# Learning Script Knowledge with Web Experiments

**Michaela Regneri** **Alexander Koller** **Manfred Pinkal**
Department of Computational Linguistics and Cluster of Excellence
Saarland University, Saarbrücken
{regneri|koller|pinkal}@coli.uni-saarland.de

## Abstract

We describe a novel approach to unsupervised learning of the events that make up a script, along with constraints on their temporal ordering. We collect natural-language descriptions of script-specific event sequences from volunteers over the Internet. Then we compute a graph representation of the script's temporal structure using a multiple sequence alignment algorithm. The evaluation of our system shows that we outperform two informed baselines.

## 1 Introduction

A script is "a standardized sequence of events that describes some stereotypical human activity such as going to a restaurant or visiting a doctor" (Barr and Feigenbaum, 1981). Scripts are fundamental pieces of commonsense knowledge that are shared between the different members of the same culture, and thus a speaker assumes them to be tacitly understood by a hearer when a scenario related to a script is evoked: When one person says "I'm going shopping", it is an acceptable reply to say "did you bring enough money?", because the SHOPPING script involves a 'payment' event, which again involves the transfer of money.

It has long been recognized that text understanding systems would benefit from the implicit information represented by a script (Cullingford, 1977; Mueller, 2004; Miikkulainen, 1995). There are many other potential applications, including automated storytelling (Swanson and Gordon, 2008), anaphora resolution (**?**), and information extraction (**?**).

However, it is also commonly accepted that the large-scale manual formalization of scripts is infeasible. While there have been a few attempts at doing this (Mueller, 1998; Gordon, 2001), efforts in which expert annotators create script knowledge bases clearly don't scale. The same holds true of the script-like structures called "scenario frames" in FrameNet (Baker et al., 1998).

There has recently been a surge of interest in automatically learning script-like knowledge resources from corpora (Chambers and Jurafsky, 2008b; Manshadi et al., 2008); but while these efforts have achieved impressive results, they are limited by the very fact that a lot of scripts – such as SHOPPING – are shared implicit knowledge, and their events are therefore rarely elaborated in text.

In this paper, we propose a different approach to the unsupervised learning of script-like knowledge. We focus on the temporal event structure of scripts; that is, we aim to learn what phrases can describe the same event in a script, and what constraints must hold on the temporal order in which these events occur. We approach this problem by asking non-experts to describe typical event sequences in a given scenario over the Internet. This allows us to assemble large and varied collections of event sequence descriptions (ESDs), which are focused on a single scenario. We then compute a *temporal script graph* for the scenario by identifying corresponding event descriptions using a Multiple Sequence Alignment algorithm from bioinformatics, and converting the alignment into a graph. This graph makes statements about what phrases can describe the same event of a scenario, and in what order these events can take place. Crucially, our algorithm exploits the sequential structure of the ESDs to distinguish event descriptions that occur at different points in the script storyline, even when they are semantically similar. We evaluate our script graph algorithm on ten unseen scenarios, and show that it significantly outperforms a clustering-based baseline.

The paper is structured as follows. We will first position our research in the landscape of related work in Section 2. We will then define how

we understand scripts, and what aspect of scripts we model here, in Section 3. Section 4 describes our data collection method, and Section 5 explains how we use Multiple Sequence Alignment to compute a temporal script graph. We evaluate our system in Section 6 and conclude in Section 7.

## 2 Related Work

Approaches to learning script-like knowledge are not new. For instance, Mooney (1990) describes an early attempt to acquire causal chains, and Smith and Arnold (2009) use a graph-based algorithm to learn temporal script structures. However, to our knowledge, such approaches have never been shown to generalize sufficiently for wide coverage application, and none of them was rigorously evaluated.

More recently, there have been a number of approaches to automatically learning event chains from corpora (Chambers and Jurafsky, 2008b; Chambers and Jurafsky, 2009; Manshadi et al., 2008). These systems typically employ a method for classifying temporal relations between given event descriptions (Chambers et al., 2007; Chambers and Jurafsky, 2008a; Mani et al., 2006). They achieve impressive performance at extracting high-level descriptions of procedures such as a CRIMINAL PROCESS. Because our approach involves directly asking people for event sequence descriptions, it can focus on acquiring specific scripts from arbitrary domains, and we can control the level of granularity at which scripts are described. Furthermore, we believe that much information about scripts is usually left implicit in texts and is therefore easier to learn from our more explicit data. Finally, our system automatically learns different phrases which describe the same event together with the temporal ordering constraints.

Jones and Thompson (2003) describe an approach to identifying different natural language realizations for the same event considering the temporal structure of a scenario. However, they don't aim to acquire or represent the temporal structure of the whole script in the end.

In its ability to learn paraphrases using Multiple Sequence Alignment, our system is related to Barzilay and Lee (2003). Unlike Barzilay and Lee, we do not tackle the general paraphrase problem, but only consider whether two phrases describe the same event in the context of the same script. Furthermore, the atomic units of our alignment process are entire phrases, while in Barzilay and Lee's setting, the atomic units are words.

