Skip N-grams and Ranking Functions for Predicting Script Events

Bram Jans
KU Leuven
Leuven, Belgium
bram.jans@gmail.com

Steven Bethard
University of Colorado Boulder
Boulder, Colorado, USA
steven.bethard@colorado.edu

Ivan Vulić
KU Leuven
Leuven, Belgium
ivan.vulic@cs.kuleuven.be

Marie Francine Moens
KU Leuven
Leuven, Belgium
sien.moens@cs.kuleuven.be

Abstract

In this paper, we extend current state-of-the-art research on unsupervised acquisition of scripts, that is, stereotypical and frequently observed sequences of events. We design, evaluate and compare different methods for constructing models for script event prediction: given a partial chain of events in a script, predict other events that are likely to belong to the script. Our work aims to answer key questions about how best to (1) identify representative event chains from a source text, (2) gather statistics from the event chains, and (3) choose ranking functions for predicting new script events. We make several contributions, introducing skip-grams for collecting event statistics, designing improved methods for ranking event predictions, defining a more reliable evaluation metric for measuring predictiveness, and providing a systematic analysis of the various event prediction models.

1 Introduction

There has been recent interest in automatically acquiring world knowledge in the form of *scripts* (Schank and Abelson, 1977), that is, frequently recurring situations that have a stereotypical sequence of events, such as a visit to a restaurant. All of the techniques so far proposed for this task share a common sub-task: given an event or partial chain of events, predict other events that belong to the same script (Chambers and Jurafsky, 2008; Chambers and Jurafsky, 2009; Chambers and Jurafsky, 2011; Manshadi et al., 2008; McIntyre and Lapata, 2009; McIntyre and Lapata, 2010; Regneri et al., 2010). Such a model can then serve as input to a system that identifies the order of the events within that script (Chambers and Jurafsky, 2008; Chambers and Jurafsky, 2009) or that generates a story using the selected events (McIntyre and Lapata, 2009; McIntyre and Lapata, 2010).

In this article, we analyze and compare techniques for constructing models that, given a partial chain of events, predict other events that belong to the script. In particular, we consider the following questions:

- How should representative chains of events be selected from the source text?
- Given an event chain, how should statistics be gathered from it?
- Given event n-gram statistics, which ranking function best predicts the events for a script?

In the process of answering these questions, this article makes several contributions to the field of script and narrative event chain understanding:

- We explore for the first time the use of skip-grams for collecting narrative event statistics, and show that this approach performs better than classic n-gram statistics.
- We propose a new method for ranking events given a partial script, and show that it performs substantially better than ranking methods from prior work.
- We propose a new evaluation procedure (using Recall@N) for the cloze test, and advocate its usage instead of *average rank* used previously in the literature.
- We provide a systematic analysis of the interactions between the choices made when constructing an event prediction model.

Proceedings of the 13th Conference of the European Chapter of the Association for Computational Linguistics, pages 336–344,
Avignon, France, April 23 - 27 2012. ©2012 Association for Computational Linguistics

Section 2 gives an overview of the prior work related to this task. Section 3 lists and briefly describes different approaches that try to provide answers to the three questions posed in this introduction, while Section 4 presents the results of our experiments and reports on our findings. Finally, Section 5 provides a conclusive discussion along with ideas for future work.

2 Prior Work

Our work is primarily inspired by the work of Chambers and Jurafsky, which combined a dependency parser with coreference resolution to collect event script statistics and predict script events (Chambers and Jurafsky, 2008; Chambers and Jurafsky, 2009). For each document in their training corpus, they used coreference resolution to identify all the entities, and a dependency parser to identify all verbs that had an entity as either a subject or object. They defined an event as a verb plus a dependency type (either subject or object), and collected for each entity, the chain of events that it participated in. They then calculated pointwise mutual information (PMI) statistics over all the pairs of events that occurred in the event chains in their corpus. To predict a new script event given a partial chain of events, they selected the event with the highest sum of PMIs with all the events in the partial chain.

The work of McIntyre and Lapata followed in this same paradigm, (McIntyre and Lapata, 2009; McIntyre and Lapata, 2010), collecting chains of events by looking at entities and the sequence of verbs for which they were a subject or object. They also calculated statistics over the collected event chains, though they considered both event bigram and event trigram counts. Rather than predicting an event for a script however, they used these simple counts to predict the next event that should be generated for a children's story.

