AI Communications 31 (2018) 465–483
DOI 10.3233/AIC-170564
IOS Press

Leveraging knowledge bases for future prediction with memory comparison networks

Daniel Andrade [a,*], Bing Bai [b], Ramkumar Rajendran [c] and Yotaro Watanabe [d]
[a] *Security Research Laboratories, NEC Corporation, Japan*
E-mail: s-andrade@cj.jp.nec.com
[b] *Department of Machine Learning, NEC Laboratories America, USA*
E-mail: bbai@nec-labs.com
[c] *Computer Science Department, Vanderbilt University, USA*
E-mail: ramkumar.rajendran@vanderbilt.edu
[d] *PKSHA Technology Inc., Japan*
E-mail: y_watanabe@pkshatech.com

Abstract. Making predictions about what might happen in the future is important for reacting adequately in many situations. For example, observing that "Man kidnaps girl" may have the consequence that "Man kills girl". While this is part of common sense reasoning for humans, it is not obvious how machines can acquire and generalize over such knowledge. In this article, we propose a new type of memory network that can predict the next future event also for observations that are not in the knowledge base. We evaluate our proposed method on two knowledge bases: Reuters KB (events from news articles) and Regneri KB (events from scripts). For both knowledge bases, our proposed method shows similar or better prediction accuracy on unseen events (or scripts) than recently proposed deep neural networks and rankSVM. We also demonstrate that the attention mechanism of our proposed method can be helpful for error analysis and manual expansion of the knowledge base.

Keywords: Future prediction, neural network, interpretability, knowledge base, reasoning

1. Introduction

Making predictions about what might happen in the future is important for reacting adequately in many situations. For example, observing that "Man kidnaps girl" may have the consequence that "Man kills girl". While this is part of common sense reasoning for humans, it is not obvious how machines can learn and generalize over such knowledge automatically.

In this article, we focus on the task of given an observed event (e.g. "Man kidnaps girl"), to score possible next future events (e.g. "Man kills girl", or "Man fires girl"). In the first part, we explain possible ways to aquire a knowledge base of temporal relations from existing resources. In the second part, we propose a new method, which we call Memory Comparison Network (MCN), for distinguishing between likely and unlikely future events. MCN is a memory network that can leverage an existing knowledge base of temporal relations for future prediction of unseen events.

Since there are only few available resources for temporal relations between events, we discuss, in Section 2, how to create a knowledge base of temporal relations. We exploit that if an entailment relation holds between two events, then the entailed event is likely to be not a new future event. For example, the phrase "man kissed woman" entails "man met woman", where "man met woman" happens before (not after) "man kissed woman". To find such entailments, we exploit the entailment relation of verbs in WordNet [11]. Verbs that tend to be in a temporal (happens-before) relation have been extracted on a large scale and are freely available in VerbOcean [8]. For example, the event (subject, buy, object) tends to be temporally preceding the event (subject, use, object). Based on these insights, we created a knowledge base of happens-before relations between events that were extracted from the Reuters news corpus (Reuters KB). Additionally, we created a knowledge base extracted from a standard data set for script learning [33] (Regneri KB).

However, all knowledge bases are incomplete, and therefore it is key to develop a method that can gener-

*Corresponding author. E-mail: s-andrade@cj.jp.nec.com.

alize over existing temporal relations. For that purpose, in Section 3, we present our proposed method MCN to predict future events given an observed event triplet (subject, verb, object). In order to allow the model to generalize to unseen events, we adopt a deep learning structure such that the semantics of unseen events can be learned through word/event embeddings. Our proposed method can learn to compare and combine the similarity of observed events to the event relations saved in memory. Furthermore, by using an attention mechanism, our method is able to provide evidence for its future prediction decision.

We note that recently several neural network architectures have been proposed to exploit word embeddings for temporal relation detection [14,23,35]. Though, these networks cannot provide any evidence for their output, which can be important for the human decision maker. We discuss the application of these existing methods to our task of future prediction in Section 4.

In Section 5, we evaluate our proposed method and existing methods on two knowledge bases: Reuters KB and Regneri KB. For both knowledge bases, our proposed method shows similar or better prediction accuracy on unseen events than the recently proposed (deep) neural networks and rankSVM [39]. Subsequently, we analyze our proposed method's design choices and its attention mechanism (Section 6). We discuss in more detail related work in Section 7. Finally, we conclude our article in Section 8.

2. Knowledge database construction

In this section, we describe our approach for extracting a knowledge base of happens-before relations with the help of lexical resources (Section 2.1) and script data (Section 2.2)

Our main focus is on distinguishing future events from other events. In texts, like news stories, an event e_l is more likely to have happened before event e_r (temporal order), if e_l occurs earlier in the text than e_r (textual order). However, there are also many situations where that is not the case: re-phrasing, introducing background knowledge, conclusions, etc. One obvious solution is discourse parsers. But without explicit temporal markers, they suffer from low recall [46]. Therefore, in practice, most script-learning systems use textual order as a proxy for temporal order [29,30,35].

However, we found that using textual order from text, leads to a high number of wrong temporal relations (details see Section 2.1). Therefore, we explore here the usage of two lexical resources, WordNet and VerbOcean, which have both high accuracy. WordNet is a carefully manually curated database of mainly synonyms [11]. VerbOcean contains temporal relations between verbs that were extracted from corpora via manually created bootstrapping rules followed by statistically filtering [8].

We assume common knowledge is given in the form of simple relations (or rules) like

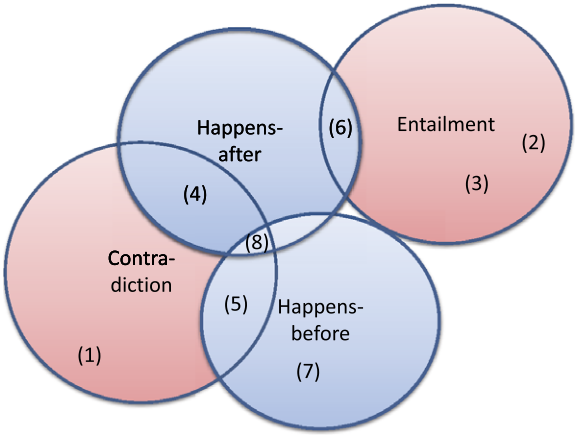
$$(\text{company, buy, share}) \rightarrow (\text{company, use, share}),$$

where "$\rightarrow$" denotes the temporal happens-before relation, meaning the event at the arrow tail happens before the event at the arrow head. In Table 1 we give the definitions of all temporal and logical relations that we use in this paper. Like the definition for entailment as in [10], all our definitions are based on defeasible reasoning.

To extract such common knowledge rules we explore the use of the lexical resources WordNet [11] and VerbOcean [8]. As also partly mentioned in [11], logical and temporal relations are not independent, but an interesting overlap exists as illustrated in Fig. 1. We emphasize that, for temporal relations, the situation is not always as clear cut as shown in Fig. 1 (e.g. repeated actions). Nevertheless, there is a tendency that a tem-

Table 1
Symbol and corresponding definition for each logical and temporal relation

Symbol	Meaning
$A \Rightarrow B$	Entailment: a human reading A would infer that B is with high probability true (same as definition given in [10]).
$A \Leftarrow B$	Backward Entailment: a human reading B would infer that A is with high probability true.
$A \Leftrightarrow B$	Paraphrase: a human reading A would infer that B is with high probability true, and vice versa.
$A \neq B$	Contradiction: a human reading A and B would infer that with high probability both cannot be true at the same time.
$A \rightarrow B$	Happens-before: a human reading A considers it likely that some time later B will happen.
$A \leftarrow B$	Happens-after: a human reading B considers it likely that some time later A will happen.
$A \nrightarrow B$	Not Happens-before: a human reading A considers it unlikely that sometimes later B will happen.



Examples
(1) "president likes media" ≠ "president hates media"
(2) "company donates money" ⇒ "company gives money"
(3) "John buys food" ⇔ "John purchases food"
(4) "minister leaves factory" ← and also ≠"minister enters factory"
(5) "John starts marathon" → and also ≠ "John finishes marathon"
(6) "governor kisses girlfriend" ← and also ⇒ "governor meets girlfriend"
(7) "John buys orange" → "John eats orange"
(8) "X's share falls 10%" →, ←, and also ≠ "X's share rises 10%"

Fig. 1. Illustration and examples of logical (entailment, contradiction) and temporal (happens-before, happens-after) relation types.

poral relation is either true or false. In particular, in the following, we consider "wrong" happens-before relations, as less likely to be true than "correct" happens-before relations.

2.1. Reuters knowledge base

For simplicity, we restrict our investigation here to events of the form (subject, verb, object). All events are extracted from around 790k news articles in Reuters [17]. We preprocessed the English Reuters articles using the Stanford dependency parser and co-reference resolution [19]. We perform lemmatization[1] of all words, and for subjects and objects, we considered only the head words.

An event is defined as a (subject, verb, object) triplet, denoted as (S, V, O). All relations are defined between two events of the form (S, V_l, O) and (S, V_r, O), where subject S and object O are the same. For extracting the temporal relations we consider only event pairs from the same article. Furthermore, the event (S, V_l, O) is written in the article before (S, V_r, O).

[1]Reduction to the base form of a word. For example, "eating" becomes "eat".

Positive samples. We extract positive samples of the form $(S, V_l, O) \rightarrow (S, V_r^{pos}, O)$, if

(1) $V_l \rightarrow V_r^{pos}$ is listed in VerbOcean as a happens-before relation.
(2) According to WordNet $V_l \Rightarrow V_r^{pos}$ is not true. That means, for example, if (S, V_r, O) is paraphrasing (S, V_l, O), then this is not considered as a temporal relation.

