경영학 석사학위논문

Identifying Semantically Similar Questions in Social Q&A Communities

소셜 질의응답 커뮤니티에서

유사한 질문 추출에 관한 연구

2018년 2월

서울대학교 대학원

경영학과 경영학 전공

김 범 수

Identifying Semantically Similar Questions in Social Q&A Communities

지도교수 박 진 수

이 논문을 경영학 석사학위논문으로 제출함

2017년 12월

서울대학교 대학원

경영학과 경영정보 전공

김 범 수

김범수의 석사학위논문을 인준함

2017년 12월

위 원 장	**(인)**
부위원장	**(인)**
위　　원	**(인)**

Abstract

Identifying Semantically Similar Questions in Social Q&A Communities

Buomsoo Kim

Management Information System

The Graduate School of Business, Seoul National University

SQA communities are an impressive instance of knowledge sharing over the Web. A tremendous number of questions are asked and answered every minute in prospering SQA communities such as Yahoo! Answers, Stack Exchange network, and Quora. However, it could be observed that a large proportion of the new questions are redundant, with a semantically similar counterpart existing in the database. There exist few thorny challenges regarding identifying semantically equivalent questions in SQA communities: (1) semantically similar questions could be rather dissimilar in terms of syntax and lexicon, (2) obtaining reliable training and test datasets is troublesome, (3) the influence of domain- or context-specific languages, and (4) severe class imbalance problem could seriously hamper the identification process. We suggest a data-driven framework that could overcome such challenges and complement existing models. Our work takes multi-disciplinary approach in building the framework, borrowing concepts and techniques from machine learning, natural language processing (NLP), deep learning, information retrieval, and etc. Our final model utilizing Word2Vec and convolutional neural networks for language modeling shows desirable level of performance, test accuracy of 0.975478 and average precision of 0.983501.

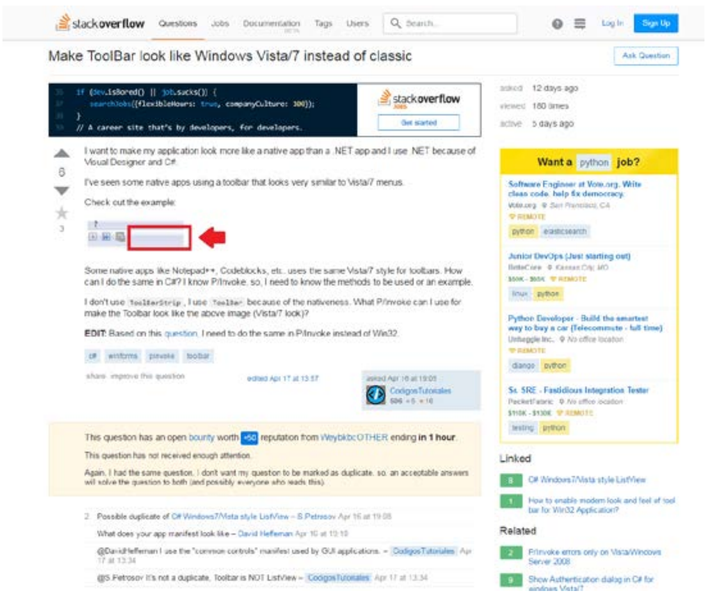
Keywords: Q&A, online communities, collective intelligence, wisdom of crowds, language modeling, word2vec, convolutional neural networks, deep learning

Student Number: 2016-20545

Table of Contents

CHAPTER 1 Introduction

The advancement of Web 2.0 has enabled extensive knowledge sharing activities among users. Especially, Social Question & Answer (SQA) communities are an impressive case of collective intelligence in which participants interact with each other vigorously. As the collective intelligence of SQA communities matures and diversifies, more users are relying on SQA services, rather than depending solely on search engines to fulfill their information needs (John et al 2016). Currently, there are many thriving SQA communities such as Yahoo! Answers, Stack Exchange network, and Quora. In such communities, a myriad of questions are asked every minute and many users endeavor to provide helpful answers. . Figure 1 exhibits one of the questions in Stack Overflow, the biggest SQA community of the Stack Exchange network. User interfaces of other SQA communities are largely similar to that of Figure 1.



Make ToolBar look like Windows Vista/7 instead of classic

```
if (dev.isBored() || job.sucks()) {
    searchJobs({flexibleHours: true, companyCulture: 100});
}
// A career site that's by developers, for developers.
```

I want to make my application look more like a native app than a .NET app and I use .NET because of Visual Designer and C#.

I've seen some native apps using a toolbar that looks very similar to Vista/7 menus.

Check out the example:

Some native apps like Notepad++, Codeblocks, etc. uses the same Vista/7 style for toolbars. How can I do the same in C#? I know P/Invoke, so, I need to know the methods to be used or an example.

I don't use `ToolStrip`, I use `Toolbar` because of the nativeness. What P/Invoke can I use for make the Toolbar look like the above image (Vista/7 look)?

EDIT: Based on this question, I need to do the same in P/Invoke instead of Win32.

c# winforms pinvoke toolbar

share improve this question edited Apr 17 at 13:57 asked Apr 16 at 19:05 CodigosTutoriales 506 •5 •16

This question has an open bounty worth +50 reputation from WeybkbcOTHER ending **in 1 hour**.

This question has not received enough attention.

Again, I had the same question. I dont want my question to be marked as duplicate, so, an acceptable answers will solve the question to both (and possibly everyone who reads this).

2 Possible duplicate of C# Windows7/Vista style ListView – S.Petrosov Apr 16 at 19:08

What does your app manifest look like – David Heffernan Apr 16 at 19:10

@DavidHeffernan I use the "common controls" manifest used by GUI applications. – CodigosTutoriales Apr 17 at 13:34

@S.Petrosov It's not a duplicate, Toolbar is NOT ListView – CodigosTutoriales Apr 17 at 13:34

asked 12 days ago
viewed 180 times
active 5 days ago

Want a python job?

Software Engineer at Vote.org. Write clean code, help fix democracy.
Vote.org San Francisco, CA
REMOTE
python elasticsearch

Junior DevOps (Just starting out)
BetaCore Kansas City, MO
$50K - $65K REMOTE
linux python

Python Developer - Build the smartest way to buy a car (Telecommute - full time)
Unhaggle Inc. No office location
REMOTE
django python

Sr. SRE - Fastidious Integration Tester
PacketFabric No office location
$110K - $130K REMOTE
testing python

Linked

8 C# Windows7/Vista style ListView

1 How to enable modern look and feel of tool bar for Win32 Application?

Related

2 P/Invoke errors only on Vista/Windows Server 2008

9 Show Authentication dialog in C# for windows Vista/7

Figure 1. Question in Stack Overflow community

As of April 2017, over 8,000 questions are asked in the Stack Overflow community on a daily basis (Stack Exchange 2017). The sheer number of new questions could imply that the community is blooming. However, it might also imply that some questions are asked without prior search efforts, creating redundant questions. For instance, if one searches “Python cosine similarity” at the Stack Overflow website, a number of semantically equivalent questions could be observed (Table 1).

Table 1. Partial query result of "Python cosine similarity" in Stack Overflow SQA community

Title	Date
Cosine Similarity [Python]	Mar 28, 2014
How to calculate cosine similarity given 2 sentence Strings? - Python	Mar 2, 2013
Computing cosine similarity using Python	Feb 1, 2017
Calculate cosine similarity of two matrices - Python	Feb 24, 2014
Vectorized cosine similarity calculation in Python	Dec 3, 2015
Finding cosine similarity between 2 numbered datasets using python	Aug 17, 2014

Such questions not only hamper searching attempts, but also lead to the dispersion of group intelligence. Surowiecki (2005) has contended that one of the four preconditions for “the wisdom of crowds” is “aggregation,” which assures the existence of certain mechanisms that can alter individual judgments into a collective one. Thus, detecting semantically similar questions and “aggregating” their corresponding contents is a critical issue in enhancing collective intelligence of a SQA community.