Manshadi and colleagues were concerned about the scalability of running parsers and coreference over a large collection of story blogs, and so used a simplified version of event chains – just the main verb of each sentence (Manshadi et al., 2008). Rather than rely on an ad-hoc summation of PMIs, they apply language modeling techniques (specifically, a smoothed 5-gram model) over the sequence of events in the collected chains. However, they only tested these language models on sequencing tasks (e.g. is the real sequence better than a random sequence?) rather than on prediction tasks (e.g. which event should follow these events?).

In the current article, we attempt to shed some light on these previous works by comparing different ways of collecting and using event chains.

3 Methods

Models that predict script events typically have three stages. First, a large corpus is processed to find event chains in each of the documents. Next, statistics over these event chains are gathered and stored. Finally, the gathered statistics are used to create a model that takes as input a partial script and produces as output a ranked list of events for that script. The following sections give more details about each of these stages and identify the decisions that must be made in each step, and an overview of the whole process with an example source text is displayed in Figure 1.

3.1 Identifying Event Chains

Event chains are typically defined as a sequence of actions performed by some actor. Formally, an event chain C for some actor a, is a partially ordered set of events (v, d) where each v is a verb that has the actor a as its dependency d. Following prior work (Chambers and Jurafsky, 2008; Chambers and Jurafsky, 2009; McIntyre and Lapata, 2009; McIntyre and Lapata, 2010), these event chains are identified by running a coreference system and a dependency parser. Then for each entity identified by the coreference system, all verbs that have a mention of that entity as one of their dependencies are collected[1]. The event chain is then the sequence of (verb, dependency-type) tuples. For example, given the sentence *A Crow was sitting on a branch of a tree when a Fox observed her*, the event chain for the *Crow* would be (*sitting*, SUBJECT), (*observed*, OBJECT).

Once event chains have been identified, the most appropriate event chains for training the model must be selected. The goal of this process is to select the subset of the event chains identified by the coreference system and the dependency parser that look to be the most reliable. Both the coreference system and the dependency parser make some errors, so not all event chains are necessarily useful for training a model. The three strategies we consider for this selection process are:

[1] Also following prior work, we consider only the dependencies *subject* and *object*.

