Ranking Kernels for Structures and Embeddings: A Hybrid Preference and Classification Model

Kateryna Tymoshenko[†] and **Daniele Bonadiman**[†] and **Alessandro Moschitti**
[†]DISI, University of Trento, 38123 Povo (TN), Italy
Qatar Computing Research Institute, HBKU, 34110, Doha, Qatar
{kateryna.tymoshenko,d.bonadiman}@unitn.it
amoschitti@gmail.com

Abstract

Recent work has shown that Tree Kernels (TKs) and Convolutional Neural Networks (CNNs) obtain the state of the art in answer sentence reranking. Additionally, their combination used in Support Vector Machines (SVMs) is promising as it can exploit both the syntactic patterns captured by TKs and the embeddings learned by CNNs. However, the embeddings are constructed according to a classification function, which is not directly exploitable in the preference ranking algorithm of SVMs. In this work, we propose a new hybrid approach combining preference ranking applied to TKs and pointwise ranking applied to CNNs. We show that our approach produces better results on two well-known and rather different datasets: WikiQA for answer sentence selection and SemEval cQA for comment selection in Community Question Answering.

1 Introduction

Recent work on learning to rank (L2R) has shown that deep learning and kernel methods are two very effective approaches, given their ability of engineering features. In particular, in question answering (QA), Convolutional Neural Networks (CNN), e.g., (Severyn and Moschitti, 2015; Miao et al., 2016; Yin et al., 2016) can automatically learn the representation of question and answer passage (Q/AP) in terms of word embeddings and their non-linear transformations. These are then used by the other layers of the network to measure Q/AP relatedness. In contrast, Convolution Tree Kernels (CTK) can be applied to relational structures built on top of syntactic/semantic structures derived from Q/AP text (Tymoshenko et al., 2016a). CNNs as well as CTKs can achieve the state of the art in ranking APs or also questions. Considering their complementary approach for generating features, studying ways to combine them is very promising. In (Tymoshenko et al., 2016a), we investigated the idea of extracting layers from CNNs and using them in a kernel function to be further combined with CTKs in a composite reranking kernel. This was used in an SVM^{Rank} (Joachims, 2002) model, which obtained a significant improvement over the individual methods. However, the simple use of CNN layers as vectors in a preference ranking approach is intutively not optimal since such layers are basically learnt in a classification model, thus they are not optimized for SVM^{Rank}.

In this work, we further compare and investigate different ways of combining CTKs and CNNs in reranking settings. In particular, we follow the intuition that as CNNs learn the embeddings in a classification setting they should be used in the same way for building the reranking kernel, i.e., we need to use the embeddings in a pointwise reranking fashion. Therefore, we propose a hybrid preference-pointwise kernel, which consists in (i) a standard reranking kernel based on CTKs applied to the Q/AP structural representations; and (ii) a classification kernel based on the embeddings learned by neural networks. The intuition about the hybrid models is to add CNN layer vectors, not their difference, to the preference CTK. That is, CNN layers are still used as they were used in a classification setting whereas CTKs follow the standard SVM^{Rank} approach.

We tested our proposed models on the answer sentence selection benchmark, WikiQA (Yang et al., 2015), and the benchmark from cQA SemEval-2016 Task 3.A[1] corpus. We show that

[1] http://alt.qcri.org/semeval2016/

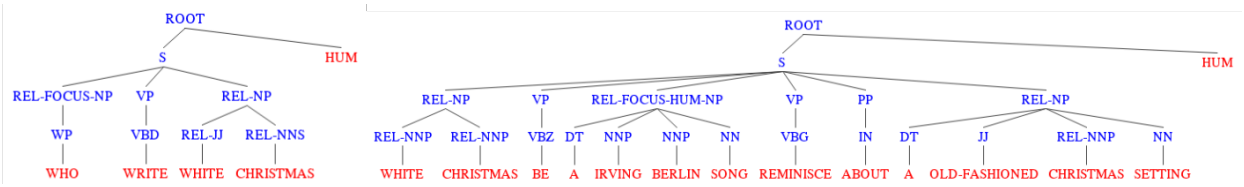

Figure 1: Shallow chunk-based tree representation of a question in the Q/AP pair: Q: *"Who wrote white Christmas?"*, AP: *"White Christmas is an Irving Berlin song"*.

the proposed hybrid kernel consistently outperforms standard reranking models in all settings.

