Learning Hybrid Representations to Retrieve Semantically Equivalent Questions

Cícero dos Santos[1], **Luciano Barbosa**[1], **Dasha Bogdanova**[2], **Bianca Zadrozny**[1]
[1]IBM Research, 138/146 Av. Pasteur, Rio de Janeiro, Brazil
{cicerons,lucianoa,biancaz}@br.ibm.com
[2]ADAPT centre, School of Computing, Dublin City University, Dublin, Ireland
dbogdanova@computing.dcu.ie

Abstract

Retrieving similar questions in online Q&A community sites is a difficult task because different users may formulate the same question in a variety of ways, using different vocabulary and structure. In this work, we propose a new neural network architecture to perform the task of semantically equivalent question retrieval. The proposed architecture, which we call BOW-CNN, combines a bag-of-words (BOW) representation with a distributed vector representation created by a convolutional neural network (CNN). We perform experiments using data collected from two Stack Exchange communities. Our experimental results evidence that: (1) BOW-CNN is more effective than BOW based information retrieval methods such as TFIDF; (2) BOW-CNN is more robust than the pure CNN for long texts.

1 Introduction

Most Question-answering (Q&A) community sites advise users before posting a new question to search for similar questions. This is not always an easy task because different users may formulate the same question in a variety of ways.

We define two questions as semantically equivalent if they can be adequately answered by the exact same answer. Here is an example of a pair of such questions from Ask Ubuntu community, which is part of the Stack Exchange Q&A community site: (q_1)"*I have downloaded ISO files recently. How do I burn it to a CD or DVD or mount it?*" and (q_2)"*I need to copy the iso file for Ubuntu 12.04 to a CD-R in Win8. How do I do so?*". Retrieving semantically equivalent questions is a challenging task due to two main factors: (1) the same question can be rephrased in many different ways; and (2) two questions may be different but may refer implicitly to a common problem with the same answer. Therefore, traditional similarity measures based on word overlap such as shingling and Jaccard coefficient (Broder, 1997) and its variations (Wu et al., 2011) are not able to capture many cases of semantic equivalence. To capture the semantic relationship between pair of questions, different strategies have been used such as machine translation (Jeon et al., 2005; Xue et al., 2008), knowledge graphs (Zhou et al., 2013) and topic modelling (Cai et al., 2011; Ji et al., 2012).

Recent papers (Kim, 2014; Hu et al., 2014; Yih et al., 2014; dos Santos and Gatti, 2014; Shen et al., 2014) have shown the effectiveness of convolutional neural networks (CNN) for sentence-level analysis of *short texts* in a variety of different natural language processing and information retrieval tasks. This motivated us to investigate CNNs for the task of semantically equivalent question retrieval. However, given the fact that the size of a question in an online community may vary from a single sentence to a detailed problem description with several sentences, it was not clear that the CNN representation would be the most adequate.

In this paper, we propose a hybrid neural network architecture, which we call BOW-CNN. It combines a traditional bag-of-words (BOW) representation with a distributed vector representation created by a CNN, to retrieve semantically equivalent questions. Using a ranking loss function in the training, BOW-CNN learns to represent questions while learning to rank them according to their semantic similarity. We evaluate BOW-CNN over two different Q&A communities in the Stack Exchange site, comparing it against CNN and 6 well-established information retrieval algorithms based on BOW. The results show that our proposed solution outperforms BOW-based information retrieval methods such as the *term frequency - inverse document frequency* (TFIDF) in all evalu-

Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Short Papers), pages 694–699, Beijing, China, July 26-31, 2015.

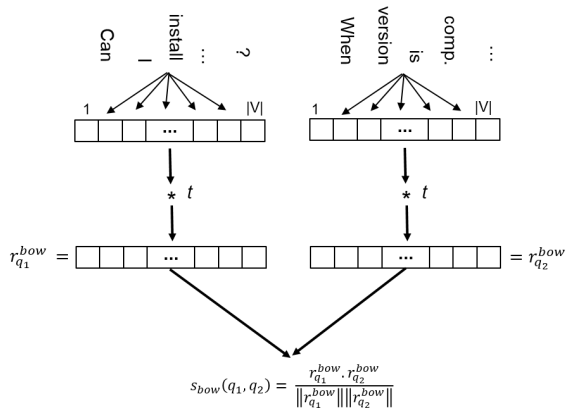

Figure 1: Representing and scoring questions with weighted bag-of-words.

