IIT-UHH at SemEval-2017 Task 3: Exploring Multiple Features for Community Question Answering and Implicit Dialogue Identification

Titas Nandi[1], **Chris Biemann**[2], **Seid Muhie Yimam**[2],
Deepak Gupta[1], **Sarah Kohail**[2], **Asif Ekbal**[1] and **Pushpak Bhattacharyya**[1]

[1]Indian Institute of Technology Patna
[2]Universität Hamburg Germany
{titas.ee13,deepak.pcs16,asif,pb}@iitp.ac.in
{biemann,yimam,kohail}@informatik.uni-hamburg.de

Abstract

In this paper we present the system for Answer Selection and Ranking in Community Question Answering, which we build as part of our participation in SemEval-2017 Task 3. We develop a Support Vector Machine (SVM) based system that makes use of textual, domain-specific, word-embedding and topic-modeling features. In addition, we propose a novel method for dialogue chain identification in comment threads. Our primary submission *won* subtask C, outperforming other systems in all the primary evaluation metrics. We performed well in other English subtasks, ranking *third* in subtask A and *eighth* in subtask B. We also developed open source toolkits for all the three English subtasks by the name cQARank[1].

1 Introduction

This paper presents the system built for participation in the SemEval-2017 Shared Task 3 on Community Question Answering (CQA). The task aims to classify and rank a candidate text c in relevance to a target text t. Based on the nature of the candidate and target texts, the main task is subdivided into three subtasks in which the teams are expected to solve the problem of Question-Comment similarity, Question-Question similarity and Question-External Comment similarity (Nakov et al., 2017).

In this work, we propose a rich feature-based system for solving these problems. We create an architecture which integrates textual, semantic and domain-specific features to achieve good results in the proposed task. Due to the extremely noisy nature of the social forum data, we also develop a customized preprocessing pipeline, rather than using the standard tools. We use Support Vector Machine (SVM) (Cortes and Vapnik, 1995) for classification, and its confidence score for ranking.

We initially define a generic set of features to develop a robust system for all three subtasks, then include additional features based on the nature of the subtasks. To adapt the system to subtasks B and C, we include features extracted from the scores of the other subtasks, propagating meaningful information essential in an incremental setting. We propose a novel method for identification of dialogue groups in the comment thread by constructing a user interaction graph and also incorporate features from this graph in our system. Our algorithm outputs mutually disjoint groups of users who are involved in conversation with each other in the comment thread.

The rest of the paper is organized as follows: Section 2 describes the related work. Sections 3, 4, and 5 elucidate the system architecture, features used and algorithms developed. Section 6 provides experimentation details and reports the official results.

2 Related Work

In Question Answering, answer selection and ranking has been a major research concern in Natural Language Processing (NLP) during the past few years. The problem becomes more interesting for Community Question Answering due to the highly unstructured and noisy nature of the data. Also, domain knowledge plays a major role in such an environment, where meta data of users and context based learning can capture trends well. The task on Community Question Answering in SemEval began in 2015, where the objective was to classify comments in a thread as *Good*, *Bad* or *PotentiallyUseful*. In subsequent years, the task

[1]https://github.com/TitasNandi/cQARank

Proceedings of the 11th International Workshop on Semantic Evaluations (SemEval-2017), pages 90–97,
Vancouver, Canada, August 3 - 4, 2017.

was extended and modified to focus on ranking and duplicate question detection in a cross domain setting.

In their 2015 system, Belinkov (2015) used word vectors of the question and of the comment, various text-based similarities and meta data features. Nicosia (2015) derived features from a comment in the context of the entire thread. They also modelled potential dialogues by identifying interlacing comments between users. Establishing similarity between Questions and External comments (subtask C) is quite challenging, which can be tackled by propagating useful context and information from other subtasks. Filice (2016) introduced an interesting approach of stacking classifiers across subtasks and Wu & Lan (2016) proposed a method of reducing the errors that propagated as a result of this stacking.

