as

$$\nu_0 = N + \nu_0 \tag{5.32}$$

The number of previously encountered inversions for event type e is computed individually for each event type

$$v_{e,0} = \frac{\sum_d v_{d,e} + v_{e,0}\nu_0}{N + \nu_0}, \tag{5.33}$$

where N is the number of documents in our corpus. We resample

$$P(\rho_e^{t+1}|...) \propto GMM_0\Big(\rho_e^t; \nu_0, v_{e,0}\Big), \tag{5.34}$$

where we use Equation 3.17 to evaluate GMM_0, and ρ_e^t is the previous value of parameter ρ_e. In contrast to the previously defined posteriors, which are categorical distributions over possible labelings or inversion counts, the resulting posterior distribution over possible values for ρ_e is continuous and not straightforward to sample from, since the normalizing constant is unknown. We use slice sampling, as defined in Section 3.5.2 for sampling a new value from the posterior distribution.

Note that resampling GMM parameters does not have a direct influence on our corpus labeling, which is why there is no document likelihood factor in the posterior equation.

5.4 Posterior Regularization

Regneri et al. (2011) encode a "one sense per participant" constraint in their model for participant clustering. This means that they assign all token-identical participant descriptions the same participant type. We integrate this constraint into our model as well, through *posterior regularization*.

Generally, we aim to learn sparse posterior distributions, by specifying hyperparameters which encourage parameterizations which result in sparse hidden variable distributions. Graça et al. (2009) propose a form of posterior regularization which encourages sparsity *directly* in the posterior distribution, by specifying linear constraints on the posterior distribution of each hidden variable $z \in \phi$. A desired distribution $q(z)$ is defined for each hidden variable, and the original objective, maximizing the log likelihood $p(D|\phi)$, is regularized as follows

$$p(D|\phi) + \epsilon * \sum_{z\in\phi} KL\big(q(z)||p_\phi(z)\big), \tag{5.35}$$

where $KL\big(q(z)||p_\phi(z)\big)$ is the Kullback-Leibler divergence between the desired distribution $q(z)$ of a hidden variable z, and its actual distribution $p_\phi(z)$. ϵ is a parameter for the influence of the regularization component. A small KL-distance will minimize the penalization term, and thus parameter distributions close to the desired distributions will be preferred. By specifying a sparse set of desired distributions $\mathcal{Q}_\phi$ the posterior distributions of each variable will tend to be sparse too.

While Graça et al introduce their posterior regularization technique within the framework of variational inference, we adapt the idea and re-formulate it for the Bayesian sampling setting. We can define our notion of sparse distributions as distributions which concentrate their probability mass over one single component. We can thus reformulate the KL-distance in the above formula as:

$$\arg\max_l P(z = l|D). \tag{5.36}$$

Here, we are only interested in one particular $z \in \phi$, namely labels of participant words with types $p \in \{1...P\}$, given observed words w_p. Of the posterior distributions defined above, only the distribution over participant types is influenced by this factor. We will regularize this posterior distribution, by taking into account in our Gibbs updating step how the regularization term in 5.36 changes given a hypothesized participant labeling. We use the regularized objective as the posterior probability distribution over participant labels, and modify term 5.29 accordingly:

$$P(\kappa_j^{d,i} = l|...) \propto \prod_e^E \prod_p^P BBM_e^p \times DCM(p; \boldsymbol{\delta}_p) \times P(\kappa_j^{d,i} = l|w_{d,i}^j)^\epsilon. \tag{5.37}$$

The scaling factor makes the posterior distribution sharper, as it boosts the probability of labels which are already likely under the distribution. Eventually, all occurences of one term t in the vocabulary will be labeled as the same participant type l. Once the model has converged to this situation, with a high value for ϵ, we practically take the *maximum a posteriori*, the most likely, label for t.

Coming back to our motivation, we include posterior regularization in our model, in order to trigger, for each participant-realizing term in the vocabulary, a very strong tendency to assign all mentions of that term the same participant type. This essentially encodes the "one sense per participant" constraint. We will start with a low value for ϵ in order to ensure flexibility in the early learning phase, e.g. to allow assignment of different terms to the same participant type, and raise its value continuously during the inference process.

5.5 Summary

In this chapter, we derived an inference algorithm for the script model introduced in Chapter 4. We reversed the generative story, and made the independence assumptions included in the generative story explicit. Using the independence assumptions, we factorized the joint probability of all hidden variables in our model into independent components. We integrated over the parameters of the Binomial distributions and the Multinomial distributions in our model.

Let us define the remaining set of hidden variables $\boldsymbol{\phi} = [\boldsymbol{\tau}, \boldsymbol{\kappa}, \mathbf{v}, \boldsymbol{\rho}]$, and the hyperparameters $\boldsymbol{\psi} = [\alpha^{+}, \alpha^{-}, \beta^{+}, \beta^{-}, \boldsymbol{\gamma}, \boldsymbol{\delta}, \rho_0, \boldsymbol{\nu}_0]$.

Our general Bayesian procedure of estimating the probability of our parameters $\boldsymbol{\phi}$ from prior belief $P(\boldsymbol{\phi}|\boldsymbol{\psi})$ and the likelihood of the data $P(\mathbf{D}|\boldsymbol{\phi})$:

$$P(\boldsymbol{\phi}; \boldsymbol{\psi}|\mathbf{D}) \propto \frac{P(\boldsymbol{\phi}|\boldsymbol{\psi})P(\mathbf{D}|\boldsymbol{\phi})}{\int_{\boldsymbol{\phi}} P(\mathbf{D}|\boldsymbol{\phi})P(\boldsymbol{\phi})\mathrm{d}\boldsymbol{\phi}}. \tag{5.38}$$

The computation of the posterior above, using the joint probability of all parameters $\boldsymbol{\phi}$, is intractable. In particular, the integration over all possible parameters in the denominator above is not possible to compute. We thus use Gibbs sampling for approximate inference. We defined separate resampling steps for every hidden variable in $\boldsymbol{\phi}$, under the constraints imposed by the update procedure in the Gibbs sampler. Using these definitions, we can define a posterior distribution for each hidden variable over all possible values it can take, and update the label from this distribution. In the actual learning procedure we will repeatedly iterate over all hidden variables in $\boldsymbol{\phi}$ and update them in turn conditioned on all other hidden variables $\boldsymbol{\phi}'$ until the sampler converges.

We also briefly introduced the posterior regularization framework. We use posterior regularization in order to impose additional constraints on our participant type-specific language models, which encourage all mentions of a term in the participant vocabulary to be labeled with the same participant type.

In the next chapter, we will present an extension to the model, and the corresponding changes to the inference procedure.

Chapter 6

Extending the Script Model with Informed Prior Knowledge

In this chapter, we present an extension to the proposed script model. One challenge we are faced with is the very limited size of the corpora available for training and testing. For completely unsupervised learning settings, like the one presented, sufficient amount of data is inevitable for the model to induce underlying structure from the data, without any external cues. We introduce a way of alleviating this problem by including knowledge about word similarities, which will guide the inference process, in a completely unsupervised way.

Regneri et al. (2011), who we follow closely in our task definition, uses WordNet-based semantic similarity, in particular Lin's similarity measure (Lin, 1998), as one factor in their model for participant clustering. We will also use WordNet-based similarity scores, but obtain them in a different way, because the way we integrate prior knowledge into our model poses restrictions on the way similarity scores can be computed, as we explain below.

We start by describing how we obtain semantic knowledge in Section 6.1, and continue by formalizing the modified generative story in Section 6.2. Finally, we extend the inference procedure by defining posterior probability distributions for the additional hidden variables in Section 6.3.

6.1 Defining the Prior Knowledge

In order to alleviate the problem of a limited amount of available data, we augment our model by providing it with automatically induced external cues to guide the inference process. In particular, we inject knowledge of semantic similarity, automatically obtained from WordNet (Fellbaum, 1998), into our model. From this knowledge we learn informed prior parameters for our language models.

Intuitively, we would like to be able to "tie" the probabilities of semantically similar words together, within each cluster we infer. Returning to the example of the `Cooking Pasta` scenario, we would like words similar to "cook" and "boil" to

	Food	Order	Table	Counter	Menu	Drink	Board	Meal
Food	13	0	3	0	3	3	3	2
Order	0	32	0	0	0	0	0	0
Table	3	0	20	6	3	0	8	0
Counter	0	0	6	22	0	0	0	0
Menu	3	0	3	0	15	0	3	0
Drink	3	0	0	0	0	13	0	0
Board	3	0	8	0	3	0	25	0
Meal	2	0	0	0	0	0	0	6

Table 6.1: Excerpt from the participant covariance matrix Σ_ξ of the `Fastfood` scenario, based on the most frequent terms in the vocabulary.

have a high realization probability in a cluster corresponding to the event type "boiling water", while we want the same set of words to have a low probability in most other clusters.

The basic Dirichlet distribution assumes independence between the components of the parameterizations it defines a distribution over. In our case, this leads to the unrealistic assumption, that the realization probabilities of all terms in the vocabulary are independent of each other within each cluster. We will relax this assumption by introducing correlation among parameters for semantically similar terms. We use the variance-covariance matrix of a multivariate Normal distribution to encode these correlations.