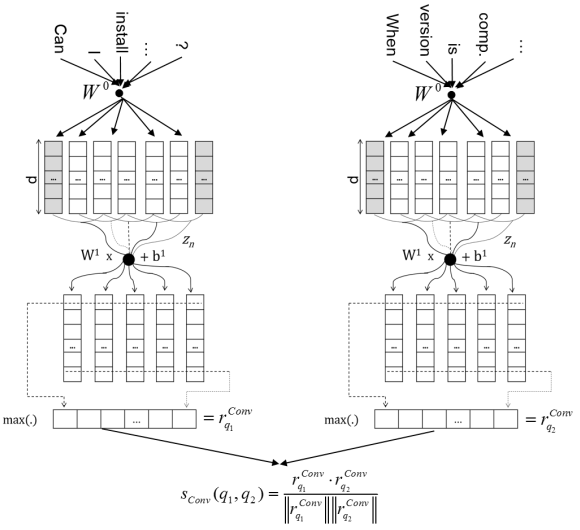

Figure 2: Representing and scoring questions with a convolutional approach.

ated scenarios. Moreover, we were able to show that for short texts (title of the questions), an approach using only CNN obtains the best results, whereas for long texts (title and body of the questions), our hybrid approach (BOW-CNN) is more effective.

2 BOW-CNN

2.1 Feed Forward Processing

The goal of the feed forward processing is to calculate the similarity between a pair of questions (q_1, q_2). To perform this task, each question q follows two parallel paths (BOW and CNN), each one producing a distinct vector representations of q. The BOW path produces a weighted bag-of-words representation of the question, r_q^{bow}, where the weight of each word in the vocabulary V is learned by the neural network. The CNN path, uses a convolutional approach to construct a distributed vector representations, r_q^{conv}, of the question. After producing the BOW and CNN representations for the two input questions, the BOW-CNN computes two partial similarity scores $s_{bow}(q_1, q_2)$, for the CNN representations, and $s_{conv}(q_1, q_2)$, for the BOW representations. Finally, it combines the two partial scores to create the final score $s(q_1, q_2)$.

2.2 BOW Path

The generation of the bag-of-words representation for a given question q is quite straightforward. As detailed in Figure 1, we first create a sparse vector $q^{bow} \in \mathbb{R}^{|V|}$ that contains the frequency in q of each word of the vocabulary. Next, we compute the weighted bag-of-words representation by performing the element-wise vector multiplication:

$$r_q^{bow} = q^{bow} * t \quad (1)$$

where the vector $t \in \mathbb{R}^{|V|}$, contains a weight for each word in the vocabulary V. The vector t is a parameter to be learned by the network. This is closely related to the TFIDF text representation. In fact, if we fix t to the vector of IDFs, this corresponds to the exact TFIDF representation.

2.3 CNN Path

As detailed in Figure 2, the first layer of the CNN path transforms words into representations that capture syntactic and semantic information about the words. Given a question consisting of N words $q = \{w_1, ..., w_N\}$, every word w_n is converted into a real-valued vector r^{w_n}. Therefore, for each question, the input to the next NN layer is a sequence of real-valued vectors $q^{emb} = \{r^{w_1}, ..., r^{w_N}\}$. Word representations are encoded by column vectors in an embedding matrix $W^0 \in \mathbb{R}^{d \times |V|}$, where V is a fixed-sized vocabulary.

The next step in the CNN path consists in creating distributed vector representations $r_{q_1}^{conv}$ and $r_{q_2}^{conv}$ from the word embedding sequencies q_1^{emb} and q_2^{emb}. We perform this by using a convolutional layer in the same way as used in (dos Santos and Gatti, 2014) to create sentence-level representations.