3 System Description

3.1 System Pipeline

The system architecture of our submission to subtask A is depicted in Figure 1. We explain the pre-processing pipeline in the next subsection. The cleaned data is fed into our supervised machine learning framework. We train our word-embedding model on the unannotated and training data[2] provided by the organizers, and train a probabilistic topic model on the training data. The detailed description of features is provided in the following section. After obtaining the feature vectors, we perform feature selection using wrapper methods to maximize the accuracy on the development set. We Z-score normalize the feature vectors and feed them to a SVM. We tune the hyper-parameters of SVM and and generate classification labels and probabilities, the latter being used for computing the MAP score.

3.2 Preprocessing Pipeline

Due to the highly unstructured, spelling and grammatical error-prone nature of the data, adaptation of any standard tokenization pipeline was not well motivated. We customized the preprocessing according to the nature of the data. We unescaped HTML special characters and removed URLs, e-mails, HTML tags, image description tags, punctuations and slang words (from a defined dictionary). Finally, we expanded apostrophe words and removed stopwords.
The cleaned data is then used in all further experiments.

[2] http://alt.qcri.org/semeval2017/task3/index.php?id=data-and-tools

4 Features

We use a rich set of features to capture the textual and semantic relevance between two snippets of text. These features are categorized into several broad classes:

4.1 String Similarity Features

This set of features makes use of a number of string matching algorithms to compute the string similarity between the question and comment. This generates a continuous set of values for every comment, and is apt for a baseline system. The bag of algorithms used is a careful combination of various string similarity, metric distances and normalized string distance methods, capturing the overall profiling of texts. The string similarity functions used include Longest Common Subsequence (LCS), Q-Gram (q = *1,2,3*), Weighted Levenshtein and Optimal String Alignment. The normalized similarity algorithms used are Jaro-Winkler, Normalized Levenshtein, n-gram (n = *1,2,3*), cosine-similarity (n = *1,2,3*), Jaccard Index (n = *1,2,3*), and Sorensen-Dice coefficient (n = *1,2,3*). The metric distance methods implemented are Levenshtein, Damerau, and Metric LCS.

4.2 Word Embedding Features

Semantic features constitute the core of our feature engineering pipeline. These try to capture the proximity between the meanings encoded in the word sequences of question and comments. We train word embeddings using Word2Vec (Mikolov et al., 2013) on the unannotated and given training data. The unannotated data is in the same format as the training data, except that the comments are not annotated. We performed experiments with different vector sizes (N = *100, 200, 300*), and finally settled on using 100 dimensional word vectors. We also used a pre-trained model on *Google News* dataset in order to compare the performance of the two models. Interestingly, the domain-specific model trained on the unannotated and training data proved to be better than the one trained on Google News dataset, hence we used the former in building our final system.
Since we wanted a feature vector corresponding to each comment in the thread, we had to transform