This way, we were able to extract 1699 positive samples. Examples are shown in Table 2.

Negative samples. We extracted negative samples of the form $(S, V_l, O) \nrightarrow (S, V_r^{neg}, O)$, if $V_l \rightarrow V_r^{neg}$ is not listed in VerbOcean. This way, we extracted 1177 negative samples.

There are several reasons for a relation not being in a temporal relation. Using VerbOcean and WordNet we analyzed the negative samples, and found that the majority (1030 relations) could not be classified

Table 2
Examples of happens-before relations of Reuters KB

Examples
(company, buy, share) → (company, use, share)
(ex-husband, stalk, her) → (ex-husband, kill, her)
(farmer, plant, acre) → (farmer, harvest, acre)

Table 3
Examples of happens-before and not happens-before event relations from the Regneri KB

Examples
([protagonist], insert, money, machine) → ([protagonist], receive, ticket)
([protagonist], pay ride) → ([protagonist], sit)
([protagonist], order, desert) ↛ ([protagonist], eat, entree)
([protagonist], open, door) ↛ ([protagonist], check, who)

into any relation with VerbOcean or WordNet. We estimated conservatively that around 27% of these relations are false negatives: for a sub-set of 100 relations, we labeled a sample as a false negative, if it can have an interpretation as a happens-before relation.[2]

To simplify the task, we created a balanced data set, by pairing all positive and negative samples: each sample pair contains one positive and one negative sample, and the task is to build a system that gives a higher score to a correct happens-before relation than to a wrong happens-before relation. The resulting data set contains in total 1765 pairs.

We emphasize that this data set is different from the settings considered for example in [29], where the events' order in text is used as a proxy for their temporal order. We analyzed a random subset of 100 events of the form $(S, V_l, O) \rightarrow_{text} (S, V_r, O)$, where $\rightarrow_{text}$ means "occurs earlier in text". We found that only around 30% can be considered as happens-before relations.

2.2. Regneri knowledge base

As an another data-set for evaluation, we created a knowledge base from the data set prepared in [33]. The released data-set contains 14 scripts, prototypical event sequences, like "eating at restaurant" and "going to laundry". We extracted the predicate arguments from the plain text event descriptions by using the Stanford dependency parser and lemmatization (same as before in Section 2.1).

Some examples of the event relations are shown in Table 3.

Unlike Reuters KB, the events in the Regneri data always have a fixed human subject (protagonist) that is omitted. Furthermore, the data set has a flexible number of arguments: no arguments, only direct or only indirect object, both direct and indirect objects.[3] Every happens-before (positive) relation is paired with a happens-after (negative) relation. In total, this way we were able to extract 17939 positive/negative happens-before event relations that are grouped into the 14 different scripts.

3. Similarity-based reasoning for happens-before relation scoring

In this section, we describe in detail our proposed method and its underlying assumptions.

In the following, let r be a happens-before relation of the form:

$$r : e_l \rightarrow e_r,$$

where e_l and e_r are two events, respectively.

Given an unseen event e' (i.e. an event that is not in the knowledge base), we can find possible future events using the following two assumptions:

(I) If $(e' \sim e_l) \wedge (e_l \rightarrow e_r)$, then $e' \rightarrow e_r$.

(II) If $(e' \sim e_r) \wedge (e_l \rightarrow e_r)$, then $e_l \rightarrow e'$.

Where $\sim$ denotes some appropriate similarity relation between two events.

For example, let us assume we have the following relation in our knowledge base

"John acquires computer" → "John uses computer".

Furthermore, let us assume that the similarity $\sim$ is defined such that the following statement is true

"John buys computer" $\sim$ "John acquires computer".

[2] Therefore, this over-estimates the number of false negatives. This is because it also counts relations like "X's share falls 10%" ↔ "X's share rises 10%" as a false negative.

[3] A few events had more than two arguments which we ignored for evaluation.

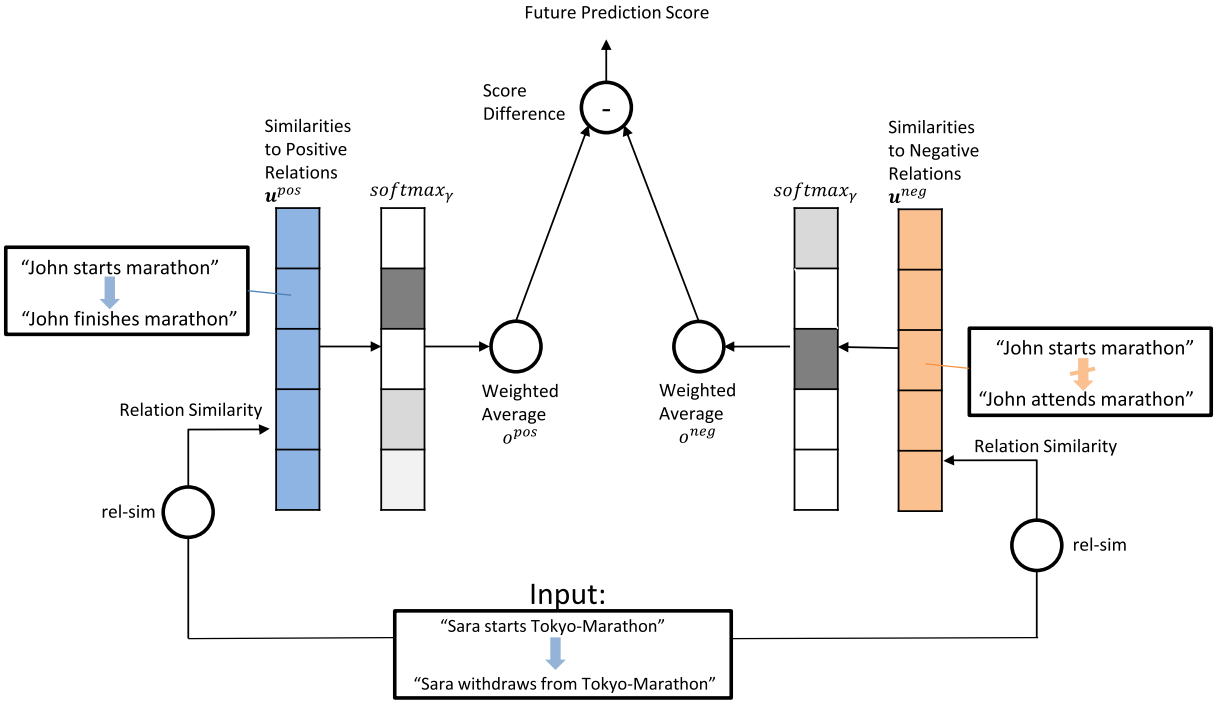


Fig. 2. Illustration of proposed model.

Then, using (I), we can reason that:

"John buys computer" → "John uses computer".

This example also shows that using the paraphrase relation ⇔ for the similarity relation ∼ would be too restrictive.[4]

It is not clear what the best measure for the similarity ∼ is. Therefore, instead of defining the similarity ∼ manually, we propose to learn a measure for this similarity from data such that it optimizes future prediction accuracy.

3.1. Memory comparison network

Using the assumptions (I) and (II), we propose a memory-based network, which we name Memory Comparison Network (MCN). It bases its decision on one (or more) training samples that are similar to a test sample. In contrast to other methods like neural networks for script learning [14,23], and (non-linear) SVM ranking models [39], it has the advantage of giving an explanation of why a relation is considered (or not considered) as a happens-before relation. Compared to k-nearest neighbor approaches, our proposed method does not need a fixed k, and instead uses a trainable softmax to learn attention weights. Furthermore, our proposed method can learn the similarity relation ∼ from data.

The high-level architecture of our proposed model is shown in Fig. 2. All training data is saved in memory and accessed during testing. First, we calculate similarity weights between the test input relation and each happens-before relation (left hand side in Fig. 2) and each not happens-before relation (right hand side in Fig. 2). We denote the resulting similarity weights as $\mathbf{u}^{pos}$ and $\mathbf{u}^{neg}$, respectively. Next, we use a trainable softmax function, denoted as softmax_γ, to calculate attention weights $\text{softmax}_\gamma(\mathbf{u}^{pos})$ and $\text{softmax}_\gamma(\mathbf{u}^{neg})$ (shown in Fig. 2 on the left and right hand side in gray scale). Finally, using these attention weights, we calculate two weighted averages o^{pos} and o^{neg} that represent the score that the input relation is a happens-before (positive) and not a happens-before relation (negative), respectively.

The attention weights help to focus only on a few relevant training samples. Our analysis, in Section 6, shows that compared to using the similarity weights $\mathbf{u}^{pos}$ and $\mathbf{u}^{neg}$ directly, the attention weights help to improve accuracy.

[4] Let e_1 and e_2 be two events. It holds that if $[e_1 \Leftrightarrow e_2]$, then $[e_1 \sim e_2]$. But it does not hold that if $[e_1 \sim e_2]$, then $[e_1 \Leftrightarrow e_2]$.

In the following, we explain in more detail the components of our network architecture and its trainable parameters.

Similarity between events. In order to calculate the similarity between two relations, we first need to calculate the similarity between events that occur in the relations. For calculating the similarity between two events we use a parameterized cosine similarity that can learn the importance of all arguments. Our parameterization ensures that the number of trainable parameters is independent of the size of the word embeddings, which proves helpful for exploiting high-dimensional word embeddings (see experimental results and discussion in Section 6.2).