Nevertheless, identifying similar questions remains a challenging problem despite constant research efforts (John et al 2016; Zhou et al 2015). One of the major challenges is related to the “lexico-syntactic gap,” which makes semantically equivalent questions syntactically and lexically unlike (Das et al 2016). Recently, a few related works have attempted to resolve this problem with neural network language models, Word2Vec (Chahuara et al 2016; Zhou et al 2015). Word2Vec, proposed by Mikolov et al (2013a), is an efficient way to estimate word representations in a finite vector space. Moving slightly forward, we have implemented a Doc2Vec model, an extension of Word2Vec, proposed by Le and Mikolov (2014). Besides, we have created a prototypical model based on convolutional neural networks (CNN). According to Kim (2014), CNNs with word embedding are useful model structure for classifying sentences.

Another challenge relates to labeling training and test data. Even though it is relatively easy to obtain SQA-related datasets, there still exist difficulties in labeling them, i.e., classifying question pairs into similar and dissimilar categories. Literature review has revealed that most previous studies have resorted to a small number of human annotators. However, it is suggested that this is a highly costly procedure requiring considerable amount of time and human effort. Furthermore, due to fatigue from a lengthy process and the lack of domain-specific knowledge, human judges are prone to make cognitive errors (Lorist et al 2005). Thus, we rely on collective intelligence of SQA community users rather than human annotators.

Finally, domain-specific languages make the identification of the meaning of questions highly complicated and demanding. In contemporary English, there exist a number of words that convey different meanings depending on the context and domain. Furthermore, sometimes people call the same concept with different wordings. For instance, according to Merriam Webster's, the word derivative has a distinct meaning in each area of linguistics, mathematics, chemistry, and finance – for details, refer to Table 2 (Derivative 2017). Hence, we propose a generic framework that can be used to match similar questions and merge them in any pre-specified domain level.

Table 2. Diverse meanings of the word "derivative"

Domain	Meaning
General	Something derived
Linguistics	A word formed from another word or base: a word formed by derivation
Mathematics	The limit of the ratio of the change in a function to the corresponding change in its independent variable as the latter change approaches zero
Chemistry	A chemical substance related structurally to another substance and theoretically derivable from it
	A substance that can be made from another substance
Finance	A contract or security that derives its value from that of an underlying asset (such as another security) or from the value of a

	rate (as of interest or currency exchange) or index of asset value (such as a stock index)

Empirical evaluation results with real-world datasets from Stack Exchange SQA communities have revealed that our models demonstrate a practical level of performance in finding semantically equivalent questions, irrespective of the domain of interest. Our best model (model 5B) shows test accuracy of 0.925, precision of 0.95778, recall of 0.889366, and F-score of 0.922274.

Furthermore, it is confirmed that CNN-based text classification model proposed in Kim (2014) is effective even with cross-domain identifications and under severe class imbalance problems. By building a global model using CNNs, we were able to boost the performance. Whereas the global model based on Doc2Vec resulted in classification accuracy of merely 0.507, our CNN based deep learning model was able to achieve over 0.97 of classification accuracy and over 0.98 of average precision under class balance circumstances.

The rest of the article is organized as follows. In next section, we review related works. In the following section, our overall framework and methodologies are explained. Then, the results and findings are presented. In the final section, we conclude our study by summarizing the contributions and limitations of our study.

CHAPTER 2 Related Works

There exist a number of prior attempts to identify duplicate questions in SQA communities. Translation models with basic similarity measures were widely employed at the outset. Jeon et al (2005) have set up a foundation for the translation model, which was initially proposed to support machine translation, for instance, from French to English. In such models, the similarity between different questions is equivalent to the probability of translating one question into another one. In the paper, "the IBM model 1," which does not require prior linguistic knowledge is adopted for its simplicity (Brown et al 1993). In the experiment, as the source and target languages are the same (Korean), word translation probabilities can be interpreted as "semantic similarities of words."

The translation probability (i.e., semantic similarity) from word *s* (source word) to word *t* (target word) is:

$$P(t|s) = \lambda_s^{-1} \sum_{i=1}^{N} c\left(t\middle|s;J^i\right)$$

in which λ_s is a normalization factor that makes the sum of probabilities equal to 1, *N* refers to the number of training instances, and J^i is the *i*th pair in the training set. In each J^i, there are two sentences – a source sentence and a target sentence. So, what the model calculates is the semantic proximity of words in source sentence and words in target sentence in each data instance. Finally,

$$c\left(t\middle|s;J^i\right) = \frac{P(t|s)}{P(t|s_1) + \cdots + P(t|s_n)} \#(t,J^i)\#(s,J^i)$$

where $\{s_1, s_2, \ldots, s_n\}$ are words in the source sentence in J^i and $\#(\mathrm{t}, J^i)$ is the number of occurrence of the word t in J^i. Table 3 is a partial result of such computations; the first row (header) demonstrates the source words and below rows should semantically similar words to them, ranked based on similarity (Jeon et al 2015). Their model shows comparable result to those of other approaches - mean average precision of 0.314.

Table 3. Similar Words to Keywords

Word / Rank	**music**	**intel**	**excel**	**font**	**watch**
1	music	pentium	excel	font	watch
2	file	4	korean	korean	time
3	tag	celeron	function	97	background
4	sound	amd	novice	add	start
5	background	intel	cell	download	date

Jeon et al (2005) have obtained dataset from Naver KnowledgeIN archive, which is the biggest SQA community service in South Korea. In generating training samples, they see the corresponding answers to questions. Their bold assumption is that if answers to questions are similar, the questions should be semantically similar as well. However, this could be somewhat problematic since an answerer might have interpreted a question erroneously and gave an answer unrelated to the question. Furthermore, if there are multiple answers to a single question, a measure to deal with conflict between answers should be addressed.

Xue et al (2008) adopted the basic algorithm of translation model used in Jeon et al (2006), IBM Translation Model 1. However, they extend the approach by taking into account both question part and answer part. In other words, word to word translation probabilities between both question-answer pairs, $P(Q|A)$, and answer-question pairs, $P(A|Q)$, are utilized. Both probabilities are computed and combined to generated pooled probabilities. Table 4 is the partial result of querying similar words using both probability measures.

Table 4. Partial Result of Querying Similar Words

source word	**everest**			**xp**		
probability used	$P(A\|Q)$	$P(Q\|A)$	Pooled	$P(A\|Q)$	$P(Q\|A)$	Pooled
1	everest	mountain	everest	xp	xp	xp
2	29,035	tallest	mountain	drive	window	window
3	ft	everest	tallest	install	computer	install
4	mount	highest	29,035	click	system	drive
5	8,850	mt	highest	system	pc	computer

Lee et al (2008) have proposed improved translation model, namely "compact translation model," from IBM model 1 used in Jeon et al (2005). One of the problems of IBM translation model 1 is that it solely depends on word co-occurrence statistics between source and target sentences. Hence, as it cannot take into account the effect of "context," unimportant words such as stopwords are comprised in interpretation, creating a great noise. In compact translation model, two different term weighting strategies, Term Frequency-Inverse Document Frequency (TF-IDF) and TextRank schemes are employed.

TF-IDF is a classical method to weight terms based on their relative status in documents (Salton and Buckley 1988). To calculate the weight of word *w* in document *D*, two measures, term frequency (*tf*) and inverse document frequency (*idf*) are employed.

$$tf_{w,D} = \frac{frequency_{w,D}}{|D|}$$

$$idf_w = \log(\frac{|C|}{df_w})$$

where $frequency_{w,D}$ is the number of occurrences of *w* in *D* (i.e., how many times *w* occurs in *D*). |*D*| and |*C*| refers to the size of document *D* (i.e., how many unique words are comprised in *D*) and the size of document collection (i.e., how many documents are there, in total). Lastly, df_w is the number of documents in which *w* occurs.