6.1.1 Obtaining Word Similarities from WordNet

We use WordNet to obtain a semantic similarity score for each pair of words in our vocabulary. Since we work on limited domains, we define a subset of WordNet based on which we compute the similarity scores. The sub domain is defined as all WordNet synsets that any word in our vocabulary is a member of, plus the hypernym sets of all these synsets. We create a feature vector for each term v_i, $f(v_i)$, with dimensions n corresponding to the synsets of our WordNet sub domain, as follows:

$$f(v_i)_n = \begin{cases} 1 & \text{if any sense of } v_i \in \text{synset } n \\ 0 & \text{otherwise} \end{cases}$$

The similarity of two terms v_i and v_j is then defined as the dot product of their respective feature vectors $f(v_i) \cdot f(v_j)$.

Recall that we treat the event vocabulary V_η and the participant vocabulary V_ξ completely separately, such that we will obtain two independent sets of similarity scores. We use the two sets to define the respective covariance matrices Σ_η, capturing similarity between event-describing terms, and Σ_ξ, capturing similarity between participant-describing terms. They have V_η and V_ξ dimensions, corresponding to the number of terms in the event and participant vocabulary, respectively. Each cell (i, j) contains the similarity between terms v_i and v_j as defined above. Tables 6.1 and 6.2 show two example excerpts of the respective covariance matrix for events and participants for the `Eating in a fastfood restaurant` scenario. It can be seen that the event term similarity

	eat	order	pay	wait	walk	go	make
eat	26	0	5	0	0	0	0
order	0	40	0	0	0	0	1
pay	5	0	66	0	0	17	25
wait	0	0	0	19	0	10	0
walk	0	0	0	0	31	12	4
go	0	0	17	10	12	124	29
make	0	1	25	0	4	29	202

Table 6.2: Excerpt from the event covariance matrix Σ_η of the `Fastfood` scenario, based on the most frequent terms in the vocabulary.

scores are much denser than the participant term similarity scores, especially for very generic verbs which have many senses, such as “make” or “go”.

We will use each matrix as the variance-covariance matrix of a multivariate normal distribution. Covariance matrices of Multivariate Normal distributions need to be positive semidefinite. Computing the value of each cell (i, j) as the dot product of the feature vectors of elements i and j guarantees positive semidefiniteness of the resulting matrix.

6.2 The Modified Generative Process

In order to relax the independence assumption inherent in the Dirichlet distribution, we add another level to the model hierarchy: instead of specifying priors $Dirichlet(\boldsymbol{\gamma})$ and $Dirichlet(\boldsymbol{\delta})$ directly, we sample them for each event type e and participant type p from a Logistic Normal distribution (see Section 3.4).

Since the event vocabulary V_η and participant vocabulary V_ξ are completely separate in our model, the construction of the language model priors works independently, but equivalently, for the event and the participant component. For illustration, we will describe the modified generative process for event type language models below. A plate diagram representation of the full model is displayed in Figure 6.1, and the formalized generative story of the model extension is displayed in Figure 6.2.

In the extended model version each event type e is assigned an individual associated Dirichlet prior vector $\boldsymbol{\gamma}^e$. The dimensions of $\boldsymbol{\gamma}^e$ correspond to the terms in the event vocabulary $v \in V_\eta$, such that for each term an individual parameter (pseudo count) is constructed for each event type. The resulting priors are thus non-symmetric, or informed.

The prior vectors $\boldsymbol{\gamma}$ are generated as follows. For each event type e, we draw a parameter vector $\boldsymbol{\eta}_e$ from the logistic normal $\boldsymbol{\eta}_e \sim N(\Sigma_\eta, \boldsymbol{\mu})$ with mean $\boldsymbol{\mu} = 0$. The covariance matrix Σ_η is defined as described in Section 6.1.1, and is globally defined across all event types e. The dimensions of $\boldsymbol{\eta}_e$ correspond to the words of the event vocabulary, and the correlations among words encoded in Σ_η are reflected in $\boldsymbol{\eta}_e$. The vector $\boldsymbol{\eta}_e$ is normalized using logistic transformation, to yield the Dirichlet parameter vector $\boldsymbol{\gamma}^e$.

The Dirichlet parameter vectors $\boldsymbol{\delta}^p$ are generated with the exact same procedure from $N(\Sigma_\xi, 0)$, by first sampling $\boldsymbol{\xi}_p \sim N(\Sigma_\xi, 0)$ for each participant type p, and subsequent normalization to yield $\boldsymbol{\delta}_p$. Σ_ξ is the globally defined co-

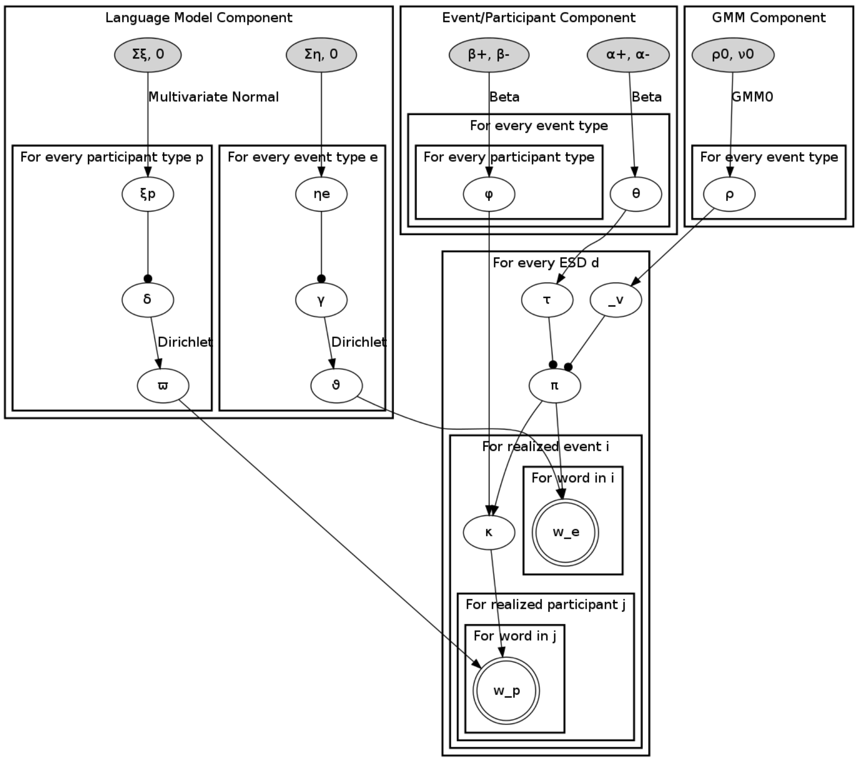

Figure 6.1: The plate diagram for the full script model. Dirichlet parameters for event language models and participant language models are now drawn from the type-specific Multivariate Normal distribution. Shaded circles indicated manually optimized hyperparameters. Double circles indicate observed variables, and round arrow heads indicate deterministic computations.

variance matrix encoding semantic similarity between words in the participant vocabulary V_ξ.

6.3 The Modified Inference Procedure

First note, that the inference procedure as described in the previous chapter stays exactly the same. The only difference is that the Dirichlet parameters $\boldsymbol{\gamma}_e$ and $\boldsymbol{\delta}_p$ are not manually specified, but learnt. After extending the model as described above, there are two additional sets of hidden variables in the model, namely parameter vectors for the Dirichlet parameters for each event type e, $\boldsymbol{\eta}_e$, and for each participant type p, $\boldsymbol{\xi}_p$. We will explain the inference procedure taking the event prior parameters as an example. The update procedure for the participant prior parameters is exactly equivalent. We resample the vectors for each event type as follows:

$$P(\boldsymbol{\eta}_e|\Sigma_\eta, \mathbf{w}) \propto N(\boldsymbol{\eta}_e|\Sigma_\eta) DCM(e; (\boldsymbol{\gamma}_e \leftarrow \boldsymbol{\eta}_e)) \tag{6.1}$$

Generation of parameters ϑ_e and ϖ_p

```
for event type $e = 1, \dots, E$ do
  $\boldsymbol{\eta}^e \sim N(\Sigma_\eta, 0)$
  for all words $w$ do
    $\gamma^e_w = \exp(\eta^e_w) / \sum_{w'} \exp(\eta^e_{w'})$        [ Dir prior]
  $\vartheta_e \sim Dirichlet(\boldsymbol{\gamma}^e)$        [event lang mod]
for participant type $p = 1, \dots, P$ do
  $\boldsymbol{\xi}^p \sim N(\Sigma_\xi, 0)$
  for all words $w$ do
    $\delta^p_w = \exp(\xi^p_w) / \sum_{w'} \exp(\xi^p_{w'})$        [ Dir prior]
  $\varpi_p \sim Dirichlet(\boldsymbol{\delta}^p)$        [ ptcpt lang mod ]
```

Figure 6.2: The generative story of the modification to the parameter generation procedure for $\boldsymbol{\vartheta}_e$ and $\boldsymbol{\varpi}_p$ to encode word correlations.

