Question Relatedness on Stack Overflow: The Task, Dataset, and Corpus-inspired Models

Amirreza Shirani[†], Bowen Xu[‡], David Lo[‡], Thamar Solorio[†] and Amin Alipour[†]
[†]University of Houston
[‡]Singapore Management University

Abstract

Domain-specific community question answering is becoming an integral part of professions. Finding related questions and answers in these communities can significantly improve the effectiveness and efficiency of information seeking. Stack Overflow is one of the most popular communities that is being used by millions of programmers. In this paper, we analyze the problem of predicting knowledge unit (question thread) relatedness in Stack Overflow. In particular, we formulate the question relatedness task as a multi-class classification problem with four degrees of relatedness.

We present a large-scale dataset with more than $300K$ pairs. To the best of our knowledge, this dataset is the largest domain-specific dataset for Question-Question relatedness. We present the steps that we took to collect, clean, process, and assure the quality of the dataset. The proposed dataset on Stack Overflow is a useful resource to develop novel solutions, specifically data-hungry neural network models, for the prediction of relatedness in technical community question-answering forums.

We adapt a neural network architecture and a traditional model for this task that effectively utilize information from different parts of knowledge units to compute the relatedness between them. These models can be used to benchmark novel models, as they perform well in our task and in a closely similar task.

Introduction

Community question answering (cQA) is becoming an integral part of professions allowing users to tap on crowds' wisdom and find answers to their questions. Techniques, such as answer summarization (Chan et al. 2012; Xu et al. 2017; Demner-Fushman and Lin 2006; Liu et al. 2008), question answer matching (Tan et al. 2016; Shen et al. 2015) and question semantic matching (Bogdanova et al. 2015; Wu, Zhang, and Huang 2011; Nakov et al. 2017), have been devised to improve users' experience by accelerating finding relevant information and enhancing the information presentation to users.

We refer to the collection of a question along with all its answers as a *knowledge unit* (KU). Finding related knowledge unit in these communities can significantly improve the

effectiveness and efficiency of information seeking. It allows users to navigate between knowledge units, prune unrelated knowledge units from the information search space. Finding related knowledge units can be quite time-consuming due to the fact that even the same question can be rephrased in many different ways. Therefore automated techniques to identify related knowledge units are desirable.

In this work, we describe the task of prediction of *relatedness* in Stack Overflow, the most popular resource for topics related to software development. Knowledge in Stack Overflow is dispersed and developers usually need to explore several related knowledge units to gain insights into the problem at hand and possible solutions. Stack Overflow has become an indispensable tool for programmers; about 50 million developers visit it monthly, and over 85% of users visit Stack Overflow almost daily.[1] The reputation of this webiste has attracted many developers to actively participate and contribute to the forum. A study showed that most questions on Stack Overflow are answered within 11 minutes of posting them (Mamykina et al. 2011).

We formulate the problem of identification of related KUs, as a multi-class classification problem by breaking relatedness into multiple classes. More precisely, a model has to classify the degree of relatedness of two KUs into one of four classes: *duplicate*, *direct*, *indirect*, or *isolated*.

Predicting relatedness in Stack Overflow poses an interesting challenge because in addition to natural text, KUs contain a huge amount of programming terms which is of a different nature, and like many other cQA websites, different users exhibit different discursive habits in posting questions and answers; e.g., some provide minimal details in their questions or answers, while some tend to include a sizable amount of information.

We create a large, reliable dataset for training and testing models for this task. It contains more than 300K knowledge unit pairs annotated with their corresponding relatedness class. We report all steps to collect, clean, process, and assure the quality of the dataset. We rely on URL sharing in Stack Overflow to decide on the relatedness of KUs, as that programmers facing a specific problem are the best ones to judge the degree of relatedness of questions. We verified the

---

---

[1]Stack Overflow 2018 Developer Survey, `https://insights.stackoverflow.com/survey/2018/`

reliability of our approach by conducting a user study.

To establish a baseline for future evaluations, we present two successful neural network and traditional machine learning models. we adapt a lightweight Bidirectional Long Short-term Memory (BiLSTM) model tailored to our proposed dataset. We also investigate so-called soft-cosine similarity features in a Support Vector Machine (SVM) model. To investigate the adequacy of these models, we evaluate them on a closely related duplicate detection task. Our experiments show that our models outperform the state-of-the-art techniques in a duplicate detection task, suggesting that our models are potent benchmarks for our task.

Contributions. This paper makes the following contributions.

- We present the task of question relatedness in Stack Overflow, with four degrees of similarity.

- We present a reliable, large dataset for knowledge units relatedness in Stack Overflow.

- We adapt a corpus-inspired BiLSTM architecture for relatedness detection.

- We evaluate the performance of SVM models with several hand-crafted features to predict the relatedness in Stack Overflow.

Related Work

There are several tasks related to identifying semantically relevant questions such as Duplicate Question Detection (DQD), Question-Question similarity, and paraphrase identification.