TF-IDF weight of word w in D ($tf - idf_{w,D}$) is simply a product of $tf_{w,D}$ and idf_w. Simply put, words with high TF-IDF weights are regarded as important and low weights less important.

$$tf - idf_{w,D} = tf_{w,D} * idf_w$$

TextRank is an adoption of Google's PageRank algorithm – a graph-based ranking model for keyword generation. Words in entire corpora constitute vertices in graph and the number of co-occurrence between each word pair become the weight of the edge. At the outset, the scores of vertices are initialized as 1, and the PageRank algorithm is applied until convergence. Major assumption of TextRank algorithm is that a word with importance co-occurs frequently with other words with importance in corpora. Hence, under such assumption, it could be said that words with high TextRank score has crucial importance.

Xue et al (2008) and Lee et al (2008) performed experiment using external archive of SQA communities, respectively Wondir and Yahoo! Answers. Both models in Xue et al (2008) and Lee et al (2008) seem to outperform Jeon et al (2005)'s naïve translation model.

As mentioned above, considering all words in a question with equal importance would be inefficient and ineffective; it would be desirable to attend to certain class of words that have distinctive status in corpora. With recent developments in NLP and machine learning, various methodologies and models were suggested to perform such functionality. Zhang et al (2014), Das et al (2015), and Chahuara et al (2016) have taken topic modeling approach.

Topic models attempt to find certain themes, i.e., "topics," from a text corpus by inspecting hidden structures in the corpus. Latent Dirichlet Allocation (LDA), one of the simplest and most popular topic modeling methods, is a generative model that infers probability about each observation in training data

(Blei 2012).

Intuitively, not only different words are “highlighted” in different documents, but also different classes of words exist in a single document as well. For instance, words about *data analysis*, such as “computer” and “prediction” and words about *evolutionary biology*, such as “life” and “organism” can co-exist in an article. Here, *data analysis* and *evolutionary biology* can be regarded as “topics” of the article. A topic is formally defined as “a distribution over a fixed vocabulary.” The combination of topics in a corpus leads to guess the overall characteristic of a document – in this case, scientific one. In short, LDA is a language model that attempts to capture and formalize such intuition (Blei 2012).

LDA is a generative probabilistic model that regards data as arising from a generative process with *hidden variables*. Further, *joint probability distribution* over the observed and hidden random variables is defined. Mathematically defined, the generative process can be represented as below:

$$p(\beta_{1:K}, \theta_{1:D}, z_{1:D}, w_{1:D}) \\ = \prod_{i=1}^{K} p(\beta_i) \prod_{d=1}^{D} p(\theta_d) \ \ (\prod_{n=1}^{N} p(z_{d,n}|\theta_d) p(w_{d,n}|\beta_{1:K}, z_{d,n}))$$

where $\beta_{1:K}$ are topics (each β_k distribution over the vocabulary) and topic distribution for the *d*th document is θ_d ($\theta_{d,k}$ topic distribution for topic *k* in document *d*). z_d is the topic assignments for the *d*th document ($z_{d,n}$ topic assignment for the *n*th word in *d*th document) and w_d is the observed words for document *d* ($w_{d,n}$ the *n*th word in *d*th document). From such distribution, hidden topic structure of a document is inferred.

Zhang et al (2014) have attempted to induce semantic similarity between query and question by taking topic modeling approach. Based on features extracted from topic models, clustering is performed and similar questions are filtered. Approaches taken by Das et al (2015) and Chahuara et al (2016) are in similar vein. Das et al (2015) have proposed Deep Structured Topic Model (DSTM), a novel process that comprises two steps – retrieving similar questions in latent topic vector space and re-ranking them with a deep layered semantic model. Chahuara et al (2016) have combined topic modeling and multinomial regression. On top of the topic model, multinomial nonlinear regression is performed to retrieve and rank similar questions. The instantiation is evaluated on Yahoo! Answers dataset, relying on human annotators. Their results are reported to perform better than benchmark models employing translation models or basic ranking algorithms.

Finally, Zhou et al (2015) and Wang and Poupart (2016) have proposed approaches based on neural network language models, Word2Vec. Such models convert each word into a vector, enabling arithmetic operations and topological analysis among word vectors. Such vector space models not only reduce the dimensionality of input space dramatically, but also enable algebraic operations between word vectors based on their semantic and syntactic similarities. For instance, relationships such as *vector(“King”) – vector(“Man”) + vector(“Woman”) = vector(“Queen”), vector(“Madrid”) – vector(“Spain”) + vector(“France”) = vector(“Paris”),* and *vector(“apple”) – vector(“apples”) = vector(“car”) – vector(“apple”)* can be deduced (Mikolov et al 2013a, Mikolov et al 2013b).

There are two variations of Word2Vec model that are commonly used -

the Continuous Skip-gram (CS) and the Continuous Bag-Of-Words (CBOW). The CS model attempts to predict the distribution of neighboring words (i.e., window) using center word, while the CBOW model attempts to predict a center word using neighboring words (Mikolov et al 2013a, Mikolov et al 2013b). However, in practice, both models show no significant difference in terms of performance.

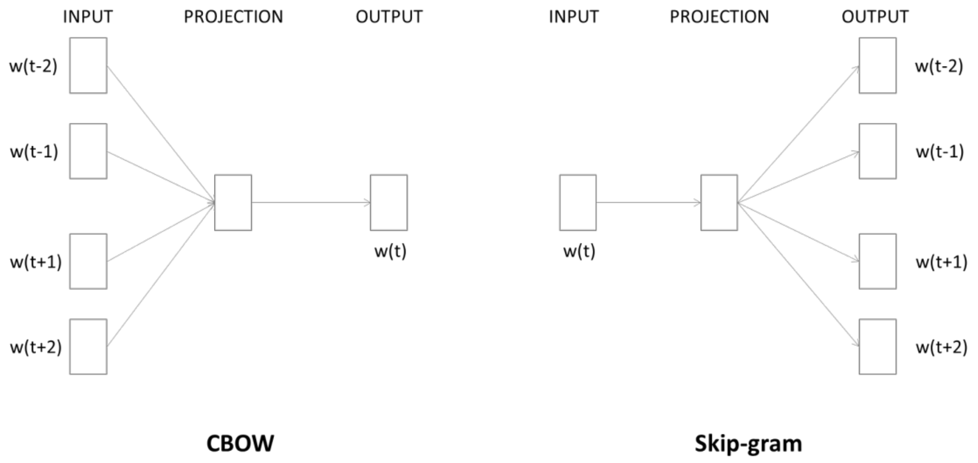


Figure 2. Architectures of CBOW and CS models

Both models resemble a shallow neural network with a single hidden layer (projection layer). Weights of hidden layer are randomly initialized and continuously updated using back propagation and such weights are used to infer vector representation of each word in corpus (Mikolov et al 2013a, Mikolov et al 2013b). Figure 2 is an outline of model architectures of CBOW and CS models (Adapted from Mikolov et al 2013a).

Zhou et al (2015) have created a CS model with metadata of category information, called M-NET. Simply put, on top of word embedding model, category information of each word is appended to help learning. For instance, in a question "What are the security issues with *java*?" under the category "Computers & Internet → Security," the corresponding category of word java also becomes "Computers & Internet → Security." Then, words that belong to similar category would have similar vector representations in embedding space. For instance, words Java and Python are likely to be more proximate than words Java and French in vector space.

Wang and Poupart (2016) also adopted the CS model, but with different approach in terms of representing questions as vectors. After they trained Word2Vec model, they have performed TF-IDF computation to weight-average word vectors. Most questions in corpus have different number of words that have finite-dimensional vector representation. So one simple and intuitive way to get vector representation for each question would be averaging all word vectors element-wise. However, one problem with such approach would be encompassing too much information; as mentioned above, unimportant words would be counted as same as important words. Hence, a fix to this problem that Wang and Poupart (2016) adopted was to generate weights (i.e., relative importance) of words in each question and weight-average them to take into account more information.