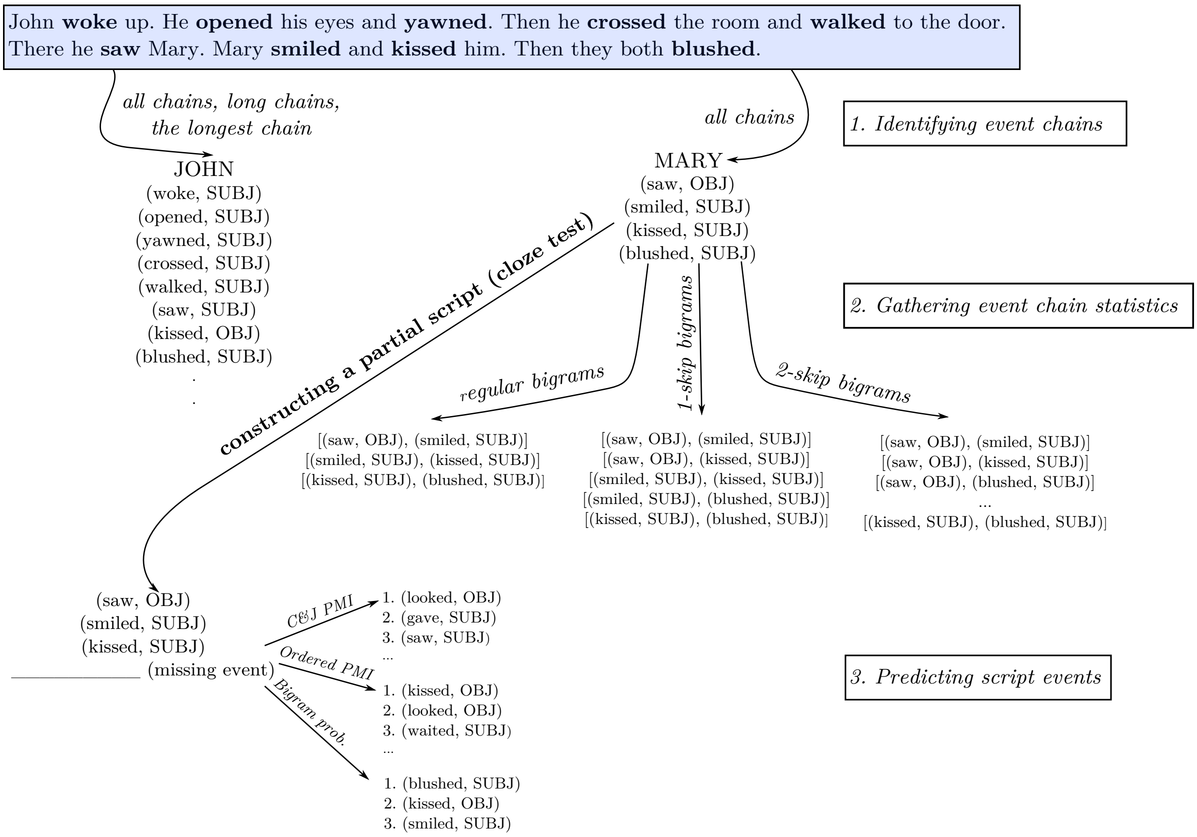
John **woke** up. He **opened** his eyes and **yawned**. Then he **crossed** the room and **walked** to the door. There he **saw** Mary. Mary **smiled** and **kissed** him. Then they both **blushed**.
all chains, long chains, the longest chain
all chains
1. Identifying event chains
JOHN
(woke, SUBJ)
(opened, SUBJ)
(yawned, SUBJ)
(crossed, SUBJ)
(walked, SUBJ)
(saw, SUBJ)
(kissed, OBJ)
(blushed, SUBJ)
MARY
(saw, OBJ)
(smiled, SUBJ)
(kissed, SUBJ)
(blushed, SUBJ)
2. Gathering event chain statistics
constructing a partial script (cloze test)
regular bigrams
1-skip bigrams
2-skip bigrams
[(saw, OBJ), (smiled, SUBJ)]
[(smiled, SUBJ), (kissed, SUBJ)]
[(kissed, SUBJ), (blushed, SUBJ)]
[(saw, OBJ), (smiled, SUBJ)]
[(saw, OBJ), (kissed, SUBJ)]
[(smiled, SUBJ), (kissed, SUBJ)]
[(smiled, SUBJ), (blushed, SUBJ)]
[(kissed, SUBJ), (blushed, SUBJ)]
[(saw, OBJ), (smiled, SUBJ)]
[(saw, OBJ), (kissed, SUBJ)]
[(saw, OBJ), (blushed, SUBJ)]
...
[(kissed, SUBJ), (blushed, SUBJ)]
(saw, OBJ)
(smiled, SUBJ)
(kissed, SUBJ)
(missing event)
C&J PMI
Ordered PMI
Bigram prob.
1. (looked, OBJ)
2. (gave, SUBJ)
3. (saw, SUBJ)
...
1. (kissed, OBJ)
2. (looked, OBJ)
3. (waited, SUBJ)
...
1. (blushed, SUBJ)
2. (kissed, OBJ)
3. (smiled, SUBJ)
3. Predicting script events

Figure 1: An overview of the whole linear work flow showing the three key steps – identifying event chains, collecting statistics out of the chains and predicting a missing event in a script. The figure also displays how a partial script for evaluation (Section 4.3) is constructed. We show the whole process for Mary's event chain only, but the same steps are followed for John's event chain.

- Select **all event chains**, that is, all sequences of two or more events linked by common actors. This strategy will produce the largest number of event chains to train a model from, but it may produce noisier training data as the very short chains included by this strategy may be less likely to represent real scripts.
- Select all **long event chains** consisting of 5 or more events. This strategy will produce a smaller number of event chains, but as they are longer, they may be more likely to represent scripts.
- Select only the **longest event chain**. This strategy will produce the smallest number of event chains from a corpus. However, they may be of higher quality, since this strategy looks for the key actor in each story, and only uses the events that are tied together by that key actor. Since this is the single actor that played the largest role in the story, its actions may be the most likely to represent a real script.