Perhaps, one of the best-known general-domain DQD dataset is Quora [2] with more than 400K question pairs. Quora dataset was released on Kaggle competition platform in January 2017. Most of the questions on Quora are asked in one piece without any further description and are not restricted to any domain. Another well-known DQD dataset is AskUbuntu (Rodrigues et al. 2017). Similar to our Stack Overflow dataset, AskUbuntu dataset is acquired from Stack Exchange data dump [3] (September 2014). The differences are that AskUbuntu dataset only provides binary classes (DQD), it is 11 times smaller than our proposed dataset and only consist of titles and bodies in a concatenated form. Many solutions are proposed to address the DQD problem. (Bogdanova et al. 2015) utilized a convolutional neural network (CNN) to address the DQD problem on AskUbuntu and Meta datasets. (Silva et al. 2018) applied the same model on the cleaned version of datasets and showed that after removing Stack Exchange clues, the results drop by 20%. A more advanced architecture introduced in (Rodrigues et al. 2017) on AskUbuntu and Quora datasets. This model can be considered as the state-of-the-art model on AskUbuntu dataset which utilizes the combination of a *MayoNLP* model introduced in (Afzal, Wang, and Liu 2016) and a *CNN* model introduced in (Bogdanova et al. 2015). We use the same AskUbuntu dataset to evaluate our models on a secondary

dataset. There are two major differences between our approach and the works in (Bogdanova et al. 2015) and (Rodrigues et al. 2017). First, we improve the performance of our model by computing the distance between title, body, and answers of the two knowledge units, whereas (Bogdanova et al. 2015) and (Rodrigues et al. 2017) only compute the similarity between title+body of the two knowledge units. Second, the hybrid architectures developed by (Rodrigues et al. 2017) is a complex CNN model along with 30k dense neural network followed by two hidden multi-layers. However, our model uses shared layers bidirectional LSTMs with the limited number of parameters which results in a lightweight architecture.

Question-Question similarity introduced in subtask B of SemEval-2017 Task 3 on Community Question Answering [4] (Nakov et al. 2017) is one of the closest topics to our task. Although this task contains multi-classes of relatedness between two questions (i.e., PerfectMatch, Related, Irrelevant), unlike our task, the problem is formulated as a re-ranking Question_Question+Thread Similarity task. Various features were investigated to address Question-Question similarity introduced in subtask B of SemEval-2017 Task 3 such as neural embedding similarity features (Goyal 2017) and Kernel-based features (Filice, Da San Martino, and Moschitti 2017) (Galbraith, Pratap, and Shank 2017). The winner of this task is (Charlet and Damnati 2017) which utilized soft-cosine similarity features within a Logistic Regression model. Note that we employ the similar soft-cosine features in our traditional SVM model.

Duplicate detection between questions on Stack Overflow has been studied before. An approach named *DupPredictor* takes a new question as an input and tries to find potential duplicates of the question by considering multiple information sources (i.e., title, description and tags) (Zhang et al. 2015). *DupPredictor* computes the latent topics of each question by using a topic model. For each pair of questions, it computes four similarity scores by comparing their titles, descriptions, latent topics, and tags and then combined together to result in a new similarity score. In another similar work, (Xu et al. 2016) introduced a dataset for knowledge unit relatedness and proposed a convolutional neural network for predicting the relatedness. Unfortunately, the limited number of knowledge units (KUs) were collected heuristically and tend to have low quality. The presented dataset does not cover different parts of a knowledge unit, instead, it merges title+body into a single sequence. Clearly, mixing all parts together does not provide an opportunity to perform an experiment on separate parts of KUs independently. Moreover, this dataset contains some extra information (signals) which leads to a biased dataset. As explained in "Data Quality" section, we remove these unwanted clues from the data.

Description of The Dataset

Questions in the real world are supposed to have more relationships than only duplicate or non-duplicate. For example,

one question in Stack Overflow talks about *The time complexity of array function*[5], while another question is about *How to find time complexity of an algorithm*[6]. These two questions are linked by Stack Overflow users as related but not duplicate.

Relatedness Between Knowledge Units

Knowledge units often contain semantically-related knowledge, and thus they are linkable for different purposes, such as explaining certain concepts, approaches, background knowledge or describing a sub-step for solving a complex problem (Ye, Xing, and Kapre 2016). Figure 1 shows an example of how knowledge units are linked to each other on Stack Overflow. One of the answers of a knowledge unit (short for *KU1*) guides the asker to refer to another knowledge unit (short for *KU2*) which is helpful to solve the problem. These two knowledge units are linked through URL sharing. URL sharing is strongly encouraged by Stack Overflow to link related knowledge units (StackOverflow 2018). A network of linkable knowledge units constitutes a *knowledge unit network* (KUNet) over time through URL sharing (Ye, Xing, and Kapre 2016). Relationships between any two knowledge units in KUNet can be divided into four classes: *duplicate*, *direct*, *indirect* and *isolated* (Xu et al. 2016). Duplicate KUs discuss the same question and can be answered by the same answer. Direct relatedness between KUs means that the content of one KU can help solve the problem in the other KU, for example, by explaining certain concepts, providing examples, or covering a sub-step for solving a complex problem. Indirect relatedness means that contents of KUs are related but they are not immediately applicable to each other. Isolated KUs are not semantically related. The order of relatedness of each class is *duplicate > direct > indirect > isolated.*

Dataset Creation

Figure 2 depicts the steps that we took to create a relatedness dataset. We describe each step below.