The component $(\boldsymbol{\gamma}_e \leftarrow \boldsymbol{\eta}_e)$ indicates that $\boldsymbol{\eta}_e$ is normalized in an intermediate step to yield $\boldsymbol{\gamma}_e$ using the logistic transformation:

$$\gamma^i_e = k * \frac{exp(\eta^i_e)}{\sum_{i'} exp(\eta^{i'}_e)}. \tag{6.2}$$

We introduce a parameter k, which allows us to scale the resulting Dirichlet parameters, and thus to regulate sparsity of the posterior distributions.

Note that when resampling a parameter vector $\boldsymbol{\eta}_e$ for a particular event type e, only the DCM component affecting event type e is relevant. Unlike in the previously defined resampling steps, we need to consider the normalizing constant here, because it depends on the hyperparameter vector $\boldsymbol{\gamma}_e$ we want to learn:

$$DCM(e; (\boldsymbol{\gamma}_e \leftarrow \boldsymbol{\eta}_e)) = \frac{\Gamma(\sum_v \gamma^v_e)}{\prod_v \Gamma(\gamma^v_e)} \frac{\prod_{v=1}^{|V_\eta|} \Gamma(N^e_v + \gamma^v_e)}{\Gamma(\sum_{v=1}^{|V_\eta|} N^e_v + \gamma^v_e)}. \tag{6.3}$$

The posterior distribution over prior parameter vectors, as defined in Equation 6.1, consists of two terms. The first term assigns high probability to parameter vectors which are likely under the Multivariate Gaussian distribution parameterized with the respective WordNet-based covariance matrix. The second term returns the document likelihood of all observed data currently labeled with the class for which the prior is being resampled. While the first term triggers correlating hyperparameter values for semantically similar words, the second term regulates the magnitude of the hyperparameters: a likely hyperparameter vector for class e should have a high prior value for a term t which is often observed with label e in the data, and also for all terms similar to t according to the covariance matrix.

The resulting posterior distribution over possible vectors $\boldsymbol{\eta}_e$ is a continuous distribution over vector-valued variables. Direct sampling from it is intractable, because the normalizing constant is unknown. We use multivariate Slice sampling to sample from this distribution, by successively resampling each

Term v_i	$simSet(v_i)$
select	decide
eat	consume
enter	want, look, enter, count
cup	container, receptacle
food	table, drink, menu, board, meal
counter	table
garbage	trash, container, tray, receptacle

Table 6.3: Examples for *simSets* obtained from the event covariance matrix (top), and the participant covariance matrix (bottom) for the scenario `Eating in a fastfood restaurant`. Whenever observed word i is assigned a new class label we update the components corresponding to all words in the set in w_i's old class and w_i's new class.

component of the vector based on the probability of the full vector under the Multivariate Gaussian.

6.3.1 Efficient Prior Resampling

Taking posterior samples from the Logistic Normal distribution, thus resampling the full prior parameter vector, is computationally expensive. We employ a few heuristics to make updates more efficient.

First, we heuristically define a set $simSet(v_i)$ of closely related terms for each term v_i in a vocabulary. We build this set based on the similarity scores in the covariance matrix by including all terms in the vocabulary whose similarity to term v_i exceeds an empirically determined threshold. We define this threshold as $0.1 * \Sigma_\xi(i,i)$ for each participant term v_i, meaning that any term $v_j \in simSet(v_i)$ must share 10% of all senses of term v_i. For event terms the threshold is defined as $0.3 * \Sigma_\eta(k,k)$, 30% of all senses of term v_k must be shared. As indicated above, similarity scores among event-realizing terms tend to be higher, and less distinctive, than those among participant-realizing terms, such that we set a higher threshold for the former. Examples for the *simSets* of closely related terms we obtain with this method are displayed in Table 6.3.

Whenever the label of an observed word w was changed from the old label x to the new label y in the previous Gibbs update, we resample specifically the components corresponding to w in the parameter vectors for both classes, η_x^w and η_y^w. To encourage the correlations as defined in the covariance matrix, we furthermore resample components η_x^u and η_y^u for each $u \in simSet(w)$, the synonym set of word w. Finally, we renormalize $\boldsymbol{\eta}_x$ and $\boldsymbol{\eta}_y$ in order to update the corresponding Dirichlet parameter vectors $\boldsymbol{\gamma}_x$ and $\boldsymbol{\gamma}_y$.

Intuitively, the component η_{class}^w should increase for the new class of w, since the DCM component should return a high probability for parameter vectors in which component w has an increased value. Increasing values for all words $u \in simSet(w)$ should be triggered through the influence of the covariance matrix. Consequently the corresponding components in the prior vector $\boldsymbol{\gamma}_{class}$ should increase, while all other components decrease, due to re-normalization. Following the same reasoning, relevant components in the prior vector for the class w was previously assigned should decrease.

After every n^{th} iteration of the Gibbs sampler, we fully resample all parameter vectors in order to capture the globally defined correlations from the covariance matrix.

6.4 Summary

One bottleneck for the performance of our model is the limited amount of training data we have available. In this chapter, we have described our approach towards this problem, through an extension to the script model, which allows for incorporation of prior knowledge. We construct word similarity scores from WordNet in a completely unsupervised way, and use them for modeling correlation of probability of semantically related words across all type-specific language models. We described our way of incorporating the prior knowledge into our Bayesian model, and explained the extension of the inference procedure.

The description of our Bayesian script model, and the corresponding inference process is now complete. In the following chapters we will evaluate our model, and provide a quantitative and qualitative analysis of its performance.

Chapter 7

Evaluation

After describing the script model and its inference process, we will now evaluate its performance. We will evaluate our system on three tasks,

1. event cluster induction
2. induction of a scenario-specific canonical event ordering
3. participant cluster induction.

Our evaluation will closely follow the work presented in Regneri et al. (2010) (henceforth R10) and Regneri et al. (2011) (henceforth R11), and we will compare our model directly against the two systems developed in R10 and R11. R10 present a graph-based system for learning event paraphrases and ordering constraints. They collect scenario-specific corpora of ESDs, and create a gold standard for evaluation of both tasks. R11 present a system for participant clustering, which builds on R10, and they construct a gold standard for this task. Both systems are described in detail in Section 2.1. In order to be able to compare our system to the systems presented in R10 and R11, we evaluate our model on the same tasks, and use the same data and the same metrics for evaluation.

We start by giving an overview over the data we use for development and testing in Section 7.1. We describe our experimental setup in Section 7.2, by explaining the data preprocessing procedure, the evaluation metrics, and the gold standard that we use, as well as the setup for the various experiments we run.

7.1 Data

Our model learns event types and participant types from event-sequence descriptions (ESDs), which are explicit instantiations of scripts. Our data consists of a number of corpora, each of which contains a set of ESDs for a particular scenario.

We use data from different sources, which we describe below. A commonality that our resources share is that they are collected through crowd sourcing (e.g. Singh et al. (2002)). We deal with common sense knowledge about frequent every day situations, that every member (of the Western culture) knows about,

Iron Clothes	**Take a bus**
Get wrinkled clothes	Walk to the bus stop
Put it on the ironing board	Wait in line
Go over with hot iron	Get on the bus
Do again with other clothes	Pay for the ticket
Make sure they aren't wrinkled	Seat if there is room to do so
Put iron away	

Cook scrambled Eggs	**Pay with credit card**
Turn on the oven	Slide card through reader
Put pan on oven	Wait for cashier to process
Put a bit of oil in the pan	Sign the receipt
Throw some eggs into the pan	Get my receipt
Stir eggs	Put my card in my wallet
Put scrambled eggs on plate	Leave
Eat eggs	

Table 7.1: One example ESD for each of the four scenarios `Iron clothes`, `Take a bus`, `Cook scrambled eggs` and `Pay with credit card`.

and that, for this reason, is seldom made explicit. One promising way of capturing this implicit knowledge in a general way is to ask a representative number of volunteers to explicitly describe a common scenario-specific event chain. From the variety of obtained descriptions, it is possible to extract a generic idea about the stereotypical scenario-specific event chain.

R10 collect corpora of ESDs for 22 different scenarios of varying complexity, which we will call the R10 corpus. The data was collected via a web experiment through Amazon Mechanical Turk[1]. For each scenario, 25 non-expert annotators were asked to describe the stereotypical sequence of events involved in the scenario in temporal order and "bullet point-style" language. Table 7.1 shows one example ESD for each of the scenarios `Eating in a restaurant`, `Taking a bus`, `Cook scrambled eggs` and `Paying with a credit card`. The R10 corpus was manually postprocessed. Nonsense- or inappropriate ESDs were removed, some spelling mistakes were manually corrected, and ESDs were structurally changed (e.g. by splitting event descriptions including two conjuncted events).

In addition to the corpus they collected, R10 use data from the Open Mind Indoor Common Sense (OMICS) corpus (Gupta and Kochenderfer, 2004). The corpus has a very similar format to the R10 corpus, but is restricted to indoor activities. The corpus was originally collected to provide robots with common sense knowledge. Like the R10 corpus, the OMICS corpus was collected through crowd sourcing. The OMICS corpus contains on average around twice as many ESDs per scenario as the R10 corpus.

Tables 7.2 and 7.3 provide an overview over the scenario types we use for development and testing, respectively, including the size of the respective corpora and the average length of an ESD in event descriptions.