More specifically, given a question q_1, the convolutional layer applies a matrix-vector operation to each window of size k of successive windows

in $q_1^{emb} = \{r^{w_1}, ..., r^{w_N}\}$. Let us define the vector $z_n \in \mathbb{R}^{dk}$ as the concatenation of a sequence of k word embeddings, centralized in the n-th word:

$$z_n = \left(r^{w_{n-(k-1)/2}}, ..., r^{w_{n+(k-1)/2}}\right)^T$$

The convolutional layer computes the j-th element of the vector $r_{q_1}^{conv} \in \mathbb{R}^{cl_u}$ as follows:

$$[r_{q_1}^{conv}]_j = f\left(\max_{1<n<N}\left[W^1 z_n + b^1\right]_j\right) \quad (2)$$

where $W^1 \in \mathbb{R}^{cl_u \times dk}$ is the weight matrix of the convolutional layer and f is the hyperbolic tangent function. Matrices W^0 and W^1, and the vector b^1 are parameters to be learned. The word embedding size d, the number of convolutional units cl_u, and the size of the word context window k are hyper-parameters to be chosen by the user.

2.4 Question Pair Scoring

After the bag-of-words and convolutional-based representations are generated for the input pair (q_1, q_2), the partial scores are computed as the cosine similarity between the respective vectors:

$$s_{bow}(q_1, q_2) = \frac{r_{q_1}^{bow} . r_{q_2}^{bow}}{\|r_{q_1}^{bow}\| \|r_{q_2}^{bow}\|}$$

$$s_{conv}(q_1, q_2) = \frac{r_{q_1}^{conv} . r_{q_2}^{conv}}{\|r_{q_1}^{conv}\| \|r_{q_2}^{conv}\|}$$

The final score for the input questions (q_1, q_2) is given by the following linear combination

$$s(q_1, q_2) = \beta_1 * s_{bow}(q_1, q_2) + \beta_2 * s_{conv}(q_1, q_2)$$

where β_1 and β_2 are parameters to be learned.

2.5 Training Procedure

Our network is trained by minimizing a ranking loss function over the training set D. The input in each round is two pairs of questions $(q_1, q_2)^+$ and $(q_1, q_x)^-$ where the questions in the first pair are semantically equivalent (positive example), and the ones in the second pair are not (negative example). Let Δ be the difference of their similarity scores, $\Delta = s_\theta(q_1, q_2) - s_\theta(q_1, q_x)$, generated by the network with parameter set θ. As in (Yih et al., 2011), we use a logistic loss over Δ

$$L(\Delta, \theta) = log(1 + exp(-\gamma\Delta))$$

where γ is a scaling factor that magnifies Δ from [-2,2] (in the case of using cosine similarity) to a larger range. This helps to penalize more on the prediction errors. Following (Yih et al., 2011), in our experiments we set γ to 10.

Sampling informative negative examples can have a significant impact in the effectiveness of the learned model. In our experiments, before training, we create 20 pairs of negative examples for each positive pair $(q_1,q_2)^+$. To create a negative example we (1) randomly sample a question q_x that is not semantically equivalent to q_1 or q_2; (2) then create negative pairs $(q_1,q_x)^-$ and $(q_2,q_x)^-$. During training, at each iteration we only use the negative example x that produces the smallest different $s_\theta(q_1, q_2)^+ - s_\theta(q_1, q_x)^-$. Using this strategy, we select more representative negative examples.

We use stochastic gradient descent (SGD) to minimize the loss function with respect to θ. The backpropagation algorithm is used to compute the gradients of the network. In our experiments, BOW-CNN architecture is implemented using Theano (Bergstra et al., 2010).

3 Experimental Setup

3.1 Data

A well-structured source of semantically equivalent questions is the Stack Exchange site. It is composed by multiple Q&A communities, whereby users can ask and answer questions, and vote up and down both questions and answers. Questions are composed by a title and a body. Moderators can mark questions as duplicates, and eventually a question can have multiple duplicates.

For this evaluation, we chose two highly-accessed Q&A communities: Ask Ubuntu and English. They differ in terms of content and size. Whereas Ask Ubuntu has 29510 duplicated questions, English has 6621. We performed experiments using only the title of the questions as well as title + body, which we call *all* for the rest of this section. The average size of a title is very small (about 10 words), which is at least 10 times smaller than the average size of *all* for both datasets. The data was tokenized using the tokenizer available with the Stanford POS Tagger (Toutanova et al., 2003), and all links were replaced by a unique string. For Ask Ubuntu, we did not consider the content inside the tag code, which contains some specific Linux commands or programming code.