2 Answer Sentence/Comment Selection

We focus on two question answering subtasks: answer sentence selection task (AST) and the comment selection task from cQA.

AST consists in selecting correct answer sentences (i.e., an AP composed of only one sentence) for a question Q from a set of candidate sentences, $S = \{s_1, ..., s_N\}$. In factoid question answering, Q typically asks for an entity or a fact, e.g., time location and date. S is typically a result of so-called *primary search*, a result of fast-recall/low-precision search for potential answer candidates. For example, it could be a set of candidate APs returned when running a search engine over a large corpus using Q as a search query. Many such APs are typically not pertinent to the original question, thus automatic approaches for selecting those useful are very valuable.

cQA proposes a task similar to AST, where Q is a question asked by a user in a web forum and S are the potential answers to the question posted as comments by other users. Again, many comments in a cQA thread do not contain an answer to the original question, thus raising the need for automatic comment selection.

The crucial features for both tasks capture information about the relations between Q and an AP. Manual feature engineering can provide competitive results (Nicosia et al., 2015), however, it requires significant human expertise in the specific domain and is time-consuming. Thus, machine learning methods for automatic feature engineering are extremely valuable.

3 CTK and CNN models

Our baselines are the standalone CTK and CNN models originally proposed in (Severyn et al., 2013; Severyn and Moschitti, 2015) and further advanced in (Tymoshenko et al., 2016a,b). The following subsections provide a brief overview of these models.

3.1 CTK structures

The CTK models are applied to syntactic structural representations of Q and AP. We used shallow chunk-based and constituency tree representations in AST (Tymoshenko et al., 2016a) and cQA (Tymoshenko et al., 2016b), respectively. We follow the tree construction algorithms provided in the work above. Due to the space restrictions, we present only high-level details below.

A shallow chunk-based representations of a text contains lemma nodes at leaf level and their part-of-speech (POS) tag nodes at the preterminal level. The latter are further grouped under the chunk and sentence nodes.

A constituency tree representation is an ordinary constituency parse tree. In all representations, we mark lemmas that occur in both Q and AP by prepending the **REL** tag to the labels of the corresponding preterminal nodes and their parents.

Moreover, in the AST setting, often question and focus classification information is used (Li and Roth, 2002), thus we enrich our representation with the question class and focus information, when is available.

Additionally, we mark AP chunks containing named entities that match the expected answer type of the question by prepending *REL-FOCUS-*$<QC>$ to them. Here, the $< QC >$ placeholder is substituted with the actual question class. Fig. 1 illustrates a shallow chunk-based syntactic structure enriched with relational tags.

3.2 Convolutional Neural Networks

A number of NN-based models have been proposed in the research line of answer selection (Hu et al., 2014; Yu et al., 2014). Here, we employ

task3/

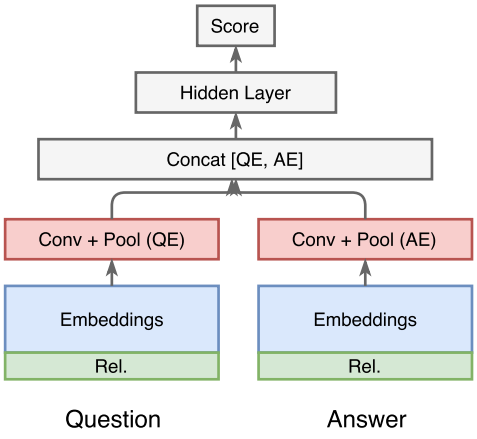

Figure 2: CNN architecture to compute the similarity between question and answer.

the NN model described in (Tymoshenko et al., 2016a) and depicted in Fig. 2. It includes two main components (i) two sentence encoders that map input documents i into fixed size m-dimensional vectors x_{s_i}, and (ii) a feed forward NN that computes the similarity between the two sentences in the input.

We use a sentence model built with a convolution operation followed by a k-max pooling layer with $k = 1$. The sentence vectors, x_{s_i}, are concatenated together and given in input to standard NN layers, which are constituted by a non-linear hidden layer and a sigmoid output layer. The sentence encoder, $x_{s_i} = f(s_i)$ outputs a fixed-size vector representation of the input sentence s_i (we will refer to $f(s_i)$ as question embedding, QE, and answer embedding, AE, respectively).