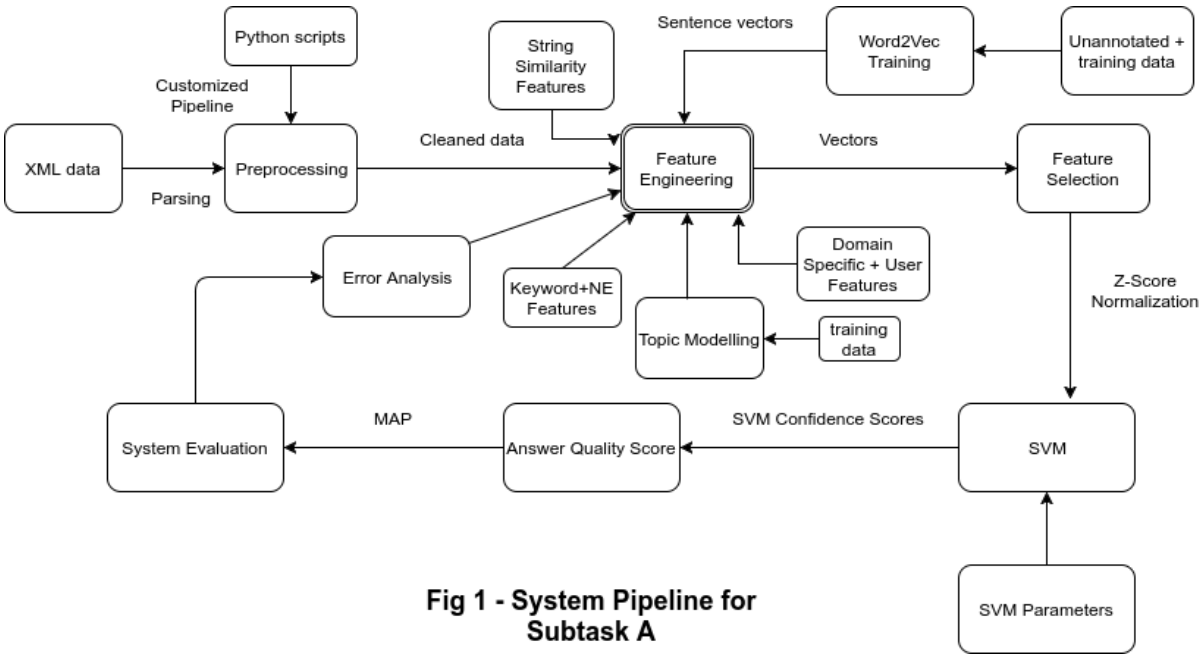

Fig 1 - System Pipeline for Subtask A

these trained word vectors into sentence vectors. Two approaches were considered for this:

- Construct the sentence vector by taking an average of the vectors of all words that constitute the sentence.
- Construct the sentence vector as a weighted average of all the word vectors constituting that sentence. Here the weight corresponds to the *Inverse Document Frequency* (IDF) value of the word in the thread.

Although the first approach has an evident disadvantage of not assigning importance to the keywords in the sentence (which is why we resorted to the IDF-based weighted averaging), it yielded better results, which is why we included it in our final system.
We extract two sets of features from these sentence vectors:

- The vector subtraction of the comment vector from the vector of the question at the head of the thread is used as the scoring vector for that comment.
- We calculate the cosine similarity, Euclidean and Manhattan distances between question and comment vectors.

4.3 Topic Modeling Features

To capture the thematic similarity between the question and comment texts, we train a LDA topic model on the training data using Mallet (McCallum, 2002). We perform different experiments by varying the number of topics (*n = 10, 20, 50, 100*) and obtain the best performance with 20 topics. We generate a topic vocabulary of 50 words for each topic class. The following features were entailed from these topic distributions and words:

- The vector subtraction of question and comment topic vectors, measuring the topical distance between the two snippets of text.
- Cosine, Euclidean and Manhattan distance between the topic vectors.
- We generate a vocabulary for each text by taking the union of topic words of its first 10 most probable topic classes.
  $\text{Vocabulary}(T) = \bigcup_{i=1}^{10} topic\_words(t_i)$
  where each t_i represents one of the top 10 topic classes for comment or question T.
  We then determine the word overlap of the topic vocabulary of the question with *(i)* the entire comment string and *(ii)* the topic vocabulary of the comment.

4.4 Domain Specific Features

In CQA sites, comments in a thread typically reflect an underlying discussion about a question, and there is usually a strong correlation among the nearby comments in the thread. Users reply to each other, ask for further details, can acknowl-

edge others' answers or can tease other users. Therefore, as discussed in (Barrón-Cedeño et al., 2015), comments in a common thread are strongly interconnected.

We extract various features from the meta data of the thread and from our surface observation of the thread's structure and properties. We extract if the comment is written by the asker of the question. In the case of repeated comments by the asker, we monitor if the comment is an acknowledgement (*thanks*, *thank you*, *appreciate*) or a further question. With the likely assumption that the comments at the beginning of a thread will be more relevant to the question, we have a feature capturing the position of comment in the thread. We also compute the coverage (the ratio of the number of tokens) of question by the comment and comment by the question.

We further try to model explicit conversations among users in the thread. We do it in two ways:

- Repeated and interlacing comments by a user in the same thread
- Explicitly mentioning the name of some previous user in the comment

The case of implicit dialogues (where the intent of the conversation has to be inferred solely from the context of the comment by a user) is discussed in a separate section later. These domain-specific features proved to be quite effective in classification, and thus form an integral part of our system.

4.5 Keyword and Named Entity Features

Finding the focus of the question and comment is important in measuring if the comment specifically covers the aspects of the question. We extract keywords from the texts using the RAKE keyword extraction algorithm (Rose et al., 2010), and derive features from the keyword match between question and comment. We also use the relative importance of common keywords as feature values.