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

Scenario Name	Abbreviation	♯ESDs	Avg len
OMICS corpus			
Answer Doorbell	Doorbell	49	3.59
Do Laundry	Laundry	32	5.69
R10 corpus			
Make Omelet	Omelet	25	6.16
Eat in Restaurant	Restaurant	19	10.4

Table 7.2: The list of scenarios from the R10 and the OMICS corpus used for parameter tuning. The column named ♯ESDs specifies the number of ESDs in the corpus for the respective scenario. The average length of an ESD in event descriptions is given in the rightmost column.

Scenario	Abbreviation	♯ESDs	Avg len
OMICS corpus			
Cook food using microwave	Microwave	59	5.03
Answer the telephone	Telephone	55	4.47
Buy from vending machine	Vending	32	4.53
Make coffee	Coffee	38	5.00
R10 corpus			
Iron clothes	Iron	19	8.79
Make scrambled eggs	Scr. Eggs	20	10.3
Eat in fast food restaurant	Fastfood	15	8.93
Return food (in a restaurant)	Ret. Food	15	5.93
Take a shower	Shower	21	11.29
Take the bus	Bus	19	8.53

Table 7.3: Scenario types from the R10 corpus and the OMICS corpus used as test scenarios in our experiments. The column named ♯ESDs specifies the number of ESDs in the corpus for the respective scenario. The average length of an ESD in event descriptions is given in the rightmost column.

7.2 Experimental Setup

We start by describing the preprocessing steps we apply to our data, the gold standards we use for evaluation, and the evaluation metrics we employ. Finally, we describe the design of our experiments.

Preprocessing

We preprocess our data in a number of ways. First, we filter participant descriptions out of every event description. Regneri et al. (2011) parsed all corpora they used for evaluating their R10 and R11 system, using a dependency parser, specifically trained for the "bullet point-style" phrases our data consists of. On the basis of the resulting dependency parses, they identified all noun phrases in all event descriptions. We use this set of noun phrases, and take each identified noun phrase as one participant description. We thus extract all participant descriptions from the original event description, which leaves us, for each event with the description of the event, usually a verb phrase lacking its arguments, and a set of participant descriptions, each corresponding to one noun phrase. We do not include any information on the ordering of participants or any information about semantic roles in this set.

We remove all pronouns from the data, because we currently do not have any component for dealing with pronoun resolution in our model. We furthermore remove all articles. Finally, we reduce all participant descriptions, as defined above, to their head words.

As an example, the event description "put the pasta into boiling water" would be represented as the description of the event "put into" and the two participant descriptions "pasta" and "water", after the preprocessing step.

Gold Standard

We use the gold standard data sets presented in R10 and R11 in our evaluation.

R10 presents two gold standards, one for the task of event clustering, and one for learning ordering constraints. Both gold standards consist of pairs of event descriptions with a binary annotation. For the event clustering gold standard, 30 event description pairs that their system classified as paraphrases, and 30 completely random pairs were collected. Annotators decided for each pair whether the two phrases are paraphrases, describing the same event type. Similarly, for the ordering gold standard 30 pairs of event descriptions (a, b) that were classified as *a happens before b* by their system were collected, plus 30 random pairs. Annotators were asked to decide, for each ordered pair of event descriptions, whether the descriptions are described in their natural order. As the final annotation the majority vote of the 5 annotators was taken.

R11 presents a gold standard for participant clusters. Annotators created participant description sets (PDS) of noun phrases referring to the same type of participant. Sets of aligned event descriptions were sequentially presented to the annotators. The annotators extracted all noun phrases referring to some participant, and grouped them into PDSs. Note that the set of extracted NPs by the annotators does not always correspond to the set of NPs automatically identified by the dependency parser.

Evaluation Metrics

We use two different evaluation metrics, namely the precision/recall metric for event paraphrase and event ordering evaluation, and the b^3 metric for evaluation of the inferred participant clusters.

Precision, Recall and F1-Measure

We use the following standard definitions for precision, recall and F1 measure:

$$precision = \frac{true_{sys} \cap true_{gold}}{true_{sys}} \tag{7.1}$$

$$recall = \frac{true_{sys} \cap true_{gold}}{true_{gold}} \tag{7.2}$$

$$F1 = \frac{2 * precision * recall}{precision + recall}, \tag{7.3}$$

where $true_{sys}$ stands for the number of pairs labeled as positive as classified by our system, and $true_{gold}$ stands for the number of pairs labeled as positive in the gold standard. The intersection $true_{sys} \cap true_{gold}$ is thus the number of cases that were correctly recognized as true by the system. Precision is a measure for the extent to which the pairs labeled as true by the system are really true as defined in the gold standard. Recall is a measure for the extent to which the pairs labeled as true in the gold standard are captured in the true pairs as labeled by the system. Since the two measures stand in a trade-off relation, the F1 measure is defined as their harmonic mean.

The b^3 Metric

Inferring participant description sets can be viewed as a form of co-reference resolution. The b^3 metric is a commonly used evaluation metric in this field (Bagga and Baldwin, 1998), and we follow R11 in using the b^3 metric for participant cluster evaluation. The b^3 measure computes an individual precision and recall score for all mentions of a participant description in the corpus. Precision and recall for a particular mention i are defined as:

$$precision(i) = \frac{|sys_i \cap gold_i|}{|sys_i|} \tag{7.4}$$

$$recall(i) = \frac{|sys_i \cap gold_i|}{|gold_i|}, \tag{7.5}$$

where sys_i refers to the PDS mention i is assigned to by the system, and $gold_i$ refers to the gold PDS mention i is assigned to. $|sys_i|$ and $|gold_i|$ refer to the length of the PDS to which i belongs in the gold annotation and the system labeling, respectively, and $|sys_i \cap gold_i|$ refers to the overlap between the two PDSs.

A final score for the system's performance is computed by averaging over all mention-specific precision and recall scores. We compute a final F1 measure taking the average precision and recall scores, and using the definition in Equation 7.3.

Since the participant mentions provided to the model are automatically extracted by a dependency parser, they do not always correspond to the manually extracted mentions included in the gold standard. This is problematic because b^3 precision is only defined on mentions occurring in the system sets ($|sys_i| > 0$), and b^3 recall is only defined for mentions occurring in the gold standard ($|gold_i| > 0$). Cai and Strube (2010) extended the b^3 metric to deal with this problem in an accurate way. All mentions included in the gold standard, but not in the system mentions are copied as singleton PDSs to the latter. Conversely, all mentions included in some non-singleton PDS inferred by the system, but not in the gold standard, are included as singleton sets in the gold standard. Mentions which only occur in the system sets as a singleton PDS are discarded.

Parameters

In all experiments, we run the Gibbs sampler for 6000 iterations, and report results based on the posterior distributions after the 6000^{th} iteration. One iteration is defined as one sweep over all ESDs in the corpus, where we decide randomly for each ESD which random variable $\in \{\boldsymbol{\tau}, \mathbf{v}, \boldsymbol{\kappa}\}$ to resample. We resample $\boldsymbol{\eta}$ and $\boldsymbol{\xi}$ component-wise as described in Section 6.3.1, and resample the full vectors for every class in the model after every 100^{th} iteration.

We use a development set of four scenarios, the details of the corpora are displayed in Table 7.2. We tune our hyperparameters on these scenarios, and obtain the following values. The Beta parameters for event realization are set to $\alpha^+ = 0.7$, $\alpha^- = 0.1$, and the Beta parameters for participant realization $\beta^+ = 0.5$, $\beta^- = 0.001$. We set the prior parameters of the GMM $\rho_0 = 1.0$ and $\nu_0 = 3.0$. We finally tune the scaling parameter for the Dirichlet priors of event language models $k_\eta = 0.7$ and participant language models $k_\xi = 1.0$. We initialize the regularization parameter $\epsilon = 0.0$, and increase it from the 500^{th} sampling iteration on after every 100^{th} iteration by 0.1.

Finally, we specify the number of event types E and the number of participant types P manually to a number higher than the expected number of event types and participant types present in the data. The model will converge towards using a subset of all possible types. We set $E = 25$ and $P = 30$.

Experiments

In our experiments, we use 10 scenario types which we did not use for development, as displayed in Table 7.3.

We will conduct two sets of experiments. First, we will compare the performance of our model to the performance of the R10 system and the R11 system in order to get a notion of how well our model performs in comparison to existing systems for unsupervised script modeling. We compare the performance on the tasks of event clustering, and ordering constraint learning to the R10 system. The performance on participant clustering of our system will be compared to the performance of the R11 system. Note that the R10 system and the R11 system together tackle the three problems in a pipeline-based architecture. Our model induces all three tasks jointly, in one learning step.

In a second set of experiments, we will examine the influence of the components in our system. We compare the performance of the full model, to a model variant in which we omit the Generalized Mallows Model, in order to assess the contribution of ordering constraints on all three tasks. Additionally, we evaluate a model which lacks the informed prior knowledge of word similarities (but includes the GMM). Comparing the full model to this variant will shed light on the usefulness of the prior knowledge as we define it.

We finally provide a qualitative analysis of the clusters induced by our script model.

Chapter 8

Results and Discussion

We present the results for the various experiments described in the previous chapter. We start by comparing our model to the R10 and R11 systems in Section 8.1, and continue by comparing different variants of our model against each other in Section 8.2. Finally, in Section 8.3 we will provide a qualitative analysis by looking into the clusters induced by our model.