For each community, we created training, vali-

Community	Training	Validation	Test
Ask Ubuntu	9802	1991	3800
English	2235	428	816

Table 1: Partition of training, validation and test sets for the experiments.

dation and test sets. In Table 1, we inform the size of each set. The number of instances in the training set corresponds to the number of positive pairs of semantically equivalent questions. The number of instances in the validation and the test sets correspond to the number of questions which are used as queries. All questions in the validation and test set contain at least one duplicated question in the set of all questions. In our experiments, given a query question q, all questions in the Q&A community are evaluated when searching for a duplicate of q.

3.2 Baselines and Neural Network Setup

In order to verify the impact of jointly using BOW and CNN representations, we perform experiments with two NN architectures: the BOW-CNN and the CNN alone, which consists in using only the CNN path of BOW-CNN and, consequently, computing the score for a pair of questions using $s(q_1, q_2) = s_{conv}(q_1, q_2)$.

Additionally, we compare BOW-CNN with six well-established IR algorithms available on the Lucene package (Hatcher et al., 2004). Here we provide a brief overview of them. For further details, we refer the reader to the citation associated with the algorithm.

- **TFIDF** (Manning et al., 2008) uses the traditional Vector Space Model to represent documents as vectors in a high-dimensional space. Each position in the vector represents a word and the weight of words are calculated using TFIDF.

- **BM25** (Robertson and Walker, 1994) is a probabilistic weighting method that takes into consideration term frequency, inverse document frequency and document length. Its has two free parameters: k1 to tune term-frequency saturation; and b to calibrate the document-length normalization.

- **IB** (Clinchant and Gaussier, 2010) uses information-based models to capture the importance of a term by measuring how much its behavior in a document deviates from its behavior in the whole collection.

- **DFR** (Amati and Van Rijsbergen, 2002) is based on divergence from randomness framework. The relevance of a term is measured by the divergence between its actual distribution and the distribution from a random process.

- **LMDirichlet** and **LMJelinekMercer** apply probabilistic language model approaches for retrieval (Zhai and Lafferty, 2004). They differ in the smoothing method: LMDirichlet uses Dirichlet priors and LMJelinekMercer uses the Jelinek-Mercer method.

Param. Name	BOW-CNN	CNN
Word Emb. Size	200	200
Context Winow Size	3	3
Convol. Units	400	1000
Learning Rate	0.01	0.05

Table 2: Neural Network Hyper-Parameters

The word embeddings used in our experiments are initialized by means of unsupervised pre-training. We perform pre-training using the skip-gram NN architecture (Mikolov et al., 2013) available in the `word2vec` tool. We use the English Wikipedia to train word embeddings for experiments with the English dataset. For the AskUbuntu dataset, we use all available Ask-Ubuntu community data to train word embeddings.

The hyper-parameters of the neural networks and the baselines are tuned using the development sets. In Table 2, we show the selected hyper-parameter values. In our experiments, we initialize each element $[t]_i$ of the bag-of-word weight vector t with the IDF of $i-$th word w_i computed over the respective set of questions Q as follows

$$[t]_i = IDF(w_i, Q) = \log \frac{|Q|}{|q \in Q : w_i \in q|}$$

4 Experimental Results

Comparison with Baselines. In Tables 3 and 4, we present the question retrieval performance (Accuracy@k) of different algorithms over the AskUbuntu and English datasets for the *title* and *all* settings, respectively. For both datasets, BOW-CNN outperforms the six IR algorithms for both *title* and *all* settings. For the AskUbuntu *all*, BOW-CNN is four absolute points larger than the

Algorithm	AskUbuntu			English		
	@1	@5	@10	@1	@5	@10
TFIDF	8.3	17.5	22.5	10.0	18.1	21.6
BM25	7.3	17.1	21.8	10.0	18.9	23.2
IB	8.1	18.1	22.6	10.1	18.4	22.7
DFR	7.7	17.8	22.4	10.5	19.0	23.0
LMD	5.6	14.1	19.0	10.9	20.1	24.2
LMJ	8.3	17.5	22.5	10.3	18.5	22.1
CNN	**11.5**	**24.8**	**31.4**	**11.6**	**23.0**	**26.9**
BOW-CNN	10.9	22.6	28.7	11.3	21.4	26.0

Table 3: Question *title* retrieval performance (Accuracy@k) for different algorithms.