Finally, it is worth pointing out that our work is placed in the growing landscape of research that attempts to learn linguistic information out of data directly collected from users over the Internet. Some examples are the general acquisition of commonsense knowledge (Singh et al., 2002), the use of browser games for that purpose (von Ahn and Dabbish, 2008), and the collaborative annotation of anaphoric reference (Chamberlain et al., 2009). In particular, the use of the Amazon Mechanical Turk, which we use here, has been evaluated and shown to be useful for language processing tasks (Snow et al., 2008).

## 3 Scripts

Before we delve into the technical details, let us establish some terminology. In this paper, we distinguish *scenarios*, as classes of human activities, from *scripts*, which are stereotypical models of the internal structure of these activities. Where EATING IN A RESTAURANT is a scenario, the script describes a number of events, such as ordering and leaving, that must occur in a certain order in order to constitute an EATING IN A RESTAURANT activity. The classical perspective on scripts (Schank and Abelson, 1977) has been that next to defining some events with temporal constraints, a script also defines their participants and their causal connections.

Here we focus on the narrower task of learning the events that a script consists of, and of modeling and learning the temporal ordering constraints that hold between them. Formally, we will specify a script (in this simplified sense) in terms of a directed graph $G_s = (E_s, T_s)$, where $E_s$ is a set of nodes representing the events of a scenario $s$, and $T_s$ is a set of edges $(e_i, e_k)$ indicating that the event $e_i$ typically happens before $e_k$ in $s$. We call $G_s$ the *temporal script graph (TSG)* for $s$.

Each event in a TSG can usually be expressed with many different natural-language phrases. As the TSG in Fig. 3 illustrates, the first event in the script for EATING IN A FAST FOOD RESTAURANT can be equivalently described as 'walk to the counter' or 'walk up to the counter'; even phrases like 'walk into restaurant', which would not usually be taken as paraphrases of these, can be accepted as describing the same event in the context

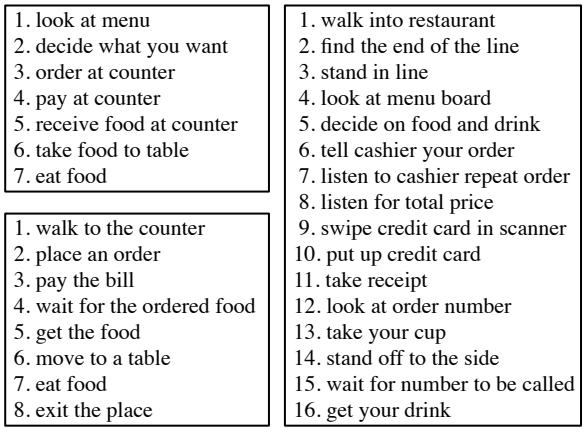


Figure 1: Three event sequence descriptions

of this scenario. We call a natural-language realization of an individual event in the script an *event description*, and we call a sequence of event descriptions that form one particular instance of the script an *event sequence description (ESD)*. Examples of ESDs for the FAST FOOD RESTAURANT script are shown in Fig. 1.

One way to look at a TSG is thus that its nodes are equivalence classes of different phrases that describe the same event; another is that valid ESDs can be generated from a TSG by randomly selecting phrases from some nodes and arranging them in an order that respects the temporal precedence constraints in $T_s$. Our goal in this paper is to take a set of ESDs for a given scenario as our input and then compute a TSG that clusters different descriptions of the same event into the same node, and contains edges that generalize the temporal information encoded in the ESDs.

## 4 Data Acquisition

In order to automatically learn TSGs, we selected 22 scenarios for which we collect ESDs. We deliberately included scenarios of varying complexity, including some that we considered hard to describe (CHILDHOOD, CREATE A HOMEPAGE), scenarios with highly variable orderings between events (MAKING SCRAMBLED EGGS), and scenarios for which we expected cultural differences (WEDDING).

We used the Amazon Mechanical Turk[1] to collect the data. For every scenario, we asked 25 people to enter a typical sequence of events in this scenario, in temporal order and in "bullet point style". We required the annotators to enter at least 5 and at most 16 events. Participants were allowed to skip a scenario if they felt unable to enter events for it, but had to indicate why. We did not restrict the participants (e.g. to native speakers).

In this way, we collected 493 ESDs for the 22 scenarios. People used the possibility to skip a form 57 times. The most frequent explanation for this was that they didn't know how a certain scenario works: The scenario with the highest proportion of skipped forms was CREATE A HOMEPAGE, whereas MAKING SCRAMBLED EGGS was the only one in which nobody skipped a form. Because we did not restrict the participants' inputs, the data was fairly noisy. For the purpose of this study, we manually corrected the data for orthography and filtered out forms that were written in broken English or did not comply with the task (e.g. when users misunderstood the scenario, or did not list the event descriptions in temporal order). Overall we discarded 15% of the ESDs.

Fig. 1 shows three of the ESDs we collected for EATING IN A FAST-FOOD RESTAURANT. As the example illustrates, descriptions differ in their starting points ('walk into restaurant' vs. 'walk to counter'), the granularity of the descriptions ('pay the bill' vs. event descriptions 8–11 in the third sequence), and the events that are mentioned in the sequence (not even 'eat food' is mentioned in all ESDs). Overall, the ESDs we collected consisted of 9 events on average, but their lengths varied widely: For most scenarios, there were significant numbers of ESDs both with the minimum length of 5 and the maximum length of 16 and everything in between. Combined with the fact that 93% of all individual event descriptions occurred only once, this makes it challenging to align the different ESDs with each other.

