Help Me Understand This Conversation: Methods of Identifying Implicit Links Between CSCL Contributions

Help Me Understand This Conversation: Methods of Identifying Implicit Links Between CSCL Contributions

Mihai Masala[1,3(✉)], Stefan Ruseti[1,3], Gabriel Gutu-Robu[1], Traian Rebedea[1,3], Mihai Dascalu[1,2], and Stefan Trausan-Matu[1,2]

[1] University Politehnica of Bucharest,
313 Splaiul Independentei, 060042 Bucharest, Romania
mihai.masala@gmail.com, {stefan.ruseti,gabriel.gutu,traian.rebedea,
mihai.dascalu,stefan.trausan}@cs.pub.ro
[2] Academy of Romanian Scientists,
54 Splaiul Independentei, 050094 Bucharest, Romania
[3] Autonomous Systems, 22 Tudor Vladimirescu, 050883 Bucharest, Romania

Abstract. Multi-participant chat conversations are one of the most frequently employed Computer Supported Collaborative Learning tools due to their ease of use. Moreover, chats enhance knowledge sharing, sustain creativity and aid in collaborative problem solving. Nevertheless, the manual analysis of multi-participant chats is a difficult task due to the mixture of different topics and the inter-twinning of multiple discussion threads during the same conversation. Several tools that employ Natural Language Processing techniques have been developed to automatically identify links between contributions in order to facilitate the tracking of topics and of discussion threads, as well as to highlight key contributions in terms of follow-up impact. This paper proposes a novel method for detecting implicit links based on features computed using string kernels and word embeddings, combined with neural networks. This method significantly outperforms previous results on the same dataset. Due to its smaller size, our model represents an alternative to more complex deep neural networks, especially when limited training data is available as is the case of CSCL chats in a specific domain.

Keywords: Computer Supported Collaborative Learning
Implicit links identification · Natural Language Processing
String kernels · Neural networks

1 Introduction

In an ubiquitous digital and online connected society, a significant part of communication between individuals has shifted to online messaging, either in social networking platforms or standalone chat applications. These technologies are no longer used just for entertainment or staying in touch with kin, as they are

V. Pammer-Schindler et al. (Eds.): EC-TEL 2018, LNCS 11082, pp. 482–496, 2018.
https://doi.org/10.1007/978-3-319-98572-5_37

employed for more and more complex activities. In education, online chats have been used for distant and lifelong learning, serious games, but also as a supplement to traditional learning activities. One of the most up-to-date use cases resides within Massive Open Online Course (MOOC) platforms in which chat and online discussion forums allow participants to communicate with the tutors and among themselves. From a broader perspective, chats have been often used for Computer Support Collaborative Learning (CSCL) tasks [1] as they support frequent changes of context and interest, thus potentially generating multiple discussion threads within the same conversation fostering collaboration and creativity [2]. However, exactly these multiple discussion threads make chat conversations difficult to understand and follow, especially as the number of participants increases. To solve this problem, environments were designed to support multi-participant collaborative chats by allowing users to manually annotate a set of referred contributions [3]. We call these annotations *explicit links* between chat utterances as they are added by participants when issuing an utterance.

Explicit links are useful for adding some structure to CSCL chats, but complex conversations with several parallel discussion threads are still hard to follow. Despite this explicit annotation facility, in practice participants do not annotate every utterance as this process is tedious and it interrupts the conversation flow. Thus, a mechanism for discovering unannotated links between utterances is useful to facilitate the understanding of CSCL chats. These are called *implicit links* and they are important to improve the readability of muti-participant chats. The labeling of implicit links can be done with the help of Natural Language Processing (NLP) techniques which support the automated analysis of texts [4].

Our main objectives have been two-fold. First, we highlight that detection of implicit links is a similar, albeit more complex task to sentence selection from question answering. Second, we present a supervised approach using string kernels and neural networks previously used for answer selection [5] that improves the performance for detecting implicit links when compared to previous studies employing different semantic models and semantic distances in the WordNet ontology [6,7]. In sentence selection for question answering, the most suitable answer is considered the sentence most (semantically) similar to the question. Similarly, given the current utterance and a list of previous contributions within a specific (time or distance) frame, an implicit link can be considered as the most similar utterance to the current one. In a first simplification, we eliminate the context of the conversation and only compute the similarity between the current utterance and each candidate, followed by the selection of the one with the highest score. Another important difference between question answering and implicit link detection is that the datasets for the former are an order of magnitude larger than the ones for the latter. This means that simpler supervised models can achieve better results than more complex, deep learning solutions.