Additionally, we encode the relational information between Q and AP, by injecting relation features into the network. In particular, we associate each word w of the input sentences with a *word overlap* binary feature indicating if w is shared by both Q and AP.

4 Hybrid learning to rank model

We represent a Q/AP pair as $p = (q, a, \vec{x})$, where q and a are the structural representations of Q and AP (as described in Sec. 3), and $\vec{x}$ is a feature vector that incorporates the features characterizing the Q/AP pair (e.g., similarity features between Q and AP or their embeddings learned by an NN).

Reranking kernel. This kernel captures differences between two Q/AP pairs, p_1 and p_2, and predicts which pair should be ranked higher, i.e., in which pair, AP has higher probability to provide a correct answer to Q. In the reranking setting, a training/classification instance is a pair of Q/AP pairs, $\langle p_1 = (q, a_1, \vec{x}_1), \quad p_2 = (q, a_2, \vec{x}_2)\rangle$. The instance is positive if p_1 is ranked higher than p_2, and negative otherwise. One approach for producing training data is to form pairs both using $\langle p_1, p_2\rangle$ and $\langle p_2, p_1\rangle$, thus generating both positive and negative examples.

However, since these are clearly redundant as formed by the same members, it is more efficient training with a reduced set of examples such that members are not swapped. Algorithm 1 describes how we generate a more compact set of positive (E_+) and negative (E_-) training examples for a specific Q.

Given a pair of examples, $\langle p_1, p_2\rangle$ and $\langle p'_1, p'_2\rangle$, we used the following preference kernel (Shen and Joshi, 2003):

$$R_K(\langle p_1, p_2\rangle, \langle p'_1, p'_2\rangle) = K(p_1, p'_1) + \\ +K(p_2, p'_2) - K(p_1, p'_2) - K(p_2, p'_1), \quad (1)$$

which is equivalent to the dot product between vector subtractions, i.e., $\big(\phi(p_1) - \phi(p_2)\big) \cdot \big(\phi(p'_1) - \phi(p'_2)\big)$, used in preference reranking, where ϕ is a feature map. Additionally, we indicate (i) with R_{TK} the preference kernel using TKs applied to q and a trees, i.e., $TK(p_i, p_j) = TK(q_i, q_j) + TK(a_i, a_j)$; and (ii) with R_V, the preference kernel applied to vectors, i.e., $V(p_i, p_j) = V(\vec{x}_i, \vec{x}_j)$. Our final reranking kernel is:

$$RK(\langle p_1, p_2\rangle, \langle p'_1, p'_2\rangle) = R_{TK}(\langle p_1, p_2\rangle, \langle p'_1, p'_2\rangle) + \\ +R_V(\langle p_1, p_2\rangle, \langle p'_1, p'_2\rangle) \quad (2)$$

Now, if we substitute the explicit form for R_V, we have:

$$RK(\langle p_1, p_2\rangle, \langle p'_1, p'_2\rangle) = R_{TK}(\langle p_1, p_2\rangle, \langle p'_1, p'_2\rangle) + \\ +V(p_1, p'_1) + V(p_2, p'_2) - V(p_1, p'_2) - V(p_2, p'_1)$$

Since our vectors are internal network layers in order to not lose important information with differences (operated by R_V), we only keep $V(p_1, p'_1)$ (or equivalently $V(p_2, p'_2)$), i.e.,

$$RK(\langle p_1, p_2\rangle, \langle p'_1, p'_2\rangle) = R_{TK}(\langle p_1, p_2\rangle, \langle p'_1, p'_2\rangle) + \\ V(p_1, p'_1) \quad (3)$$

Note that our approach also works when using Alg. 1.