As generating and classifying training and test datasets are an important issue, we also surveyed labeling methods. Most prior studies relied on a small number of human annotators deciding whether two distinct questions are semantically equivalent (Song et al 2007, Lee et al 2008, Xue et al 2008, Wang et al 2009, Hao and Agichtein 2012, Zhang et al 2014, Das et al 2015, Zhou et al 2015, John et al 2016, Chahuara et al 2016). However, this point might constitute limitations for such studies. Manually labeling data is immensely time-consuming and laborious. Thus, it is highly costly and it might lead to human error arising from fatigue and boredom of menial work, leaving some room for improvement (Lorist et al 2005). Table 5 is the summary of related works.

Table 5. Summary of Related Works

Author(s)	**Methodology**	**Dataset**	**Labeling**
Jeon et al (2005)	Translation model	Naver KnowledgeIN archive	Ranking algorithm (LM-HRANK measure)
Song et al. (2007)	Similarity measures	FAQ system archive	Human Annotator
Achananuparp et al. (2008)	Similarity measures	Sample corpus (TREC-9)	Unspecified
Lee et al (2008)	Translation model	Yahoo! Answers Archive	Human Annotator
Xue et al (2008)	Translation model	Wondir archive, Sample corpus (TREC-9)	Human Annotator
Wang et al (2009)	Syntactic tree matching	Yahoo! Answers Archive	Human Annotator
Hao and Agichtein	Equivalent pattern	Yahoo! Answers	Human Annotator

(2012)	learning	Archive	
Shtok et al (2012)	Query performance prediction	Yahoo! Answers Archive	Crowdsourcing (Amazon Mechanical Turk)
Zhang et al (2014)	Topic Modeling	Yahoo! Answers archive, Sample corpus, Twitter dataset	Human Annotator
Das et al (2015)	Topic Modeling	Yahoo! Answers Archive	Ranking algorithm (BM25), Human judge
Han et al (2015)	Rule-based approach	Sample corpus (ICHI 2015)	Unspecified
Zhou et al (2015)	Neural network language model	Yahoo! Answers, Baidu Zhidao Archive	Human Annotator
John et al (2016)	Graph-based cluster analysis	Yahoo! Answers Archive	Human Annotator
Chahuara et al (2016)	Topic modeling	Yahoo! Answers, Stack Exchange Archive	Human Annotator
Hoogeveen et al (2016)	Machine learning	Sample corpus	Unspecified
Wang and Poupart (2016)	Neural network language model	Sample corpus	Unspecified

CHAPTER 3 Methodology

In order to create a classification model, we employ techniques from diverse disciplines, including natural language processing, machine learning, and information retrieval. Our proposed methodology comprises four steps: (1) Data collection and preprocessing, (2) Language Modeling, (3) Identification (classification), and (4) Evaluation. Our overall framework is summarized in Table 6 and Figure 3 is the bird eye's view of our model. Visualization of our model using *Keras* library in Python is provided in Appendix 1.

Table 6. Overall Framework

Steps	**Disciplines referred to**	**Techniques employed**
Data Preprocessing	· Natural Language Processing	· Tokenization · Lemmatization · Stopwords removal
Language Modeling	· Neural language modeling	· Word2Vec · Doc2Vec
Identification (Classification)	· Machine learning · Deep learning	· Logistic regression classifier · Deep learning (convolutional neural networks)
Model Selection & Evaluation	· Information retrieval · Machine learning	· Evaluation metrics (accuracy/precision/recall/F-

		score/average precision)

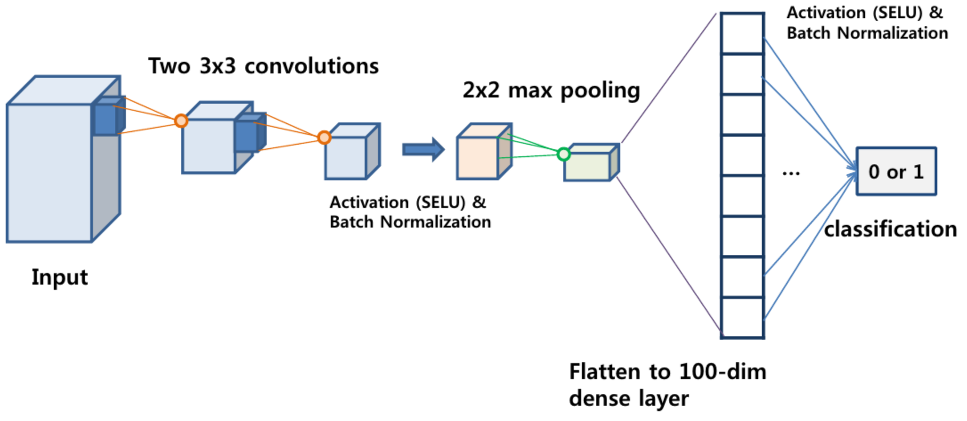
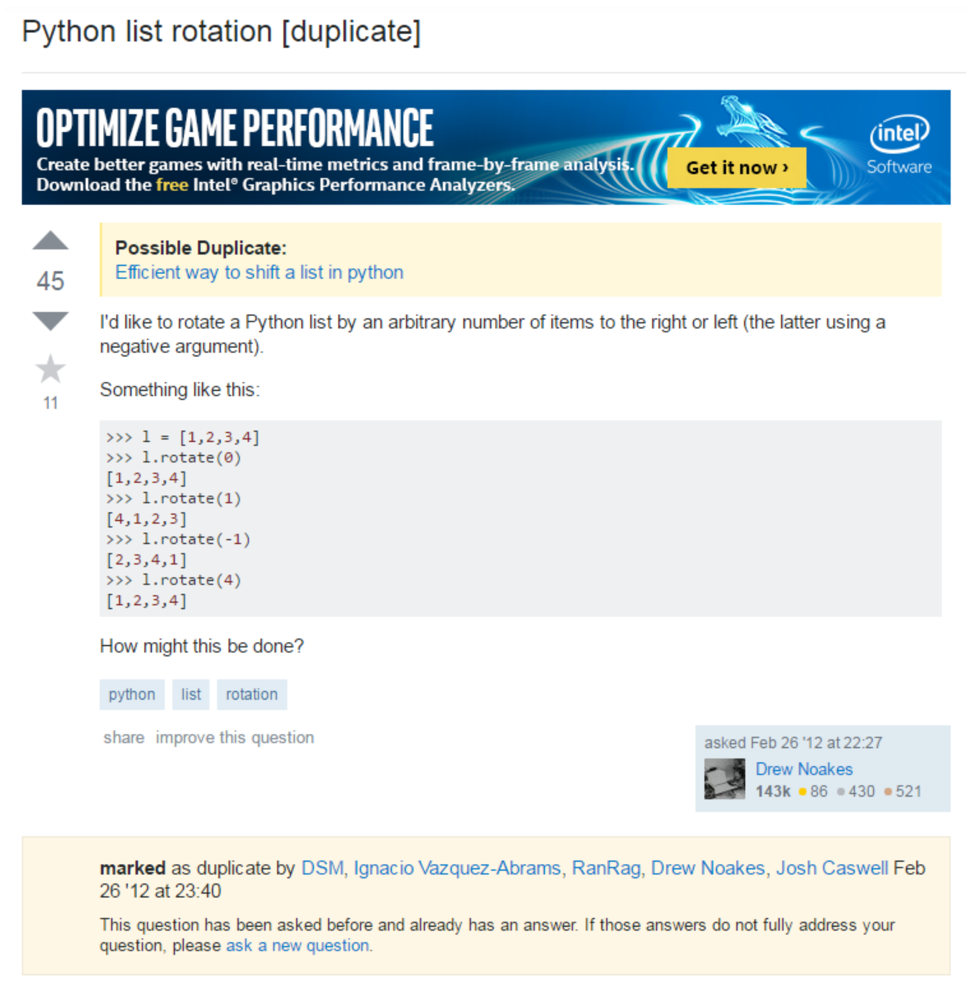


Figure 3. Bird Eye's View of Final Model (CNN)

3.1 Data Collection & Preprocessing

As briefly mentioned above, obtaining a reliable dataset for training is a nontrivial issue in identifying similar questions in SQA community archives. In particular, it is difficult to label the question pairs, classifying them as a match or non-match. We took an alternative approach to obtaining and labeling a dataset: to rely on the collective intelligence of active SQA communities. Some of the questions in Stack Exchange communities are marked as "duplicate." If a questions gains five votes from moderators or users with a certain level of reputation, the question is closed as duplicate and the title is appended with "[Duplicate]" mark. Figure 4 is one of duplicate marked questions in Stack Overflow community.