8.1 Comparison to Existing Systems

In the following evaluations, we will present results on a subset of the scenario types presented in the R10 and the R11 evaluation. In addition to this subset, we were able to obtain results of the R10 system on scenarios which are not mentioned in the publication. Those scenarios are marked with an asterisk (*) in the relevant Tables 8.1 and 8.2.

The Event Paraphrase Task

Table 8.1 displays the evaluation results of our model (Bayesian Scripts; BS) and the R10 model on the event paraphrase task. Overall, the performance of both systems is very similar. While our system achieves better precision on average across all evaluation scenarios, the R10 system achieves higher recall scores, and a slightly higher average F1 score. For most scenarios the performance of both systems correlate, meaning that the systems tend to perform well/poorly on the same set of scenarios. There are the two notable exceptions of the `Bus` scenario where our system performs comparably very poorly, and the `Fastfood` scenario where our system clearly outperforms the R10 system. We note that the `Bus` scenario corpus contains an unusual variety of event types. It might be that the ordering constraints imposed by the GMM are too strict to capture this variety. Conversely, the ESDs in the `Fastfood` corpus are much more regular, so that the event clustering component can take advantage of the ordering constraints posed by the GMM in this case.

The Event Ordering Task

The results of the system evaluation on the event ordering task are displayed in Table 8.2. We can observe a similar pattern as before. Our model achieves a

Scenario	**Event Paraphrase Task**					
	Precision		Recall		F1	
	R10	BS	R10	BS	R10	BS
Coffee	0.50	0.47	0.94	0.58	**0.65**	0.52
Telephone	0.93	0.92	0.85	0.72	**0.89**	0.81
Bus	0.65	0.52	0.87	0.43	**0.74**	0.47
Iron	0.52	0.65	0.94	0.56	**0.67**	0.60
Scr. Eggs	0.58	0.92	0.86	0.65	0.69	**0.76**
Vending	0.59	0.72	0.83	0.78	0.69	**0.75**
Microwave*	0.75	0.85	0.75	0.80	0.75	**0.82**
Shower*	0.70	0.68	0.88	0.67	**0.78**	0.67
Fastfood*	0.50	0.74	0.73	0.87	0.59	**0.80**
Ret. Food*	0.73	0.90	0.68	0.87	0.71	**0.89**
Average	0.645	**0.737**	**0.833**	0.693	**0.716**	0.709

Table 8.1: Results of the evaluation on the event ordering tasks. Our system (BS) is compared to the system presented in Regneri et al. (2010) (R10). We report precision, recall and F1 measure for 10 scenario types.

Scenario	**Event Ordering Task**					
	Precision		Recall		F1	
	R10	BS	R10	BS	R10	BS
Coffee	0.70	0.68	0.78	0.57	**0.74**	0.62
Telephone	0.83	0.92	0.86	0.87	0.84	**0.89**
Bus	0.80	0.76	0.80	0.76	**0.80**	0.76
Iron	0.78	0.87	0.72	0.69	0.75	**0.77**
Scr. Eggs	0.67	0.77	0.64	0.59	0.66	**0.67**
Vending	0.84	0.86	0.85	0.75	**0.84**	0.80
Microwave*	0.47	0.91	0.83	0.74	0.60	**0.82**
Shower*	0.48	0.85	0.82	0.84	0.61	**0.85**
Fastfood*	0.53	0.97	0.81	0.65	0.64	**0.78**
Ret. Food*	0.48	0.84	0.75	0.75	0.58	**0.79**
Average	0.658	**0.843**	**0.786**	0.721	0.706	**0.775**

Table 8.2: Results of the evaluation on the event type ordering task, of our system (BS) and the R10 system. Precision, Recall and F1-scores are reported for 10 scenario types.

Scenario	Precision		Recall		F1	
	R11	BS	R11	BS	R11	BS
Coffee	0.85	0.72	0.80	0.54	**0.82**	0.62
Fastfood	0.82	0.60	0.82	0.57	**0.82**	0.59
Microwave	0.89	0.73	0.84	0.49	**0.86**	0.59
Ret. Food	0.80	0.48	0.52	0.29	**0.57**	0.36
Shower	0.87	0.72	0.83	0.61	**0.85**	0.66
Vending	0.80	0.60	0.78	0.34	**0.79**	0.43

Table 8.3: Results on the participant clustering task, comparing the system presented in Regneri et al. (2011) (R11), to our model (BS). Results are computed under the Precision/Recall metric in the context of the b^3 metric.

significantly higher Precision score averaged across all scenarios, while the R10 system, again, achieves higher Recall scores. Our model outperforms the R10 model in the overall F1 score by a margin of 7 percent points.

The general pattern of higher Precision but lower Recall scores for our system suggests that the event clusters we infer are accurate, and the underlying event ordering is captured well, but that they are not general enough and the order is too restricted. One explanation might, again, be the very strong preference for canonical ordering that the GMM component imposes in our model. We want to experiment with varying flexibility of this component in future work.

The R10 system aligns event descriptions based on semantic similarity in a first step, and creates a temporal script graph (TSG), a graph representation of the script, on the basis of this alignment. Our model *jointly* induces both aspects, which is a more ambitious learning objective. Under this consideration, the results our model achieves compare favorably to the results of the R10 system overall on the event paraphrase task and the event ordering task.

The Participant Paraphrase Task

We also evaluate the quality of the participant clusters induced by our model. We compare the performance of our model to the performance of the R11 system. Table 8.3 displays all results.

Our model is clearly outperformed by the existing R11 model. A look into the induced clusters showed that our model induces singleton participant clusters. Each participant cluster contains all mentions of a particular type of participant description. The latter phenomenon is triggered by our posterior regularization component, which encourages all mentions of a term in the participant vocabulary to be assigned to the same cluster (cf. Section 5.4). While this is a useful initial assumption, as shown in R11, our model fails to learn that different participant descriptions can refer to the same type of participant. It is possible that our regularization factor is too strong and thus prohibits clustering of terms once all mentions have been assigned to one cluster. However, we experimented with different parameterizations and did not observe any improvement in performance.

The R11 system is based on the R10 system, taking the output of the R10 system, and computing participant equivalence sets on the basis of this output. The event clusters are thus fixed already, and participants are inferred on the

Scenario	Intra Cluster	Inter Cluster
Ret. Food	0.60	0.40
Laundry	0.61	0.25
Vending	0.48	0.30

Table 8.4: Analysis of correlation of participant mentions across event type assigned to the same gold event cluster (Intra Cluster) and of correlation of participant mentions across gold event clusters (Inter Cluster). The reported scores are cosine-similarities.

basis of fairly confident event clusters. In contrast, our model aims to learn participant clusters *jointly* with the two other tasks. We assume that participant mentions provide strong cues for event type clustering, and vice versa. However, the model fails to induce reasonable clusters of equivalent participant descriptions, referring to the same participant type.

We conducted a few tentative experiments in which we fixed the participant types as specified in the gold standard, expecting that this information would facilitate induction of event types and event orderings. However, we did not observe improved performance on these tasks, which suggests that participant mentions do not provide reliable clues for event clustering in our model.

One reason for this behavior might be the fact that event types and participant types do not necessarily correlate strongly in the data. One example would be the most frequent participant type "food" in the scenario `Eating in a Restaurant`. Participant type "food" occurs with a great variety of distinct event types such as "decide for","order", "eat", "like", "pay for", which also occur in different positions in the event chain. Introducing participants as a factor into the joint model, might thus add noise to the inference procedure, rather than useful information.

We conducted an experiment on the correlations of types of participant descriptions within each gold event cluster, compared them to correlations across gold event clusters. A feature vector was constructed for each event mention included in the gold standard, with all participant description types of the whole corpus as dimensions, containing a count for each participant description with this event mention. We computed the average cosine similarity between those vectors for all event mentions within one gold event cluster in order to obtain intra-cluster similarities. For inter-cluster similarities we obtained one feature vector for each gold cluster by averaging the vectors of all cluster members, and we computed the average cosine similarity between all average vectors. Our results are displayed in Table 8.4. While intra-cluster correlation is consistently higher than inter-cluster correlation, the difference is not as clear as it might be expected.

8.2 Influence of Model Components

In the previous section we compared our model to the existing systems R10 and R11. In this section we present an intrinsic evaluation of our model, in order to examine the influence of different components. We will compare the performance of the full script model to two model variants:

- `-GMM` We omit the Generalized Mallows Model. This will allow us to evaluate the benefit of ordering constraints on the tasks of event clustering and participant clustering.
- `-COVAR` In this model variant, we do not provide the informed prior knowledge, based on semantic word similarity. Comparing this model to the full script model allows us to judge to what extent we minimize the problem of small training data sets with our approach of injecting external knowledge for guiding the inference process.

We compare the performance of the model variants on four scenario types, and the results are displayed in Table 8.5.

Influence of the Generalized Mallows Model

Focusing on the event paraphrase task first, the results clearly show that the GMM has a strong positive influence on the model performance across all four test scenario types. The `-GMM` model variant has no preference towards capturing temporal structure in the induced clustering, such that a comparison on the event ordering task is not really possible. The IDs assigned to the particular clusters are random in this case, and do not capture any ordering information.