Algorithm	AskUbuntu			English		
	@1	@5	@10	@1	@5	@10
TFIDF	16.9	31.3	38.3	25.9	42.0	48.1
BM25	18.2	33.1	39.8	29.4	45.7	52.5
IB	14.9	28.2	34.8	25.4	42.3	48.0
DFR	18.0	32.6	39.2	28.6	45.4	52.5
LMD	13.7	26.8	34.4	23.0	40.2	46.0
LMJ	18.3	33.4	40.7	28.5	45.7	52.3
CNN	20.0	33.8	40.1	17.2	29.6	33.8
BOW-CNN	**22.3**	**39.7**	**46.4**	**30.8**	**47.7**	**54.9**

Table 4: Question *title* + *body* (*all*) retrieval performance for different algorithms.

best IR baseline (LMJ) in terms of Accuracy@1, which represents an improvement of 21.9%. Since the BOW representation we use is closely related to TFIDF, an important comparison is the performance of BOW-CNN vs. TFIDF. In Tables 3 and 4, we can see that BOW-CNN consistently outperforms the TFIDF model in the two datasets for both cases *title* and *all*. These findings suggest that BOW-CNN is indeed combining the strong semantic representation power conveyed by the convolutional-based representation to, jointly with the BOW representation, construct a more effective model.

Another interesting finding is that CNN outperforms BOW-CNN for short texts (Table 3) and, conversely, BOW-CNN outperforms CNN for long texts (Table 4). This demonstrates that, when dealing with large input texts, BOW-CNN is an effective approach to combine the strengths of convolutional-based representation and BOW.

Impact of Initialization of BOW Weights. In the BOW-CNN experiments whose results are presented in tables 3 and 4 we initialize the elements of the BOW weight vector t with the IDF of each word in V computed over the question set Q. In this section we show some experimental results that indicate the contribution of this initialization.

In Table 5, we present the performance of BOW-CNN for the English dataset when different configurations of the BOW weight vector t are used. The first column of Table 5 indicates the type of initialization, where *ones* means that t is initialized with the value 1 (one) in all positions. The second column informs whether t is allowed to be updated (*Yes*) by the network or not (*No*). The numbers suggest that letting BOW weights free to be updated by the network produces better results than fixing them to IDF values. In addition, using IDF to initialize the BOW weight vector is better than using the same weight (ones) to initialize it. This is expected, since we are injecting a prior knowledge known to be helpful in IR tasks.

t initial	t updated	Title		All	
		@1	@10	@1	@10
IDF	Yes	**11.3**	**26.0**	**30.8**	**54.9**
IDF	No	10.6	25.3	29.7	**54.9**
Ones	Yes	10.7	24.2	26.3	51.2

Table 5: BOW-CNN performance using different methods to initialize the BOW weight vector t.

5 Conclusions

In this paper, we propose a hybrid neural network architecture, BOW-CNN, that combines bag-of-words with distributed vector representations created by a CNN, to retrieve semantically equivalent questions. Our experimental evaluation showed that: our approach outperforms traditional bow approaches; for short texts, a pure CNN obtains the best results, whereas for long texts, BOW-CNN is more effective; and initializing the BOW weight vector with IDF values is beneficial.

Acknowledgments

Dasha Bogdanova's contributions were made during an internship at IBM Research. Her work was partially supported by Science Foundation Ireland through the CNGL Programme (Grant 12/CE/I2267) in the ADAPT Centre (www.adaptcentre.ie) at DCU.

References

Gianni Amati and Cornelis Joost Van Rijsbergen. 2002. Probabilistic models of information retrieval based on measuring the divergence from randomness. *ACM Transactions on Information Systems (TOIS)*, 20(4):357–389.

James Bergstra, Olivier Breuleux, Frédéric Bastien, Pascal Lamblin, Razvan Pascanu, Guillaume Des-

jardins, Joseph Turian, David Warde-Farley, and Yoshua Bengio. 2010. Theano: a CPU and GPU math expression compiler. In *Proceedings of the Python for Scientific Computing Conference*.