Given two events e_1 and e_2, we measure the similarity relation $e_1 \sim e_2$ using the following non-linear transformation of the parameterized cosine similarity:

$$\text{sim}_{\boldsymbol{\theta},A}(e_1, e_2) = g_{\boldsymbol{\theta}}\big(\cos_A(\mathbf{x}, \mathbf{y})\big), \tag{1}$$

where $\mathbf{x}$ and $\mathbf{y}$ are the vector representations of the event e_1 and e_2, respectively.[5] The function $g_{\boldsymbol{\theta}}$ is an artificial neuron with $\boldsymbol{\theta} = \{\sigma, \beta\}$, a scale $\sigma \in \mathbb{R}$, and a bias $\beta \in \mathbb{R}$ parameter, followed by a non-linearity. We use as non-linearity the sigmoid function. The parameterized cosine similarity (see e.g. [3]), is defined as follows:

$$\cos_A(\mathbf{x}, \mathbf{y}) = \frac{(A^{1/2}\mathbf{x})^T (A^{1/2}\mathbf{y})}{\|A^{1/2}\mathbf{x}\|_2 \|A^{1/2}\mathbf{y}\|_2},$$

where A is a positive definite matrix, $A^{1/2}$ is its square root, and $\|\cdot\|_2$ denotes the l2-norm. Here, we represent each event by the concatenation of its predicate and arguments' word embeddings. An event with one predicate and two arguments is represented as $\mathbf{x}^T = (\mathbf{x}_1^T, \mathbf{x}_2^T, \mathbf{x}_3^T) \in \mathbb{R}^{3d}$. The first component $\mathbf{x}_1$ is the word embedding of the predicate. For the Reuters dataset, $\mathbf{x}_2$ and $\mathbf{x}_3$ correspond to the word embedding of the subject and object, respectively. For the Regneri dataset, $\mathbf{x}_2$ and $\mathbf{x}_3$ correspond to the word embedding of the direct and indirect object, respectively. If an event has a missing argument, then the corresponding word embedding is set to the constant zero vector. In the following, we assume that all word embeddings, except the all zero vector, are l2-normalized, i.e. $\|\mathbf{x}_i\|_2 = 1$.

[5] Remark about our notation: we use bold fonts, like $\mathbf{v}$ to denote a column vector; $\mathbf{v}^T$ to denote the transpose, and v_i to denote the ith dimension of $\mathbf{v}$. If $\mathbf{v}$ is a block vector, then $\mathbf{v}_i$, denotes the ith sub-vector.

For $A^{1/2}$, we suggest to use the following block diagonal matrix

$$A^{1/2} = \begin{pmatrix} a_1 \cdot I & 0 & 0 \\ 0 & a_2 \cdot I & 0 \\ 0 & 0 & a_3 \cdot I \end{pmatrix},$$

where $a_i \in \mathbb{R}$ are trainable parameters, and $I \in \mathbb{R}^{d\times d}$ denotes the identity matrix, and $0 \in \mathbb{R}^{d\times d}$ is the all zero matrix. This has the advantage that

$$w_i := a_i^2 \tag{2}$$

can be interpreted as the weight of argument i. Furthermore, this choice ensures that the number of trainable parameters does not increase with the size of the word embedding d.

Since all word embeddings are l2-normalized, we have $\cos_A(\mathbf{x}, \mathbf{y})$ equals

$$\begin{aligned} &\frac{a_1^2 \cdot \mathbf{x}_1^T\mathbf{y}_1 + a_2^2 \cdot \mathbf{x}_2^T\mathbf{y}_2 + a_3^2 \cdot \mathbf{x}_3^T\mathbf{y}_3}{(\sqrt{a_1^2 \cdot \mathbf{x}_1^T\mathbf{x}_1 + a_2^2 \cdot \mathbf{x}_2^T\mathbf{x}_2 + a_3^2 \cdot \mathbf{x}_3^T\mathbf{x}_3}\sqrt{a_1^2 \cdot \mathbf{y}_1^T\mathbf{y}_1 + a_2^2 \cdot \mathbf{y}_2^T\mathbf{y}_2 + a_3^2 \cdot \mathbf{y}_3^T\mathbf{y}_3}} \\ &= \frac{w_1 \cdot \mathbf{x}_1^T\mathbf{y}_1 + w_2 \cdot \mathbf{x}_2^T\mathbf{y}_2 + w_3 \cdot \mathbf{x}_3^T\mathbf{y}_3}{w_1 + w_2 + w_3} \\ &= \frac{w_1}{w_1 + w_2 + w_3} \cdot \mathbf{x}_1^T\mathbf{y}_1 + \frac{w_2}{w_1 + w_2 + w_3} \cdot \mathbf{x}_2^T\mathbf{y}_2 \\ &\quad + \frac{w_3}{w_1 + w_2 + w_3} \cdot \mathbf{x}_3^T\mathbf{y}_3. \end{aligned}$$

We note that the parameterized cosine similarity has the advantage that it appropriately re-normalizes the predicate/argument weights when there is a missing argument. To understand this, consider the case where both events have no object. In that case, using the same derivation as before, we get

$$\begin{aligned} &\cos_A(\mathbf{x}, \mathbf{y}) \\ &\quad = \frac{w_1}{w_1 + w_2}\mathbf{x}_1^T\mathbf{y}_1 + \frac{w_2}{w_1 + w_2}\mathbf{x}_2^T\mathbf{y}_2, \end{aligned} \tag{3}$$

since the embeddings $\mathbf{x}_3$ and $\mathbf{y}_3$ are set to the all zero vector.

As a computationally less expensive alternative, one might consider the following weighted dot product without re-normalization:

$$\text{dot}_A(\mathbf{x}, \mathbf{y}) = \big(A^{1/2}\mathbf{x}\big)^T \big(A^{1/2}\mathbf{y}\big). \tag{4}$$

However, note that dot_A has the disadvantage that it does not re-normalize the argument weights, when some arguments missing. As a consequence, if two events have missing arguments, the resulting similarity,

will be smaller than the similarity between the same events, but with all arguments. Our experimental analysis in Section 6 shows that this leads to inferior performance.

Similarity between relations. In the following, let r_i be a happens-before relations of the form:

$$r_i : e_{l_i} \rightarrow e_{r_i},$$

where, e_{l_i} and e_{r_i}, represents the observed and future event, respectively. We define the similarity between two relations r_1 and r_2 as:

$$\begin{aligned}&\text{rel-sim}(r_1, r_2)\\ &\quad = \text{sim}_{\boldsymbol{\theta},A}(e_{l_1}, e_{l_2}) + \text{sim}_{\boldsymbol{\theta},A}(e_{r_1}, e_{r_2}).\end{aligned} \tag{5}$$

Given an input relation $r^{\text{in}} : e_l^{\text{in}} \rightarrow e_r^{\text{in}}$, we test whether the relation r^{in} is correct or wrong as follows.

Let n_{pos} and n_{neg} denote the number of positive and negative samples in memory, respectively. First, we compare to all positive and negative training relations in memory, and denote the resulting vectors as $\boldsymbol{u}^{pos} \in \mathbb{R}^{n_{pos}}$ and $\boldsymbol{u}^{neg} \in \mathbb{R}^{n_{neg}}$, respectively. That is formally

$$\begin{aligned}u_t^{pos} &= \text{rel-sim}\big(r^{in}, r_t^{pos}\big) \quad \text{and}\\ u_t^{neg} &= \text{rel-sim}\big(r^{in}, r_t^{neg}\big),\end{aligned} \tag{6}$$

where r_t^{pos} and r_t^{neg} denotes the tth positive/negative sample in memory.

Attention weights. Next, we define the score that r^{in} is correct/wrong as the weighted average of the relation similarities:

$$o^{pos} = \text{softmax}_\gamma\big(\mathbf{u}^{pos}\big)^T \mathbf{u}^{pos}, \tag{7}$$

$$o^{neg} = \text{softmax}_\gamma\big(\mathbf{u}^{neg}\big)^T \mathbf{u}^{neg}, \tag{8}$$

where $\text{softmax}_\gamma(\mathbf{u})$ returns a column vector with the tth output defined as

$$\text{softmax}_\gamma(\mathbf{u})_t = \frac{\exp(\gamma u_t)}{\sum_i \exp(\gamma u_i)},$$

and $\gamma \in \mathbb{R}$ is a weighting parameter. Note that for $\gamma \rightarrow \infty$, $\text{softmax}_\gamma(\mathbf{u}) = \max(\mathbf{u})$, and for $\gamma = 0$, o^{pos} and o^{neg} is the average of $\mathbf{u}$. We refer to $\text{softmax}_\gamma(\mathbf{u}^{pos})_i$ and $\text{softmax}_\gamma(\mathbf{u}^{neg})_j$ as the attention weight of the ith positive and jth negative training sample, respectively.

Finally, we define the happens-before score for r^{in} as

$$l\big(r^{\text{in}}\big) = o^{pos}\big(r^{\text{in}}\big) - o^{neg}\big(r^{\text{in}}\big). \tag{9}$$

The score $l(r^{\text{in}})$ can be considered as an unnormalized log probability that relation r^{in} is a happens-before relation.