## 5 Temporal Script Graphs

We will now describe how we compute a temporal script graph out of the collected data. We proceed in two steps. First, we identify phrases from different ESDs that describe the same event by computing a Multiple Sequence Alignment (MSA) of all ESDs for the same scenario. Then we postprocess the MSA and convert it into a temporal script graph, which encodes and generalizes the temporal information contained in the original ESDs.

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

| row | $\mathbf{s}_1$ | $\mathbf{s}_2$ | $\mathbf{s}_3$ | $\mathbf{s}_4$ |
|---|---|---|---|---|
| **1** | ⊘ | walk into restaurant | ⊘ | enter restaurant |
| **2** | ⊘ | ⊘ | walk to the counter | go to counter |
| **3** | ⊘ | find the end of the line | ⊘ | ⊘ |
| **4** | ⊘ | stand in line | ⊘ | ⊘ |
| **5** | look at menu | look at menu board | ⊘ | ⊘ |
| **6** | decide what you want | decide on food and drink | ⊘ | make selection |
| **7** | order at counter | tell cashier your order | place an order | place order |
| **8** | ⊘ | listen to cashier repeat order | ⊘ | ⊘ |
| **9** | pay at counter | ⊘ | pay the bill | pay for food |
| **10** | ⊘ | listen for total price | ⊘ | ⊘ |
| **11** | ⊘ | swipe credit card in scanner | ⊘ | ⊘ |
| **12** | ⊘ | put up credit card | ⊘ | ⊘ |
| **13** | ⊘ | take receipt | ⊘ | ⊘ |
| **14** | ⊘ | look at order number | ⊘ | ⊘ |
| **15** | ⊘ | take your cup | ⊘ | ⊘ |
| **16** | ⊘ | stand off to the side | ⊘ | ⊘ |
| **17** | ⊘ | wait for number to be called | wait for the ordered food | ⊘ |
| **18** | receive food at counter | get your drink | get the food | pick up order |
| **19** | ⊘ | ⊘ | ⊘ | pick up condiments |
| **20** | take food to table | ⊘ | move to a table | go to table |
| **21** | eat food | ⊘ | eat food | consume food |
| **22** | ⊘ | ⊘ | ⊘ | clear tray |
| **22** | ⊘ | ⊘ | exit the place | ⊘ |

Figure 2: A MSA of four event sequence descriptions

### 5.1 Multiple Sequence Alignment

The problem of computing Multiple Sequence Alignments comes from bioinformatics, where it is typically used to find corresponding elements in proteins or DNA (Durbin et al., 1998).

A sequence alignment algorithm takes as its input some *sequences* $s_1, \ldots, s_n \in \Sigma^*$ over some alphabet $\Sigma$, along with a *cost function* $c_m : \Sigma \times \Sigma \rightarrow \mathbb{R}$ for substitutions and *gap costs* $c_{gap} \in \mathbb{R}$ for insertions and deletions. In bioinformatics, the elements of $\Sigma$ could be nucleotides and a sequence could be a DNA sequence; in our case, $\Sigma$ contains the individual event descriptions in our data, and the sequences are the ESDs.

A *Multiple Sequence Alignment* $A$ of these sequences is then a matrix as in Fig. 2: The $i$-th column of $A$ is the sequence $s_i$, possibly with some gaps ("⊘") interspersed between the symbols of $s_i$, such that each row contains at least one non-gap. If a row contains two non-gaps, we take these symbols to be *aligned*; aligning a non-gap with a gap can be thought of as an insertion or deletion.

Each sequence alignment $A$ can be assigned a *cost* $c(A)$ in the following way:

$$c(A) = c_{gap} \cdot \Sigma_{\oslash} + \sum_{i=1}^{n} \sum_{\substack{j=1,\\ a_{ji} \neq \oslash}}^{m} \sum_{\substack{k=j+1,\\ a_{ki} \neq \oslash}}^{m} c_m(a_{ji}, a_{ki})$$

where $\Sigma_{\oslash}$ is the number of gaps in $A$, $n$ is the number of rows and $m$ the number of sequences. In other words, we sum up the alignment cost for any two symbols from $\Sigma$ that are aligned with each other, and add the gap cost for each gap. There is an algorithm that computes cheapest pairwise alignments (i.e. $n = 2$) in polynomial time (Needleman and Wunsch, 1970). For $n > 2$, the problem is NP-complete, but there are efficient algorithms that approximate the cheapest MSAs by aligning two sequences first, considering the result as a single sequence whose elements are pairs, and repeating this process until all sequences are incorporated in the MSA (Higgins and Sharp, 1988).

### 5.2 Semantic similarity

In order to apply MSA to the problem of aligning ESDs, we choose $\Sigma$ to be the set of all individual event descriptions in a given scenario. Intuitively, we want the MSA to prefer the alignment of two phrases if they are semantically similar, i.e. it should cost more to align 'exit' with 'eat' than 'exit' with 'leave'. Thus we take a measure of semantic (dis)similarity as the cost function $c_m$.