Extract preliminary data from Stack Overflow data dump. We mainly focus on Java-related knowledge units on Stack Overflow because Java is one of the top-3 most popular tags in Stack Overflow [7]. Moreover, questions with this tag not only are about Java programming language, but they cover a broad spectrum of topics that Java technology provides, such as web and mobile programming, and embedded systems. First, we extracted *all* knowledge units tagged by "Java" from Stack Overflow data dump. Next, all *duplicate* and *direct* links between knowledge unit pairs are extracted from Stack Overflow data dump.

Knowledge unit network. Knowledge unit network (KUNet) is a network in which each KU is represented as a node and an edge between two nodes exists if a *duplicate* or *direct* link exists between the two corresponding KUs. We construct a KUNet based on the extracted links from a table named *PostLinks* from Stack Overflow data dump.

---

[5] https://goo.gl/dJwmuE
[6] https://goo.gl/S8lBjE
[7] https://stackoverflow. com/tags

Identifying duplicate and direct pairs As shown in Figure 2(a), the link between (A and B) and (B and C) are labeled as a *duplicate*. We also consider a *duplicate* link between A and C by transitivity. We apply transitivity rule until no new duplicate relation is found among knowledge units.

Identifying indirect and isolated pairs Four types linkable KU pairs are extracted from the KUNet based on their definitions. *Indirect* KU pairs are pairs of nodes that are indirectly connected in the network. More specifically, they are connected in the KUNet with a certain range of distance (in this case, length of shortest path $\in$ [2,5]), but the relationship between them belongs neither to duplicate nor direct. Finally, *isolated* KU pairs are pairs of nodes that are completely disconnected in the network.

Statistical Characteristics of the Dataset

Using the steps described in the previous section we created a dataset. Table 1 depicts the statistical characteristics of the dataset. The dataset contains 160,161 distinct knowledge units and 347,372 pairs of knowledge units with four types of relationships. Among all knowledge units, $117,139$ (i.e., 73%) of them have at least one code snippet in their body. The average number of words in code snippets in body is 118.46. There are $318,491$ answers in our dataset and each knowledge unit has 1.99 answers on average. $140,122$ (i.e., 87%) of knowledge units contain at least one answer and $90,672$ (i.e., 57%) of them contain one accepted answer. Moreover, $96,707$ (60%) of knowledge units have at least one code snippet in their answers which means that more than half of solutions are code related.

Training, Development, and Test Sets We split the dataset into three parts, train, development, and test, to facilitate the development, and evaluation of classification models. We assigned 60% of knowledge units to train set, 10% to development set, and 30% to test set. To have the same number of KU pairs for each class, by using under-sampling techniques, we make this dataset balanced.

Table 1: Brief statistics of the dataset

Scope	Indicator	Size
Whole KU	# of distinct KUs	160,161
	# of four types of KU pairs	347,372
Title	avg. # of words in title	8.52
Body	avg. # of words in body(exclude code snippets)	97.02
	# of distinct KUs whose body has at least one code snippet	117,139(73%)
	avg. # of code snippets in one body	1.46
	avg. # of words in single code snippet in one body	118.46
Answers	# of distinct answers	318,491
	avg. # of answers within single KU	1.99
	# of distinct KUs contain at least one answer	140,122(87%)
	# of distinct KUs contain an accepted answer	90,672(57%)
	# of distinct KUs whose answers has at least one code snippet	96,707(60%)
	avg. # of words in an answer (exclude code snippets)	68.39
	avg. # of code snippets within one answer	0.60
	avg. # of words in single code snippet	81.98

Instructions to Use The Dataset

Table 2 presents the overall structure of our dataset. There are 24 attributes in our dataset for each pair of knowledge units. The first 23rd attributes include all the content of the first and second knowledge units, they are id, title, body, accepted answer, answers, and tags.
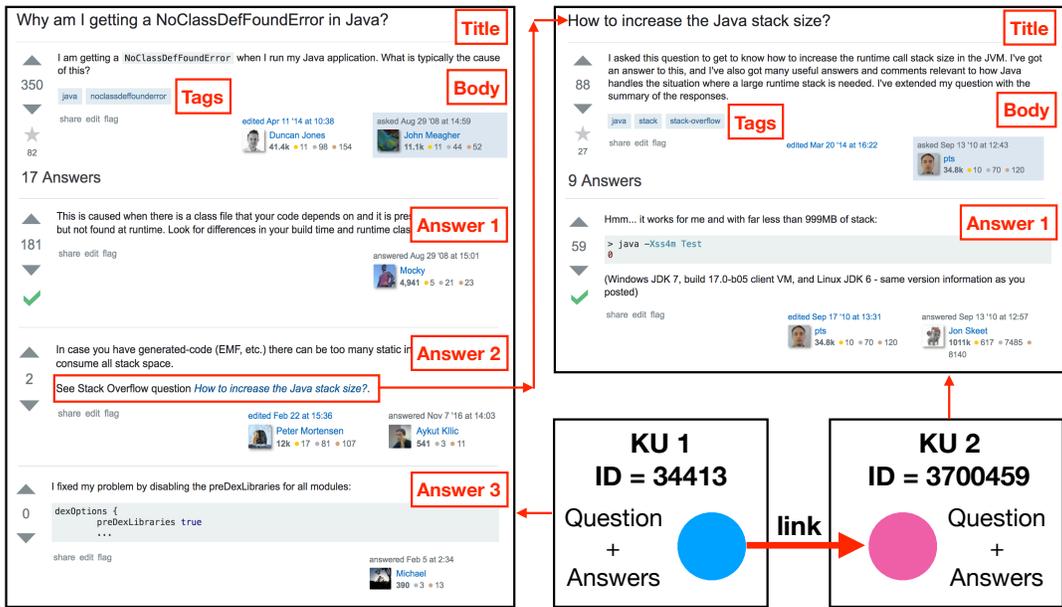
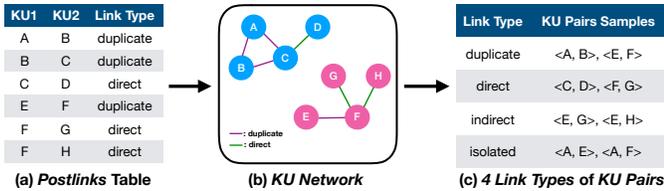Figure 1: A pair of linkable knowledge units on Stack Overflow