3.2 Gathering Event Chain Statistics

Once event chains have been collected from the corpus, the statistics necessary for constructing the event prediction model must be gathered. Following prior work (Chambers and Jurafsky, 2008; Chambers and Jurafsky, 2009; Manshadi et al., 2008; McIntyre and Lapata, 2009; McIntyre and Lapata, 2010), we focus on gathering statistics about the n-grams of events that occur in the collected event chains. Specifically, we look at strategies for collecting bigram statistics, the most common type of statistics gathered in prior work. We consider three strategies for collecting bigram statistics:

- **Regular bigrams**. We find all pairs of events that are adjacent in an event chain and collect the number of times each event pair was observed. For example, given the chain of events (*saw*, SUBJ), (*kissed*, OBJ), (*blushed*, SUBJ), we would extract the two event bigrams: ((*saw*, SUBJ), (*kissed*, OBJ)) and ((*kissed*, OBJ), (*blushed*, SUBJ)). In addition to the event pair counts, we also collect the number of times each event was observed individually, to allow for various conditional probability calculations. This strategy follows the classic approach for most language models.

- **1-skip bigrams**. We collect pairs of events that occur with 0 or 1 events intervening between them. For example, given the chain (*saw*, SUBJ), (*kissed*, OBJ), (*blushed*, SUBJ), we would extract three bigrams: the two regular bigrams ((*saw*, SUBJ), (*kissed*, OBJ)) and ((*kissed*, OBJ), (*blushed*, SUBJ)), plus the 1-skip-bigram, ((*saw*, SUBJ), (*blushed*, SUBJ)). This approach to collecting n-gram statistics is sometimes called *skip-gram* modeling, and it can reduce data sparsity by extracting more event pairs per chain (Guthrie et al., 2006). It has not previously been applied in the task of predicting script events, but it may be quite appropriate to this task because in most scripts it is possible to skip some events in the sequence.

- **2-skip bigrams**. We collect pairs of events that occur with 0, 1 or 2 intervening events, similar to what was done in the 1-skip bigrams strategy. This will extract even more pairs of events from each chain, but it is possible the statistics over these pairs of events will be noisier.

3.3 Predicting Script Events

Once statistics over event chains have been collected, it is possible to construct the model for predicting script events. The input of this model will be a partial script c of n events, where $c = c_1 c_2 \ldots c_n = (v_1, d_1), (v_2, d_2), \ldots, (v_n, d_n)$, and the output of this model will be a ranked list of events where the highest ranked events are the ones most likely to belong to the event sequence in the script. Thus, the key issue for this model is to define the function f for ranking events. We consider three such ranking functions:

- **Chambers & Jurafsky PMI**. Chambers and Jurafsky (2008) define their event ranking function based on pointwise mutual information. Given a partial script c as defined above, they consider each event $e = (v', d')$ collected from their corpus, and score it as the sum of the pointwise mutual informations between the event e and each of the events in the script:

$$f(e, c) = \sum_{i}^{n} \log \frac{P(c_i, e)}{P(c_i)P(e)}$$

Chambers and Jurafsky's description of this score suggests that it is unordered, such that $P(a, b) = P(b, a)$. Thus the probabilities must be defined as:

$$P(e_1, e_2) = \frac{C(e_1, e_2) + C(e_2, e_1)}{\sum_{e_i} \sum_{e_j} C(e_i, e_j)}$$

$$P(e) = \frac{C(e)}{\sum_{e'} C(e')}$$

where $C(e_1, e_2)$ is the number of times that the ordered event pair (e_1, e_2) was counted in the training data, and $C(e)$ is the number of times that the event e was counted.

- **Ordered PMI**. A variation on the approach of Chambers and Jurafsky is to have a score that takes the order of the events in the chain into account. In this scenario, we assume that in addition to the partial script of events, we are given an insertion point, m, where the new event should be added. The score is then defined as:

$$f(e, c) = \sum_{k=1}^{m} \log \frac{P(c_k, e)}{P(c_k)P(e)} + \sum_{k=m+1}^{n} \log \frac{P(e, c_k)}{P(e)P(c_k)}$$

where the probabilities are defined as:

$$P(e_1, e_2) = \frac{C(e_1, e_2)}{\sum_{e_i} \sum_{e_j} C(e_i, e_j)}$$

$$P(e) = \frac{C(e)}{\sum_{e'} C(e')}$$

This approach uses pointwise mutual information but also models the event chain in the order it was observed.