Python list rotation [duplicate]

OPTIMIZE GAME PERFORMANCE
Create better games with real-time metrics and frame-by-frame analysis.
Download the free Intel® Graphics Performance Analyzers.
Get it now ›
intel Software

45

11

Possible Duplicate:
Efficient way to shift a list in python

I'd like to rotate a Python list by an arbitrary number of items to the right or left (the latter using a negative argument).

Something like this:

```
>>> l = [1,2,3,4]
>>> l.rotate(0)
[1,2,3,4]
>>> l.rotate(1)
[4,1,2,3]
>>> l.rotate(-1)
[2,3,4,1]
>>> l.rotate(4)
[1,2,3,4]
```

How might this be done?

python list rotation

share improve this question

asked Feb 26 '12 at 22:27
Drew Noakes
143k ●86 ●430 ●521

marked as duplicate by DSM, Ignacio Vazquez-Abrams, RanRag, Drew Noakes, Josh Caswell Feb 26 '12 at 23:40

This question has been asked before and already has an answer. If those answers do not fully address your question, please ask a new question.

Figure 4. Example of Duplicate Question

We have chosen eight varied, active communities: Ask Ubuntu, Arqade, Geographic Information Systems, Home Improvement, Super User, Server Fault, TeX-LaTeX, and, Unix & Linux. Ask Ubuntu (https://askubuntu.com/) is a SQA platform for Ubuntu users and developers. It is one of the oldest (7 years and 5 months) and largest (280,000 questions) community in the Stack Exchange network and about 130 questions are asked daily on average. Arqade (https://gaming.stackexchange.com/) is a place for passionate video gamers;

gamers enjoying video games via PC, Playstation, mobile, etc. communicate and socialize in the Arqade community. Geographic Information Systems (https://gis.stackexchange.com/) is a site for devoted cartographers, geographers, and other related professionals. Although it is relatively small in number of users (80,000), it is highly active in light of number of questions (38 daily, 92,000 total) and answers (109,000 total). Home Improvement (https://diy.stackexchange.com/) is a site for contractors and serious DIYers (Do-It-Yourself). Many users who are wanting to renovate their places by themselves exchange ideas and thoughts at the Home Improvement community.

Super User (https://superuser.com/) and Server Fault (https://serverfault.com/) are also two of the oldest (8 years and 5 months and 8 years and 8 months, respectively) and largest (366,000 questions and 254,000 questions, respectively) communities. Lastly, TeX-LaTeX (https://tex.stackexchange.com/) and Unix & Linux (https://unix.stackexchange. com/) are sites specialized for Latex users and Unix/Linux users respectively. There are also great number of committed users to TeX-LaTeX and Unix & Linux communities.

A brief glance at the data revealed that there are a number of duplicate question pairs in Stack Exchange communities – 1,263,520 questions in total and 14,713 questions in total. Thus, we conjectured that it would be sufficient to generate enough training data and test data to learn and evaluate the model. Basic statistics of eight communities of interest are summarized in Table 7.

Table 7. Number of Questions and Duplicate Questions for Each Community

Community (Topic)	Number of Questions	Number of Duplicate Questions
Ask Ubuntu	257173	5531
Arqade	75696	1021
Geographic Information Systems	79194	250
Home Improvement	28973	75
Super User	343033	4398
Server Fault	238764	1320
Tex – Latex	129182	1609
Unix & Linux	111505	509

As our data is in unstructured format, preprocessing is necessary. We have utilized NLP techniques in order to maximize the performance of the word embedding model constructed. After converting all the letters to lowercase, we removed all the stopwords in the text, and tokenized and lemmatized each title and body corpus. Stopwords are routine words in English that has negligible effect on the meaning of a text (Perkins 2010). Examples of stopwords include ‘is’, ‘at’, ‘any’, ‘a’, and ʻdo’. Tokenization involves breaking down the text into indivisible parts, i.e., tokens (Bird and Loper 2006). Lemmatization is the process of finding the canonical form of a word, namely lemma. Usually a set of words share a lemma (Perkins 2010).

Finally, the whole dataset is split into 7 to 3 ratio of training and test dataset. The training set is cross-validated for model selection and hyperparameter tuning. Then the final model is trained using the training set and evaluated with test data.

3.2 Language Modeling

Doc2Vec is a method to retrieve a fixed-dimensional vector representation for each document. The overall framework is largely similar to that of Word2Vec, but paragraph matrix (or document matrix) is added in learning process and each paragraph (document) can be represented in same finite-dimensional space with words. Figure 5 is an abstracted framework for learning word vectors and Figure 6 is one for learning paragraph vectors (Le and Mikolov 2014).

After preprocessing, we have constructed a Doc2Vec model with the question corpus and computed similarities between questions for each domain. As mentioned earlier, we have built domain-specific models to evade the potential problem of domain-specific vocabularies and jargons.