Figure 2: Overview of the data collection process

The last attribute (i.e., Attr. Id =24) represents the relationship between the two knowledge units (i.e., $< KU1, KU2, Relationship >$). More information is available at `https://anonymousaaai2019.github.io`

Table 2: The structure of the dataset

Attr. Id	Attr. Name	Attr. Description
1	Id	KU Pair ($< KU1, KU2 >$) Id
2/13	q1/2_Id	Id of *KU*'s Question on SO
3/14	q1/2_Title	*KU*'s Title
4/15	q1/2_Body	The text of *KU*'s Body (Exclude Code Snippets)
5/16	q1/2_BodyCode	Code Snippets in *KU*'s Body
6/17	q1/2_AcceptedAnswerId	Ids of *KU*'s Accepted Answers on SO
7/18	q1/2_AcceptedAnswerBody	The text of *KU*'s Accepted Answer (Exclude Code Snippets)
8/19	q1/2_AcceptedAnswerCode	Code Snippets in *KU*'s Accepted Answer
9/20	q1/2_AnswersIdList	Ids of *KU*'s Answers on SO
10/21	q1/2_AnswersBody	The text of *KU*'s Answers (Exclude Code Snippets)
11/22	q1/2_AnswersCode	Code Snippets in *KU*'s Answers
12/23	q1/2_Tags	Tags of *KU*
24	Class	Relationship (i.e., *duplicate*, *direct*, *indirect* or *isolated*)

Quality Control

Data Cleaning

We perform three operations to further improve the quality of our dataset. Natural language and programming language snippets are mixed in the text. To deal with this, first, we extract programming language snippets (aka. code snippets) from HTML formatted text by using the regular expression $\langle pre \rangle \langle code \rangle (.*?) \langle /code \rangle \langle /pre \rangle$ Note that, it is possible that multiple code snippets exist in body or multiple answers of one knowledge unit, so we store them into a list. Next, since text attributes (e.g., body, answer body) provided by Stack Overflow data dump are in HTML format, we clean the content by removing HTML tags and escape characters, e.g., $\langle p \rangle \langle /p \rangle$, $\&\#xA;$ and $\<$. Second, we observe and remove some extra information added by Stack Exchange API that can be considered as a signal. For example, at the beginning of the body content of some duplicate and direct questions, it includes the string `Possible Duplicate:`, followed by the topic content of the possible duplicate question. The inclusion of signals in training can result in a biased dataset and unreliable models. This problem was first observed by (Silva et al. 2018) in AskUbuntu dataset.

Third, we found that there is an overlap between some *duplicate* and *direct* links in the Stack Overflow data dump, since it provides knowledge unit pairs as long as two knowledge units are linked through URL sharing. To solve this, if a link belongs to *duplicate* and *direct* at the same time, we label it as a *duplicate*.

User Study

This dataset is extracted from Stack Overflow forum that is managed and maintained by volunteer domain experts who serve as moderators and contributors. Links between knowledge units (i.e., Stack Overflow posts) are validated in a crowdsourced process by domain experts. To asses the reliability of the crowdsourced process and our data collection procedure, we perform a user study. We ask three experts (who are not authors of this paper) to label relationships between pairs of knowledge units that we have in our dataset. The participants analyze a statistically significant sample

size (i.e., 96 pairs) that is representative of the population of knowledge units in our dataset (at 95% confidence level, and 10% margin or error). Each participant can provide his/her assessment of the degree of relatedness of two knowledge units in a 4 point Likert scale: 1 (unrelated/isolated), 2 (indirect), 3 (direct), and 4 (duplicate). The user study highlights that the participant labels are the same as the labels in our dataset 82% of the time. The average absolute difference between the Likert scores and the labels in our dataset is only 0.2 (out of 4). This highlights that the links in our dataset are of high-quality.