The paper continues with a review of the linguistic techniques and features used for identifying implicit links. The following section contains a presentation of the proposed supervised method, with additional details about the corpus of conversations and the neural network model. Afterwards, results are presented

together with a comparison to previous studies in order to highlight the performance increase of the proposed supervised method. The last section concludes the paper and includes a discussion on the advantages of our approach.

2 Related Work

2.1 Implicit Links Detection

The process of manually annotating explicit links has two main limitations: (a) it is time consuming and it breaks down the conversation flow; (b) it may be subjective to the particularities of the user. Mechanisms for automated annotation of links have been designed to replace the manual labour performed by chat participants. As the links discovered by such algorithms and techniques are not explicitly added by users, the process is called implicit links detection [8] or chat disentanglement [9]. Multiple methods can be employed for solving this task.

Semantic Models and Ontologies. Previous experiments used semantic distances based on the WordNet ontology, together with several semantic models [6,7] to determine the optimal window (in terms of distance or time) to identify implicit references. The utterance belonging to a considered window that had the highest semantic similarity score with the referred utterance was chosen as its implicit reference. A corpus consisting of 55 chat conversations manually annotated with explicit links was used to evaluate the performance of this approach [6].

The experiments used three semantic distances computed using the WordNet [10] lexical ontology. In addition, three semantic models widely used in NLP tasks were also employed in the experiment. Latent Semantic Analysis (LSA) [11] builds a matrix of term-document occurrences which is decomposed using Singular Value Decomposition and then the dimensionality is reduced to a latent semantic space; the semantic relatedness between words is computed using cosine similarity in this space. Latent Dirichlet Allocation (LDA) [12] stores each word or text as a probability distribution over latent topics; the Jensen-Shannon dissimilarity is used to compute the relatedness between two units of texts (e.g. utterances). Word2vec [13] is based on neural word embeddings which are computed starting from word n-gram co-occurrences; the similarity between words is computed within the embedded space by means of cosine similarity.

Neural Networks. Neural networks have greatly contributed to recent advancements in various NLP tasks as they are able to automatically model complex combinations of simple inputs such as word embeddings. One relevant experiment for our study considered both meta information (e.g. time and distance between utterances, same/different author for the utterances, mentioning the author's name in the other utterance) and the content of the utterances. State-of-the-art results were obtained on chat disentanglement tasks by using Recurrent Neural Networks (RNNs) [14]. Our method is aimed at using a slightly

simpler neural network, with fewer parameters, that receives as input meta information, semantic features (e.g. word embeddings) and lexical features computed using string kernels.

2.2 Lexical and Semantic Models for Text Similarity

In this section we present two recent methods for computing semantic similarity between text documents, namely string kernels and neural models over word embeddings. Both are seen in the NLP community as powerful alternatives to ontologies and semantic models like LSA and LDA frequently used in the educational community for processing different types of texts, including CSCL chats.

Word Embeddings. Several alternatives for computing word embeddings on very large datasets have been proposed in recent years. Word embeddings are a method for representing words in a lower dimensional space based on their context of appearance in the corpus. While word2vec [15] is a generative neural model, GloVe embeddings [16] are computed using a count-based approach. However, both models are working on the word level as opposed to fastText [17] which is considered an extension of word2vec working on character n-grams.

String Kernels. String kernels [18] are kernel functions that work at the character level. Instead of projecting the documents into a high-dimensional space and performing computations in that space, string kernels employ a kernel function that simulates the dot-product of two elements in that high-dimensional space; the more similar the documents are, the higher the value of the kernel function. String kernels assume that any good measure of similarity between two documents is strongly related to the number of shared sub-strings of a given size in those documents. String kernels are obtained by varying the sizes of the n-grams (usually between 2 and 10 characters) and the function used for computing the n-gram overlap. The most common string kernels (i.e. intersection, presence and spectrum [19]) are based on the number of co-occurrences of shared n-grams. Spectrum kernel (see Eq. 1) is computed as the dot-product of shared n-grams frequencies. Instead of multiplying the frequencies, intersection kernel (see Eq. 2) uses the minimum of these frequencies. In contrast, the presence kernel (see Eq. 3) uses presence bits to encode if a n-gram is present or not in a string. For a fair comparison of strings of different sizes, normalized versions of these kernels are used in follow-up experiments.