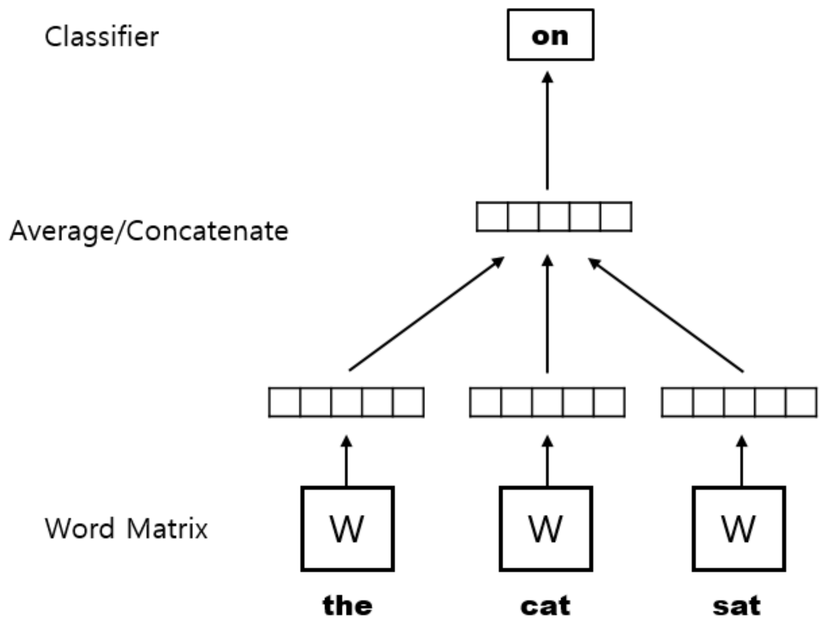


Figure 5. Framework for Learning Word Vectors

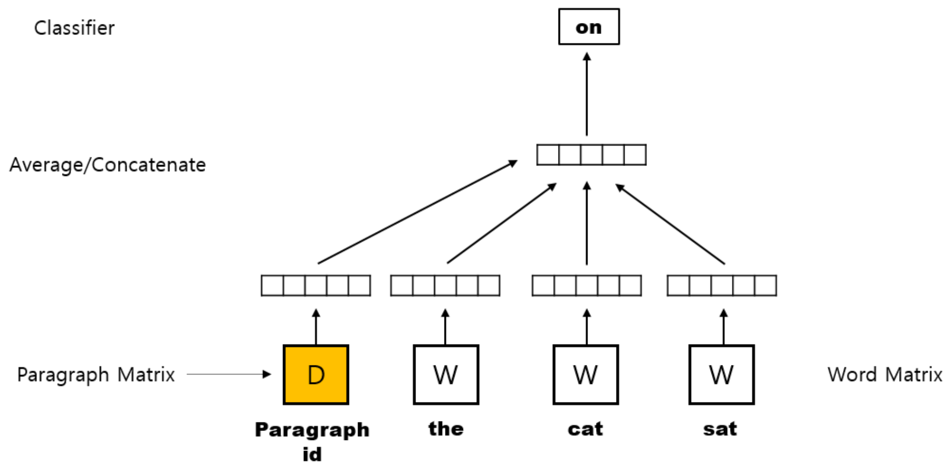


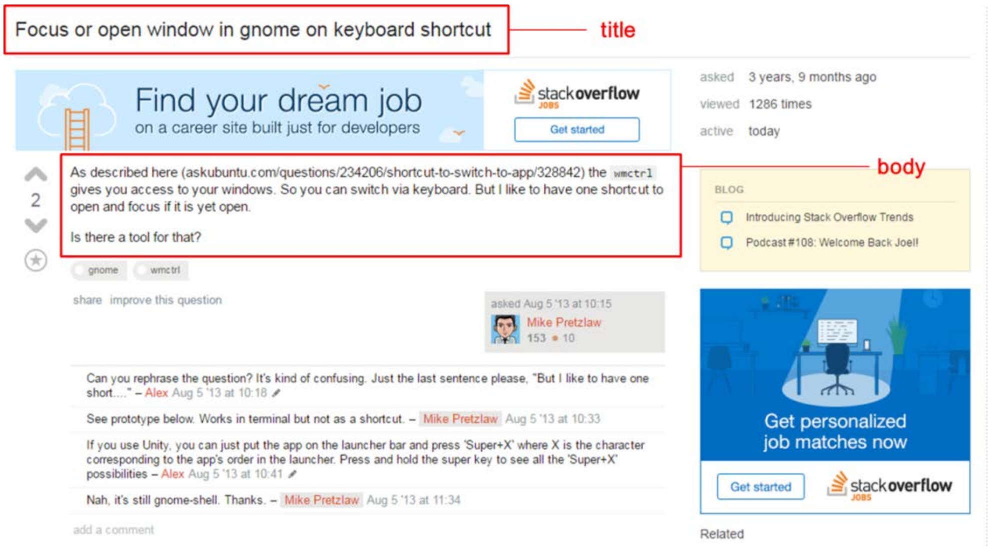
Figure 6. Framework for Learning Paragraph Vectors

After the Doc2Vec model is generated, we have calculated cosine similarity measures between questions. As each question has two parts, a body and a title, we have computed two similarity measures for a question pair. Cosine similarity between two vectors is simply the cosine of vectors. Cosine similarity of two vectors A and B (with θ an angle between two vectors) can be calculated as below:

$$similarity(A,B) = \cos(\theta) = \frac{A \cdot B}{|A||B|} = \frac{\sum_i A_i B_i}{\sqrt{\sum_i A_i^2}\sqrt{\sum_i B_i^2}}$$

Where here A_i, B_i are components of A and B, respectively.

Usually, the title is a one-sentence summary of the question and the body is a detailed explanation of the question (Figure 7). Thus, one question pair has two similarity measures, body similarity and title similarity.



Focus or open window in gnome on keyboard shortcut

asked 3 years, 9 months ago
viewed 1286 times
active today

As described here (askubuntu.com/questions/234206/shortcut-to-switch-to-app/328842) the `wmctrl` gives you access to your windows. So you can switch via keyboard. But I like to have one shortcut to open and focus if it is yet open.

Is there a tool for that?

2

gnome wmctrl

share improve this question

asked Aug 5 '13 at 10:15
Mike Pretzlaw
153 ● 10

BLOG
Introducing Stack Overflow Trends
Podcast #108: Welcome Back Joel!

Can you rephrase the question? It's kind of confusing. Just the last sentence please, "But I like to have one short...." – Alex Aug 5 '13 at 10:18

See prototype below. Works in terminal but not as a shortcut. – Mike Pretzlaw Aug 5 '13 at 10:33

If you use Unity, you can just put the app on the launcher bar and press 'Super+X' where X is the character corresponding to the app's order in the launcher. Press and hold the super key to see all the 'Super+X' possibilities – Alex Aug 5 '13 at 10:41

Nah, it's still gnome-shell. Thanks. – Mike Pretzlaw Aug 5 '13 at 11:34

add a comment

Related

Figure 7. Title and Body of Question

However, in our revised attempt adopting CNN for Sentence Classification approach (Kim 2014), we have computed vector representation of each word in corpus with Word2Vec (Mikolov 2013). And when creating representation for each sentence (i.e., question), we have concatenated vector representations for words in a sentence, converting it into a matrix. More details are covered in Section 3.3 for the revised approach.

3.3 Identification (Classification)

As we take the machine learning approach in developing the framework, the identification process can be deemed as a classification problem. Thus, we want to classify each question pair as similar or non-similar.

For each Doc2Vec model corresponding to a specific domain of interest, we have created a predictive model based on the logistic regression algorithm, one of the simplest, yet powerful, classification algorithms (Shmueli et al 2016). We have tried to avoid overfitting by not only reducing the number of parameters by employing a simple algorithm, but also restricting the number of features used. We have initially attempted to classify question pairs with only the title similarity feature, which is computationally cheaper. However, we were able to obtain better results when considering both features.

It should be noted that logistic classification model based on Doc2Vec similarity measures fail to achieve a reliable performance in a global context; it ends up in correctly classifying only half of the question pairs. In other words, it fails to grasp the effects of differing domain-specific languages and topics. Hence, we build a global model using CNNs, whose prototype is proposed in Kim (2014).

As explained in above, data instances with “match” labels were created by resorting to the collective intelligence of each community. We artificially created data with ‘non-match’ labels by randomly picking question pairs in the question dataset without duplicates. To evade the class imbalance problem, we have set the portion of each class equally. Thus, the half of the dataset for each domain is labeled ‘1’ (i.e., similar), and the rest is labeled ‘0’ (i.e., not similar).

There are some reasons why we have employed CNN structure. To start with, CNN is comparatively cheaper in terms of computation resources than Recurrent Neural Networks (RNN) or Multi-Layer Perceptron (MLP) structures. Characteristics of CNNs such as single feature extraction from convolution operations, dimensionality reduction from pooling operations, weight sharing significantly reduce computational efforts.

Furthermore, in light of language modeling context, CNNs can extract information regarding “context” of sentence with sliding filters and local connectivity (Young et al 2017). In other words, CNNs effectively capture the local context of each word occurrence in sentence. Consider trivial case of interpreting a sentence “I had two *hamburgers* at *McDonald’s* for lunch today, so I do not want to eat at *Wendy’s* now.” At the beginning of the sentence, words “*hamburgers*” and “*McDonald’s*” imply that the speaker is going to talk about something related to fast foods, in this case “*Wendy’s*.” And CNN structure can take into account this contextual information when interpreting the sentence by taking advantage of sliding filters, which function as “windows.” Hence, CNNs are employed in a number of prior studies in NLP domain and applications of such (Kalchbrenner et al 2014; Yih et al 2014; Ruder et al 2016; Poria et al 2016).

We have extended and ameliorated the CNN structure for sentence classification proposed in Kim 2014. To start with, we perform square convolution and pooling operations, rather than rectangular ones. In NLP domain, rectangular convolution/pooling operations, or one-dimensional convolution/pooling operations, are commonly performed (Figure 8). However, as empirical results show no big difference in terms of accuracy, we have performed square convolution & pooling operation, which is common in image recognition field (LeCun et al 1998; Krizhevsky et al 2012).