Method

In this section, we describe models to predict relatedness between knowledge units. We extensively explore different neural network and traditional models for this task and report the best-performing models. First, we investigate a BiLSTM architecture which progressively learns and compares the semantic representation of different parts of two knowledge units. The description of our model is presented in the next section. We then compare the BiLSTM model with a support vector machine model. We also apply these models to a closely similar task, duplicate detection in AskUbuntu, and compare the results with the state-of-the-art models in that task.

Data Pre-processing

We apply some simple pre-processing steps on all text parts, Title, Body and Answers. Since there are many technical terms in Stack Overflow, we apply more specific pre-processing steps: First, we split words with punctuation marks. For example, `javax.persistence.Query javax_query` changes to `javax persistence Query javax query`. Then, we split camel case words, for example, `EntityManage` is changed to `Entity Manage`. In the end, we take several standard steps in pre-processing data including: normalizing URLs and numbers, removing punctuation marks and stop-words, and changing all words to lowercase.

LSTM Model

We use bidirectional long short-term memory (BiLSTM) (Hochreiter and Schmidhuber 1997) as a sentence encoder to capture long-term dependencies in forward and backward directions. In a simple form, an LSTM unit contains a memory cell with self-connections, as well as three multiplicative gates to control information flow. Given input vector x_t, previous hidden outputs h_{t-1}, and previous cell state c_{t-1}, LSTM units operate as Figure 3, where i_t, f_t, o_t are input, forget, and output gates, respectively. The sigmoid function $\sigma()$ is a soft gate function controlling the amount of information flow. W_s and bs are model parameters to learn.

Figure 4 describes the overall architecture of the BiLSTM model (DOTBILSTM). Unlike previous studies (i.e. (Rodrigues et al. 2017)(Bogdanova et al. 2015)), this model utilizes the information in Title, Body and Answers parts of each knowledge unit. Each word (w_i) is represented as a vector, $\mathbf{w} \in \mathbb{R}^d$, looked up into an embedding matrix,

$$X = \left[\begin{array}{c} x_t \\ h_{t-1} \end{array} \right]$$
$$i_t = \sigma(W_{iX}X + W_{ic}ct - 1 + b_i)$$
$$f_t = \sigma(W_{fX}X + W_{fc}ct - 1 + b_f)$$
$$o_t = \sigma(W_{oX}X + W_{oc}c_t - 1 + b_o)$$
$$c_t = f_t \odot c_{t-1} + i_t \odot tanh(W_{cX}X + b_c)$$
$$h_t = o_t \odot tanh(c_t)$$

Figure 3: LSTM Unit

$\mathbf{E} \in \mathbb{R}^{d \times |V|}$. A shared layer BiLSTM as a sentence encoder takes all the six inputs, embeds and transforms them into fixed-sized vectors. Then in order to compute the distance between each two knowledge units, we compute the inner dot product between all the three representations of the first knowledge unit and all three representations of the second knowledge unit. As a result, it maps a pair of knowledge units into a low dimensional space, where their distance is small if they are similar. In the next step, we concatenate computed values together. Our results show that concatenating the BiLSTM representations at the last layer increases the performance slightly. We feed these values to a fully-connected layer followed by a ReLU activation function, a dropout layer and then a *SoftMax* output layer for classification. The objective function is the Categorical cross-entropy objective over four class target labels.
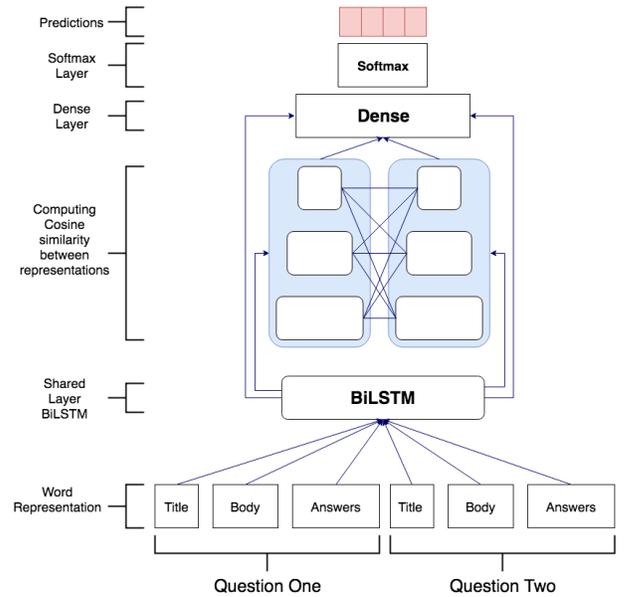

Figure 4: Main architecture of DOTBILSTM

Implementation Details (DOTBILSTM)

This section describes implementation details which are empirically chosen after running several models with different values and keeping the one that gives us the best results in the validation set.

We initialize word embeddings with pre-trained GloVe (Pennington, Socher, and Manning 2014) vectors of size 300. Compared to pre-trained Google news word2vec (Mikolov et al. 2013a) and word embedding trained on Stack Overflow, GloVe performed slightly better in this task. We choose the size of each sentence based on the average size over the training set. Titles are truncated or padded to 10 words, bodies to 60 words and answers to 180. BiLSTMs with 128 units is used as the encoder. In our experiments, we observed that using shared parameters for BiLSTMs boosts the model. The network uses Adam optimizer (Kingma and Ba 2014), and the learning rate is set to 0.001. The last layer is a dense layer with ReLu activation and 50 units. In order to have a better training and force the network to find different activation paths which leads to a better generalizing, a dropout layer with the rate of 0.2 is used. All the models are trained for 25 epochs and the reported test accuracy corresponds to the best accuracy obtained on the validation set.