In case of factoid questions, or especially in subtask B, *Named Entity Recognition* becomes an important tool for computing the relevance of a text. We extract named entities using the Stanford Named Entity Recognizer (Finkel et al., 2005) and classify words into seven entity categories including PERSON, LOCATION, ORGANIZATION, DATE, MONEY, PERCENT, and TIME. We compute if both question and comment have named entities, and if these belong to the same classes, if the named entity is an answer to a *Wh-type* question or not.

4.6 Implicit Dialogue Identification

Data driven error analysis on the *Qatar Living Data* indicated the presence of implicit dialogue chains. Users were almost always engaging in conversations with each other but this could only be captured by the content of their comment. Here we propose a novel algorithm based on construction of a user interaction graph to model these potential dialogues. The components of our construction are as follows:

- **Vertices** - Users in the comment thread and the Question
- **Edges** - Directed edges showing interaction
- **Edge Weights** - Numerical estimate of the interaction

Algorithm 1 Dialogue Group Detection

Initialize:
  $U \rightarrow$ User Graph ▷ Initially Empty
  $D \rightarrow$ Dialogue Graph ▷ Initially Empty
  $Q \rightarrow$ Question node ▷ Node indexed 0
procedure DIALOGUE IDENTIFICATION
  $V(U) \leftarrow V(U) \cup \{Q\}$ ▷ Add Q to vertex set of U
  for each comment c_x in thread **do**
    u_i commented c_x
    if u_i is a new user **then**
      $V(U) \leftarrow V(U) \cup \{u_i\}$
      $V(D) \leftarrow V(D) \cup \{u_i\}$
    end if
    for Q and each previous comment c_y **do**
      u_j commented c_y
      if $i \neq j$ and e_{ij} doesn't exist in $E(U)$ **then**
        Add e_{ij} in $E(U)$ ▷ Add e_{ij} in edge set of U
      end if
      $w(e_{ij}) \leftarrow$ Compute_Weight(c_x, c_y, i, j)
    end for
    $e \leftarrow max_j \ \ w(e_{ij})$ ▷ Pick max outgoing edge
    if $j \neq 0$ and e does not exist in $E(D)$ **then**
      Add e in $E(D)$
    end if
  end for
  Find weakly connected components in D
end procedure

The algorithm to construct this dynamic graph is given in Algorithm 1. We simultaneously construct two graphs, a user graph and a dialogue graph. Initially, the user graph has the question node and the dialogue graph is empty. We add new users to the graphs according to the timestamp of their occurrence in the thread. For each new comment, we add edges to each previous user and the question, in the user graph for the user

who commented. Then we pick the maximum outgoing edge to some previous user from the user who commented, and add that edge in the dialogue graph. Finally, we find the *weakly connected components* (WCCs) in the dialogue graph and the users in each such WCC are in mutual dialogue. Note that the user graph at the end of each iteration depicts the current conversational interaction of the user who commented, with respect to all other users in the thread.

Algorithm 2 Compute Weight Function

procedure COMPUTE_WEIGHT(c_x, c_y, i, j)
  u_i commented c_x
  u_j commented c_y
  $e_{ij} \leftarrow 0.0$
  if user u_i explicitly mentions user u_j in comment **then**
    $e_{ij} \leftarrow e_{ij} + 1.0$ ▷ Explicit dialogue
  end if
  $c_x \rightarrow \{w_1, w_2, ..., w_k\}$ ▷ w_i is the i^{th} word in c_x
  $c_y \rightarrow \{w_1^{'}, w_2^{'}, ..., w_l^{'}\}$
  $tr\_score \leftarrow (\sum_{1 \leq m \leq k} \max_{1 \leq n \leq l} cos(\vec{v_{w_m}}, \vec{v_{w_n^{'}}}))/k$
  $\vec{t_x} \leftarrow$ topic vector for c_x
  $\vec{t_y} \leftarrow$ topic vector for c_y
  $to\_score \leftarrow cos(\vec{t_x}, \vec{t_y})$ ▷ Topic similarity score
  $e_{ij} \leftarrow e_{ij} + tr\_score + to\_score$ ▷ Edge weight
  return e_{ij}
end procedure

The main part of the algorithm is where we compute the edge weights between a pair of users after some comment, see Algorithm 2 for details. We have three components that constitute the weight:

- if the user mentions the other user explicitly
- we calculate the score of reformulating one comment from the other by closest word match based on cosine scores of word vectors *(tr_score)*
- cosine of the topic vectors of a pair of comments *(to_score)*

In addition to identifying latent dialogue groups, we also extract features from this graph and these features prove to be very helpful in classification.