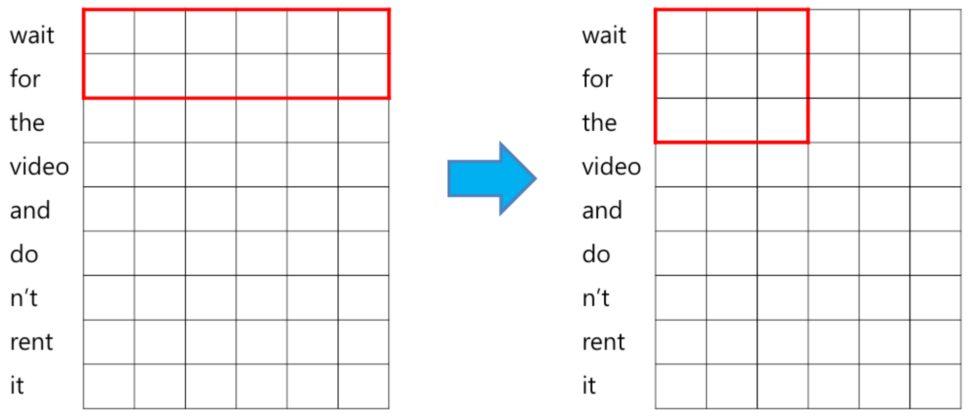


Figure 8. Rectangular and Square Convolution Operations

To effectively manage training process, we have adopted Scaled Exponential Linear Units (SELU) activation function and Batch Normalization (BN) technique. SELU is one of the most recent developments in research domain regarding activation functions. As outputs of the function tend to converge to distribution of zero mean and unit variance, it is claimed to be effective in handling noise and perturbations (Klaumbauer et al 2017).

$$selu(x) = \lambda \begin{cases} x & (if \;\; x > 0) \\ \alpha e^{x} - \alpha & (if \;\; x \leq 0) \end{cases}$$

BN, which is a normalization scheme for each training batch, is an effective method for model training. With BN, it is reported that higher learning rates are permissible and weight initialization schemes can be ignored. Further, it also can prevent overfitting, acting as a regularizer (Ioffe and Szegedy 2015).

Finally, as the CNN structure in Kim 2014 is a sentence classification model, we convert it into sentence-pair classification model. Say that we have sentences of maximum length *l* (i.e., *l* words for each sentence, at maximum) and embedding dimension of *d*. Then we get *2l* X *d* matrix for each sentence pair. If one of sentences has smaller length than *l*, remaining elements are zero-padded to preserve the dimensionality. Conversely, if we have a sentence with number of words bigger than *l*, such sentence is pre-truncated to fit in.

Created embedding weights, i.e., vector representations, are not updated by back-propagation; they are kept static during the training process. This is similar to CNN-static model implementation in the original paper (Kim 2014).

Let assume that we have two questions, "*Merge two arrays*" and "*Array merging in NumPy*" and set *l* = 4, *d* = 100. Resulting matrix for such question pair would be 8 X 100 matrix, as in Figure 9. Note that the third row of matrix is all-zero, to preserve the dimensionality of data instances. If we have another question

"*How can I merge arrays in Python*," such question would be truncated to "*merge arrays in Python*" to fit into 8 X 100 matrix.

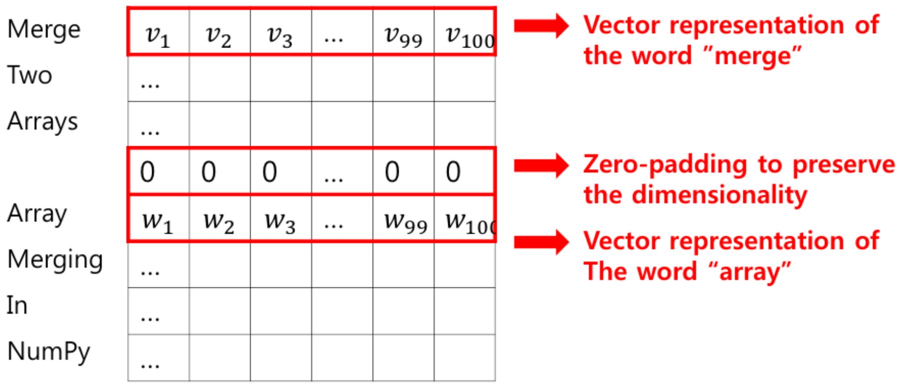

Merge
Two
Arrays
Array
Merging
In
NumPy
v_1 v_2 v_3 … v_{99} v_{100}
0 0 0 … 0 0
w_1 w_2 w_3 … w_{99} w_{100}
Vector representation of the word "merge"
Zero-padding to preserve the dimensionality
Vector representation of The word "array"

Figure 9. Example of Sentence Pair Representation

3.4 Model Selection & Evaluation

For each Doc2Vec model corresponding to a specific domain of interest, we have created a predictive model based on the logistic regression algorithm, one of the simplest classification algorithms (Shmueli et al 2016). We have tried to avoid overfitting by not only reducing the number of parameters by employing a simple algorithm, but also restricting the number of features used (Hawkins 2004). Features used as an input for the predictive model are similarity measures calculated in Section 4.1, body similarity and title similarity. In fact, we have initially attempted to classify question pairs with only the title similarity feature, which is computationally cheaper. However, we were able to obtain better results by considering both features. Detailed results are shown in Section 4. As a result, we have created 16 classification models, summarized in Table 8.

Table 8. Classification Models of Initial Attempt

Model	**Community (Domain)**	**Features Used**	
		Title Similarity	**Body Similarity**
1A	Ask Ubuntu	O	
1B		O	O
2A	Arqade	O	
2B		O	O
3A	Geographic Information Systems	O	
3B		O	O
4A	Home Improvement	O	
4B		O	O
5A	Super User	O	
5B		O	O
6A	Server Fault	O	
6B		O	O
7A	TeX-LaTeX	O	
7B		O	O
8A	Unix & Linux	O	
8B		O	O

The classification result is cross-validated to assess the generalizability of the algorithm. *k*-fold cross-validation is a popular and reliable model selection tool to assess the accuracy and generalizability of a classifier (Kohavi 1995;

Refaeilzadeh 2009). We have conducted a five-fold cross-validation to gauge the overall performance of each classifier.

Evaluation of a design artifact is a critical part of design science research (Hevner et al 2004). To conduct evaluation, we have calculated two types of error for L: type-1 and type-2 errors. As mentioned, recall and precision are metrics that are often used to gauge type-1 error (false positive) and type-2 error (false negative) in classification problems. Recall is the proportion of real positive cases that are predicted positive. Precision is the proportion of predicted positive cases that are actual positives among total predicted positive cases. F-score, the harmonic mean of precision and recall, is calculated. In addition, average precision, which is the mean of precision measures, is calculated when evaluating the revised model (Shmueli et al 2016).

$$Recall = \frac{True\ \ positive}{True\ \ Positive + False\ \ Negative}$$

$$Precision = \frac{True\ \ positive}{True\ \ Positive + False\ \ Positive}$$

$$F - score = \frac{2 * Precision * Recall}{Precision + Recall}$$

For each domain, we have randomly partitioned the initial dataset into two disjoint sets, a training set and a test set. The ratio of training set to test set is 7 to 3. First, a predictive model using logistic regression algorithm is generated using the training set. Then, the results are validated by metrics mentioned above, which are calculated using the test set. The process is iterated for 30 times and metrics are averaged to verify the generalizability of our model.

When evaluating the CNN model and selecting appropriate model, grid search and random search are employed. Grid search is trying all possible combinations in the hyperparameter “grid,” which is one of the most common methods for hyperparameter tuning. In contrast, random search evaluates models with randomly initialized parameters in pre-specified distributions. It is reported that random search is more effective for finding an appropriate model with a limited computation resource (Bergstra and Bengio 2012).

We perform both random search and grid search, and compare the results in various scenarios. First scenarios is identical to the original one – class ratio of 0 (non-match) to 1 (match) is set to 1:1. Second and third scenarios are planned to take into account the skewedness of real-world dataset; in second, ration is 10 to 1 and in third, 100 to 1.

CHAPTER 4 Results

4.1 Initial Attempt

In this section, we display our analysis results. Five-fold cross-validation results are summarized in Table 9. Performances of the models 1A to 8B are compared. The models differ in the domain of interest, i.e., the community, and the features used for the classification process.