SVM model

In this section, we explain the design of SOFTSVM, an SVM model for question relatedness task. We investigate different features as well as different data selections to achieve the best possible results.

We extract three types of features from knowledge units: *Number of common n-grams* which is simply the number of common word n-grams, and common character n-grams in a pair of text sequences. *Cosine similarity measure* to determine the similarity between two vectors (Kenter and De Rijke 2015; Levy, Goldberg, and Dagan 2015). This feature is obtained by TF-IDF weighting, computed over the training and development datasets. And, *Soft-cosine similarity measures* that unlike the traditional cosine similarity, takes into account word-level relations by computing a relation matrix (Sidorov et al. 2014). Given two N-dimension vectors a and b, the soft cosine similarity is calculated as follows.

$$soft-cosine(a,b) = \frac{\sum_{i,j}^{N} a_i m_{ij} b_j}{\sqrt{\sum_{i,j}^{N} a_i m_{ij} a_j}\sqrt{\sum_{i,j}^{N} b_i m_{ij} b_j}} \tag{1}$$

Unlike cosine similarity, soft-cosine similarity between two texts without any words in common is not null as soon as the two texts share related words. For computing the matrix M, we followed the same implementation presented in (Charlet and Damnati 2017), the winner of SemEval-2017 Task 3, Question-Question similarity. We create three variants of soft-cosine similarity feature. One is computed based on Levenshtein distance (Soft_Lev), and the other two features are based on two different word embeddings: Google News pre-trained word2vec (Mikolov et al. 2013a)(Soft_Google) and Stack Overflow domain-specific word2vec (Soft_SO).

Implementation Details (SOFTSVM)

We build an SVM model with the linear kernel using `sklearn` package (Pedregosa et al. 2011). In total, for each KU pair, we extract ten different hand-crafted features:

three common word n-grams (for n=1,2 and 3), three common character n-grams (for n=3,4 and 5), cosine similarity and three soft-cosine similarity features Soft_SO, Soft_Lev and Soft_Google. We compute the features between titles, bodies and answers separately. For computing Soft_SO, we train word2vec on text parts of the dataset using skip-gram model (Mikolov et al. 2013b) with vectors dimension 200 and minimum word frequency of 20.

Feature Selection

In this section, we compare and select important features by building SVM models using each feature separately. As shown in Figure 5, cosine and three Soft-cosine features outperform other features. Therefore, we choose cosine similarity, Soft_SO, Soft_Google, and Soft_Lev, as the final feature set in the SOFTSVM because they perform better than other features.
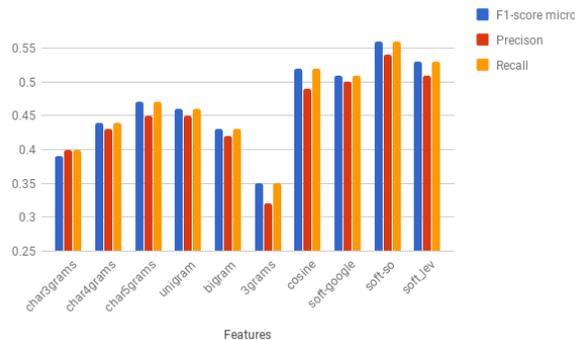

Figure 5: Performance of SVM models using individual features

To compare and select the important text selection parts, we build the SVM model by considering only title, body or answers. As shown in Table 3, the model with different parts perform similarly and the best performance is achieved when we consider all three, title, body and answers.

Table 3: Results of choosing different text selections.

Text selection/metrics	F-micro	Precision	Recall
Title	0.47	0.44	0.48
Body	0.51	0.49	0.51
Answers	0.51	0.5	0.51
Title, Body, Answers	0.59	0.58	0.59

Results and Discussion

Analysis of Results   Table 4 compares results for both SOFTSVM and DOTBILSTM on Stack Overflow dataset. Comparing the obtained results, we realize that DOTBILSTM substantially outperforms SOFTSVM by more than 16 absolute percentage point in F-micro. This suggests that the BiLSTM model can utilize the large amount of training data in Stack Overflow dataset and predict the relatedness between knowledge units more effectively than our traditional model.

Table 4: Results for SOFTSVM and DOTBiLSTM models

Model/Metrics	F-micro	Precision	Recall
SOFTSVM	0.59	0.58	0.59
DOTBiLSTM	0.75	0.75	0.75

Tables 5 shows F-micro scores for predicting individual classes. Comparing results of the individual classes, DOT-BiLSTM performs better than SOFTSVM in predicting Isolated, Duplicate and Indirect classes.