Moving on to the participant paraphrase task, the benefit of the GMM does not emerge clearly. On the one hand, many participant types tend to occur at many different points of time in the script, such that we would not expect participants to benefit from ordering constrains directly. On the other hand, the quality of induced event clusters clearly improves under the GMM, such that, given our original modeling assumptions, we would expect the quality of participant clusters to improve with it. The fact that this is not the case, and the surprisingly low participant type-event type correlations presented in the previous section, suggest that our modeling assumptions might not fully accurately resemble the data.

Influence of the Informed Prior Knowledge

We move on to comparing the full model to a variant which does not have access to the informed prior knowledge. Instead we specify uniform Dirichlet parameters, $\boldsymbol{\gamma} = 0.1$ and $\boldsymbol{\delta} = 0.1$. We can see from Table 8.5 that, for the event-related objectives, the informed prior knowledge boosts performance for the scenarios `Return Food` and `Shower`, while it harms performance for the `Microwave` scenario, and has a mixed influence on the `Vending` scenario.

Looking at the overview over our test corpora, displayed in Table 7.3, we can see that one explanation for this result can be the size of the training corpora for the respective scenarios. The `Microwave` corpus consists of approximately three times as many ESDs as the training corpora available for the `Shower` scenario and the `Return food` scenario. It might be the case that the prior knowledge does provide useful guidance when the available training corpus is very limited in size. However, we obtain the word covariances in a completely unsupervised way, and did not experiment with optimizing this process. For scenarios for which we have a larger training corpus available, it might thus be the case that the prior knowledge is too noisy, and disturbs the inference process rather than providing useful guidance.

Model	**Event Paraphrase**			**Event Ordering**			**Ptcpt. Paraphrase**		
	P	R	F	P	R	F	P	R	F
	Ret. Food								
`Full`	0.90	0.87	**0.89**	0.84	0.75	**0.79**	0.48	0.29	0.36
`-GMM`	0.50	0.25	0.33	0.41	0.38	0.39	0.49	0.29	0.36
`-COVAR`	0.86	0.72	0.65	0.76	0.69	0.72	0.50	0.29	**0.37**
	Vending								
`Full`	0.72	0.78	0.75	0.86	0.75	**0.80**	0.60	0.34	**0.43**
`-GMM`	0.84	0.35	0.49	0.65	0.45	0.53	0.60	0.30	0.40
`-COVAR`	0.85	0.71	**0.77**	0.81	0.66	0.73	0.55	0.36	**0.43**
	Shower								
`Full`	0.68	0.67	**0.67**	0.85	0.84	**0.85**	0.72	0.61	0.66
`-GMM`	0.36	0.17	0.23	0.40	0.37	0.38	0.71	0.68	**0.69**
`-COVAR`	0.64	0.44	0.52	0.75	0.71	0.73	0.65	0.68	0.67
	Microwave								
`Full`	0.85	0.80	0.82	0.91	0.74	0.82	0.73	0.49	0.59
`-GMM`	0.88	0.31	0.45	0.67	0.62	0.64	0.73	0.53	0.61
`-COVAR`	0.81	0.99	**0.90**	0.92	0.83	**0.88**	0.72	0.68	**0.70**

Table 8.5: Results of three model variants on our three evaluation tasks. The first line per scenario shows the performance of the full model. `-GMM` is a model version without the GMM component. In the `-COVAR` version the prior covariance information is replaced with a uniform Dirichlet prior (but it includes the GMM). We show standard Precision, Recall and F1-Measure for the event paraphrase task, and the event ordering task, and b^3 scores for the participant paraphrase task. We evaluate on four scenario types.

Cluster	Words
1	go to locate walk load move coffee
2	find get open grind
3	in pour from into get boil identity plug
4	find put in turn on place clean identify close
5	put in into place inside do install add measure
6	to get grind if have make sure press start add measure coffee require filter
7	put fill in on with plug down run through
8	fill turn on wait with until be off do watch up
9	to in pour from into place under brew when do add spout out as desire
10	turn on off stir switch as desire audd
11	to turn on wait for stop drip be and brew when full remove serve set auto then

Table 8.6: Event type clusters induced for the `Coffee` scenario.

There are several factors in the prior knowledge construction process that can be optimized. First, there are many ways to construct the word-specific feature vectors. Our simplistic approach of binary features depending on whether a word sense is member of a SynSet, can be refined. Additionally, it is possible to optimize the influence covariance matrix itself, by tuning the relative influence of the Logistic Normal component, and the document likelihood component in the posterior probability over Dirichlet parameter vectors.

The performance of the participant component varies very little across model variants, such that it is not possible to draw confident conclusions about the influence of our modifications on participant learning. This observation supports our previous assumption that the resulting participant clusters are mainly triggered by strong hyperpriors and posterior regularizations, and suggests that this process receives only little influence from the other components in our model.

8.3 Qualitative Analysis

In order to illustrate the previously presented results, we show examples of the posterior event clusters our model induced, by looking into the induced language models, i.e. the terms of the event vocabulary assigned to each event type. We do not show induced participant clusters here, because our model currently infers exclusively singleton participant clusters. We thus obtain one individual cluster for each participant description type, containing all mentions of this type in the data. Tables 8.6-8.8 present the posterior event clusters for various scenario types.

For all scenarios the model converges on a subset of the maximum number of 25 event types we specified a priori. Generally, the members of a cluster are not necessarily synonyms, but, especially given the restricted domain of a particular scenario, the words are clearly semantically related in most cases. One commonality between the members of a particular cluster is their position in the ESDs: cluster IDs correlate with the position at which the event type

Cluster	Words
1	walk to order at go counter up inside find look
2	walk into enter
3	to listen repeat
4	tell
5	to order eat decide what want park car on count proceed
6	wait in to order get place stand line examine give
7	order pay for at listen confirm
8	into order at collect make from look food dispense
9	in swipe
10	wait pay for at pick up put expect receive with
11	to keep go take find move
12	wait to for get pick up when ready be call recieve select do
13	seat at eat sit down look consume exit
14	to stand off dispose of
15	in eat place take put away throw
16	get eat leave clear return

Table 8.7: Event type clusters induced for the `Fastfood` scenario.

described by the words in the cluster tends to occur in an ESD. This confirms our previously presented experiments on the benefit of the GMM. Taking a closer look at the clusters learnt for the `Fastfood` scenario, displayed in Table 8.7, we can extrapolate the following abstract ordered event chain from the output, which resembles an accurate description of the scenario:

1. enter restaurant (cluster 1-2)
2. order/pay (cluster 3-9)
3. receive food (cluster 10-12)
4. eat food (cluster 13)
5. dispose of trash (cluster 14-15)
6. leave restaurant (cluster 16)

In the clusters inferred for the `Fastfood` scenario, displayed in Table 8.7, the event description "wait" occurs in multiple clusters, namely cluster 6, whose topic broadly corresponds to "waiting to order the food", as well as in clusters 10 and 12, which broadly describe the event type of "waiting for the food". Our model thus captures the fact that one term of the event vocabulary may describe different types of events, depending on their position in the ESD.

Table 8.8 shows, for comparison, the clusters induced for the `Microwave` scenario by the `-GMM` version of the model in the bottom part, and the clusters inferred by the best performing model variant `-COVAR`. In contrast to the clusters inferred by the `-COVAR` model, which includes the GMM component, it is difficult to find any coherence within the clusters inferred by the `-GMM` model variant. Note that the event type distributions inferred by the `-GMM` model variant are less sparse, meaning that more distinct clusters are learnt by model. Furthermore, the cluster-specific language models are less sparse, which results in the fact that the probability mass is distributed across more distinct terms of the vocabulary and thus that all clusters tend to contain more realizing words. Conversely, each term in the vocabulary is assigned to more distinct event types, which results in topically less clear-cut clusters.

For illustration, we highlight in boldface all occurences of the event-realizing term "remove" in the `-COVAR` clusters and in the `-GMM` clusters. The former model infers two event types including the verb "remove": it is included in cluster 1, in the sense of "removing the package from the food", as well as in cluster 8 where it refers to "remove the food from the microwave". In contrast, through the more uniform language model parameters inferred by the `-GMM` model variant, the word appears in the majority of the inferred clusters, and no contextually meaningful sense can be extrapolated.

The clusters also reveal some problems of the automatic preprocessing steps we apply. First, the event type clusters contain some nouns, which should have been automatically extracted from the dependency parses of each event description (e.g. cluster 1 in Table 8.6 contains the noun "coffee"). Some closer inspection revealed that the dependency parser tends to make errors on words which are ambiguous in their part of speech, since the "bullet point-style" phrases often provide little cues for disambiguation. Many ESDs in the `Microwave` scenario contain the event description "Press start.", where "start" is used as a participant, as an abbreviation for "start button". The parser, however, analyses "press" as a noun, and "start" as the verb of the phrase. We did not do any manual correction of erroneous parser output. Another level of noise in the data stems from misspelling. Since the data was collected in written form from volunteers over the Internet, it is noisy with respect to grammar as well as spelling. Although the R10 corpus was manually post-processed to some extent, the data still contained many orthographic and grammatical errors. We did not do any further manual cleaning.

8.4 Summary

In this section, we presented and discussed the performance of the Bayesian script model in various experiments. We started by showing that the performance of our joint model of event types and event orderings performs favorably in comparison to existing models which learn the same targets in a pipeline based architecture. We also showed that the model is not able to jointly induce equivalence classes of participants.