The phrases to be compared are written in bullet-point style. They are typically short and elliptic (no overt subject), they lack determiners and use infinitive or present progressive form for the main verb. Also, the lexicon differs considerably from usual newspaper corpora. For these reasons, standard methods for similarity assessment are not straightforwardly applicable: Simple bag-of-words approaches do not provide sufficiently good results, and standard taggers and parsers cannot process our descriptions with sufficient accuracy.

We therefore employ a simple, robust heuristics, which is tailored to our data and provides very shallow dependency-style syntactic information.

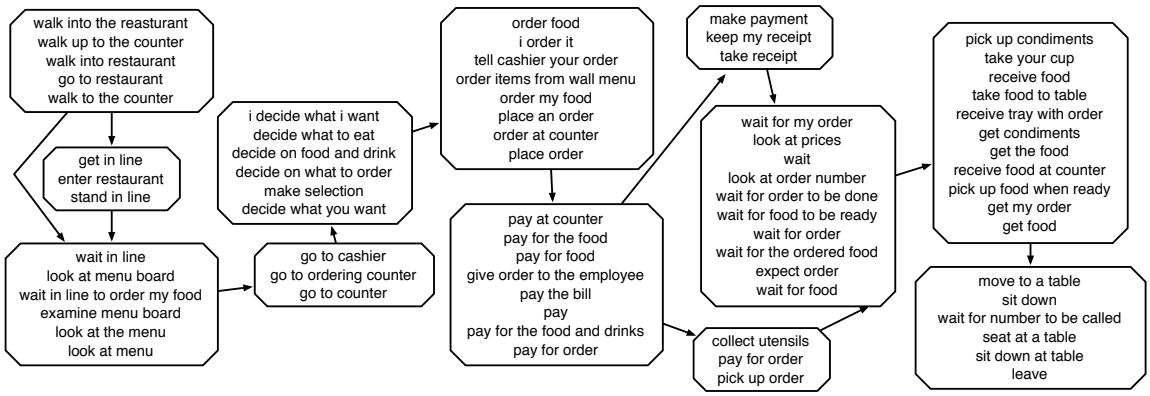


Figure 3: An extract from the graph computed for EATING IN A FAST FOOD RESTAURANT

We identify the first potential verb of the phrase (according to the POS information provided by WordNet) as the predicate, the preceding noun (if any) as subject, and all following potential nouns as objects. (With this fairly crude tagging method, we also count nouns in prepositional phrases as "objects".)

On the basis of this pseudo-parse, we compute the similarity measure $sim$:

$$sim = \alpha \cdot pred + \beta \cdot subj + \gamma \cdot obj$$

where $pred$, $subj$, and $obj$ are the similarity values for predicates, subjects and objects respectively, and $\alpha, \beta, \gamma$ are weights. If a constituent is not present in one of the phrases to compare, we set its weight to zero and redistribute it over the other weights. We fix the individual similarity scores $pred$, $subj$, and $obj$ depending on the WordNet relation between the most similar WordNet senses of the respective lemmas (100 for synonyms, 0 for lemmas without any relation, and intermediate numbers for different kind of WordNet links).

We optimized the values for $pred$, $subj$, and $obj$ as well as the weights $\alpha$, $\beta$ and $\gamma$ using a held-out development set of scenarios. Our experiments showed that in most cases, the verb contributes the largest part to the similarity (accordingly, $\alpha$ needs to be higher than the other factors). We achieved improved accuracy by distinguishing a class of verbs that contribute little to the meaning of the phrase (i.e., support verbs, verbs of movement, and the verb "get"), and assigning them a separate, lower $\alpha$.

### 5.3 Building Temporal Script Graphs

We can now compute a low-cost MSA for each scenario out of the ESDs. From this alignment, we extract a temporal script graph, in the following way. First, we construct an initial graph which has one node for each row of the MSA as in Fig. 2. We interpret each node of the graph as representing a single event in the script, and the phrases that are collected in the node as different descriptions of this event; that is, we claim that these phrases are paraphrases in the context of this scenario. We then add an edge $(u, v)$ to the graph iff (1) $u \neq v$, (2) there was at least one ESD in the original data in which some phrase in $u$ directly preceded some phrase in $v$, and (3) if a single ESD contains a phrase from $u$ and from $v$, the phrase from $u$ directly precedes the one from $v$. In terms of the MSA, this means that if a phrase from $u$ comes from the same column as a phrase from $v$, there are at most some gaps between them. This initial graph represents exactly the same information as the MSA, in a different notation.

The graph is automatically post-processed in a second step to simplify it and eliminate noise that caused MSA errors. At first we prune spurious nodes which contain only one event description. Then we refine the graph by merging nodes whose elements should have been aligned in the first place but were missed by the MSA. We merge two nodes if they satisfy certain structural and semantic constraints.