4.7 Classifier

We use an SVM classifier as implemented in LibSVM (Chang and Lin, 2011) for classification. We experiment with different kernels (Hsu et al., 2003), and achieve the best results with the RBF kernel, which we use to train the model for our primary submission. We also achieve comparable results with the linear kernel and L2-regularized logistic regression. The ranking score for a question-comment pair in subtask A is the calculated probability of the pair to be classified as *Good*.
The ranking score for subtask B is the SVM probability score for the original question-related question pair multiplied by the reciprocal search engine rank provided in the data.
For subtask C, the scoring value is the sum of the log probabilities of the SVM scores of all subtasks $final\_score = \log(svm\_A) + \log(svm\_B) + \log(svm\_C)$

5 Stacking features for other subtasks

We implemented a generic system for tackling semantic similarity for any two snippets of text. We further fine tuned it with domain specific features for subtask A. For subtasks B and C, we again adopted this generic system with slight modifications. But, the strong interconnectivity and incremental nature of the subtasks motivated the development of a stacking strategy where we propagate useful information from other subtasks as features for the present subtask and re-run the classifier. Filice (2016) developed a stacking strategy that we adopt with modifications.
For subtask B, we consider the scores for subtasks A and C as probability distributions and calculate various features and correlation coefficients (*Spearmann, Kendall, Pearson*) over these distributions.
For subtask C, we calculate feature values from the SVM scores of all three subtasks, and re-run our system with these stacking features. These features include average, minimum and maximum of subtask A and B scores, and binary features capturing if these probability scores are above 0.5.

6 Experimentation and Results

We extensively experimented with a lot of feature engineering. Notable features that were discarded in the feature ablation process are:

- **Statistical Paraphrasing**: We found the top 10 semantically related words corresponding to every word in the comment, based on word vectors and did an n-gram matching (n = *1,2,3*) on the extended wordlist.
- **Doc2Vec**: We also used Doc2Vec (Le and Mikolov, 2014) to generate sentence vectors directly, but these degraded the results.

Features	Development Set 2017						
Subtask A	**MAP**	**AvgRec**	**MRR**	**P**	**R**	**F1**	**Accuracy**
All Features	**65.50**	**84.86**	**71.96**	**58.43**	**62.71**	**60.50**	**72.54**
All — string features	65.53	84.90	72.19	57.84	62.71	60.18	72.17
All — embedding features	62.11	81.23	69.00	53.03	53.42	53.23	68.52
All — domain features	61.85	81.06	69.80	54.46	54.52	54.49	69.47
All — topic features	65.15	84.79	72.37	59.02	61.98	60.47	72.83
All — keyword features	65.73	84.65	71.94	57.98	62.59	60.20	72.25
IR Baseline	**53.84**	**72.78**	**63.13**	-	-	-	-
Subtask B							
All Features	**73.03**	**88.77**	**78.33**	**72.39**	**45.33**	**55.75**	**69.20**
All — string features	73.46	88.83	78.95	72.87	43.93	54.81	69.00
All — embedding features	73.91	89.11	79.33	71.53	45.79	55.84	69.00
All — domain features	73.07	88.77	78.33	71.77	41.59	52.66	68.00
All — topic features	72.95	88.07	78.17	67.86	44.39	53.67	67.20
All — keyword features	73.55	88.99	79.33	72.93	45.33	55.91	69.40
All — stacking features	72.95	88.64	78.67	71.90	40.65	51.94	67.80
IR Baseline	**71.35**	**86.11**	**76.67**	-	-	-	-
Subtask C							
All Features	**36.09**	**41.13**	**39.89**	**18.42**	**37.10**	**24.62**	**84.32**
All — string features	36.85	40.27	39.72	16.81	35.07	22.72	83.54
All — embedding features	39.39	45.09	45.01	17.48	47.83	25.60	80.82
All — domain features	36.83	40.68	39.69	17.21	35.07	23.09	83.88
All — topic features	35.89	41.18	40.50	16.98	38.84	23.63	82.68
All — keyword features	35.39	41.17	38.57	18.58	37.97	24.95	84.24
All — stacking features	36.57	41.85	40.80	16.80	36.81	23.07	83.06
IR Baseline	**30.65**	**34.55**	**35.97**	-	-	-	-
Runs	**Test Set 2017**						
Subtask A	**MAP**	**AvgRec**	**MRR**	**P**	**R**	**F1**	**Accuracy**
Primary	**86.88**	**92.04**	**91.20**	**73.37**	**74.52**	**73.94**	**72.70**
Contrastive 1	86.35	91.74	91.40	79.42	51.94	62.80	68.02
Contrastive 2	85.24	91.37	90.38	81.22	57.65	67.43	71.06
IR Baseline	**72.61**	**79.32**	**82.37**	-	-	-	-
Subtask B							
Primary	**43.12**	**79.23**	**47.25**	**26.85**	**71.17**	**38.99**	**58.75**
Contrastive 1	42.29	78.41	46.40	32.66	59.51	42.17	69.77
Contrastive 2	42.38	78.59	46.82	32.99	59.51	42.45	70.11
IR Baseline	**41.85**	**77.59**	**46.42**	-	-	-	-
Subtask C							
Primary	**15.46**	**33.42**	**18.14**	**08.41**	**51.22**	**14.44**	**83.03**
Contrastive 1	15.43	33.78	17.52	09.45	54.07	16.08	84.23
Contrastive 2	14.00	30.53	14.65	05.98	85.37	11.17	62.06
IR Baseline	**09.18**	**21.72**	**10.11**	-	-	-	-