Table 5: Comparing the results (f-score) of SOFTSVM and DOT-BiLSTM models

Models/Classes	Duplicate	Direct	Indirect	Isolated	Overall: Micro
SOFTSVM	0.53	0.57	0.44	0.79	0.59
DOTBiLSTM	0.92	0.55	0.67	0.87	0.75

Reformulating the problem to the binary format of Duplicate Detection*:* For having a better comparison between our task and other typical duplicate/non-duplicate classification studies (some mentioned in "Related Work"), we *reformulate* the task to Duplicate Question Detection (DQD) and report the results of our models in the 2-class scenario. DQD is to predict if two given knowledge units are either duplicate or non-duplicate. To evaluate the models under the DQD scenario, we need to map four relatedness classes into two Duplicate and Not-duplicate classes. We consider duplicate class from the original dataset as *duplicate* and the rest as *non-duplicates* instances. To address the imbalanced class problem, we apply under-sampling techniques for non-duplicate class. More precisely, we randomly choose instances from all other three classes (direct, indirect and isolated) to have an equal number of both classes. By reformulating the task from multi-class to binary classification task, we expect our models to achieve higher results. We evaluate both DOTBiLSTM and SOFTSVM models using the reformulated dataset. DOTBiLSTM and SOFTSVM prediction performances increase to 0.91 and 0.70 f-score respectively. As we expected, by having two classes instead of four, in a relatively simpler problem, DOTBiLSTM and SOFTSVM results increase by 16% percent and 11% respectively.

Comparing with AskUbuntu Dataset: We take a further step and expand our work by investigating AskUbutu DQD dataset for two specific reasons: (1) to show the robustness of the used models, and (2) To show the challenging nature of the proposed dataset on Stack Overflow compared to others. We expect to observe a different behavior of our models on this data due to the different nature and structure. For example, unlike Stack Overflow, the inputs of AskUbuntu dataset are only limited to title+body of each question. Moreover, AskUbuntu data contains a fewer number of instances, that is 24K pairs for training, 6K for testing and 1K for validation part. We use the cleaned version of AskUbuntu dataset (without signal) prepared by (Rodrigues et al. 2017). Using the same splitting used in (Rodrigues et al. 2017), our both models perform similarly. DOTBiLSTM model achieves 0.88 f-score and 0.87 accuracy, and SOFTSVM model achieves 0.90 f-score and 0.90 accuracy.

Our models outperform the state-of-the-art *Hybrid DCNN* model on this dataset introduced in (Rodrigues et al. 2017) with the accuracy of 0.79. This shows that not only our lightweight BiLSTM and traditional SVM model perform well in the Stack Overflow dataset, these models also outperform the complex *Hybrid DCNN* model on AskUbuntu dataset. Note that in order to evaluate the models on this dataset, we need to customize models to only have 2 inputs (title+body pairs).

More To Explore: In our experiments, we purposely confined our models to only utilize information in *Title*, *Body* and *Answers*. However, relying on other parts of the dataset like *BestAnswer*, *Tags* and *Code parts* can boost performance further for this task. As future work, we intend to investigate to utilize code parts as they are considered as informative resources about the content of the knowledge units.

Conclusion

This paper presents the task along with a large-scale dataset for identifying relatedness of knowledge unit (question thread) pairs in Stack Overflow. We reported all the steps for creating this dataset and a user study to evaluate the quality of the dataset. We devised two models, DOTBiLSTM and SOFTSVM for this task and their performances for future evaluations. We also compared the performance of DOT-BiLSTM and SOFTSVM models with the state-of-the-art model on AskUbuntu dataset and found that these models outperform the state-of-the-art model. We made the dataset and models available online.

References

[Afzal, Wang, and Liu 2016] Afzal, N.; Wang, Y.; and Liu, H. 2016. Mayonlp at semeval-2016 task 1: Semantic textual similarity based on lexical semantic net and deep learning semantic model. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*, 674–679.

[Bogdanova et al. 2015] Bogdanova, D.; dos Santos, C.; Barbosa, L.; and Zadrozny, B. 2015. Detecting semantically equivalent questions in online user forums. In *Proceedings of the Nineteenth Conference on Computational Natural Language Learning*, 123–131.

[Chan et al. 2012] Chan, W.; Zhou, X.; Wang, W.; and Chua, T.-S. 2012. Community answer summarization for multisentence question with group l 1 regularization. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers-Volume 1*, 582–591. Association for Computational Linguistics.

[Charlet and Damnati 2017] Charlet, D., and Damnati, G. 2017. Simbow at semeval-2017 task 3: Soft-cosine semantic similarity between questions for community question answering. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, 315–319.

[Demner-Fushman and Lin 2006] Demner-Fushman, D., and Lin, J. 2006. Answer extraction, semantic clustering,

and extractive summarization for clinical question answering. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics*, 841–848. Association for Computational Linguistics.

[Filice, Da San Martino, and Moschitti 2017] Filice, S.; Da San Martino, G.; and Moschitti, A. 2017. Kelp at semeval-2017 task 3: Learning pairwise patterns in community question answering. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, 326–333.

[Galbraith, Pratap, and Shank 2017] Galbraith, B.; Pratap, B.; and Shank, D. 2017. Talla at semeval-2017 task 3: Identifying similar questions through paraphrase detection. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, 375–379.

[Goyal 2017] Goyal, N. 2017. Learningtoquestion at semeval 2017 task 3: Ranking similar questions by learning to rank using rich features. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, 310–314.