We continued by examining the influence of different components in our model, by testing different model variants and comparing their performance. We were able to show the benefit of the Generalized Mallows Model, in our *joint* learning setting of event types and constraints on their orderings. An evaluation on the benefit of the informed prior knowledge was conducted, and we showed that the prior knowledge is helpful for scenario types for which comparatively little training data is available, but is not robust enough to boost performance on scenario types for which we are equipped with more data.

We concluded by a qualitative analysis, and showing example event clusters induced by the model, which supported the findings of the quantitative

Cluster	Words
1	get put in to **remove** place take out of keep on heat locate from find be with switch microwave-safe on/in
2	to close open take press locate find walk off unwrap cover input
3	put in to place keep into on enter inside with replace loosely cover wrap
4	set for close select shut check choose need
5	in set accord to for press on enter select up input stir cook choose as punch indicate
6	start wait for push press and turn on select enjoy cook
7	set to wait for when do open until s finish heat up be serve let repeat till acheived cook over stop timer/power as appropriate
8	get **remove** when open take out of and after from go off up be carefully buzz eat beep ding

Cluster	Words
1	in set to for **remove** push take out of press on from find enter input
2	set accord to **remove** open after locate from walk let
3	in to start wait for on until finish select be cover repeat till acheived cook over stop punch
4	start wait **remove** take out press and carefully eat
5	get put close press turn on find shut cook
6	put in set place into inside
7	set **remove** push take on locate from select off unwrap serve cover stir cook
8	in wait when open take out of and turn on until s finish heat enter select up be check choose need as indicate
9	close open
10	put in **remove** place open keep and into on with replace loosely on/in
11	set to start wait for close finish heat up cook
12	put in start **remove** place when do open keep press into on select inside buzz choose wrap ding
13	get put in set to for **remove** place close when take out on heat go off up be with beep microwave-safe timer/power as appropriate
14	start for turn on enter select enjoy switch

Table 8.8: Event type clusters induced for the `Microwave` scenario. Top: Clusters induced by the best performing model variant `-COVAR`. Bottom: Clusters induced by the `-GMM` model variant.

evaluations.

As a last remark, we would like to point out that it is generally difficult to induce reliable models on small data sets in completely unsupervised learning setting, as the one presented here. Even our prior knowledge module cannot completely balance out this fact, because it only provides specific knowledge for one module, the language model component, in our model. It is to be expected that the performance of the model would increase with more available training data. It might also be the case that learning three objectives from the small corpora is over-ambitious. Predictive patterns of participant type occurrences might emerge more clearly when more data is available, and it might be possible to reliably infer participant types together with the event types and orderings with our model in that case.

Chapter 9

Conclusion

Through years of experience, humans internalize knowledge about stereotypical courses of events involved in every day situations. For NLP applications, such as question answering or automatic summarization, it has been shown that this kind of *script knowledge* improves performance (Cullingford, 1978; Miikkulainen, 1995).

In this thesis, we proposed a new way of learning script knowledge from explicit event-sequence descriptions in a fully unsupervised way. We presented a model that jointly learns event types, constraints on their ordering, and participant types for various common scenarios. In particular, we presented a hierarchical Bayesian script model, which allows us to formulate our modeling decisions for the three objectives on one consistent theoretical foundation. We derived an inference algorithm for our model, using Gibbs sampling for approximate sampling.

We argued that our three learning objectives are highly connected and should thus provide cues for each other in inference. We thus proposed a joint model. In addition to this joint formulation we presented two further contributions: First, we incorporated the Generalized Mallows Model (GMM), a flexible, statistical model over permutations, for modeling event ordering constraints. In contrast to previously employed approaches, the GMM is able to infer event type-specific temporal flexibility. Secondly, we incorporated informed prior knowledge in order to alleviate the problem of learning from small data sets. In particular we encoded WordNet-based semantic similarities in a covariance matrix and thus triggered correlations of probabilities of semantically related words across all induced types.

We test the effect of our contributions in an extensive quantitative and qualitative evaluation. First, we evaluate our system, by comparing it to two existing systems R10 (Regneri et al., 2010), which learns event types and orderings, and R11 (Regneri et al., 2011), which infers participant types. The two systems infer the three objectives in a pipeline-based architecture, employing different methodologies at each stage. The R10 model uses a less flexible ordering model, i.e. multiple sequence alignment.

On the task of event type learning, the performance of our system is very similar to the performance of R10. However, we tackle the more complex task of inferring this objective jointly with the two other objectives. Given this consideration our model compares favorably.

Our model outperforms the R10 system on the task of learning ordering constraints. This shows the effectiveness of our more flexible ordering model. We conducted an experiment in which we compare our full model to a model variant lacking the GMM component. The results significantly degraded for the task of event type learning, which empirically confirms our theoretical motivation for using the GMM for modeling event ordering constraints in our model.

We compare our model on the task of participant type learning to the R11 system. We were not able to obtain results comparable to the good performance of the existing system. Although we assumed that participant types should provide clues for event type induction, and vice versa, the participant component in our model does currently not have a positive influence. We provided a discussion of this phenomenon, suspecting - based on an analysis of the data - that event types and participant types do not necessarily correlate, and that our modeling assumptions might thus be not entirely accurate.

We furthermore evaluated the influence of the prior knowledge component by comparing to a model variant lacking this component. We were able to show that our prior knowledge component boosts performance of the model on scenario types for which particularly little training sets are available. For scenario types for which larger amounts of training data are available, the model results did not improve with the prior knowledge. This suggests that the prior knowledge we define might not be robust enough, yet.

We finally provided a qualitative analysis of the event clusters inferred by our model. Overall, the inferred clusters correspond to separate event types involved in a particular scenario. Furthermore, the underlying temporal order is captured in the inferred clustering.

9.1 Future Work

We conclude with pointing at possible directions for future work. Our evaluation revealed that our model is currently not able to induce participant types jointly with event types and event ordering constraints. One possible area for further investigation would be considering the semantic roles of the participants. We would expect that the discriminative power of participant types for event types increases with this change. However, we have to keep in mind the problem of the small data sets we are faced with, and the information in the data may become too sparse if we impose further distinctions.

In addition to modeling event type-specific temporal flexibility with the GMM, our model learns a measure for whether an event type is obligatory or rather optional in a script, through the event type-specific realization probabilities. The evaluations we presented in Chapter 8 do not capture these effects. We want to design an evaluation specifically targeted at these phenomena. For evaluating the latter phenomenon we could collect human judgments on the level of optionality for specific event types and compare these to the parameters induced by the model.

Our modeling assumptions currently consist of a number of simplifications, which could be relaxed in a number of ways. Our generative story includes simplifications, such as independence between event types in an ESD and independence among participant types within and across events. Clearly, these assumptions are not present in the real data and we expect that loosening those independence assumptions will lead to a more accurate model. However, currently this is prohibitive due to the limited amount of available training data.

It would be interesting to examine to what extent we can alleviate the problem of small data sets by optimizing our prior knowledge component. We currently construct the similarity scores for our covariance matrix by counting SynSet membership overlap for every pair of words. The similarity metric can be easily tuned and refined, for example by considering the distance of any sense of the word to a particular SynSet. Note, however, that we are somewhat restricted in the ways we can compute word similarities, because the resulting matrix must be positive semidefinite. Any similarity score of a word pair we can use, should thus be computable as the dot product of two word feature vectors, in order to guarantee that we meet this constraint. This, for example, excludes standard WordNet similarity measures such as Lin's WordNet distance (Lin, 1998). Regneri et al. (2011) find that Lin's distance was the optimal choice for incorporating semantic similarity into their R11 system.

Another possibility for optimization of the prior knowledge component is tuning the influence of the matrix on the posterior probability distribution over parameter vectors. It is possible to skew the covariance matrix in the directions, in which the covariance between components is high already based in the WordNet similarity, and thus boost high similarities scores for two words even more. As a side effect, the relative influence between the Multivariate Normal component and the document likelihood component of the posterior can be influenced in this way.

Our evaluation suggested that the canonical ordering inferred by the GMM was too restricted for some scenarios. This suggests that one global parameterization of the GMM is too restrictive. One possibility of loosening this restriction would be to relax the GMM in the early learning phase through appropriately set hyperparameters, and change the hyperparameters during the learning period such that the preference for canonical ordering increases, and, hopefully, improves the quality of the event clusters at the same time. Another possibility for increasing flexibility of the GMM, as suggested by Chen et al. (2009), would be to draw the parameters of the GMM from an additional layer in the model hierarchy, instead of specifying it hyperparameters manually and globally a priori.

Bibliography

J. Aitchison. 1982. The statistical analysis of compositional data. *Journal of the Royal Statistical Society, Series B*, 44:139–177.

Christophe Andrieu, Nando de Freitas, Arnaud Doucet, and Michael I. Jordan. 2003. An introduction to mcmc for machine learning. *Machine Learning*, 50(1-2):5–43.

Amit Bagga and Breck Baldwin. 1998. Algorithms for scoring coreference chains. In *In The First International Conference on Language Resources and Evaluation Workshop on Linguistics Coreference*, pages 563–566.