- **Bigram probabilities**. Finally, a natural ranking function, which has not been applied to the script event prediction task (but has been applied to related tasks (Manshadi et al., 2008)) is to use the bigram probabilities of language modeling rather than pointwise mutual information scores. Again, given an insertion point m for the event in the script, we define the score as:

$$f(e,c) = \sum_{k=1}^{m} \log P(e|c_k) + \sum_{k=m+1}^{n} \log P(c_k|e)$$

where the conditional probability is defined as[2]:

$$P(e_1|e_2) = \frac{C(e_1,e_2)}{C(e_2)}$$

This approach scores an event based on the probability that it was observed following all the events before it in the chain and preceding all the events after it in the chain. This approach most directly models the event chain in the order it was observed.

4 Experiments

Our experiments aimed to answer three questions: Which event chains are worth keeping? How should event bigram counts be collected? And which ranking method is best for predicting script events? To answer these questions we use two corpora, the Reuters Corpus and the Andrew Lang Fairy Tale Corpus, to evaluate our three different chain selection methods, $\{$*all chains, long chains, the longest chain*$\}$, our three different bigram counting methods, $\{$*regular bigrams, 1-skip bigrams, 2-skip bigrams*$\}$, and our three different ranking methods, $\{$*Chambers & Jurafsky PMI, ordered PMI, bigram probabilities*$\}$.

4.1 Corpora

We consider two corpora for evaluation:

- **Reuters Corpus, Volume 1** [3] (Lewis et al., 2004) – a large collection of 806, 791 news stories written in English concerning a number of different topics such as politics, economics, sports, etc., strongly varying in length, topics and narrative structure.
- **Andrew Lang Fairy Tale Corpus** [4] – a small collection of 437 children stories with an average length of 125 sentences, and used previously for story generation by McIntyre and Lapata (2009).

In general, the Reuters Corpus is much larger and allows us to see how well script events can be predicted when a lot of data is available, while the Andrew Lang Fairy Tale Corpus is much smaller, but has a more straightforward narrative structure that may make identifying scripts simpler.

4.2 Corpus Processing

Constructing a model for predicting script events requires a corpus that has been parsed with a dependency parser, and whose entities have been identified via a coreference system. We therefore processed our corpora by (1) filtering out non-narrative articles, (2) applying a dependency parser, (3) applying a coreference resolution system and (4) identifying event chains via entities and dependencies.

First, articles that had no narrative content were removed from the corpora. In the Reuters Corpus, we removed all files solely listing stock exchange values, interest rates, etc., as well as all articles that were simply summaries of headlines from different countries or cities. After removing these files, the Reuters corpus was reduced to 788, 245 files. Removing files from the Fairy Tale corpus was not necessary – all 437 stories were retained.

We then applied the Stanford Parser (Klein and Manning, 2003) to identify the dependency structure of each sentence in each article in the corpus. This parser produces a constitutent-based syntactic parse tree for each sentence, and then converts this tree to a collapsed dependency structure via a set of tree patterns.

Next we applied the OpenNLP coreference engine[5] to identify the entities in each article, and the noun phrases that were mentions of each entity.

Finally, to identify the event chains, we took each of the entities proposed by the coreference system, walked through each of the noun phrases associated with that entity, retrieved any *subject*

[2] Note that predicted bigram probabilities are calculated in this way for both classic language modeling and skip-gram modeling. In skip-gram modeling, skips in the n-grams are only used to increase the size of the training data; prediction is performed exactly as in classic language modeling.

[3] http://trec.nist.gov/data/reuters/reuters.html

[4] http://www.mythfolklore.net/andrewlang/

[5] http://incubator.apache.org/opennlp/

or *object* dependencies that linked a verb to that noun phrase, and created an event chain from the sequence of (verb, dependency-type) tuples in the order that they appeared in the text.

4.3 Evaluation Metrics

We follow the approach of Chambers and Jurafsky (2008), evaluating our models for predicting script events in a *narrative cloze* task. The narrative cloze task is inspired by the classic psychological cloze task in which subjects are given a sentence with a word missing and asked to fill in the blank (Taylor, 1953). Similarly, in the narrative cloze task, the system is given a sequence of events from a script where one event is missing, and asked to predict the missing event. The difficulty of a cloze task depends a lot on the context around the missing item – in some cases it may be quite predictable, but in many cases there is no single correct answer, though some answers are more probable than others. Thus, performing well on a cloze task is more about ranking the missing event highly, and not about proposing a single "correct" event.