$$k_p(s,t) = \sum_{v \in \sum_p} num_v(s) \cdot num_v(t) \tag{1}$$

$$k_p^{\cap}(s,t) = \sum_{v \in \sum_p} min\{num_v(s), num_v(t)\} \tag{2}$$

$$k_p^{0/1}(s,t) = \sum_{v \in \sum_p} in_v(s) \cdot in_v(t) \tag{3}$$

where:

- $\sum_p$ = all p-grams of a given size p
- $num_v(s)$ = number of occurrences of string (n-gram) v in document s
- $in_v(s) = 1$ if string (n-gram) v occurs in document s, 0 otherwise

String kernels have also been used as a feature extraction method and combined with different classifiers to solve various problems such as native language identification [20], digit recognition and protein fold predictions [21]. Recently, Beck et al. [22] used Gaussian Process regression on string kernels to optimize the weights related to each n-gram size and the decay parameters for gaps and matches. Their model outperforms linear baselines for sentiment analysis, but lags behind a non-linear baseline, giving evidence that extending string kernels with non-linearities can provide better results. In a similar manner, Masala et al. [5] have used a neural network to assign weights to different n-gram sizes and also to non-linearly combine different kernels using a neural network. Their results show that a shallow neural network using string kernels and word embeddings can achieve very good results in question answering with a much smaller model than state-of-the-art deep models. We propose to use a similar approach for implicit links detection.

Neural Models for Text Similarity. Neural models for computing similarity between sentences have been widely used in question answering in recent years [23–25]. For the specific task of answer selection, a question and a pool of candidate answers are given and the model must discriminate the most likely answer from all other candidates. In general, neural networks computing the similarity between two sentences (or documents) generate inner representations for both text and then apply a similarity function on these representations. Usually, the representation is computed using a Bidirectional Long Short-Term Memory (Bi-LSTM) network [26] or a convolution neural network (CNN) [27].

Adding attention mechanisms to neural models proved to be a very efficient method in question answering, outperforming previous models. The intuition behind the attention mechanism is that, by looking at the question, different weights can be assigned to different parts of the candidate answer, thus allowing the model to focus on the relevant parts of the candidate. dos Santos et al. [24] combine the question and candidate representations obtained from the Bi-LSTM or CNN into a single, fixed-length matrix. Using this matrix, attention weights are extracted and used to modify both the question and the answer representations.

Instead of computing the attention weights by only looking at the inner representations of the question and the answer, Bachrach et al. [23] also use a global view of the question and of the answer, obtained using a multilayer perceptron (MLP) on a bag-of-words representation. In addition, Wang et al. [28] propose a general method for word-level sentence matching. After computing the attention weights, comparison functions (e.g. element-wise subtraction and multiplication, a simple MLP) are used for combining the representation of the

answer with the attention-weighted representation of the question, at word level. For the final classification, a CNN is used on top of this new representation.

3 Method

3.1 Corpus of CSCL Chat Conversations

The corpus used for this experiment consists of 55 chat conversations among undergraduate Computer Science students [6]. Students had to discuss about web technologies supporting collaborative work and how these can be efficiently used by a software company. While each participant had to be the supporter of a different technology, in the end they had to reach an agreement on the solution that best suited the company. To this aim, the discussions were similar to the problem-solving tasks usually encountered in other CSCL platforms - e.g., Stahl's Virtual Math Teams project [29]. Stahl demonstrated that problems which are difficult to be solved independently can be answered more effectively by groups of students involved in collaborative learning.

Two methods were considered for the matching process between the automatically detected implicit links and the manually annotated explicit links. The first one is the *perfect match* in which the two referenced utterances (explicit and predicted link) are identical. The second one is the *in-turn matching* - i.e., the implicit link belongs to a uninterrupted block of subsequent utterances written by the same participant, as the explicit link.

The conversations were performed using *ConcertChat* [3], which enables participants to explicitly refer one or more previous turns, when uttering their own contribution. These explicit annotations were used for computing the accuracy of the proposed method using both exact and in-turn matching. The corpus contains about 4500 explicit links and 17600 utterances, meaning that 29% of contributions have a corresponding explicit link. Table 1 shows fragments extracted from chat conversations depicting an exact match and an in-turn match, where the emphasized text marks the utterance which denotes the implicit link. The explicit link added by the participants within the conversation is presented in the Ref ID (reference ID) column.