The *semantic* constraints check whether the event descriptions of the merged node would be sufficiently consistent according to the similarity measure from Section 5.2. To check whether we can merge two nodes $u$ and $v$, we use an unsupervised clustering algorithm (Flake et al., 2004) to

first cluster the event descriptions in $u$ and $v$ separately. Then we combine the event descriptions from $u$ and $v$ and cluster the resulting set. If the union has more clusters than either $u$ or $v$, we assume the nodes to be too dissimilar for merging.

The *structural* constraints depend on the graph structure. We only merge two nodes $u$ and $v$ if their event descriptions come from different sequences and one of the following conditions holds:

- $u$ and $v$ have the same parent;
- $u$ has only one parent, $v$ is its only child;
- $v$ has only one child and is the only child of $u$;
- all children of $u$ (except for $v$) are also children of $v$.

These structural constraints prevent the merging algorithm from introducing new temporal relations that are not supported by the input ESDs.

We take the output of this post-processing step as the temporal script graph. An excerpt of the graph we obtain for our running example is shown in Fig. 3. One node created by the node merging step was the top left one, which combines one original node containing 'walk into restaurant' and another with 'go to restaurant'. The graph mostly groups phrases together into event nodes quite well, although there are some exceptions, such as the 'collect utensils' node. Similarly, the temporal information in the graph is pretty accurate. But perhaps most importantly, our MSA-based algorithm manages to keep similar phrases like 'wait in line' and 'wait for my order' apart by exploiting the sequential structure of the input ESDs.

## 6 Evaluation

We evaluated the two core aspects of our system: its ability to recognize descriptions of the same event (*paraphrases*) and the resulting temporal constraints it defines on the event descriptions (*happens-before relation*). We compare our approach to two baseline systems and show that our system outperforms both baselines and sometimes even comes close to our upper bound.

### 6.1 Method

We selected ten scenarios which we did not use for development purposes, five of them taken from the corpus described in Section 4, the other five from the OMICS corpus.[2] The OMICS corpus is a freely available, web-collected corpus by the Open Mind Initiative (Singh et al., 2002). It contains several *stories* ($\approx$ scenarios) consisting of multiple ESDs. The corpus strongly resembles ours in language style and information provided, but is restricted to "indoor activities" and contains much more data than our collection (175 scenarios with more than 40 ESDs each).

For each scenario, we created a *paraphrase set* out of 30 randomly selected pairs of event descriptions which the system classified as paraphrases and 30 completely random pairs. The *happens-before set* consisted of 30 pairs classified as *happens-before*, 30 random pairs and additionally all 60 pairs in reverse order. We added the reversed pairs to check whether the raters really prefer one direction or whether they accept both and were biased by the order of presentation.

We presented each pair to 5 non-experts, all US residents, via Mechanical Turk. For the paraphrase set, an exemplary question we asked the rater looks as follows, instantiating the Scenario and the two descriptions to compare appropriately:

> Imagine two people, both telling a story about SCENARIO. Could the first one say *event*$_2$ to describe the same part of the story that the second one describes with *event*$_1$ ?

For the happens-before task, the question template was the following:

> Imagine somebody telling a story about SCENARIO in which the events *event*$_1$ and *event*$_2$ occur. Would *event*$_1$ normally happen before *event*$_2$?

We constructed a gold standard by a majority decision of the raters. An expert rater adjudicated the pairs with a 3:2 vote ratio.

### 6.2 Upper Bound and Baselines

To show the contributions of the different system components, we implemented two baselines:

**Clustering Baseline:** We employed an unsupervised clustering algorithm (Flake et al., 2004) and fed it all event descriptions of a scenario. We first created a similarity graph with one node per event description. Each pair of nodes is connected

[2] http://openmind.hri-us.com/

| | Scenario | Precision | | | Recall | | | F-Score | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | *sys* | base$_{cl}$ | base$_{lev}$ | *sys* | base$_{cl}$ | base$_{lev}$ | *sys* | base$_{cl}$ | base$_{lev}$ | upper |
| MTurk | pay with credit card | 0.52 | 0.43 | 0.50 | 0.84 | 0.89 | 0.11 | **0.64** | 0.58 | • 0.17 | 0.60 |
| | eat in restaurant | 0.70 | 0.42 | 0.75 | 0.88 | 1.00 | 0.25 | **0.78** | • 0.59 | • 0.38 | • 0.92 |
| | iron clothes I | 0.52 | 0.32 | 1.00 | 0.94 | 1.00 | 0.12 | **0.67** | • 0.48 | • 0.21 | • 0.82 |
| | cook scrambled eggs | 0.58 | 0.34 | 0.50 | 0.86 | 0.95 | 0.10 | **0.69** | • 0.50 | • 0.16 | • 0.91 |
| | take a bus | 0.65 | 0.42 | 0.40 | 0.87 | 1.00 | 0.09 | **0.74** | • 0.59 | • 0.14 | • 0.88 |
| OMICS | answer the phone | 0.93 | 0.45 | 0.70 | 0.85 | 1.00 | 0.21 | **0.89** | • 0.71 | • 0.33 | 0.79 |
| | buy from vending machine | 0.59 | 0.43 | 0.59 | 0.83 | 1.00 | 0.54 | **0.69** | 0.60 | 0.57 | 0.80 |
| | iron clothes II | 0.57 | 0.30 | 0.33 | 0.94 | 1.00 | 0.22 | **0.71** | • 0.46 | • 0.27 | 0.77 |
| | make coffee | 0.50 | 0.27 | 0.56 | 0.94 | 1.00 | 0.31 | **0.65** | • 0.42 | ○ 0.40 | • 0.82 |
| | make omelette | 0.75 | 0.54 | 0.67 | 0.92 | 0.96 | 0.23 | **0.83** | • 0.69 | • 0.34 | 0.85 |
| | Average | 0.63 | 0.40 | 0.60 | 0.89 | 0.98 | 0.22 | **0.73** | 0.56 | 0.30 | 0.82 |