Table 1: Feature Ablation Results on Development Set and Runs on Test Set

Our primary submission for subtasks A and B uses SVM with an RBF kernel for classification as this yielded the best results on the dev set. We also achieved similar results with the linear and L2-regularized logistic regression classifiers and we use these for our contrastive submissions. All the submissions comprised of same number of features. For subtask C, we oversample the training data using the SMOTE (Chawla et al., 2002) technique in the ImbalancedLearn[3] toolkit, due to the highly skewed distribution of labels. We use regular SMOTE for our primary and SMOTE SVM for our first contrastive submission. For the second contrastive submission, we integrate the feature sets of subtasks A and B directly in the feature set of subtask C.

The feature ablation results on the development set and the results of different runs on the test set are presented in Table 1. It reports the system performance on all evaluation metrics including Mean Average Precision (*MAP*), Average Recall (*AvgRec*), Mean Reciprocal Rank (*MRR*), Precision (*P*), Recall (*R*), F1-score (*F1*) and Accuracy.

7 Conclusion

We establish the importance of domain specific and dialogue identification features in tackling the given task. In future work, we would like to fo-

[3] https://github.com/scikit-learn-contrib/imbalanced-learn

cus on extracting more information from inter-comment dependencies. This should improve our algorithm for dialogue group detection and model conversational activity better. We also wish to work on a Deep Learning architecture for handling this, as in (Wu and Lan, 2016) and (Guzmán et al., 2016). The problem can be modeled as a semi-supervised classification task, where the unannotated data can help supervised classification. Subtask C still presents a challenging research problem and we will investigate novel methods to integrate results from other subtasks to tackle this subtask better.

References

Alberto Barrón-Cedeño, Simone Filice, Giovanni Da San Martino, Shafiq Joty, Lluís Màrquez, Preslav Nakov, and Alessandro Moschitti. 2015. Thread-Level Information for Comment Classification in Community Question Answering. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*. Association for Computational Linguistics, Beijing, China, pages 687–693. http://www.aclweb.org/anthology/P15-2113.

Yonatan Belinkov, Mitra Mohtarami, Scott Cyphers, and James Glass. 2015. VectorSLU: A Continuous Word Vector Approach to Answer Selection in Community Question Answering Systems. In *Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015)*. Association for Computational Linguistics, Denver, Colorado, pages 282–287. http://www.aclweb.org/anthology/S15-2048.