For optimizing the parameters of our model we minimize the margin rank loss:

$$\begin{aligned}&L\big(r^{\text{in-pos}}, r^{\text{in-neg}}\big)\\ &\quad = \max\big\{0, 1 - l\big(r^{\text{in-pos}}\big) + l\big(r^{\text{in-neg}}\big)\big)\big\},\end{aligned} \tag{10}$$

where $r^{\text{in-pos}} : e_l^{\text{in}} \rightarrow e_r^{\text{in-pos}}$ and $r^{\text{in-neg}} : e_l^{\text{in}} \rightarrow e_r^{\text{in-neg}}$ are positive and negative samples from the training data that are not in memory. All parameters of the models are trained using stochastic gradient descent (SGD). The word embeddings ($\mathbf{x}_1$, $\mathbf{x}_2$, and $\mathbf{x}_3$) are kept fixed during training.

We emphasis that during training our method, we put part of the training samples into memory and calculate the loss in Equation (10) with respect to two input relations $r^{\text{in-pos}}$ and $r^{\text{in-neg}}$ that are not in memory. This is necessary, since otherwise, if we put all training samples into memory, and calculate the loss with an input relation that is also in memory, our proposed method will learn that there is always one perfect match in the memory (e.g. the trainable parameter γ of softmax_γ is likely to converge towards infinity).

4. Alternative ranking models

Here in this section, we investigate several other models that can be applied for ranking future predictions. All models that we consider are based on word embeddings in order to be able to generalize to unseen events.

Our first model is based on the bilinear model proposed in [2] for document retrieval, with scoring function $l(e_l, e_r) = \mathbf{e}_l^T M \mathbf{e}_r$, where the event representation $\mathbf{e}_l, \mathbf{e}_r \in \mathbb{R}^{3d}$ are the concatenated word embeddings from the predicate and all arguments, and $M \in \mathbb{R}^{3d \times 3d}$ is the parameter matrix. We denote this model as Bai2009.

We also compare to three neural network architectures that were proposed in different contexts. The model in [4], originally proposed for semantic parsing, is a three layer network that can learn non-linear com-

binations of (subject, verb) and (verb, object) pairs. The non-linearity is achieved by the Hadamard product of the hidden layers. The original network can handle only events (relations between verb and objects, but not relations between events). We recursively extend the model to handle relations between events. We denote the model as Bordes2012.

In the context of script learning, recently two neural networks have been proposed for detecting happens-before relations. The model proposed in [23] (denoted by Modi2014) learns event embeddings parameterized with verb and context (subject or object) dependent embedding matrices. The event embeddings are then mapped to a score. Their original method gets as input only one event, and returns a score that reflects absolute time, rather than time relative to another event. We therefore appropriately extend it: in order to score a relation between two events, we use the dot product between the two events' embeddings.[6]

The model in [14] suggests a deeper architecture than Modi2014. Their model (denoted by Granroth2016) uses two additional non-linear layers for combining the left and right events.

We train all above models in the same way as the proposed method using margin rank loss.[7] For all methods, we found fixing the word embeddings (i.e. no adjustment during training) improves performance.

Our final two models use rankSVM [39] with a linear kernel and a RBF kernel, respectively. The feature representation is the concatenation of the embeddings of all words in the relation.

5. Evaluation

In this section, we describe the details of our experimental settings and compare our proposed method to several previous methods.

Due to the relatively small size of the knowledge bases we use 10-fold and 14-fold cross-validation for Reuters KB and Regneri KB, respectively.

For Reuters KB, we use 10 different random splits with around 50%, 25% and 25% for training, validation and testing, respectively. We split the data such that the left event (observation) of the sample contains a predicate that is not contained in the training set (and also not in the validation set).

For the Regneri KB, we test on all events belonging to one script, and all events from the remaining scripts are split into training (8 scripts) and validation (5 scripts) set. This leads to a 14-fold cross-validation, since there are 14 scripts.

We implemented all methods using Torch 7 [9].

For the bilinear model and all neural networks, we performed up to 2000 epochs for 50 dimensional word embeddings and up to 200 epochs for 300 dimensional word embeddings.[8] To prevent overfitting, we used early stopping with respect to the validation set. Some models were quite sensitive to the choice of the learning rates, so we tested 0.00001, 0.0001, and 0.001, and report the best results on the validation set.

For our proposed method, for Reuters KB, we used up to 100 epochs with a fixed learning of 0.001, and for Regneri KB, we used up to 50 epochs with a fixed learning rate of 0.01. Therefore, we split the training data further, into training data in memory (two thirds) and training data for input (one third).[9] As initial parameters for this non-convex optimization problem we set $\sigma = 1.0$, $\beta = -0.5$, $\gamma = 5.0$, and the argument weights to the uniform probability.

For the rankSVM baselines we used the implementation from [40]. We tested both a linear and a RBF kernel with the hyper-parameters optimized via grid-search. For the linear kernel we tested c in the range 2^{-5} to 2^{16}, with step size 1 of the exponent. For the RBF kernel's parameters, due to the high computational costs, we had to limit the search to a course grid: $c \in \{2^{-5}, 2^{-2}, 2^{1}, 2^{4}, 2^{7}\}$, and $\gamma \in \{2^{-10}, 2^{-7}, 2^{-4}, 2^{-1}\}$.

We report accuracy, when asking the question: given observation e_l, is e_r^{pos} more likely to be a future event than e_r^{neg}?

We used the 50 and 300 dimensional word embeddings from the GloVe tool [27] trained on Wikipedia + Gigaword 5 provided by the authors. If an event has no direct or indirect object, we use the all zero vector.

All prediction accuracies (in percent) are shown in Tables 4 and 5 for Reuters KB, and Tables 6 and 7 for Regneri KB, respectively. By using the false-negative

[6] We tried two variants: left and right events with different and same parameterization. The results did not change significantly.

[7] Originally, the model in [14] is optimized with respect to negative log-likelihood, however, in our preliminary experiments we found that margin rank loss performed better.

[8] We had to reduce the maximal number of epochs to 200 since otherwise, for example, the baseline Bordes2000 with all the evaluation of different learning rates and cross-validation would have taken around 70 days on a GPU.

[9] Also see explanation at end of Section 3.1.

Table 4
Results for Reuters KB with 50 dimensional word embeddings. Mean accuracy and standard deviation (in brackets)

Method	Test data
Human estimate	76.7
Memory Comparison Network	60.3 (8.9)
Granroth2016	60.9 (5.3)
Modi2014	57.5 (7.8)
Bordes2012	58.3 (7.8)
Bai2009	58.9 (7.4)
rankSVM (RBF)	60.6 (6.2)
rankSVM (linear)	59.1 (9.4)
Random Baseline	50.0
Memory Comparison Network (all args with $\cos_A$, softmax_γ, trained)	60.3 (8.9)
Memory Comparison Network (all args with dot_A, softmax_γ, trained)	58.8 (7.7)
Memory Comparison Network (only predicate, softmax_γ, trained)	60.2 (8.9)
Memory Comparison Network (only predicate, max, trained)	59.6 (6.5)
Memory Comparison Network (only predicate, max, no parameters)	60.1 (5.8)
Memory Comparison Network (only predicate, average, no parameters)	60.1 (5.9)

Table 5
Results for Reuters KB with 300 dimensional word embeddings. Mean accuracy and standard deviation (in brackets)

Method	Test data
Human estimate	76.7
Memory Comparison Network	62.8 (7.4)
Granroth2016	61.5 (4.5)
Modi2014	59.7 (7.8)
Bordes2012	51.9 (10.0)
Bai2009	59.1 (7.0)
rankSVM (RBF)	63.6 (5.6)
rankSVM (linear)	60.3 (7.5)
Random Baseline	50.0
Memory Comparison Network (all args with $\cos_A$, softmax_γ, trained)	62.8 (7.4)
Memory Comparison Network (all args with dot_A, softmax_γ, trained)	61.9 (7.5)
Memory Comparison Network (only predicate, softmax_γ, trained)	63.2 (9.5)
Memory Comparison Network (only predicate, max, trained)	58.1 (7.4)
Memory Comparison Network (only predicate, max, no parameters)	58.3 (7.5)
Memory Comparison Network (only predicate, average, no parameters)	61.0 (6.2)

estimate from Section 2.1, we also calculated an estimate of the human performance ("Human estimate") on the task for Reuters KB.[10]

The results suggest that our proposed model provides good generalization performance that is at par or better than the recently proposed neural network Granroth2016, and SVM ranking with RBF-kernel. Our results support our claim that the happens-before relation can be detected by similarity-based reasoning. Furthermore, we can observe that the performance of our proposed method tends to increase with higher dimensional word embeddings.

6. Analysis

In this section, we analyze our proposed method and investigate the usefulness of the attention mecha-

[10] We assume that all false-negatives lead to a wrong human guess.

Table 6
Results for Regneri KB with 50 dimensional word embeddings. Mean accuracy and standard deviation (in brackets)

Method	Test data
Memory Comparison Network	63.5 (7.4)
Granroth2016	57.3 (6.8)
Modi2014	60.1 (9.3)
Bordes2012	55.9 (9.3)
Bai2009	57.5 (6.0)
rankSVM (RBF)	60.3 (8.2)
rankSVM (linear)	60.2 (7.9)
Random Baseline	50.0
Memory Comparison Network (all args with $\cos_A$, softmax_γ, trained)	63.6 (7.7)
Memory Comparison Network (all args with dot_A, softmax_γ, trained)	60.6 (10.9)
Memory Comparison Network (only predicate, softmax_γ, trained)	62.3 (6.9)
Memory Comparison Network (only predicate, max, trained)	60.4 (5.0)
Memory Comparison Network (only predicate, max, no parameters)	60.2 (5.0)
Memory Comparison Network (only predicate, average, no parameters)	59.8 (6.2)

Table 7
Results for Regneri KB with 300 dimensional word embeddings. Mean accuracy and standard deviation (in brackets)

Method	Test data
Memory Comparison Network	67.3 (9.4)
Granroth2016	62.9 (9.7)
Modi2014	56.6 (9.8)
Bordes2012	48.8 (6.7)
Bai2009	60.6 (9.2)
rankSVM (RBF)	63.6 (9.1)
rankSVM (linear)	65.3 (10.2)
Random Baseline	50.0
Memory Comparison Network (all args with $\cos_A$, softmax_γ, trained)	67.3 (9.4)
Memory Comparison Network (all args with dot_A, softmax_γ, trained)	66.0 (8.1)
Memory Comparison Network (only predicate, softmax_γ, trained)	67.0 (6.9)
Memory Comparison Network (only predicate, max, trained)	62.4 (4.6)
Memory Comparison Network (only predicate, max, no parameters)	62.3 (4.5)
Memory Comparison Network (only predicate, average, no parameters)	64.9 (7.1)

nism. Furthermore, we discuss the types of errors of our method and its current limitations.