Figure 4: Results for paraphrasing task; significance of difference to *sys*: • : $p \leq 0.01$, ○ : $p \leq 0.1$

with a weighted edge; the weight reflects the semantic similarity of the nodes' event descriptions as described in Section 5.2. To include all input information on inequality of events, we did not allow for edges between nodes containing two descriptions occurring together in one ESD. The underlying assumption here is that two different event descriptions of the same ESD always represent distinct events.

The clustering algorithm uses a parameter which influences the cluster granularity, without determining the exact number of clusters beforehand. We optimized this parameter automatically for each scenario: The system picks the value that yields the optimal result with respect to density and distance of the clusters (Flake et al., 2004), i.e. the elements of each cluster are as similar as possible to each other, and as dissimilar as possible to the elements of all other clusters.

The clustering baseline considers two phrases as paraphrases if they are in the same cluster. It claims a happens-before relation between phrases $e$ and $f$ if some phrase in $e$'s cluster precedes some phrase in $f$'s cluster in the original ESDs. With this baseline, we can show the contribution of MSA.

**Levenshtein Baseline:** This system follows the same steps as our system, but using Levenshtein distance as the measure of semantic similarity for MSA and for node merging (cf. Section 5.3). This lets us measure the contribution of the more fine-grained similarity function. We computed Levenshtein distance as the character-wise edit distance on the phrases, divided by the phrases' character length so as to get comparable values for shorter and longer phrases. The gap costs for MSA with Levenshtein were optimized on our development set so as to produce the best possible alignment.

**Upper bound:** We also compared our system to a human-performance upper bound. Because no single annotator rated all pairs of ESDs, we constructed a "virtual annotator" as a point of comparison, by randomly selecting one of the human annotations for each pair.

### 6.3 Results

We calculated precision, recall, and f-score for our system, the baselines, and the upper bound as follows, with $all_{system}$ being the number of pairs labelled as *paraphrase* or *happens-before*, $all_{gold}$ as the respective number of pairs in the gold standard and $correct$ as the number of pairs labeled correctly by the system.

$$precision = \frac{correct}{all_{system}} \qquad recall = \frac{correct}{all_{gold}}$$

$$f\text{-}score = \frac{2 * precision * recall}{precision + recall}$$

The tables in Fig. 4 and 5 show the results of our system and the reference values; Fig. 4 describes the paraphrasing task and Fig. 5 the happens-before task. The upper half of the tables describes the test sets from our own corpus, the remainder refers to OMICS data. The columns labelled *sys* contain the results of our system, *base*$_{cl}$ describes the clustering baseline and *base*$_{lev}$ the Levenshtein baseline. The f-score for the upper bound is in the column *upper*. For the f-score values, we calculated the significance for the difference between our system and the baselines as well as the upper bound, using a resampling test (Edgington, 1986). The values marked with • differ from our system significantly at a level of $p \leq 0.01$, ○ marks a level of $p \leq 0.1$. The remaining values are not significant with $p \leq 0.1$. (For the average values, no sig-

| | Scenario | Precision | | | Recall | | | F-Score | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | *sys* | base$_{cl}$ | base$_{lev}$ | *sys* | base$_{cl}$ | base$_{lev}$ | *sys* | base$_{cl}$ | base$_{lev}$ | upper |
| MTurk | pay with credit card | 0.86 | 0.49 | 0.65 | 0.84 | 0.74 | 0.45 | **0.85** | • 0.59 | • 0.53 | 0.92 |
| | eat in restaurant | 0.78 | 0.48 | 0.68 | 0.84 | 0.98 | 0.75 | **0.81** | • 0.64 | 0.71 | • 0.95 |
| | iron clothes I | 0.78 | 0.54 | 0.75 | 0.72 | 0.95 | 0.53 | **0.75** | 0.69 | • 0.62 | • 0.92 |
| | cook scrambled eggs | 0.67 | 0.54 | 0.55 | 0.64 | 0.98 | 0.69 | 0.66 | ***0.70*** | 0.61 | • 0.88 |
| | take a bus | 0.80 | 0.49 | 0.68 | 0.80 | 1.00 | 0.37 | **0.80** | • 0.66 | • 0.48 | • 0.96 |
| OMICS | answer the phone | 0.83 | 0.48 | 0.79 | 0.86 | 1.00 | 0.96 | 0.84 | • 0.64 | ***0.87*** | 0.90 |
| | buy from vending machine | 0.84 | 0.51 | 0.69 | 0.85 | 0.90 | 0.75 | **0.84** | • 0.66 | ○ 0.71 | 0.83 |
| | iron clothes II | 0.78 | 0.48 | 0.75 | 0.80 | 0.96 | 0.66 | **0.79** | • 0.64 | 0.70 | 0.84 |
| | make coffee | 0.70 | 0.55 | 0.50 | 0.78 | 1.00 | 0.55 | **0.74** | 0.71 | ○ 0.53 | ○ 0.83 |
| | make omelette | 0.70 | 0.55 | 0.79 | 0.83 | 0.93 | 0.82 | 0.76 | ○ 0.69 | ***0.81*** | • 0.92 |
| | Average | 0.77 | 0.51 | 0.68 | 0.80 | 0.95 | 0.65 | **0.78** | 0.66 | 0.66 | 0.90 |