A. Broder. 1997. On the resemblance and containment of documents. In *Proceedings of the Compression and Complexity of Sequences 1997*, SEQUENCES '97, pages 21–, Washington, DC, USA. IEEE Computer Society.

Li Cai, Guangyou Zhou, Kang Liu, and Jun Zhao. 2011. Learning the latent topics for question retrieval in community qa. In *IJCNLP*, volume 11, pages 273–281.

Stéphane Clinchant and Eric Gaussier. 2010. Information-based models for ad hoc ir. In *Proceedings of the 33rd international ACM SIGIR conference on Research and development in information retrieval*, pages 234–241. ACM.

Cícero Nogueira dos Santos and Maíra Gatti. 2014. Deep convolutional neural networks for sentiment analysis of short texts. In *Proceedings of the 25th International Conference on Computational Linguistics (COLING)*, Dublin, Ireland.

Erik Hatcher, Otis Gospodnetic, and Michael McCandless. 2004. Lucene in action.

Baotian Hu, Zhengdong Lu, Hang Li, and Qingcai Chen. 2014. Convolutional neural network architectures for matching natural language sentences. In *Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems 2014, December 8-13 2014, Montreal, Quebec, Canada*, pages 2042–2050.

Jiwoon Jeon, W Bruce Croft, and Joon Ho Lee. 2005. Finding similar questions in large question and answer archives. In *Proceedings of the 14th ACM international conference on Information and knowledge management*, pages 84–90.

Zongcheng Ji, Fei Xu, Bin Wang, and Ben He. 2012. Question-answer topic model for question retrieval in community question answering. In *Proceedings of the 21st ACM international conference on Information and knowledge management*, pages 2471–2474.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods for Natural Language Processing*, pages 1746–1751.

Christopher D Manning, Prabhakar Raghavan, and Hinrich Schütze. 2008. *Introduction to information retrieval*, volume 1. Cambridge university press Cambridge.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. In *In Proceedings of Workshop at ICLR*.

Stephen E Robertson and Steve Walker. 1994. Some simple effective approximations to the 2-poisson model for probabilistic weighted retrieval. In *Proceedings of the 17th annual international conference on Research and development in information retrieval*, pages 232–241.

Yelong Shen, Xiaodong He, Jianfeng Gao, Li Deng, and Grégoire Mesnil. 2014. A latent semantic model with convolutional-pooling structure for information retrieval. In *Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management*, pages 101–110.

Kristina Toutanova, Dan Klein, Christopher D Manning, and Yoram Singer. 2003. Feature-rich part-of-speech tagging with a cyclic dependency network. In *Proceedings of the North American Chapter of the Association for Computational Linguistics on Human Language Technology*, pages 173–180.

Yan Wu, Qi Zhang, and Xuanjing Huang. 2011. Efficient near-duplicate detection for q&a forum. In *Proceedings of 5th International Joint Conference on Natural Language Processing*, pages 1001–1009, Chiang Mai, Thailand, November. Asian Federation of Natural Language Processing.

Xiaobing Xue, Jiwoon Jeon, and W. Bruce Croft. 2008. Retrieval models for question and answer archives. In *Proceedings of the 31st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '08, pages 475–482.

Wen-tau Yih, Kristina Toutanova, John C. Platt, and Christopher Meek. 2011. Learning discriminative projections for text similarity measures. In *Proceedings of the Fifteenth Conference on Computational Natural Language Learning*, CoNLL'11, pages 247–256.

Wen-tau Yih, Xiaodong He, and Christopher Meek. 2014. Semantic parsing for single-relation question answering. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 643–648. Association for Computational Linguistics.

Chengxiang Zhai and John Lafferty. 2004. A study of smoothing methods for language models applied to information retrieval. *ACM Transactions on Information Systems (TOIS)*, 22(2):179–214.

Guangyou Zhou, Yang Liu, Fang Liu, Daojian Zeng, and Jun Zhao. 2013. Improving question retrieval in community question answering using world knowledge. In *Proceedings of the Twenty-Third international joint conference on Artificial Intelligence*, pages 2239–2245.