In order to evaluate the design choices of our proposed method, we tested five variations of our method. The results are shown in the lower half of Tables 4 and 5 for Reuters KB, and Tables 6 and 7 for Regneri KB, respectively.

The first variation, "Memory Comparison Network (all args with dot_A, softmax_γ, trained)" replaces the parameterized cosine similarity by the simplified version without normalization, see Equation (4). The results confirm that without normalization, the parameterized cosine similarity performs worse. We suspect that this is partly due to the wrong handling of missing arguments (as discussed in Section 3.1).

The next variation uses only the predicates (verbs) for representing an event. For both knowledge bases, the results suggests that using only the verbs leads to similar or only slightly lower accuracy. However, later in this section, we will also show some examples where information from arguments is actually necessary to correctly predict future events.

Finally, we evaluated the effect of using the attention weights from softmax_γ instead of either the hard max

function, or the average. In particular, we note that the method "Memory Comparison Network (only predicate, max, trained/no parameters)", which replaces the softmax_γ by max, is a kind of simple 1-nearest neighbor ranking. "no parameters" uses the inner product of the verb's embeddings instead of $\text{sim}_{\boldsymbol{\theta},A}$ from Equation (1). The variation, "Memory Comparison Network (only predicate, average, no parameters)" uses for o^{pos} and o^{neg}, in Equations (7), the average of $\mathbf{u}^{pos}$ and $\mathbf{u}^{neg}$, respectively. We can see that the choice of softmax_γ, over max or average, improves performance.

Since our model uses a kind of similarity-based reasoning, we can easily identify "supporting evidence" for the output of our system. Four examples from Reuters KB and Regneri KB are shown in Table 8 and Table 9, respectively. Here, "supporting evidence" denotes the training sample with the highest similarity rel-sim to the input. In each table, we show two examples when the input is a happens-before relation (first and second example), and two examples when the input is not a happens-before relation (third and fourth example).[11]

We also tried to quantify the usefulness of the supporting evidence using a random subset of 100 input happens-before relations from the Reuters KB test set, where we annotated for each input relation all relations in the knowledge base, and 280 input happens-before relations from Regneri KB, where we annotated for each input relation the top 50 relations in the knowledge base output by our system.[12] For Regneri KB, we annotated a random subset of 20 test relations from each split of the test and training data, i.e. in total $20 \times 14 = 280$ input relations, in order to cover all scripts (see Section 2.2).

For each input relation, we rank the happens-before relations in the knowledge base by o^{pos}. Ideally, our system should output all supporting evidence in the top ranks, and all irrelevant relations below.

For evaluation, we use three measures: recall at top n, R-Precision [18], and missing knowledge probability at top n, which we define in the following.

$$\text{Recall@n} = \frac{r_n}{m},$$

where r_n is the number of supporting evidences found in the top n, and m is the total number of supporting evidences found in the knowledge base for the input relation.

$$\text{R-Precision} = \frac{r_m}{m}.$$

We average Recall@n and R-Precision over all input relations for which $m \geqslant 1$, i.e. at least one supporting evidence in the knowledge base.

Furthermore, from a practical point-of-view, we are also interested in whether the ranking of the relations can help us to identify missing evidence in the KB. For that purpose, we calculate the probability $p(\text{missing evidence in KB} | r_n = 0)$, which we denote by Missing@n:

$$\text{Missing@n} = \frac{f(\text{missing evidence in KB} \wedge r_n = 0)}{f(r_n = 0)},$$

where $f(C)$ denotes the number of annotated input relations which fulfill condition C.[13]

All results for Reuters KB and Regneri KB are shown in Tables 10, 11 and 12.

Inspecting the R-Precision and Recall@1, we can see that the ranking of evidence by our system is not yet satisfactory. Nevertheless, from Table 12, we see that if we cannot find supporting evidences in the top 20 relations output by our system, we can conclude that the KB does not contain sufficient knowledge with probability up to 98%. This shows that the attention mechanism can be helpful for detecting missing knowledge in the KB.

We note that our method can also correctly rank future events, even if the predicates are the same. Three examples are shown in Table 13. This suggests, that our model uses also the information from the subject and objects to predict the future event. We confirm this by inspecting the actual argument weights that were learned by our proposed method (Table 14).

Furthermore, it is intriguing that rankSVM (RBF) performs comparable to our proposed method on Reuters KB, whereas on Regneri KB, rankSVM (RBF) performs worse than the proposed method. As we discuss in Section 6.1, the difference in training data size does not seem to be the reason for these differences. Instead, we suspect that the performance difference is

[11] Since we considered only the head, a unit like "percent" means "x percent", where x is some number.

[12] For Reuters there were only 83 unique happens-before relations in the KB, leading to $100 \times 83 = 8300$ annotations. However, for Regneri there were more than 2000 relations in the KB which forced us to limit the annotation to the top 50 output by our system. Therefore, the recalls shown for Regneri KB are not exact, but based on an estimate using the top 50 results.

[13] "Missing evidence in KB" means that for the input relation, we could not find any supporting evidence in the knowledge base.

Table 8

Four examples with input relations, output scores and evidences by our proposed method. Reuters KB with 300 dimensional word embeddings. An event is shown as a triplet (subject, predicate, object)

input relation: (price, climb, cent) $\rightarrow$ (price, slide, cent)			
o^{pos}:	0.977	supporting evidence:	(price, rise, percent) $\rightarrow$ (price, tumble, percent)
o^{neg}:	0.963	supporting evidence:	(price, ease, cent) $\nrightarrow$ (price, slide, cent)
input relation: (elephant, grab, him) $\rightarrow$ (elephant, throw, him)			
o^{pos}:	0.832	supporting evidence:	(ex-husband, catch, her) $\rightarrow$ (ex-husband, kill, her)
o^{neg}:	0.830	supporting evidence:	(ex-husband, catch, her) $\nrightarrow$ (ex-husband, call, her)
input relation: (price, gain, cent) $\nrightarrow$ (price, strengthen, cent)			
o^{pos}:	0.943	supporting evidence:	(investment, build, plant) $\rightarrow$ (investment, expand, plant)
o^{neg}:	0.956	supporting evidence:	(dollar, rise, yen) $\nrightarrow$ (dollar, strengthen, yen)
input relation: (farmer, plant, acre) $\nrightarrow$ (farmer, seed, acre)			
o^{pos}:	0.813	supporting evidence:	(refinery, produce, tonne) $\rightarrow$ (refinery, process, tonne)
o^{neg}:	0.820	supporting evidence:	(refinery, produce, tonne) $\nrightarrow$ (refinery, receive, tonne)

Table 9

Four examples with input relations, output scores and evidences by our proposed method. Regneri KB with 300 dimensional word embeddings. An event is shown as a 4-tuple (subject, predicate, first object, second object). Empty slots (e.g. no object) are removed from the 4-tuple. [pro] stands for the protagonist

input relation: ([pro], insert, money, machine) $\rightarrow$ ([pro], receive, ticket)			
o^{pos}:	0.029	supporting evidence:	([pro], tell, order) $\rightarrow$ ([pro], receive, order)
o^{neg}:	0.022	supporting evidence:	([pro], find, table) $\nrightarrow$ ([pro], receive, order)
input relation: ([pro], pick, handset, phone) $\rightarrow$ ([pro], press, button)			
o^{pos}:	0.841	supporting evidence:	([pro], put, food, microwave) $\rightarrow$ ([pro], press, button)
o^{neg}:	0.835	supporting evidence:	([pro], ask) $\nrightarrow$ ([pro], press, button)
input relation: ([pro], eat, desert) $\nrightarrow$ ([pro], choose, item)			
o^{pos}:	0.868	supporting evidence:	([pro], eat) $\rightarrow$ ([pro], take, trash)
o^{neg}:	0.870	supporting evidence:	([pro], eat) $\nrightarrow$ ([pro], decide, what)
input relation: ([pro], eat, desert) $\nrightarrow$ ([pro], give, order)			
o^{pos}:	0.890	supporting evidence:	([pro], look, menu) $\rightarrow$ ([pro], give, order, employee)
o^{neg}:	0.899	supporting evidence:	([pro], eat, meal) $\nrightarrow$ ([pro], tell, order)

Table 10

Evaluation of supporting evidence for Reuters KB and Regneri KB (300 dimensional word embeddings) using R-Precision shown in percent

Method	R-Precision
Reuters KB	
Memory Comparison Network	19.5
Random Baseline	9.8
Regneri KB	
Memory Comparison Network	16.2
Random Baseline	3.0

due to missing arguments. By construction, events in Reuters KB do not contain any missing arguments, whereas Regneri KB does. Analyzing Regneri KB, we find that around 19% of the events contain 0 arguments (only verb), 60% of the events contain 1 argument (direct object), and 21% of the events contain 2 arguments (direct and indirect object). As explained in Section 3.1, and in particular, the derivation of Formula (3), our proposed method accounts for missing arguments by appropriately re-weighting the remaining arguments. This is different from rankSVM, since rankSVM necessarily needs a fixed vector representation, where missing arguments are set to the zero vector. This might partly explain the performance difference of rankSVM between Reuters KB and Regneri KB.