Gutu et al. [6] have previously shown that a distance of 5 utterances covers 82% of explicit links in the dataset, a distance of 10 covers 95%, while a distance of 20 covered more than 98%. As for time, a 1 min timeframe covers 61%, whereas 93% of explicit links are covered by a 3 min timeframe, and more than 97% by 5 min window. For this reason, windows of 5 and 10 utterances, and 1, 2, and 3 min were used for the current experiments.

3.2 Network Model and Design

One of our key insights is that there is a strong resemblance between the way implicit links relate to their respective utterances and how an answer connects to a question. Therefore we propose a neural model inspired from the answer

Table 1. Fragments extracted from conversations showing exact and in-turn matching. (Implicit link is highlighted in bold)

Utt. ID	Ref. ID	Speaker	Content
			Exact matching
87		**Tibi**	**you can't rely on anyone ..there should be authorised people writing on this site**
88		Octavian	but if want to find organized information about that product wiki is the way to go
89	87	Oana	the people writing on the web site are authorized
			In-turn matching
193		Alin	and they embed only what you need
		... (several utterances of the same participant, Alin) ...	
196		**Alin**	**this is just an example of how hidden markov model can be used**
199	193	Razvan	you talked about the prior, does this mean that the method ignores the sequences that are after the word it's tagging and only takes into account the ones before it?

selection task. The goal of our model, inspired from the work of Masala et al. [5] is to find a combination of string kernels that can better capture the notion of implicit links between utterances. The previous most similar utterance to the current one is selected as the implicit link.

We combine three string kernels (spectrum, presence and intersection) with five n-gram ranges: 1–2, 3–4, 5–6, 7–8 and 9–10. We thus compute for each pair of sentences a feature vector $v \in \mathcal{R}^{15}$. A simple feed-forward multilayer perceptron (MLP) with one hidden layer is trained over these features. The MLP computes a similarity score for each utterance that is a candidate for an implicit link. The utterance that has the highest similarity score is selected as the discovered implicit link. For all experiments the hidden layer size is set to 8, using a batch size of 100 and Adam [30] optimizer for training. The objective function is the hinge loss defined in Eq. 4, similar to the one proposed by Hu and Lu [31] for finding similarities between two sentences, with the margin M set to 0.1.

$$e(u_r, u^+, u^-) = max(0, M + sim(u_r, u^-) - sim(u_r, u^+)) \tag{4}$$

where:

- u_r is the current utterance, for which the link is computed
- u^+ is the correct (explicitly) linked utterance
- u^- is an incorrect utterance from the current window
- $sim(u_r, u)$ is the similarity score computed by the MLP between the representations of two utterances
- M is the desired margin between positive and negative examples

In addition, we experiment with augmenting the features obtained using string kernels with semantic and conversation-specific features. Given two utterances, we compute the cosine similarity between the average vector computed

in the embedding space (using word2vec, FastText, and GloVe) for all words in each utterance. The information retrieved from the chat structure consists of differences expressed as counts of in-between utterances and time between the two considered utterances. Finally, for each candidate utterance (for a link) we compute two conversation specific features: if its author is the same as for the current utterance and whether the utterance contains a question.

4 Results

We evaluated the proposed methods on the previously described dataset. Our supervised neural model is compared with an unsupervised method based on string kernels and with state-of-the-art methods for implicit links detection and answer selection. For the unsupervised methods, the n-gram range (3–7) was selected to optimize the performance on a small evaluation set. For all supervised methods, a 10-fold cross-validation procedure was employed. Note that all results are reported on the test set. The word2vec [15] embeddings were pretrained on the Google News Dataset. The GloVe embeddings [16] were trained on a Wikipedia 2014 dump and Gigaword 5[1]. The FastText embeddings [17] were also trained on Wikipedia. For computing string kernels we employed an open-source library[2].