C. F. Baker, C. J. Fillmore, and J. B. Lowe. 1998. The berkeley framenet project. In *Proceedings of the 17th international conference on Computational linguistics - Volume 1*, COLING '98, pages 86–90. Association for Computational Linguistics.

A. Barr and E.A. Feigenbaum. 1986. *The handbook of artificial intelligence. 1 (1981).* The Handbook of Artificial Intelligence. Addison-Wesley.

R. Barzilay, N. Elhadad, and K. McKeown. 2002. Inferring strategies for sentence ordering in multidocument news summarization. *Journal of Artificial Intelligence Research.*

Christopher M. Bishop. 2006. *Pattern Recognition and Machine Learning (Information Science and Statistics).* Springer-Verlag New York, Inc., Secaucus, NJ, USA.

D. Blei and J. Lafferty. 2006. Correlated topic models. In *Advances in Neural Information Processing Systems 18*, pages 147–154. MIT Press, Cambridge, MA.

D. M. Blei, T. L. Griffiths, M. I. Jordan, and J. B. Tenenbaum. 2004. Hierarchical topic models and the nested chinese restaurant process. In *Advances in Neural Information Processing Systems 16.* MIT Press.

D. M. Blei and J. D. Lafferty. 2005. Correlated topic models. In *Advances in Neural Information Processing Systems 17.*

D. M. Blei, A. Y. Ng, and M. I. Jordan. 2003. Latent dirichlet allocation. *Journal of Machine Learning Research*, 3:993–1022.

Jie Cai and Michael Strube. 2010. Evaluation metrics for end-to-end coreference resolution systems. In *Proceedings of the 11th Annual Meeting of the Special Interest Group on Discourse and Dialogue*, SIGDIAL '10, pages 28–36. Association for Computational Linguistics, Stroudsburg, PA, USA.

N. Chambers and D. Jurafsky. 2008. Unsupervised learning of narrative event chains. In *Proceedings of ACL-08: HLT*, pages 789–797. Association for Computational Linguistics.

N. Chambers and D. Jurafsky. 2009. Unsupervised learning of narrative schemas and their participants. In *Proceedings of ACL-09 and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, pages 602–610. Association for Computational Linguistics.

N. Chambers and D. Jurafsky. 2011. Template-based information extraction without the templates. In *Proceedings of ACL*.

H. Chen, S. R. K. Branavan, R. Barzilay, and D. R. Karger. 2009. Content modeling using latent permutations. *J. Artif. Int. Res.*, 36(1):129–163.

R. E. Cullingford. 1978. *Script Application: Computer Understanding of Newspaper Stories*. Ph.D. thesis, Department of Computer Science, Yale University.

Christiane Fellbaum, editor. 1998. *WordNet: an electronic lexical database*. MIT Press.

M. Fligner and J. Verducci. 1986. Distance based ranking models. *Journal of the Royal Statistical Society, Series B*, 48:359–369.

M. Fligner and J. Verducci. 1990. Posterior probabilities for a consensus ordering. *Psychometrika*, 55:53–63.

Stuart Geman and Donald Geman. 1984. Stochastic relaxation, gibbs distributions, and the bayesian restoration of images. *IEEE Trans. Pattern Anal. Mach. Intell.*, 6(6):721–741.

João Graça, Kuzman Ganchev, Ben Taskar, and Fernando C. N. Pereira. 2009. Posterior vs parameter sparsity in latent variable models. In *Advances in Neural Information Processing Systems 21*, pages 664–672.

João V. Graça, Kuzman Ganchev, and Ben Taskar. 2008. Expectation maximization and posterior constraints. In *Advances in Neural Information Processing Systems 20*, pages 569–576. MIT Press.

T. L. Griffiths and M. Steyvers. 2004. Finding scientific topics. *Proceedings of the National Academy of Sciences*, 101(Suppl. 1):5228–5235.

Rakesh Gupta and Mykel J. Kochenderfer. 2004. Common sense data acquisition for indoor mobile robots. In *AAAI*, pages 605–610.

P. Hennig, D. H. Stern, R. Herbrich, and T. Graepel. 2012. Kernel topic models. *Journal of Machine Learning Research - Proceedings Track*, 22:511–519.

Jennifer A. Hoeting, David Madigan, Adrian E. Raftery, and Chris T. Volinsky. 1999. Bayesian model averaging: A tutorial. *Statistical Science*, 14(4):382–417.

M. G. Kendall. 1938. A new measure of rank correlation. *Biometrika*, 30(1/2):81–93.

A. Klementiev, D. Roth, and K. Small. 2008. Unsupervised rank aggregation with distance-based models. In *Proceedings of the 25th international conference on Machine learning*, pages 472–479. ACM.

S. Kullback and R. A. Leibler. 1951. On information and sufficiency. *Ann. Math. Statist.*, 22(1):79–86.

M. Lapata. 2003. Probabilistic text structuring: experiments with sentence ordering. In *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics - Volume 1*, pages 545–552. Association for Computational Linguistics.

G. Lebanon and J. Lafferty. 2002. Cranking: Combining rankings using conditional probability models on permutations. In *In Proceedings of the 19th International Conference on Machine Learning*, pages 363–370.

Henry Lieberman, Hugo Liu, Push Singh, and Barbara Barry. 2004. Beating common sense into interactive applications. *AI Magazine*, 25:63–76.

Dekang Lin. 1998. An information-theoretic definition of similarity. In *In Proceedings of the 15th International Conference on Machine Learning*, pages 296–304. Morgan Kaufmann.

D. J. C. MacKay. 2002. *Information Theory, Inference & Learning Algorithms.* Cambridge University Press, New York, NY, USA.

C. L. Mallows. 1957. Non-null ranking models. *Biometrika*, 44:114–130.

Christopher D. Manning and Hinrich Schütze. 1999. *Foundations of statistical natural language processing.* MIT Press, Cambridge, MA, USA.

Marina Meila, Kapil Phadnis, Arthur Patterson, and Jeff Bilmes. 2007. Consensus ranking under the exponential model. In *22nd Conference on Uncertainty in Artificial Intelligence (UAI07).* Vancouver, British Columbia.

Risto Miikkulainen. 1995. Script-based inference and memory retrieval in subsymbolic story processing. *Applied Intelligence*, pages 137–163.

Ashutosh Modi, Ivan Titov, and Alexandre Klementiev. 2012. Unsupervised induction of frame-semantic representations. In *Proceedings of the NAACL-HLT 2012 Workshop on Inducing Linguistic Structure.* Montreal, Canada.

R. M. Neal. 2003. Slice sampling. *Annals of Statistics*, 31:705–767.

Radford M. Neal. 1993. Probabilistic inference using Markov chain Monte Carlo methods. Technical Report CRG-TR-93-1, Dept. of Computer Science, University of Toronto.

E. Ovchinnikova. 2012. *Integration of World Knowledge for Natural Language Understanding.* Atlantis Thinking Machines. Atlantis Press (Zeger Karssen).

B. O'Connor. 2012. Learning frames from text with an unsupervised latent variable model. Data analysis project report, machine learning department, Carnegie Mellon University. URL `http://brenocon.com/oconnor_-dap2012.pdf`.

Simone Paolo Ponzetto and Michael Strube. 2009. Extracting world and linguistic knowledge from wikipedia. In *HLT-NAACL (Tutorial Abstracts)*, pages 7–8.

J. Pustejovsky, P. Hanks, R. Saurí, A. See, R. Gaizauskas, A. Setzer, D. Radev, B. Sundheim, D. Day, L. Ferro, and M. Lazo. 2003. The TIMEBANK corpus. In *Proceedings of Corpus Linguistics 2003*, pages 647–656.

Rajat Raina, Andrew Y. Ng, and Daphne Koller. 2006. Constructing informative priors using transfer learning. In *In Proceedings of the 23rd International Conference on Machine Learning*, pages 713–720.

M. Regneri, A. Koller, and M. Pinkal. 2010. Learning script knowledge with web experiments. In *Proceedings of ACL 2010*. Association for Computational Linguistics.

M. Regneri, A. Koller, J. Ruppenhofer, and M. Pinkal. 2011. Learning script participants from unlabeled data. In *Proceedings of RANLP 2011*.

Roger C. Schank and Robert P. Abelson. 1975. Scripts, plans and knowledge. In *Thinking: Readings in Cognitive Science, Proceedings of the Fourth International Joint Conference on Artificial Intelligence*, pages 151–157. Tbilisi, USSR.

Push Singh, Thomas Lin, Erik T. Mueller, Grace Lim, Travell Perkins, and Wan Li Zhu. 2002. Open mind common sense: Knowledge acquisition from the general public. pages 1223–1237. Springer-Verlag.

M. Steyvers, M. D. Lee, B. Miller, and P. Hemmer. 2009. The wisdom of crowds in the recollection of order information. In *Advances in Neural Information Processing Systems 19*, pages 1785–1793. Curran Associates, Inc.

Ivan Titov and Alexandre Klementiev. 2011. A bayesian model for unsupervised semantic parsing. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 1445–1455. Association for Computational Linguistics, Portland, Oregon, USA.

Ivan Titov and Alexandre Klementiev. 2012. A bayesian approach to unsupervised semantic role induction. In *Proceedings of the Conference of the European Chapter of the Association for Computational Linguistics*, pages 12–22. Avignon, France.