6.1. Impact of training data size and runtime

Since the Regneri KB is about 10 times larger than the Reuters KB, we suspected that some of the per-

Table 11

Evaluation of supporting evidence for Reuters KB and Regneri KB (300 dimensional word embeddings) using Recall@n shown in percent

Method	Recall@1	Recall@5	Recall@10	Recall@20
	Reuters KB			
Memory Comparison Network	3.3	17.7	32.9	48.7
Random Baseline	1.2	6.1	12.1	24.3
	Regneri KB			
Memory Comparison Network	3.4	26.5	47.3	74.1
Random Baseline	1.0	5.1	10.2	20.5

Table 12

Evaluation of supporting evidence for Reuters KB and Regneri KB (300 dimensional word embeddings). Missing@n denotes the probability $p(\text{missing evidence in KB}|r_n = 0)$, shown here in percent

Method	Missing@1	Missing@5	Missing@10	Missing@20
	Reuters KB			
Memory Comparison Network	69.1	79.3	87.8	91.5
Random Baseline	67.3	74.2	79.4	86.0
	Regneri KB			
Memory Comparison Network	87.3	92.0	94.9	98.0
Random Baseline	86.1	87.3	88.5	90.2

Table 13

Three examples of our proposed method. The predicates are the same, but the objects are different. Regneri KB with 300 dimensional word embeddings. An event is shown as a 4-tuple (subject, predicate, first object, second object). Empty slots (e.g. no object) are removed from the 4-tuple. Due to space limitations, we omit the protagonist

input relation: (put, amount, money) $\rightarrow$ (take, goods)	$o^{pos} - o^{neg}$: −0.001
input relation: (put, money, machine) $\nrightarrow$ (take, money)	$o^{pos} - o^{neg}$: −0.002
input relation: (press, button) $\rightarrow$ (take, item)	$o^{pos} - o^{neg}$: 0.002
input relation: (press, button, product) $\nrightarrow$ (take, money, pocket)	$o^{pos} - o^{neg}$: −0.003
input relation: (turn, laundry) $\rightarrow$ (put, clothe, dryer)	$o^{pos} - o^{neg}$: 0.007
input relation: (turn, dryer) $\nrightarrow$ (put, clothe, dryer)	$o^{pos} - o^{neg}$: −0.063

Table 14

Predicate and arguments weights learned by our proposed model, see Equation (2) for weight definition

Word embedding dimension	Predicate	Subject	Object
	Reuters KB		
50	85.5%	7.1%	7.4%
300	75.9%	11.9%	12.2%
Word embedding dimension	Predicate	Subject	Indirect object
	Regneri KB		
50	91.4%	0.8%	7.8%
300	37.6%	17.7%	44.7%

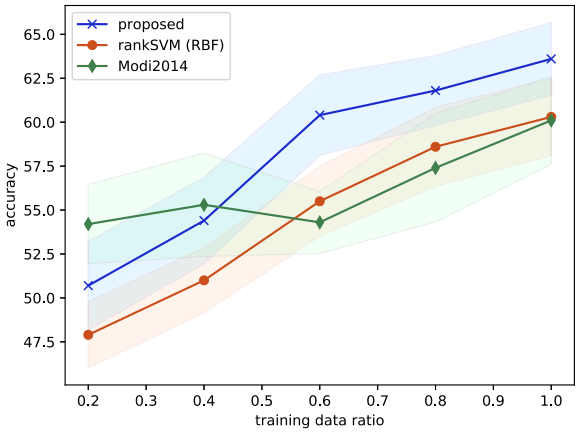


Fig. 3. Accuracy of best three methods on Regneri KB for different sizes of training data with 50 dimensional word embeddings. Width of error bars (shaded areas) is two standard errors.

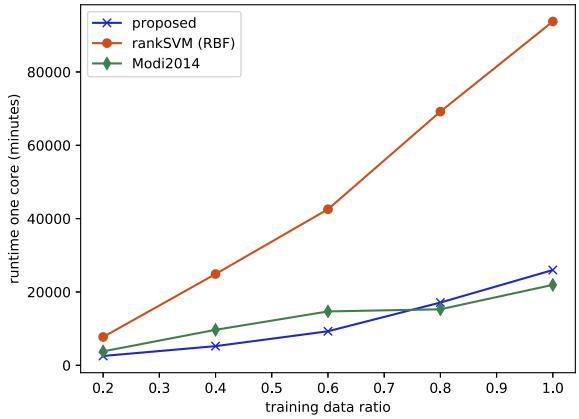


Fig. 4. Runtime for different sizes of training data on Regneri KB with 50 dimensional word embeddings. Runtime is an estimate of the time needed to perform all cross-validations on one core.

formance differences of the proposed method and the baselines are due to the different training data size. We therefore compared our proposed method, Modi2014, the best neural network baseline, and rankSVM (RBF) on Regneri KB with different training data sizes.

As can be seen from Fig. 3, for 50% or more of the training data size, the proposed method is consistently better than rankSVM (RBF) and Modi2014. When using only 20% of the training data, Modi2014 has an accuracy of around 54% whereas the performance of the proposed method and rankSVM (RBF), are at chance level and below, respectively. Comparing the low accuracy of all methods on Regneri KB with 20% training data, suggests that the task setting in Regneri KB is more difficult than Reuters KB. In fact, recall that the task setting in Regneri KB is such that testing is performed on a script scenario that is possibly unrelated to the script scenarios that were used for training.

We also investigated the impact of training data size on the training time and runtime of each method. For that purpose we ran all experiments on an Intel(R) Xeon(R) CPU 3.30GHz with 32 cores. However, we found that the usage of the number of CPU cores of the proposed method, Modi and rankSVM were considerably different.[14] In order to get a rough estimate of the runtime that can be compared across methods, we normalized all times by the maximal number of cores that were actually used. The results for running the 14-fold cross-validation on Regneri KB with different training data size is show in Fig. 4. We see that the runtime of rankSVM (RBF) is considerably higher than or proposed method and Modi2014. Furthermore we note that our method is easily parallizable, since

[14]The proposed method used between 5 and 7 cores, Modi and rankSVM used only one core.

Table 15
Number of trainable parameters of each model with respect to size of word embedding dimension d and number of training samples n

Method	Parameters
Memory Comparison Network	6
Granroth2016	$10.5d^2 + 5d + 1$
Modi2014	$3d^2$
Bordes2012	$12d^2 + 6d$
Bai2009	$9d^2$
rankSVM (RBF)	n
rankSVM (linear)	$3d$

we can compare to all training instances in memory in parallel (i.e. calculation of similarities in Equation (6)). Therefore, we conclude that our method is more scalable for training than rankSVM (RBF), while leading to similar or better performance.

6.2. Impact of number of trainable parameters

In Table 15, we list the number of trainable parameters of each neural network. For completeness, we also include rankSVM's number of trainable parameters. For the RBF-kernel it is necessary to calculate a weight for each training sample, whereas for the linear kernel it is sufficient to learn a projection vector from the feature vector dimension ($3d$) to a scalar.

Note that our method's number of parameters is independent of the size of the word embedding, whereas the other neural networks' number of parameters are in $O(d^2)$. Therefore, our method seems to be able to benefit more from large word embedddings, as suggested by the results in Tables 5 and 7.

The increase in parameter space when using large word embeddings, even harms the method of Bordes2012. Whereas the network architecture of Granroth2016 seems to be less prone to overfitting on the large parameter space.

6.3. Error analysis

We identified roughly three types of reasons for errors of our proposed method.

- Semantic parsing errors.
- Insufficient context information.
- Insufficient knowledge in the knowledge base.

Semantic parsing errors. There are some erroneous events in the knowledge base, which were caused by parsing errors of the original plain text. For example, the second event of the input relation in Table 16, confuses the syntactic and semantic subject of the event. Apparently the semantic subject of the second event should correctly be "government" or "authorities".

Insufficient context information. In some cases, we found that it is even for a human not possible to decide the correct prediction due to insufficient context information. An example from Regneri KB is shown in Table 17. In the context of buying a ticket for a theater, the input relation should be judged wrong, whereas, in the context of riding a bus, the happens-before relation should be considered correct.

Insufficient knowledge in the knowledge base. Finally, we identified prediction errors that are due to insufficient happens-before relations in memory (i.e. the knowledge base that is in memory is insufficient). An example from Regneri KB is shown in Table 18. There, we tested an input relation from the "shower" script, and the system has in memory parts of the remaining scripts ("ride bus", "make coffee", "answer doorbell", "eat fast-food", "eat in restaurant", "complain about food", "iron clothe", "do laundry", "microwave food", "make omelet", "make scrambled egg", "answer telephone", "buy from vending machine"). Since none of the remaining scripts are closely related to the "shower" script, some highest attentions are quite bizarre: like "warming up food in the microwave". Thanks to the attention scores, our method enables detecting dubious predictions that are due to insufficient knowledge. This can help to decide when and how to extend the knowledge base.