In this way, narrative cloze is like perplexity in a language model. However, where perplexity measures how good the model is at predicting a script event given the previous events in the script, narrative cloze measures how good the model is at predicting what is missing between events in the script. Thus narrative cloze is somewhat more appropriate to our task, and at the same time simplifies comparisons to prior work.

Rather than manually constructing a set of scripts on which to run the cloze test, we follow Chambers and Jurafsky in reserving a section of our parsed corpora for testing, and then using the event chains from that section as the scripts for which the system must predict events. Given an event chain of length n, we run n cloze tests, with a different one of the n events removed each time to create a *partial script* from the remaining $n-1$ events (see Figure 1). Given a partial script as input, an accurate event prediction model should rank the missing event highly in the *guess list* that it generates as output.

We consider two approaches to evaluating the guess lists produced in response to narrative cloze tests. Both are defined in terms of a test collection C, consisting of $|C|$ partial scripts, where for each partial script c with missing event e, $rank_{sys}(c)$ is the rank of e in the system's guess list for c.

- **Average rank**. The average rank of the missing event across all of the partial scripts:

  $$\frac{1}{|C|}\sum_{c\in C} rank_{sys}(c)$$

  This is the evaluation metric used by Chambers and Jurafsky (2008).

- **Recall@N**. The fraction of partial scripts where the missing event is ranked N or less[6] in the guess list.

  $$\frac{1}{|C|}|\{c : c \in C \wedge rank_{sys}(c) \leq N\}|$$

  In our experiments we use $N = 50$, but results are roughly similar for lower and higher values of N.

Recall@N has not been used before for evaluating models that predict script events, however we suggest that it is a more reliable metric than Average rank. When calculating the average rank, the length of the guess lists will have a significant influence on results. For instance, if a small model is trained with only a small vocabulary of events, its guess lists will usually be shorter than a larger model, but if both models predict the missing event at the bottom of the list, the larger model will get penalized more. Recall@N does not have this issue – it is not influenced by length of the guess lists.

An alternative evaluation metric would have been mean average precision (MAP), a metric commonly used to evaluate information retrieval. Mean average precision reduces to mean reciprocal rank (MRR) when there's only a single answer as in the case of narrative cloze, and would have scored the ranked lists as:

$$\frac{1}{|C|}\sum_{c\in C}\frac{1}{rank_{sys}(c)}$$

Note that mean reciprocal rank has the same issues with guess list length that average rank does. Thus, since it does not aid us in comparing to prior work, and it has the same deficiencies as average rank, we do not report MRR in this article.

[6] Rank 1 is the event that the system predicts is most probable, so we want the missing event to have the smallest rank possible.

	2-skip + bigram prob.	
Chain selection	**Av. rank**	**Recall@50**
all chains	**502**	**0.5179**
long chains	549	0.4951
the longest chain	546	0.4984

Table 1: Chain selection methods for the Reuters corpus - comparison of average ranks and Recall@50.

	2-skip + bigram prob.	
Chain selection	**Av. rank**	**Recall@50**
all chains	1650	0.3376
long chains	**452**	**0.3461**
the longest chain	1534	0.3376

Table 2: Chain selection methods for the Fairy Tale corpus - comparison of average ranks and Recall@50.

	all chains + bigram prob.	
Bigram selection	**Av. rank**	**Recall@50**
regular bigrams	789	0.4886
1-skip bigrams	630	0.4951
2-skip bigrams	**502**	**0.5179**

Table 3: Event bigram selection methods for the Reuters corpus - comparison of average ranks and Recall@50.

	all chains + bigram prob.	
Bigram selection	**Av. rank**	**Recall@50**
regular bigrams	2363	0.3227
1-skip bigrams	1690	**0.3418**
2-skip bigrams	**1650**	0.3376

Table 4: Event bigram selection methods for the Fairy Tales corpus - comparison of average ranks and Recall@50.