As baselines, we have used both supervised and unsupervised methods employed for detecting implicit links and answer selection, namely:

- Path Length [6]: The best results for detecting implicit links on the same dataset were achieved using WordNet Path Length as similarity distance. Path Length computes the length of the shortest path between two concepts in the WordNet ontology.
- AP-BiLSTM [24]: The current utterance and the candidate utterance are both passed through a Bidirectional LSTM network. The outputs of both Bi-LSTMs, containing the hidden states at each time step, are afterwards combined into a single matrix. From this matrix, attention vectors are extracted via column-wise and row-wise max pooling, and new representations for the utterances are computed. For the classification step, cosine similarity is used on the new representation of the utterances. AP-BiLSTM is one of the top performing deep learning models for answer selection.

The accuracy obtained by the baseline methods are presented in Table 2. While the AP-BiLSTM model is capable of capturing complex semantic relations for the answer selection task [24], its accuracy is low for our problem, offering performance just on par with the unsupervised path length semantic distance for the exact match (and even worse for in-turn match). The poor performance can be explained by the small size of the training dataset relative to the high number of parameters required by the model.

[1] https://catalog.ldc.upenn.edu/LDC2011T07.
[2] http://string-kernels.herokuapp.com/.

Table 2. Proposed baselines for implicit links detection (Exact matching accuracy - top row & In-turn matching accuracy - bottom row).

Window (utterances)	5			10		
Time (mins)	1	2	3	1	2	3
Path Length [6]	32.44%	32.44%	-	31.88%	31.88%	-
	41.49%	41.49%	-	40.78%	40.78%	-
AP-BiLSTM [24]	32.95%	32.39%	33.97%	33.86%	28.89%	24.49%
	34.53%	35.89%	37.58%	35.10%	31.82%	28.32%
Intersection kernel	31.40%	33.87%	33.58%	31.71%	32.24%	29.47%
	34.59%	39.58%	40.01%	34.78%	37.66%	35.24%
Presence kernel	31.84%	33.97%	33.58%	31.80%	32.33%	29.67%
	34.94%	39.81%	40.01%	34.89%	37.71%	35.41%
Spectrum kernel	31.21%	33.45%	33.17%	31.39%	31.56%	28.75%
	34.34%	39.12%	39.49%	34.46%	36.72%	34.26%

Similarly, string kernels as an unsupervised method provide mixed results when compared to path length: improvements are small and only for a larger time window (e.g. 2-min time window) for exact match. Furthermore there is no significant difference between any of the three string kernels functions.

Table 3 introduces the results obtained using the proposed neural model, with and without additional chat features, but without any semantic information. The results highlight the fact that chat and conversation specific features, especially the time and in-between turns distances between utterances are very important for detecting implicit links. A similar conclusion was established in previous studies as the Path Length method from Table 2 also uses a weighting for the path length semantic score, given the distance between the two utterances [6]. Nevertheless, conversation specific features (same author and whether the candidate utterance contains a question) are also relevant features improving the results both on their own and additional to window-based ones.

Compared to previous results using path length, the proposed neural model achieves a substantial improvement from 32.44% (window/time frame: 5 utterances/1 min) to 47.85% (window/time frame: 10 utterances/3 min) accuracy for exact match. Two important results should be highlighted. First, the neural network model achieves improvement for all combinations of frames considered. Second, this model is able to improve the results even when the number of candidates is higher (e.g. larger window/time frames); this was not the case with any of the baselines presented in Table 2.

The results of the experiments using semantic information are presented in Table 4. For all models involving semantic information, experiments were conducted using several word embeddings: word2vec, FastText, and Glove (embedding sizes 100 and 300). While semantic information increased the performance of our model, the gain is not significant especially compared to the performance gain obtained by adding chat specific features. This shows that

Table 3. Proposed methods without semantic information (Exact matching accuracy & In-turn matching accuracy).

Window (utterances)	5			10		
Time (mins)	1	2	3	1	2	3
NN using sk	35.21%	35.55%	35.77%	35.55%	34.08%	30.24%
	36.90%	39.39%	39.95%	37.02%	37.47%	33.74%
NN using sk + window + time [32]	33.40%	40.40%	41.87%	33.74%	41.30%	42.66%
	35.21%	44.01%	45.25%	35.44%	45.25%	47.29%
NN using sk + question + author	37.02%	39.84%	39.50%	37.35%	38.26%	35.10%
	39.05%	44.46%	44.46%	39.16%	42.32%	39.16%
NN using sk + window + time + question + author	37.92%	45.48%	47.06%	38.14%	46.27%	**47.85%**
	39.39%	49.66%	51.80%	39.50%	50.79%	**52.93%**

Note: sk - string kernels; window - # of in-between utterances; time - elapsed time between contributions; question - whether the utterance contains a question; author - if the utterance shares the same author as the utterance containing the link.