6.4. Current limitations

Our evaluations focused on the task of comparing two temporal rules r_1 and r_2, and asking which one is more likely, by ranking them. The discriminative objective training function (see Equation (10)), works well for ranking events, but it does not guarantee that the scores can be related to reasonable probabilities of future events. Reasoning in terms of probabilities of future events, would allow us to integrate our predictions into a probabilistic reasoning framework like MLN [34]. This could allow us to incorporate context information into the reasoning process, and this way help in situations like the ones show in Table 17.

In particular, including narrative and semantic frame information in a probabilistic framework as suggested in [12] is likely to help the prediction of future events.

Table 16

Error of our proposed method due to a wrongly parsed input relation. Example is from Reuters KB with 300 dimensional word embeddings

input relation: (hijackers, seek, asylum) → (hijacker, grant, asylum)			
o^{pos}:	0.787	supporting evidence:	(we, need, policy) → (we, get, policy)
o^{neg}:	0.791	supporting evidence:	(worker, have, access) ↛ (worker, deny, access)

Table 17

Illustrates the need of additional context information. The input relation is from Regneri KB with 300 dimensional word embeddings, script "riding bus" (which is unknown to the system). Correct scores of the proposed method, but due to wrong reasons

input relation: ([pro], receive, ticket) → ([pro], hold, pole)			
o^{pos}:	0.382	supporting evidence:	([pro], read, tag, clothe) → ([pro], hold)
o^{neg}:	0.065	supporting evidence:	([pro], do) ↛ ([pro], hold)

Table 18

Error of proposed method due to insufficient knowledge in Regneri KB (300 dimensional word embeddings). Input relation is from the script "shower". Due to space limitations, we omit the protagonist. For readability, we add prepositions in squared brackets

input relation: (turn [on], water) → (shave)			
o^{pos}:	0.659	supporting evidence:	(turn [on], microwave) → (remove, food, [from] microwave)
o^{neg}:	0.659	supporting evidence:	(turn [on], laundry) ↛ (put, clothe, [into] washer)

7. Related work

In this section, we summarize previous work that is related to our proposed method's network architecture, the creation/mining of future prediction rules and approaches that can generalize to unseen observations.

Memory networks and other related neural networks. We named our model a memory network, since our model has some similarity to the general memory networks framework proposed in [37,45]. Using the notation from [45], $I(\cdot)$ corresponds to the word embedding lookup, $G(\cdot)$ saves all training samples into the memory, the $O(\cdot)$ function corresponds to (o^{pos}, o^{neg}), and the output of $R(\cdot)$ equals Equation (9). Noteable differences are the symmetric architecture for *comparing* the input to positive and negative training relations, our parameterization of the similarity measure, and the trainable softmax.

Our model also has similarity to the memory-based reasoning system proposed in [36], with two differences. First, we use here a trainable similarity measure, see Equation (5), rather than a fixed distance measure. Second, we use the trainable softmax_γ rather than max.

Recently, neural networks with attention mechanism have been proposed for several tasks like one-shot learning [42], question answering [45], and machine translation [1]. In particular, our proposed method is inspired by memory networks as proposed in [45]. Our method extends memory networks to allow comparing the input with positive (happens-before) and negative (not happens-before) event pairs. Our method is also related to a k-nearest neighbor classifier, with a trainable distance metric [44]. However, in contrast to the k-nearest neighbor method there is no fixed hyperparameter k, and all parameters can be learned via gradient descent. Our method can also be considered as an extension of Siamese networks (as used e.g. in [5,16]), with which our method combines a memory and an attention mechanism.

Mining and creation of future prediction rules. The work in [32], suggests exploiting causal reasoning for future prediction. However, temporal reasoning does not equal causal reasoning. For example, "minister enters hall" happens before "minister leaves hall". However, the causal reason for "minister leaves hall" is, for example, that "the conference ended". Another difference of our approach to the method in [32] is that we exploit word embeddings for generalization rather than relying on the completeness of manually created ontologies.

Temporal annotations in context are made available by the TimeML corpora[15] and the Penn Discourse

[15] http://www.timeml.org/timebank/timebank.html and more recently by TempEval-3 https://www.cs.york.ac.uk/semeval-2013/task1/index.html.

Treebank [31]. However, we found that due to the limited size only few future prediction rules can be extracted from such annotated corpora.[16]

One line of research, pioneered by VerbOcean [8], extracts happens-before relations from large collections of texts using bootstrapping methods. In the context of script learning, corpora statistics, such as event bi-grams, are used to define a probability distribution over next possible future events [6,28]. However, such models cannot generalize to situations of new events that have not been observed before. More recent methods proposed in [14,23,35] are based on word embeddings to address this problem.

The work in [23,33] proposes to evaluate future prediction based on scripts, i.e. prototypical event sequences, that were manually created. The knowledge base Regneri KB, which we used for evaluation, was extracted from these scripts.

Another common method for evaluating future prediction is to automatically extract event sequences from text [6]. However, the disadvantage is that the order of events in a text is not always a good proxy for temporal order. For example, events later the text might actually be entailed by events previously mentioned, i.e. already known events and new events are not distinguished.

Another approach to acquiring common sense knowledge about future prediction was recently presented in [24]. They manually created a set of stories, where each story contains mainly stereotypical causal and temporal event sequences. Each story is provided with a correct and a wrong ending, and the task for the system is to detect the correct ending (Story Cloze Test). Their data and task setting are appealing, although it seems that it is still too challenging: in order to solve the task, a system needs to combine machine reading (understanding of the semantics of the story) and common sense knowledge about future events. Our method (with associated knowledge base) addresses the simpler, but arguably more ambiguous task of predicting the next future event, given only one observed event.

Other recent efforts to enrich language resources for script learning using crowdsourcing are described in [22,43]. The work in [43] improves upon [33] by increasing the number of scripts and providing alignment information between functionally similar phrases. On the other hand, [22] provides a corpus with event sequence descriptions annotated in stories.

Methods for generalizing to unseen events. Apart from the methods introduced in Section 4, we summarize here several other methods that have been recently proposed for script learning.

The work in [13] proposes a hierarchical Bayesian model for classifying two event to be in an happens-before relation or not. For generalizing to unseen words, they suggest to use WordNet. However, the results from [23] suggests that the usage of word embeddings, and in particular their model (which we denoted Modi2014) is better for happens-before classification.

Evaluating the next event in text, is very similar to language modeling, where we want to predict the next word given all previous words. Therefore, various models, including LSTM [29,30] and the Log-Bilinear Language Model [35], have also been proposed for this task. However, all these models need large amounts of training data. Therefore, theses models are trained and tested on the textual order of events, where there is no guarantee that this conforms to temporal order.

Similar to the language modeling task, [21] proposed a neural network to predict an event given a set of surrounding events in text. Their method sums the event embeddings of the surrounding events, and thus does not use the order of the events. As such the events that occurred before and after are not distinguished.

Methods for classifying temporal relations in context. Classifying temporal relations in text is addressed, among others, by [7,20]. For example, the work in [7] extracts several features from two events occurring in the same (or neighboring) sentences, including tense markers, prepositions and part-of-speech bigrams. The work in [20] additionally exploits that two events in a causal relation tend to be in a happens-before relation.

Other resources for temporal reasoning. Here, we focused on a statistical machine learning approach for reasoning about happens-before relations given a knowledge base of true and wrong happens-before relations. However, the creation of such a knowledge-base is not trivial, and exploiting other publicly available resources for temporal reasoning should help to

[16] Using the Penn Discourse Treebank, we tried to extract happens-before relations of the form $(S, V_l, O) \rightarrow (S, V_r^{pos}, O)$ as in Table 2. However, we could extract only very few general happens-before rules that can be interpreted without further context. In particular, in a preliminary experiment, we extracted 2000 temporal rules from the Penn Discourse Treebank [31] and tried to judge the correctness of each rule. We found that when fixing only the subject, then, without using further context, only around 10 temporal happens-after relations could be judged as correct. No happens-before relations could be extracted when fixing both the subject and object. We expect similar results for the TimeML corpora.

improve coverage. For example, temporal relations like "planting a tree" before "watering the tree" could also be extracted from the question-answer pairs of the MC-Script dataset [25] that was created via crowdsourcing (see e.g. Q2 in Fig. 1 of [25]).

Applications of temporal reasoning. Although, we focused on an intrinsic evaluation of our system, the usage of happens-before relations and temporal reasoning, in general, has several important applications. For example, [38] points out the critical importance of temporal reasoning in the medical domain, e.g. for information retrieval in clinical texts. A knowledge base of temporal relations can also help the human expert in the design of security rules [26], and power management systems [15]. Finally, recent work [25,41] suggests to evaluate temporal reasoning on question answering tasks which can be closer to real world applications.

8. Conclusions

In this article, we proposed the Memory Comparison Network (MCN) for distinguishing between likely and unlikely future events. MCN is a memory network that can learn how to compare and combine the similarity of input events to event relations in the knowledge base (Section 3). MCN can effectively leverage an existing knowledge base of happens-before relations for future prediction of unseen events. Key to our method is the ability to automatically learn a similarity relation between events such that future prediction accuracy is optimized.

Our evaluations on two different knowledge bases suggests that our proposed method's future prediction accuracy is at par or better than other (deep) neural networks and rankSVM (Section 5). Furthermore, our method has the advantage that it gives explanations for its future prediction (Section 6). This is not only helpful to convince human decision makers, but also allows to judge when the rules in the knowledge base are insufficient (Section 6.3).