4.4 Results

We considered all 27 combinations of our chain selection methods, bigram counting methods, and ranking methods: {*all chains, long chains, the longest chain*}x{*regular bigrams, 1-skip bigrams, 2-skip bigrams*}x{*Chambers & Jurafsky PMI, ordered PMI, bigram probabilities*}. The best among these 27 combinations for the Reuters corpus was {*all chains*}x{*2-skip bigrams*}x{*bigram probabilities*} achieving an average rank of 502 and a Recall@50 of 0.5179.

Since viewing all the combinations at once would be confusing, instead the following sections investigate each decision (selection, counting, ranking) one at a time. While one decision is varied across its three choices, the other decisions are held to their values in the best model above.

4.4.1 Identifying Event Chains

We first try to answer the question: How should representative chains of events be selected from the source text? Tables 1 and 2 show performance when we vary the strategy for selecting event chains, while fixing the counting method to *2-skip bigrams*, and fixing the ranking method to *bigram probabilities*.

For the Reuters collection, we see that using *all chains* gives a lower average rank and a higher Recall@50 than either of the strategies that select a subset of the event chains. The explanation is probably simple: using *all chains* produces more than 700,000 bigrams from the Reuters corpus, while using only the long chains produces only around 300,000. So more data is better data for predicting script events.

For the Fairy Tale collection, *long chains* gives the lowest average rank and highest Recall@50. In this collection, there is apparently some benefit to filtering the shorter event chains, probably because the collection is small enough that the noise introduced from dependency and coreference errors plays a larger role.

4.4.2 Gathering Event Chain Statistics

We next try to answer the question: Given an event chain, how should statistics be gathered from it? Tables 3 and 4 show performance when we vary the strategy for counting event pairs, while fixing the selecting method to *all chains*, and fixing the ranking method to *bigram probabilities*.

For the Reuters corpus, *2-skip bigrams* achieves the lowest average rank and the highest Recall@50. For the Fairy Tale corpus, *1-skip bigrams* and *2-skip bigrams* perform similarly, and both have lower average rank and higher Recall@50 than *regular bigrams*.

Skip-grams probably outperform regular n-grams on both of these corpora because the skip-grams provide many more event pairs over which to calculate statistics: in the Reuters corpus, *regular bigrams* extracts 737,103 bigrams, while *2-skip bigrams* extracts 1,201,185 bigrams. Though skip-grams have not been applied to predicting script events before, it seems that they are a good fit, and better capture statistics about narrative event chains than regular n-grams do.

	all bigrams + 2-skip	
Ranking method	**Av. rank**	**Recall@50**
C&J PMI	2052	0.1954
ordered PMI	3584	0.1694
bigram prob.	**502**	**0.5179**

Table 5: Ranking methods for the Reuters corpus - comparison of average ranks and Recall@50.

	all bigrams + 2-skip	
Ranking method	**Av. rank**	**Recall@50**
C&J PMI	**1455**	0.1975
ordered PMI	2460	0.0467
bigram prob.	1650	**0.3376**

Table 6: Ranking methods for the Fairy Tale corpus - comparison of average ranks and Recall@50.

4.4.3 Predicting Script Events

Finally, we try to answer the question: Given event n-gram statistics, which ranking function best predicts the events for a script? Tables 5 and 6 show performance when we vary the strategy for ranking event predictions, while fixing the selection method to *all chains*, and fixing the counting method to *2-skip bigrams*.

For both Reuters and the Fairy Tale corpus, Recall@50 identifies *bigram probabilities* as the best ranking function by far. On the Reuters corpus the *Chambers & Jurafsky PMI* ranking method achieves Recall@50 of only 0.1954, while *bigram probabilities* ranking method achieves 0.5179. The gap is also quite large on the Fairy Tales corpus: 0.1975 vs. 0.3376.

On the Reuters corpus, average rank also identifies *bigram probabilities* as the best ranking function, yet for the Fairy Tales corpus, *Chambers & Jurafsky PMI* and *bigram probabilities* have similar average ranks. This inconsistency is probably due to the flaws in the average rank evaluation measure that were discussed in Section 4.3 – the measure is overly sensitive to the length of the guess list, particularly when the missing event is ranked lower, as it is likely to be when training on a smaller corpus like the Fairy Tales corpus.