[Hochreiter and Schmidhuber 1997] Hochreiter, S., and Schmidhuber, J. 1997. Long short-term memory. *Neural computation* 9(8):1735–1780.

[Kenter and De Rijke 2015] Kenter, T., and De Rijke, M. 2015. Short text similarity with word embeddings. In *Proceedings of the 24th ACM international on conference on information and knowledge management*, 1411–1420. ACM.

[Kingma and Ba 2014] Kingma, D. P., and Ba, J. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

[Levy, Goldberg, and Dagan 2015] Levy, O.; Goldberg, Y.; and Dagan, I. 2015. Improving distributional similarity with lessons learned from word embeddings. *Transactions of the Association for Computational Linguistics* 3:211–225.

[Liu et al. 2008] Liu, Y.; Li, S.; Cao, Y.; Lin, C.-Y.; Han, D.; and Yu, Y. 2008. Understanding and summarizing answers in community-based question answering services. In *Proceedings of the 22nd International Conference on Computational Linguistics-Volume 1*, 497–504. Association for Computational Linguistics.

[Mamykina et al. 2011] Mamykina, L.; Manoim, B.; Mittal, M.; Hripcsak, G.; and Hartmann, B. 2011. Design lessons from the fastest q&a site in the west. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '11, 2857–2866. New York, NY, USA: ACM.

[Mikolov et al. 2013a] Mikolov, T.; Chen, K.; Corrado, G.; and Dean, J. 2013a. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

[Mikolov et al. 2013b] Mikolov, T.; Sutskever, I.; Chen, K.; Corrado, G. S.; and Dean, J. 2013b. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, 3111–3119.

[Nakov et al. 2017] Nakov, P.; Hoogeveen, D.; Màrquez, L.; Moschitti, A.; Mubarak, H.; Baldwin, T.; and Verspoor, K. 2017. Semeval-2017 task 3: Community question answering. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, 27–48.

[Pedregosa et al. 2011] Pedregosa, F.; Varoquaux, G.; Gramfort, A.; Michel, V.; Thirion, B.; Grisel, O.; Blondel, M.; Prettenhofer, P.; Weiss, R.; Dubourg, V.; et al. 2011. Scikit-learn: Machine learning in python. *Journal of machine learning research* 12(Oct):2825–2830.

[Pennington, Socher, and Manning 2014] Pennington, J.; Socher, R.; and Manning, C. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, 1532–1543.

[Rodrigues et al. 2017] Rodrigues, J. A.; Saedi, C.; Maraev, V.; Silva, J.; and Branco, A. 2017. Ways of asking and replying in duplicate question detection. In *Proceedings of the 6th Joint Conference on Lexical and Computational Semantics (* SEM 2017)*, 262–270.

[Shen et al. 2015] Shen, Y.; Rong, W.; Sun, Z.; Ouyang, Y.; and Xiong, Z. 2015. Question/answer matching for cqa system via combining lexical and sequential information. In *AAAI*, 275–281.

[Sidorov et al. 2014] Sidorov, G.; Gelbukh, A.; Gómez-Adorno, H.; and Pinto, D. 2014. Soft similarity and soft cosine measure: Similarity of features in vector space model. *Computación y Sistemas* 18(3):491–504.

[Silva et al. 2018] Silva, J.; Rodrigues, J.; Maraev, V.; Saedi, C.; and Branco, A. 2018. A 20% jump in duplicate question detection accuracy? replicating ibm teams experiment and finding problems in its data preparation. *META* 20(4k):1k.

[StackOverflow 2018] StackOverflow. 2018. How to ask a good question?, `http://stackoverflow.com/help/how-to-ask`.

[Tan et al. 2016] Tan, M.; dos Santos, C.; Xiang, B.; and Zhou, B. 2016. Improved representation learning for question answer matching. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, 464–473.

[Wu, Zhang, and Huang 2011] Wu, Y.; Zhang, Q.; and Huang, X. 2011. Efficient near-duplicate detection for q&a forum. In *Proceedings of 5th International Joint Conference on Natural Language Processing*, 1001–1009.

[Xu et al. 2016] Xu, B.; Ye, D.; Xing, Z.; Xia, X.; Chen, G.; and Li, S. 2016. Predicting semantically linkable knowledge in developer online forums via convolutional neural network. In *Proceedings of the 31st IEEE/ACM International Conference on Automated Software Engineering*, 51–62. ACM.

[Xu et al. 2017] Xu, B.; Xing, Z.; Xia, X.; and Lo, D. 2017. Answerbot: automated generation of answer summary to developersź technical questions. In *Proceedings of the 32nd IEEE/ACM International Conference on Automated Software Engineering*, 706–716. IEEE Press.

[Ye, Xing, and Kapre 2016] Ye, D.; Xing, Z.; and Kapre, N. 2016. The structure and dynamics of knowledge network in

domain-specific q&a sites: a case study of stack overflow. *Empirical Software Engineering*.

[Zhang et al. 2015] Zhang, Y.; Lo, D.; Xia, X.; and Sun, J.-L. 2015. Multi-factor duplicate question detection in stack overflow. *Journal of Computer Science and Technology* 30(5):981–997.