Acknowledgements

We would like to greatly thank the three anonymous reviewers for their many comments and suggestions which helped to improve the final version of this paper. We are also grateful to NEC Corporation for the support of this research. Part of this research was conducted when the third and fourth authors were affiliated with NEC.

References

[1] D. Bahdanau, K. Cho and Y. Bengio, Neural machine translation by jointly learning to align and translate, in: *International Conference on Learning Representations*, 2015.

[2] B. Bai, J. Weston, D. Grangier, R. Collobert, K. Sadamasa, Y. Qi, O. Chapelle and K. Weinberger, Supervised semantic indexing, in: *ACM International Conference on Information and Knowledge Management*, 2009, pp. 187–196.

[3] S. Basu, M. Bilenko and R.J. Mooney, A probabilistic framework for semi-supervised clustering, in: *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2004, pp. 59–68.

[4] A. Bordes, X. Glorot, J. Weston and Y. Bengio, Joint learning of words and meaning representations for open-text semantic parsing, in: *International Conference on Artificial Intelligence and Statistics*, 2012, pp. 127–135.

[5] J. Bromley, J.W. Bentz, L. Bottou, I. Guyon, Y. LeCun, C. Moore, E. Säckinger and R. Shah, Signature verification using a "Siamese" time delay neural network, *International Journal of Pattern Recognition and Artificial Intelligence* **7**(4) (1993), 669–688. doi:10.1142/S0218001493000339.

[6] N. Chambers and D. Jurafsky, Unsupervised learning of narrative event chains, in: *Annual Meeting of the Association for Computational Linguistics*, 2008, pp. 789–797.

[7] N. Chambers, S. Wang and D. Jurafsky, Classifying temporal relations between events, in: *Proceedings of the 45th Annual Meeting of the ACL on Interactive Poster and Demonstration Sessions*, Association for Computational Linguistics, 2007, pp. 173–176. doi:10.3115/1557769.1557820.

[8] T. Chklovski and P. Pantel, Verbocean: Mining the web for fine-grained semantic verb relations, in: *Conference on Empirical Methods in Natural Language Processing*, 2004, pp. 33–40.

[9] R. Collobert, K. Kavukcuoglu and C. Farabet, Torch7: A Matlab-like environment for machine learning, in: *BigLearn NIPS Workshop*, 2011.

[10] I. Dagan, O. Glickman and B. Magnini, The Pascal recognising textual entailment challenge, in: *Machine Learning Challenges. Evaluating Predictive Uncertainty, Visual Object Classification, and Recognising Tectual Entailment*, Springer, 2006, pp. 177–190.

[11] C. Fellbaum and G. Miller, *Wordnet: An Electronic Lexical Database*, MIT Press, 1998.

[12] F. Ferraro and B. Van Durme, A unified Bayesian model of scripts, frames and language, in: *AAAI*, 2016, pp. 2601–2607.

[13] L. Frermann, I. Titov and M. Pinkal, A hierarchical Bayesian model for unsupervised induction of script knowledge, in: *European Chapter of the Association for Computational Linguistics*, 2014, pp. 49–57.

[14] M. Granroth-Wilding and S. Clark, What happens next? Event prediction using a compositional neural network model, in: *AAAI Conference on Artificial Intelligence*, 2016, pp. 2727–2733.

[15] I. Haghighi, A. Jones, Z. Kong, E. Bartocci, R. Gros and C. Belta, Spatel: A novel spatial-temporal logic and its applications to networked systems, in: *Proceedings of the 18th International Conference on Hybrid Systems: Computation and Control*, ACM, 2015, pp. 189–198.

[16] G. Koch, R. Zemel and R. Salakhutdinov, Siamese neural networks for one-shot image recognition, in: *International Conference on Machine Learning*, 2015.

[17] D.D. Lewis, Y. Yang, T.G. Rose and F. Li, Rcv1: A new benchmark collection for text categorization research, *The Journal of Machine Learning Research* **5** (2004), 361–397.

[18] C.D. Manning, P. Raghavan and H. Schütze, *Introduction to Information Retrieval*, Vol. 39, Cambridge University Press, 2008.

[19] C.D. Manning, M. Surdeanu, J. Bauer, J. Finkel, S.J. Bethard and D. McClosky, The Stanford CoreNLP natural language processing toolkit, in: *ACL System Demonstrations*, 2014, pp. 55–60.

[20] P. Mirza, Extracting temporal and causal relations between events, in: *Proceedings of the ACL 2014 Student Research Workshop*, 2014, pp. 10–17. doi:10.3115/v1/P14-3002.

[21] A. Modi, Event embeddings for semantic script modeling, in: *Proceedings of the 20th SIGNLL Conference on Computational Natural Language Learning*, 2016, pp. 75–83. doi:10.18653/v1/K16-1008.

[22] A. Modi, T. Anikina, S. Ostermann and M. Pinkal, Inscript: Narrative texts annotated with script information, Preprint, 2017, arXiv:1703.05260.

[23] A. Modi and I. Titov, Inducing neural models of script knowledge, in: *Conference on Computational Natural Language Learning*, 2014, pp. 49–57.

[24] N. Mostafazadeh, N. Chambers, X. He, D. Parikh, D. Batra, L. Vanderwende, P. Kohli and J. Allen, A corpus and cloze evaluation for deeper understanding of commonsense stories, in: *Computational Linguistics*, 2016, pp. 839–849.

[25] S. Ostermann, A. Modi, M. Roth, S. Thater and M. Pinkal, Mcscript: A novel dataset for assessing machine comprehension using script knowledge, in: *LREC*, 2018.

[26] E. Paja, F. Dalpiaz and P. Giorgini, Modelling and reasoning about security requirements in socio-technical systems, *Data & Knowledge Engineering* **98** (2015), 123–143. doi:10.1016/j.datak.2015.07.007.

[27] J. Pennington, R. Socher and C.D. Manning, Glove: Global vectors for word representation, in: *Conference on Empirical Methods on Natural Language Processing*, 2014, pp. 1532–1543.

[28] K. Pichotta and R.J. Mooney, Statistical script learning with multi-argument events, in: *European Chapter of the Association for Computational Linguistics*, Vol. 14, 2014, pp. 220–229.

[29] K. Pichotta and R.J. Mooney, Learning statistical scripts with LSTM recurrent neural networks, in: *AAAI*, 2016, pp. 2800–2806.

[30] K. Pichotta and R.J. Mooney, Using sentence-level LSTM language models for script inference, in: *Proceedings of the Annual Meeting of the Association for Computational Linguistics*, Vol. 1, 2016, pp. 279–289.

[31] R. Prasad, N. Dinesh, A. Lee, E. Miltsakaki, L. Robaldo, A.K. Joshi and B.L. Webber, The penn discourse treebank 2.0, in: *LREC*, 2008.

[32] K. Radinsky, S. Davidovich and S. Markovitch, Learning to predict from textual data, *Journal of Artificial Intelligence Research* **45** (2012), 641–684. doi:10.1613/jair.3865.

[33] M. Regneri, A. Koller and M. Pinkal, Learning script knowledge with web experiments, in: *Annual Meeting of the Association for Computational Linguistics*, 2010, pp. 979–988.

[34] M. Richardson and P. Domingos, Markov logic networks, *Machine learning* **62**(1–2) (2006), 107–136. doi:10.1007/s10994-006-5833-1.

[35] R. Rudinger, P. Rastogi, F. Ferraro and B. Van Durme, Script induction as language modeling, in: *Conference on Empirical Methods in Natural Language Processing*, 2015, pp. 1681–1686.

[36] C. Stanfill and D. Waltz, Toward memory-based reasoning, *Communications of the ACM* **29**(12) (1986), 1213–1228. doi:10.1145/7902.7906.

[37] S. Sukhbaatar, J. Weston, R. Fergus et al., End-to-end memory networks, in: *Advances in Neural Information Processing Systems*, 2015, pp. 2440–2448.

[38] W. Sun, A. Rumshisky and O. Uzuner, Temporal reasoning over clinical text: The state of the art, *Journal of the American Medical Informatics Association* **20**(5) (2013), 814–819. doi:10.1136/amiajnl-2013-001760.

[39] J. Thorsten, Optimizing search engines using clickthrough data, in: *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2002, pp. 133–142.

[40] J. Thorsten, Training linear svms in linear time, in: *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2006, pp. 217–226.

[41] N. UzZaman, H. Llorens and J. Allen, Evaluating temporal information understanding with temporal question answering, in: *Semantic Computing (ICSC), 2012 IEEE Sixth International Conference on*, IEEE, 2012, pp. 79–82. doi:10.1109/ICSC.2012.34.

[42] O. Vinyals, C. Blundell, T. Lillicrap, D. Wierstra et al., Matching networks for one shot learning, in: *Advances in Neural Information Processing Systems*, 2016, pp. 3630–3638.

[43] L.D.A. Wanzare, A. Zarcone, S. Thater and M. Pinkal, Descript: A crowdsourced corpus for the acquisition of high-quality script knowledge, in: *The International Conference on Language Resources and Evaluation*, 2016.

[44] K.Q. Weinberger and L.K. Saul, Distance metric learning for large margin nearest neighbor classification, *Journal of Machine Learning Research* **10** (2009), 207–244.

[45] J. Weston, S. Chopra and A. Bordes, Memory networks, in: *International Conference on Learning Representations*, 2015.

[46] N. Xue, H.T. Ng, S. Pradhan, R. Prasad, C. Bryant and A.T. Rutherford, The CoNLL-2015 shared task on shallow discourse parsing, in: *Conference on Computational Natural Language Learning*, 2015, p. 2.