Table 4. Proposed methods enhanced with semantic information (Exact matching accuracy & In-turn matching accuracy).

Window (utterances)	5			10		
Time (mins)	1	2	3	1	2	3
NN using sk + sem [32]	36.45%	36.90%	36.00%	36.68%	35.10%	31.26%
	38.14%	40.47%	40.29%	38.14%	38.26%	34.76%
NN using sk + sem + window + time [32]	34.98%	41.64%	44.24%	35.32%	42.21%	44.48%
	36.68%	45.03%	48.53%	36.90%	45.93%	49.32%
NN using sk + sem + question + author	38.14%	41.19%	40.85%	38.48%	39.95%	36.34%
	39.84%	45.03%	45.03%	40.06%	43.56%	39.72%
NN using sk + sem + window + time + question + author	37.02%	46.38%	48.08%	37.24%	47.29%	**49.09%**
	38.60%	50.00%	52.25%	38.71%	51.46%	**53.83%**

framing the implicit link detection problem as a purely answer selection task will yield a inherently limited model. The largest gains can be observed for a longer frame (e.g. window/time frame: 10 utterances/3 min, improvement from 47.85% to 49.09%) which means that semantic information becomes relevant for capturing more distant implicit links.

Turning to a qualitative interpretation of the results obtained by the neural model, Table 5 provides examples in which our model is correctly predicting the implicit link. In the top of the table, we present two examples of utterances that represent direct answers to previously asked questions. The proposed model can also detect when an author continues his idea in a new utterance (see lower part of Table 5).

Table 5. Example of correct implicit link prediction (**explicit link**/*predicted link*).

Utt. ID	Ref. ID	Speaker	Content
74		***Cristi***	***and if it is not a free chat, then it's not that easy***
75	74	Oana	well, there are tons of free chats on the web, I don't believe that this is a real problem
76		Luis	On a chat you can have multiple thread discussions creating confusion
77		Alex	but, I don't think that the chats purpose is to store the information..
78		***Cristi***	***right, again but haw do you advertise to use the same chat?***
79	78	Oana	it's the same advertisement as with the forums, or the blogs: on-line advertisment
176		***Oana***	***can't they (wikis) be made private: i mean have groups of users who have permission to post?***
177		Florin	But if one evil man wants to make joke, anothers 10 will repair the damage
178		Mihaela	in all my experience i did not encounter such i thing...
179	176	Oana	for example, in an university: only teachers can add content

Table 6. Example of wrong implicit link prediction (**explicit link**/*predicted link*).

Utt. ID	Ref. ID	Speaker	Content
91		**Ciprian**	**we shall our know-how to make the best use of our separate products by combining them in one integrated tool**
92		Cristi	I think that we can use all of them because it's clear that they have different qualities for example ...
93		*Ionut*	*Yeah,let's think of a joint venture for our companies to create a product tocombine them all, an all-in-one learning application for large groups.*
94		Ciprian	so, wiki is the perfect tool to create semantic networks as tools for knowledge representation
95		Rudi	I propouse to give chatting options for the registrated users
96	91	Cristi	Let all think how his product will integrate better and what value it adds to the joint product.
87		**Radu**	**Blogs are also based on PHP and MySQL, and are really easy to use and put in practice**
88		*Raluca*	*do you have another ideas about the implementation?*
89	87	Radu	See WordPress or Blogger...

However, as the best accuracy is 49.09%, in about half of the cases our model is unable to detect the correct link. This is due to, but not limited, to more complex utterance interaction that can mislead even human annotators (see upper part of Table 6). In other cases, utterances simply do not provide enough information (see the last utterance in the bottom of Table 6). These limitations may be overcome by extracting more complex features from each utterance. Nevertheless some limitations are also due to the way the problem was formulated (as an answer selection task).