Chih-Chung Chang and Chih-Jen Lin. 2011. LIBSVM: A library for support vector machines. *ACM TIST* 2(3):27:1–27:27. https://doi.org/10.1145/1961189.1961199.

Nitesh V. Chawla, Kevin W. Bowyer, Lawrence O. Hall, and W. Philip Kegelmeyer. 2002. SMOTE: synthetic minority over-sampling technique. *Journal of artificial intelligence research* 16:321–357.

Corinna Cortes and Vladimir Vapnik. 1995. Support-vector networks. *Machine learning* 20(3):273–297.

Simone Filice, Danilo Croce, Alessandro Moschitti, and Roberto Basili. 2016. KeLP at SemEval-2016 Task 3: Learning Semantic Relations between Questions and Answers. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*. Association for Computational Linguistics, San Diego, California, pages 1116–1123. http://www.aclweb.org/anthology/S16-1172.

Jenny Rose Finkel, Trond Grenager, and Christopher Manning. 2005. Incorporating Non-local Information into Information Extraction Systems by Gibbs Sampling. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05)*. Association for Computational Linguistics, Ann Arbor, Michigan, pages 363–370. https://doi.org/10.3115/1219840.1219885.

Francisco Guzmán, Preslav Nakov, and Lluís Màrquez. 2016. MTE-NN at SemEval-2016 Task 3: Can Machine Translation Evaluation Help Community Question Answering? In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*. Association for Computational Linguistics, San Diego, California, pages 887–895. http://www.aclweb.org/anthology/S16-1137.

Chih-Wei Hsu, Chih-Chung Chang, and Chih-Jen Lin. 2003. A practical guide to support vector classification. Technical report, Department of Computer Science, National Taiwan University. http://www.csie.ntu.edu.tw/ cjlin/papers.html.

Quoc V. Le and Tomas Mikolov. 2014. Distributed Representations of Sentences and Documents. In *Proceedings of the 31th International Conference on Machine Learning, ICML 2014, Beijing, China, 21-26 June 2014*. pages 1188–1196. http://jmlr.org/proceedings/papers/v32/le14.html.

Andrew Kachites McCallum. 2002. MALLET: A Machine Learning for Language Toolkit. Http://mallet.cs.umass.edu. http://mallet.cs.umass.edu.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Gregory S. Corrado, and Jeffrey Dean. 2013. Distributed Representations of Words and Phrases and their Compositionality. In *Advances in Neural Information Processing Systems 26: 27th Annual Conference on Neural Information Processing Systems 2013. Proceedings of a meeting held December 5-8, 2013, Lake Tahoe, Nevada, United States.*. pages 3111–3119. http://papers.nips.cc/paper/5021-distributed-representations-of-words-and-phrases-and-their-compositionality.

Preslav Nakov, Doris Hoogeveen, Lluís Màrquez, Alessandro Moschitti, Hamdy Mubarak, Timothy Baldwin, and Karin Verspoor. 2017. SemEval-2017 Task 3: Community Question Answering. In *Proceedings of the 11th International Workshop on Semantic Evaluation*. Association for Computational Linguistics, Vancouver, Canada, SemEval '17.

Massimo Nicosia, Simone Filice, Alberto Barrón-Cedeno, Iman Saleh, Hamdy Mubarak, Wei Gao, Preslav Nakov, Giovanni Da San Martino, Alessandro Moschitti, Kareem Darwish, et al. 2015. QCRI: Answer selection for community question answering-experiments for Arabic and English. In *Proceedings of the 9th International Workshop on Semantic Evaluation, SemEval*. volume 15, pages 203–209.

Stuart Rose, Dave Engel, Nick Cramer, and Wendy Cowley. 2010. Automatic keyword extraction from individual documents. In Michael W. Berry and Jacob Kogan, editors, *Text Mining. Applications and Theory*, John Wiley and Sons, Ltd, pages 1–20. https://doi.org/10.1002/9780470689646.ch1.

Guoshun Wu and Man Lan. 2016. ECNU at SemEval-2016 Task 3: Exploring Traditional Method and Deep Learning Method for Question Retrieval and Answer Ranking in Community Question Answering. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*. Association for Computational Linguistics, San Diego, California, pages 872–878. http://www.aclweb.org/anthology/S16-1135.