5 Discussion

Our experiments have led us to several important conclusions. First, we have introduced *skip-grams* and proved their utility for acquiring script knowledge – our models that employ skip bigrams score consistently higher on event prediction. By following the intuition that events do not have to appear strictly one after another to be closely semantically related, skip-grams decrease data sparsity and increase the size of the training data.

Second, our novel *bigram probabilities* ranking function outperforms the other ranking methods. In particular, it outperforms the state-of-the-art pointwise mutual information method introduced by Chambers and Jurafsky (2008), and it does so by a large margin, more than doubling the Recall@50 on the Reuters corpus. The key insight here is that, when modeling events in a script, a language-model-like approach better fits the task than a mutual information approach.

Third, we have discussed why Recall@N is a better and more consistent evaluation metric than *Average rank*. However, both evaluation metrics suffer from the strictness of the narrative cloze test, which accepts only one event being the correct event, while it is sometimes very difficult, even for humans, to predict the missing events, and sometimes more solutions are possible and equally correct. In future research, our goal is to design a better evaluation framework which is more suitable for this task, where credit can be given for proposed script events that are appropriate but not identical to the ones observed in a text.

Fourth, we have observed some differences in results between the Reuters and the Fairy Tale corpora. The results for Reuters are consistently better (higher Recall@50, lower average rank), although fairy tales contain a plainer narrative structure, which should be more appropriate to our task. This again leads us to the conclusion that more data (even with more noise as in Reuters) leads to a greater coverage of events, better overall models and, consequently, to more accurate predictions. Still, the Reuters corpus seems to be far from a perfect corpus for research in the automatic acquisition of scripts, since only a small portion of the corpus contains true narratives. Future work must therefore gather a large corpus of true narratives, like fairy tales and children's stories, whose simple plot structures should provide better learning material, both for models predicting script events, and for related tasks like automatic storytelling (McIntyre and Lapata, 2009).

One of the limitations of the work presented here is that it takes a fairly linear, n-gram-based approach to characterizing story structure. We think such an approach is useful because it forms a natural baseline for the task (as it does in many other tasks such as named entity tagging and language modeling). However, story structure is seldom strictly linear, and future work should consider models based on grammatical or discourse links that can capture the more complex nature of script events and story structure.

Acknowledgments

We would like to thank the anonymous reviewers for their constructive comments. This research was carried out as a master thesis in the framework of the TERENCE European project (EU FP7-257410).

References

Nathanael Chambers and Dan Jurafsky. 2008. Unsupervised learning of narrative event chains. In *Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 789–797.

Nathanael Chambers and Dan Jurafsky. 2009. Unsupervised learning of narrative schemas and their participants. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the Association for Computational Linguistics and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, pages 602–610.

Nathanael Chambers and Dan Jurafsky. 2011. Template-based information extraction without the templates. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 976–986.

David Guthrie, Ben Allison, W. Liu, Louise Guthrie, and Yorick Wilks. 2006. A closer look at skip-gram modelling. In *Proceedings of the Fifth international Conference on Language Resources and Evaluation (LREC)*, pages 1222–1225.

Dan Klein and Christopher D. Manning. 2003. Accurate unlexicalized parsing. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics*, pages 423–430.

David D. Lewis, Yiming Yang, Tony G. Rose, and Fan Li. 2004. RCV1: a new benchmark collection for text categorization research. *Journal of Machine Learning Research*, 5:361–397.

Mehdi Manshadi, Reid Swanson, and Andrew S. Gordon. 2008. Learning a probabilistic model of event sequences from internet weblog stories. In *Proceedings of the Twenty-First International Florida Artificial Intelligence Research Society Conference*.

Neil McIntyre and Mirella Lapata. 2009. Learning to tell tales: A data-driven approach to story generation. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the Association for Computational Linguistics and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, pages 217–225.

Neil McIntyre and Mirella Lapata. 2010. Plot induction and evolutionary search for story generation. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 1562–1572.

Michaela Regneri, Alexander Koller, and Manfred Pinkal. 2010. Learning script knowledge with web experiments. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 979–988.

Roger C. Schank and Robert P. Abelson. 1977. *Scripts, plans, goals, and understanding: an inquiry into human knowledge structures*. Lawrence Erlbaum Associates.

Wilson L. Taylor. 1953. Cloze procedure: a new tool for measuring readibility. *Journalism Quarterly*, 30:415–433.