5 Discussions and Conclusions

Chat conversation have been used in CSCL tasks especially for solving difficult problems in larger groups of students. These conversations foster multiple parallel discussions threads and competing discussion topics that make the conversations hard to follow. Automated NLP techniques come to help by interpreting chats and detecting links between utterances. This process aims at supporting or even replacing the time-consuming work of explicit annotation. For example, it would be great to have a tool that suggests an implicit link for each utterance in a conversation (either chat, but maybe even a discussion forum within a MOOC). As the accuracy is slightly below 50%, the participants would still need to correct the automatic suggestion in half of the situations. On the bright side, it means that half of the time the predicted link is correct and the conversation flow will not be interrupted to manually pick an explicit link.

This paper proposes answer selection techniques for implicit links detection in chats. We explored a supervised neural model using string kernels, as well as additional domain-specific and semantic features. While string kernels alone performed similarly to semantic similarity methods used in previous studies, the neural network learned how to combine efficiently lexical, semantic and chat related features, and significantly increased the accuracy for the detection of implicit links. The method was also compared with state-of-the-art deep-learning models for question answering and achieved better results, proving to be a viable solution for smaller datasets. To our knowledge, this is the first approach of its kind.

Performance was not improved by a large margin by adding semantic information. More experiments need to be conducted with other semantic similarity measures as features, considering that each model might capture different facets of the relations between sentences. Another improvement can be achieved by also considering the context of the conversation, and not only a pair of utterances. This highlights a limitation of our current assumption which oversimplifies the problem, albeit that implicit links can be modelled as a sentence selection task, ignoring the context of the conversation in which utterances occur. Models that use the whole conversation for link detection might be more suitable in this case, but require a larger dataset for training.

The described approach has multiple practical implications. First, it introduces the possibility to split the conversation and easily follow multiple conversation threads, a functionality of great benefits for modelling online conversations in education and beyond. Second, summarizing relevant contributions for each participant by taking into account inter-dependencies between contributions enables the generation of an overview of their involvement. This process also creates a strong basis for assessing the degree of collaboration between participants. Third, implicit links also model cohesive links among contributions; thus, the avoidance of a high inter-twining of multiple concurrent discussion threads and keeping a cohesive discourse makes the conversation easier to follow.

Acknowledgements. This research was partially supported by the FP7 2008-212578 LTfLL, EC H2020-644187 *Realising an Applied Gaming Eco-system* (RAGE), and POC-2015 P39-287 IAVPLN projects.

References

1. Stahl, G.: Group Cognition: Computer Support for Building Collaborative Knowledge. MIT Press, Cambridge, MA (2006)
2. Trausan-Matu, S.: Computer support for creativity in small groups using chats. Annal. Acad. Rom. Sci. Ser. Sci. Technol. Inf. **3**(2), 8190 (2010)
3. Holmer, T., Kienle, A., Wessner, M.: Explicit referencing in learning chats: needs and acceptance. In: Nejdl, W., Tochtermann, K. (eds.) EC-TEL 2006. LNCS, vol. 4227, pp. 170–184. Springer, Heidelberg (2006). https://doi.org/10.1007/11876663_15
4. Manning, C.D., Schtze, H.: Foundations of Statistical Natural Language Processing. MIT Press, Cambridge, MA (1999)
5. Masala, M., Ruseti, S., Rebedea, T.: Sentence selection with neural networks using string kernels. Procedia Comput. Sci. **112**, 1774–1782 (2017)
6. Gutu, G., Dascalu, M., Rebedea, T., Trausan-Matu, S.: Time and semantic similarity what is the best alternative to capture implicit links in CSCL conversations? In: 12th International Conference on Computer-Supported Collaborative Learning (CSCL 2017), pp. 223–230. ISLS (2017)
7. Gutu, G., Dascalu, M., Ruseti, S., Rebedea, T., Trausan-Matu, S.: Unlocking the power of word2vec for identifying implicit links. In: 17th IEEE International Conference on Advanced Learning Technologies (ICALT 2017), p. 199200. IEEE (2017)
8. Trausan-Matu, S., Rebedea, T.: A polyphonic model and system for inter-animation analysis in chat conversations with multiple participants. In: Gelbukh, A. (ed.) CICLing 2010. LNCS, vol. 6008, pp. 354–363. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-12116-6_29
9. Elsner, M., Charniak, E.: Disentangling chat. Comput. Linguist. **36**(3), 389–409 (2010)
10. Miller, G.A.: Wordnet: a lexical database for English. Commun. ACM **38**(11), 39–41 (1995)
11. Landauer, T.K., Dumais, S.T.: A solution to plato's problem: the latent semantic analysis theory of acquisition, induction and representation of knowledge. Psychol. Rev. **104**(2), 211240 (1997)