Figure 5: Results for happens-before task; significance of difference to *sys*: • : $p \leq 0.01$, ○ : $p \leq 0.1$

nificance is calculated because this does not make sense for scenario-wise evaluation.)

**Paraphrase task:** Our system outperforms both baselines clearly, reaching significantly higher f-scores in 17 of 20 cases. Moreover, for five scenarios, the upper bound does not differ significantly from our system. For judging the precision, consider that the test set is slightly biased: Labeling all pairs with the majority category (*no paraphrase*) would result in a precision of 0.64. However, recall and f-score for this trivial lower bound would be 0.

The only scenario in which our system doesn't score very well is BUY FROM A VENDING MACHINE, where the upper bound is not significantly better either. The clustering system, which can't exploit the sequential information from the ESDs, has trouble distinguishing semantically similar phrases (high recall, low precision). The Levenshtein similarity measure, on the other hand, is too restrictive and thus results in comparatively high precisions, but very low recall.

**Happens-before task:** In most cases, and on average, our system is superior to both baselines. Where a baseline system performs better than ours, the differences are not significant. In four cases, our system does not differ significantly from the upper bound. Regarding precision, our system outperforms both baselines in all scenarios except one (MAKE OMELETTE).

Again the clustering baseline is not fine-grained enough and suffers from poor precision, only slightly better than the majority baseline. The Levenshtein baseline gets mostly poor recall, except for ANSWER THE PHONE: to describe this scenario, people used very similar wording. In such a scenario, adding lexical knowledge to the sequential information makes less of a difference.

On average, the baselines do much better here than for the paraphrase task. This is because once a system decides on paraphrase clusters that are essentially correct, it can retrieve correct information about the temporal order directly from the original ESDs.

Both tables illustrate that the task complexity strongly depends on the scenario: Scripts that allow for a lot of variation with respect to ordering (such as COOK SCRAMBLED EGGS) are particularly challenging for our system. This is due to the fact that our current system can neither represent nor find out that two events can happen in arbitrary order (e.g., 'take out pan' and 'take out bowl').

One striking difference between the performance of our system on the OMICS data and on our own dataset is the relation to the upper bound: On our own data, the upper bound is almost always significantly better than our system, whereas significant differences are rare on OMICS. This difference bears further analysis; we speculate it might be caused either by the increased amount of training data in OMICS or by differences in language (e.g., fewer anaphoric references).

## 7 Conclusion

We conclude with a summary of this paper and some discussion along with hints to future work in the last part.

### 7.1 Summary

In this paper, we have described a novel approach to the unsupervised learning of temporal script information. Our approach differs from previous work in that we collect training data by directly asking non-expert users to describe a scenario, and

then apply a Multiple Sequence Alignment algorithm to extract scenario-specific paraphrase and temporal ordering information. We showed that our system outperforms two baselines and sometimes approaches human-level performance, especially because it can exploit the sequential structure of the script descriptions to separate clusters of semantically similar events.

### 7.2 Discussion and Future Work

We believe that we can scale this approach to model a large numbers of scenarios representing implicit shared knowledge. To realize this goal, we are going to automatize several processing steps that were done manually for the current study. We will restrict the user input to lexicon words to avoid manual orthography correction. Further, we will implement some heuristics to filter unusable instances by matching them with the remaining data. As far as the data collection is concerned, we plan to replace the web form with a browser game, following the example of von Ahn and Dabbish (2008). This game will feature an algorithm that can generate new candidate scenarios without any supervision, for instance by identifying suitable sub-events of collected scripts (e.g. inducing data collection for PAY as *sub*-event sequence of GO SHOPPING)

On the technical side, we intend to address the question of detecting participants of the scripts and integrating them into the graphs, Further, we plan to move on to more elaborate data structures than our current TSGs, and then identify and represent script elements like optional events, alternative events for the same step, and events that can occur in arbitrary order.

Because our approach gathers information from volunteers on the Web, it is limited by the knowledge of these volunteers. We expect it will perform best for general commonsense knowledge; culture-specific knowledge or domain-specific expert knowledge will be hard for it to learn. This limitation could be addressed by targeting specific groups of online users, or by complementing our approach with corpus-based methods, which might perform well exactly where ours does not.