Algorithm 1 Generating training data for reranking

Require: S_{q+}, S_{q-} - $(q, a, \vec{x})$-triplets for correct and wrong answer sentences per question Q

$E_+ \leftarrow \emptyset$, $E_- \leftarrow \emptyset$, $flip \leftarrow$ **true**
for all $s_+ \in S_{q+}$ **do**
  for all $s_- \in S_{q-}$ **do**
    if $flip ==$ **true then**
      $E_+ \leftarrow E_+ \cup (s_+, s_-)$
      $flip \leftarrow$ **false**
    else
      $E_- \leftarrow E_- \cup (s_-, s_+)$
      $flip \leftarrow$ **true**
return E_+, E_-

5 Experiments

In our experiments, we compare various methods of combining CTKs and CNNs, using standard and our hybrid reranking kernels. The software for reproducing our experimental results is available at https://github.com/iKernels/RelTextRank.

5.1 Experimental setup

WikiQA, sentence selection dataset: this was created for open domain QA. Table 1 provides the statistics regarding this dataset. Following Yang et al. (2015), we discard questions that have either only correct or only incorrect answers.

cQA, SemEval-2016 dataset: we used the English data from Task 3, Subtask A[2]. We can exactly compare with the state of the art in SemEval. It contains questions collected from the *Qatar Living forum*[3] and the first ten comments per question manually annotated. The train, dev. and test sets contain 1790, 244 and 327 questions, respectively.

Text Preprocessing: we used the Illinois chunker (Punyakanok and Roth, 2001) and the Stanford CoreNLP (Manning et al., 2014) toolkit, v3.6.0. When experimenting with SemEval-2016, we perform preprocessing as in (Tymoshenko et al., 2016a), e.g., we truncate all the comments to 2000 symbols and sentences to 70 words.

CTKs: we trained our models with SVM-Light-TK[4] using the partial tree kernel (PTK) and the subset tree kernel (STK). We use PTK for WikiQA and STK for SemEval as suggested in our previous work (Tymoshenko et al., 2016a) with default parameters and the polynomial kernel (P) of degree 3 on all feature vectors, which are embeddings learned as described in Section 3.2.

Dataset	Q	AP	Q with AP
WikiQA-train	2,118	20,360	873
WikiQA-test	633	6,165	243
WikiQA-dev	296	2,733	126

Table 1: WikiQA statistics.

Neural Network (CNN) setup: we used the same setup and parameters as (Tymoshenko et al., 2016a): we pre-initialize the word embeddings with skipgram embedding of dimensionality 50 trained on the English Wikipedia dump (Mikolov et al., 2013). We used a single non-linear hidden layer (with hyperbolic tangent activation, Tanh), whose size is equal to the size of the previous layer, i.e., the join layer. The network is trained using SGD with shuffled mini-batches using the Rmsprop update rule (Tieleman and Hinton, 2012). The model is trained until the validation loss stops improving. The size of the sentences embedding (QE and AE) and of the join layer is set as 200.

QA metrics: we report our results in terms of Mean Average Precision (MAP), Mean Reciprocal Rank (MRR) and P@1.

5.2 Ranking with trees and embeddings

We evaluate the combination techniques proposed in Sec. 4 on the SemEval-2016 and WikiQA development (DEV) and test (TEST) sets. Additionally, to have more reliable results, it is standard practice to apply n-fold cross-validation. However, we cannot do this on the training (TRAIN) sets, since the embeddings learned in Sec. 3.2 are trained on TRAIN by construction, and therefore cross-validation on TRAIN would exhibit unrealistically high performance. Thus, we employed the following disjoint *Cross Validation* approach: we train 5 models as in traditional 5-fold cross-validation on TRAIN. Then, we merged WikiQA DEV and TEST sets, split the resulting set into 5 subsets, and use *i*-th subset to test the model trained in *i*-th fold (*i*=1,..,5).

Table 2 reports the performance of the models. Here, Rank corresponds to the traditional reranking model described by Eq. 2 in Sec. 4. Hybrid refers to our new reranking/classification kernels described by Eq. 3. V means that the model uses a kernel applied to the embedding feature vectors only. T specifies that the model employs struc-

[2] http://alt.qcri.org/semeval2016/task3/index.php?id=description-of-tasks
[3] http://www.qatarliving.com/forum
[4] http://disi.unitn.it/moschitti/Tree-Kernel.htm