12. Blei, D.M., Ng, A.Y., Jordan, M.I.: Latent dirichlet allocation. J. Mach. Learn. Res. **3**(4–5), 9931022 (2003)
13. Mikolov, T., Chen, K., Corrado, G., Dean, J.: Efficient estimation of word representation in vector space. In: Workshop at ICLR (2013)
14. Mehri, S., Carenini, G.: Chat disentanglement: Identifying semantic reply relationships with random forests and recurrent neural networks. In: Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 1: Long Papers). vol. 1, pp. 615–623 (2017)
15. Mikolov, T., Sutskever, I., Chen, K., Corrado, G.S., Dean, J.: Distributed representations of words and phrases and their compositionality. In: Advances in Neural Information Processing Systems, pp. 3111–3119 (2013)
16. Pennington, J., Socher, R., Manning, C.D.: Glove: global vectors for word representation. In: Empirical Methods in Natural Language Processing (EMNLP), pp. 1532–1543 (2014). http://www.aclweb.org/anthology/D14-1162
17. Bojanowski, P., Grave, E., Joulin, A., Mikolov, T.: Enriching word vectors with subword information. arXiv preprint arXiv:1607.04606 (2016)
18. Lodhi, H., Saunders, C., Shawe-Taylor, J., Cristianini, N., Watkins, C.: Text classification using string kernels. J. Mach. Learn. Res. **2**(Feb), 419–444 (2002)
19. Ionescu, R.T., Popescu, M., Cahill, A.: Can characters reveal your native language? a language-independent approach to native language identification. In: EMNLP, pp. 1363–1373 (2014)
20. Ionescu, R.T., Popescu, M., Cahill, A.: String kernels for native language identification: insights from behind the curtains. Comput. Linguist. **42**, 491–525 (2016)
21. Gönen, M., Alpaydın, E.: Multiple kernel learning algorithms. J. Mach. Learn. Res. **12**(Jul), 2211–2268 (2011)
22. Beck, D., Cohn, T.: Learning kernels over strings using Gaussian processes. In: Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 2: Short Papers), vol. 2, pp. 67–73 (2017)
23. Bachrach, Y., Zukov-Gregoric, A., Coope, S., Tovell, E., Maksak, B., McMurtie, C.: An attention mechanism for answer selection using a combined global and local view. arXiv preprint arXiv:1707.01378 (2017)
24. dos Santos, C.N., Tan, M., Xiang, B., Zhou, B.: Attentive pooling networks. CoRR, abs/1602.03609 (2016)
25. Tan, M., dos Santos, C.N., Xiang, B., Zhou, B.: Improved representation learning for question answer matching. In: ACL, vol. 1 (2016)
26. Graves, A., Schmidhuber, J.: Framewise phoneme classification with bidirectional LSTM networks. In: Proceedings of 2005 IEEE International Joint Conference on Neural Networks, IJCNN 2005, vol. 4, pp. 2047–2052. IEEE (2005)
27. Kim, Y.: Convolutional neural networks for sentence classification. In: Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP 2014), pp. 1746–1751, August 2014
28. Wang, S., Jiang, J.: A compare-aggregate model for matching text sequences. arXiv preprint arXiv:1611.01747 (2016)
29. Stahl, G.: Studying Virtual Math Teams. Springer, New York, NY (2009). https://doi.org/10.1007/978-1-4419-0228-3
30. Kingma, D.P., Adam, B.J.: A method for stochastic optimization. arXiv preprint arXiv:1412.6980 (2014)

31. Hu, B., Lu, Z., Li, H., Chen, Q.: Convolutional neural network architectures for matching natural language sentences. In: Advances in neural information processing systems, pp. 2042–2050 (2014)
32. Masala, M., Ruseti, S., Gutu-Robu, G., Rebedea, T., Dascalu, M., Trausan-Matu, S.: Identifying implicit links in CSCL chats using string kernels and neural networks. In: Penstein Rosé, C., Martínez-Maldonado, R., Hoppe, H.U., Luckin, R., Mavrikis, M., Porayska-Pomsta, K., McLaren, B., du Boulay, B. (eds.) AIED 2018, Part II. LNCS (LNAI), vol. 10948, pp. 204–208. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-93846-2_37