### Acknowledgements

We want to thank Dustin Smith for the OMICS data, Alexis Palmer for her support with Amazon Mechanical Turk, Nils Bendfeldt for the creation of all web forms and Ines Rehbein for her effort with several parsing experiments. In particular, we thank the anonymous reviewers for their helpful comments. – This work was funded by the Cluster of Excellence "Multimodal Computing and Interaction" in the German Excellence Initiative.

## References

Collin F. Baker, Charles J. Fillmore, and John B. Lowe. 1998. The berkeley framenet project. In *Proceedings of the 17th international conference on Computational linguistics*, pages 86–90, Morristown, NJ, USA. Association for Computational Linguistics.

Avron Barr and Edward Feigenbaum. 1981. *The Handbook of Artificial Intelligence, Volume 1*. William Kaufman Inc., Los Altos, CA.

Regina Barzilay and Lillian Lee. 2003. Learning to paraphrase: An unsupervised approach using multiple-sequence alignment. In *Proceedings of HLT-NAACL 2003*.

Jon Chamberlain, Massimo Poesio, and Udo Kruschwitz. 2009. A demonstration of human computation using the phrase detectives annotation game. In *KDD Workshop on Human Computation*. ACM.

Nathanael Chambers and Dan Jurafsky. 2008a. Jointly combining implicit constraints improves temporal ordering. In *Proceedings of EMNLP 2008*.

Nathanael Chambers and Dan Jurafsky. 2008b. Unsupervised learning of narrative event chains. In *Proceedings of ACL-08: HLT*.

Nathanael Chambers and Dan Jurafsky. 2009. Unsupervised learning of narrative schemas and their participants. In *Proceedings of ACL-IJCNLP 2009*.

Nathanael Chambers, Shan Wang, and Dan Jurafsky. 2007. Classifying temporal relations between events. In *Proceedings of ACL-07: Interactive Poster and Demonstration Sessions*.

Richard Edward Cullingford. 1977. *Script application: computer understanding of newspaper stories.* Ph.D. thesis, Yale University, New Haven, CT, USA.

Richard Durbin, Sean Eddy, Anders Krogh, and Graeme Mitchison. 1998. *Biological Sequence Analysis*. Cambridge University Press.

Eugene S Edgington. 1986. *Randomization tests*. Marcel Dekker, Inc., New York, NY, USA.

Gary W. Flake, Robert E. Tarjan, and Kostas Tsioutsiouliklis. 2004. Graph clustering and minimum cut trees. *Internet Mathematics*, 1(4).

Andrew S. Gordon. 2001. Browsing image collections with representations of common-sense activities. *JASIST*, 52(11).

Desmond G. Higgins and Paul M. Sharp. 1988. Clustal: a package for performing multiple sequence alignment on a microcomputer. *Gene*, 73(1).

Dominic R. Jones and Cynthia A. Thompson. 2003. Identifying events using similarity and context. In *Proceedings of CoNNL-2003*.

Inderjeet Mani, Marc Verhagen, Ben Wellner, Chong Min Lee, and James Pustejovsky. 2006. Machine learning of temporal relations. In *COLING/ACL-2006*.

Mehdi Manshadi, Reid Swanson, and Andrew S. Gordon. 2008. Learning a probabilistic model of event sequences from internet weblog stories. In *Proceedings of the 21st FLAIRS Conference*.

Risto Miikkulainen. 1995. Script-based inference and memory retrieval in subsymbolic story processing. *Applied Intelligence*, 5(2), 04.

Raymond J. Mooney. 1990. Learning plan schemata from observation: Explanation-based learning for plan recognition. *Cognitive Science*, 14(4).

Erik T. Mueller. 1998. *Natural Language Processing with Thought Treasure*. Signiform.

Erik T. Mueller. 2004. Understanding script-based stories using commonsense reasoning. *Cognitive Systems Research*, 5(4).

Saul B. Needleman and Christian D. Wunsch. 1970. A general method applicable to the search for similarities in the amino acid sequence of two proteins. *Journal of molecular biology*, 48(3), March.

Roger C. Schank and Robert P. Abelson. 1977. *Scripts, Plans, Goals and Understanding*. Lawrence Erlbaum, Hillsdale, NJ.

Push Singh, Thomas Lin, Erik T. Mueller, Grace Lim, Travell Perkins, and Wan L. Zhu. 2002. Open mind common sense: Knowledge acquisition from the general public. In *On the Move to Meaningful Internet Systems - DOA, CoopIS and ODBASE 2002*, London, UK. Springer-Verlag.

Dustin Smith and Kenneth C. Arnold. 2009. Learning hierarchical plans by reading simple english narratives. In *Proceedings of the Commonsense Workshop at IUI-09*.

Rion Snow, Brendan O'Connor, Daniel Jurafsky, and Andrew Y. Ng. 2008. Cheap and fast—but is it good?: evaluating non-expert annotations for natural language tasks. In *Proceedings of EMNLP 2008*.

Reid Swanson and Andrew S. Gordon. 2008. Say anything: A massively collaborative open domain story writing companion. In *Proceedings of ICIDS 2008*.

Luis von Ahn and Laura Dabbish. 2008. Designing games with a purpose. *Commun. ACM*, 51(8).