	WikiQA									SemEval-2016, Task 3.A			
	DEV			TEST			Cross Validation			DEV		TEST	
	MRR	MAP	P@1	MRR	MAP	P@1	MRR	MAP	P@1	MAP	MRR	MAP	MRR
Rank:T	72.33	71.96	59.02	71.25	69.71	56.54	71.57±2.09	70.56±1.98	57.56±3.43	65.19	73.16	75.14	82.90
Class:V	70.88	70.55	58.20	66.98	65.18	53.59	68.54±1.78	67.45±1.64	55.95±1.54	65.63	72.69	75.16	82.37
Rank:V	69.39	68.73	54.92	67.56	66.42	54.43	66.64±3.21	65.66±2.19	51.83±2.66	65.29	72.65	74.49	81.77
Rank:T+V	71.54	71.05	59.02	71.56	69.99	57.38	70.23±2.63	69.33±2.15	56.79±3.19	66.22	73.74	74.79	81.69
Hybrid:T+V	**75.05**	**74.02**	**63.93**	**74.08**	**72.19**	**61.60**	**74.36±2.67**	**72.69±1.73**	**62.16±3.31**	**68.08**	**75.09**	**77.10**	**83.45**
CNN	72.04	71.73	59.84	70.34	68.73	56.12	—	—	—	66.48	73.46	76.17	81.32
Rank':T+V	71.29	70.79	57.94	72.51	71.29	59.26	—	—	—	—	—	—	—
ABCNN	—	—	—	71.27	69.14	—	—	—	—	—	—	—	—
KeLP (#1)	—	—	—	—	—	—	—	—	—	—	—	79.19	86.42
ConvKN (#2)	—	—	—	—	—	—	—	—	—	—	—	77.66	84.93

Table 2: Experimental results on WikiQA and SemEval-2016 Task 3.A corpora

tural representations with a tree kernel. Finally, $V+T$ means that both embedding feature vectors and trees are used.

The experiments show that: in general, a standalone model with CTKs applied to the syntactic structures (*Rank:T*) outperforms the standalone feature-based models using embeddings as feature vectors (V).

Then, the straightforward combination of tree and polynomial kernels applied to the syntactic structural representations and embeddings (*Rank: T+V*) does not improve over the *Rank: T* model.

At the same time, the Hybrid model consistently outperforms all the other models in all possible experimental configurations, thus confirming our hypothesis that the classification setting is more appropriate when using embeddings as feature vectors in the kernel-based ranking models.

Additionally, for reference, we report the performance of the CNN we used to obtain the embeddings. It is consistently outperformed by the Hybrid model on all the datasets.

Finally, in the last four lines of Tab. 2, we report the performance of the state-of-the-art models from previous work, measured on exactly the same experimental settings we used.

Here *Rank':T+V* is our model described in (Tymoshenko et al., 2016a), based on the traditional reranking model. Our updated version obtains comparable performance on WikiQA-DEV and slightly lower performance on WikiQA-TEST (probably, just due to differences in preprocessing after we updated our pre-processing pipelines).

ABCNN (Yin et al., 2016) is another state-of-the-art system based on advanced attention-based convolutional networks. All our models involving CTKs outperform it.

KeLP (#1) (Filice et al., 2016) and ConvKN (#2) (Barrón-Cedeño et al., 2016) are the two top-performing SemEval 2016, Task 3.A competition systems (Nakov et al., 2016). ConvKN (#2) is an earlier version of our approach, which also employs CTKs and embeddings. Both KeLP and ConvKN (i) employ cQA-domain-specific handcrafted features, which also consider the thread-level information, while in this work, we do not use manually engineered features; (ii) they employ PTK, which is capable of learning more powerful features than SST, but it is more computationally complex; (iii) KeLP system parameters were optimized in cross-validation on the training set, while, in this work, we perform no parameter optimization. Nevertheless, the performance of our *Hybrid:T+V* models on SemEval TEST is comparable to that of ConvKN (#2).

6 Conclusion

In this paper, we have studied and compared state-of-the-art feature engineering approaches, namely CTKs and CNNs, on two different QA tasks, AST and cQA. We investigated the ways of combining the two approaches into a single model and proposed a hybrid reranking-classification kernel for combining the structural representations and embeddings learned by CNNs.

We have shown that the combination of CTKs and CNNs with a hybrid kernel in the reranking setting outperforms the state of the art on AST and is comparable to the state of the art in cQA. In particular, in cQA, a combination of CTKs and CNNs performs comparably to the systems using domain specific features that were manually engineered.

Acknowledgments

This work has been partially supported by the EC project CogNet, 671625 (H2020-ICT-2014-2, Research and Innovation action). We would like to thank Raniero Romagnoli for helping us to review an early draft of this paper. Many thanks to the anonymous reviewers for their professional work and valuable suggestions.

References

Alberto Barrón-Cedeño, Giovanni Da San Martino, Shafiq Joty, Alessandro Moschitti, Fahad Al-Obaidli, Salvatore Romeo, Kateryna Tymoshenko, and Antonio Uva. 2016. ConvKN at SemEval-2016 Task 3: Answer and Question Selection for Question Answering on Arabic and English Fora. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*, pages 896–903, San Diego, California. Association for Computational Linguistics.

S. Filice, D. Croce, A. Moschitti, and R. Basili. 2016. Kelp at semeval-2016 task 3: Learning semantic relations between questions and answers. *Proceedings of SemEval*, 16:1116–1123.

Baotian Hu, Zhengdong Lu, Hang Li, and Qingcai Chen. 2014. Convolutional neural network architectures for matching natural language sentences. In *NIPS*.

T. Joachims. 2002. Optimizing search engines using clickthrough data. In *Proceedings of ACM KDD*, pages 133–142. ACM.

Xin Li and Dan Roth. 2002. Learning question classifiers. In *Proceedings of COLING*, pages 1–7. Association for Computational Linguistics.

Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J. Bethard, and David McClosky. 2014. The Stanford CoreNLP natural language processing toolkit. In *Proceedings of ACL: System Demonstrations*, pages 55–60.

Yishu Miao, Lei Yu, and Phil Blunsom. 2016. Neural variational inference for text processing. In *Proc. ICML*.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in NIPS 26*, pages 3111–3119.

P. Nakov, L. Márquez, A. Moschitti, W. Magdy, H. Mubarak, A. A. Freihat, J. Glass, and B. Randeree. 2016. SemEval-2016 task 3: Community question answering. In *Proceedings of SemEval '16*. ACL.

M. Nicosia, S. Filice, A. Barrón-Cedeño, I. Saleh, H. Mubarak, W. Gao, P. Nakov, G. Da San Martino, A. Moschitti, K. Darwish, L. Màrquez, S. Joty, and W. Magdy. 2015. QCRI: Answer selection for community question answering - experiments for Arabic and English. In *SemEval*.

V. Punyakanok and D. Roth. 2001. The use of classifiers in sequential inference. In *NIPS*, pages 995–1001.

A. Severyn, M. Nicosia, and A. Moschitti. 2013. Learning adaptable patterns for passage reranking. *CoNLL-2013*, page 75.

Aliaksei Severyn and Alessandro Moschitti. 2015. Learning to rank short text pairs with convolutional deep neural networks. In *SIGIR*. ACM.

L. Shen and A. K. Joshi. 2003. An SVM based voting algorithm with application to parse reranking. In *CONLL*, CONLL '03, pages 9–16, Edmonton, Canada.

Tijmen Tieleman and Geoffrey Hinton. 2012. Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. *COURSERA: Neural Networks for Machine Learning*, 4.

Kateryna Tymoshenko, Daniele Bonadiman, and Alessandro Moschitti. 2016a. Convolutional neural networks vs. convolution kernels: Feature engineering for answer sentence reranking. In *Proceedings of NAACL-HLT*, pages 1268–1278.

Kateryna Tymoshenko, Daniele Bonadiman, and Alessandro Moschitti. 2016b. Learning to rank non-factoid answers: Comment selection in web forums. In *CIKM*, pages 2049–2052. ACM.

Yi Yang, Wen-tau Yih, and Christopher Meek. 2015. Wikiqa: A challenge dataset for open-domain question answering. In *EMNLP*. ACL.

Wenpeng Yin, Hinrich Schütze, Bing Xiang, and Bowen Zhou. 2016. Abcnn: Attention-based convolutional neural network for modeling sentence pairs. *Transactions of the Association for Computational Linguistics*, 4:259–272.

Lei Yu, Karl Moritz Hermann, Phil Blunsom, and Stephen Pulman. 2014. Deep Learning for Answer Sentence Selection. In *NIPS Deep Learning Workshop*.