Table 9. Cross-validation Results

Model	Accuracy	Standard Deviation
1A	0.845761	0.029341
1B	0.897675	0.027395
2A	0.853553	0.025580
2B	0.910813	0.021823
3A	0.837868	0.052829
3B	0.904867	0.057153
4A	0.892444	0.087446
4B	0.919359	0.087210
5A	0.880266	0.014106
5B	0.926470	0.015834
6A	0.839293	0.063470
6B	0.902929	0.070452
7A	0.804261	0.032912
7B	0.855272	0.026580
8A	0.820760	0.036140
8B	0.879022	0.048658

Cross-validation results have revealed that our models show a practical level of performance over all eight domains – accuracy around 0.90. Hence, it could be argued that our proposed framework is quite generalizable to various domains. Overall, models considering both features, *title similarity* and *body similarity*, outperform models considering only one feature, *title similarity*, by accuracy of about 0.05. Thus, we have considered both features in further evaluation process.

Hence, we have calculated the test accuracy, precision, recall, and F-score of nine models, including the global model that comprises all data from eight domains. As mentioned, the metrics are averaged during the iteration process of 30 times to assure the generalizability of the model. Evaluation results are summarized in Table 10.

Table 10. Summary of Evaluation Results (Initial Attempt)

Model	**Community (Domain)**	**Test accuracy**	**Precision**	**Recall**	**F-score**
1B	Ask Ubuntu	0.897339	0.930675	0.859006	0.893331
2B	Arqade	0.907516	0.945810	0.865160	0.903528
3B	GIS	0.907862	0.956520	0.856595	0.901823
4B	Home Improvement	0.859184	0.890310	0.857216	0.856703
5B	Super User	0.925000	0.957780	0.889366	0.922274
6B	Server Fault	0.891725	0.967703	0.737892	0.836916
7B	Tex-Latex	0.853264	0.885034	0.809969	0.845570
8B	Unix & Linux	0.874578	0.932291	0.808427	0.865512
9	**Global**	**0.507437**	**0.563082**	**0.509670**	**0.535046**

Evaluation results demonstrate the applicability of our proposed framework. All models have shown a satisfactory level of test accuracy over 0.85. Especially, models 2, 3, and 5 have shown a high level of test accuracy over 0.90.

Nevertheless, it could be observed that model 9, namely the Global model, fails to produce useful results. Model 9 gathers all questions in eight domains, creates a global Doc2Vec model, and with vector representations from the model, classifies each question pair. Hence, when using a simple structure for identification, it could be asserted that building a distinct model for each domain is effective.

4.2 Revised Approach

In building a global model that can be applied in the presence of a large training dataset, we take the approach proposed in Kim (2014). First, with the training set composed of all questions in eight domains, we create a Word2Vec model with dimensionality of 100. Then, we excluded questions that have lengths of over 10 words. Such questions were minimal in proportion; holding only 1% of the questions. Then, we created a convolutional neural network (CNN) model for question pair classification.

Hyperparameters are tuned with grid search and random search methods and mentioned above; i.e., individual models are created and cross-validated with specified hyperparameter settings and chosen based on its performances. Hyperparameters of interest are batch size, dropout rate, epochs, and number of filters. Re-evaluation results of best models in each scenario are summarized in

Table 11. More detailed results are available in Appendix 2 and 3.

Table 11. Summary of Evaluation Results (Revised Attempt)

Method	Scenario	Test Accuracy	Precision	Recall	F1-score	Average Precision
Grid Search	1	0.9754	0.97	0.98	0.98	0.9835
	2	0.9903	0.90	0.99	0.95	0.9514
	3	**0.9958**	**0.74**	**0.81**	**0.77**	**0.7768**
Random Search	2	0.9939	0.95	0.98	0.97	0.9687
	3	**0.9966**	**0.71**	**0.92**	**0.81**	**0.8209**

As expected, average precision measures under class imbalance scenarios suffer, especially in scenario 3. However, there is a good news – when performing random search, we could see a slight boost in terms of average precision (0.8209), compared to result of grid search (0.7768). It could be asserted that random search, as claimed by Bergstra and Bengio (2012), performs better than grid search.

CHAPTER 5 Conclusion

In this study, we have proposed a framework that can contribute to the collective intelligence in SQA communities by identifying similar questions. Even though finding semantically similar questions is a critical issue and garnered interest in both practice and academia, it remains a thorny problem. Thus, prior studies show some limitations regarding the implementation and evaluation of the models. We have addressed such limitations and provided partial solutions for them.

We have taken multidisciplinary approach to the creation and evaluation of the framework and models. Techniques and ideas from NLP, neural network language modeling, machine learning, and deep learning are utilized to implement the models.

As it is difficult to obtain reliable test datasets, we fell back on “the wisdom of crowds” already established in active SQA communities. By doing so, we were able to reduce human effort in labeling the training and test datasets, while obtaining reliable data for learning. Finally, by formulating a Doc2Vec model for each domain, we were able to ease the problems of the “lexico-syntactic gap” and domain-specific terminologies.

Classification and evaluation results show that our framework and models show an applicable level of performance, with the possibility of solving the real-world problem of identifying semantically equivalent questions in SQA communities. Nevertheless, our study has some limitations. To start with, as we have built a Doc2Vec model for each domain, it requires a certain amount of training data regarding the domain of interest. However, in building a global model, a CNN-based model structure suggested by Kim (2014) is highly effective.

Furthermore, the levels of collective intelligence in SQA communities might vary depending on the domain. It is one of our assumptions that the collective intelligence of each SQA community excels the intellectual ability of an individual. However, in some cases this might turn out to be unrealistic and require further validation, though not evident in our chosen communities with a number of active users. For instance, some SQA communities might suffer from a lack of participation among users as they are concerned with unpopular topics. Hence, it would be crucial to check the participatory status of a community to warrant the collective intelligence of the users before applying the framework.

Even though there are plenty of previous works devoted to solving the problem of identifying similar questions in SQA domain, there still exist many questions to be answered in depth. Such questions provide room for future work, challenging researchers in NLP and machine learning.

One of the potential pitfalls and directions for future research would be regarding imbalance among classes, which is slightly covered in this study. In reality, there are much more dissimilar question pairs than similar question pairs, as indicated in Table 5. One interesting research area would be testing the model with not only basic measures such as accuracy, precision, and recall, but also other suggestions such as receiver operating characteristics (ROC) and bookmaker's informedness (Fawcett 2006; Powers 2011). Such attempts are expected to be highly beneficial for both research and practice. Otherwise, it could be effective to perform class-weight learning with highly skewed datasets, especially penalizing *false negatives* very harshly to prevent all-zero classification.

Also, more refined methods to model sentences (questions) could be utilized. Nowadays, there are some state-of-the-art word embedding techniques, apart from Word2Vec and Doc2Vec. For instance, GloVe (Pennington 2014) or geometry of sentence (Mu et al 2017) could be good choices for future research.

Moreover, adopting state-of-the-art methods not only in language modeling contexts, but also in computational learning could be meaningful. Even though we have stressed the desirability of CNN-based methods, RNN-based models such as Long-Short Term Memory (LSTM) models or Gated Recurrent Units (GRU) models can be effective as well (Hochreiter et al 1997; Chung et al 2014). Using such models with state-of-the art techniques such as sequence-to-sequence (Sutskever et al 2014; Cho et al 2014) and attention mechanisms (Luong et al 2015) can be meaningful research direction.

Finally, applications on more real-world datasets and feedbacks from such attempts would boost the generalizability and applicability of the model. First, we could try out with data from other domains, i.e., communities, in stack exchange network. Then, taking out and testing our model for more large-scale datasets in other networks such as Quora or Yahoo! Answers could be desirable.

References

Bergstra, J., & Bengio, Y. (2012). Random search for hyper-parameter optimization. Journal of Machine Learning Research, 13(Feb), 281-305.

Bird, S. 2006. “NLTK: the natural language toolkit,” In *Proceedings of the COLING/ACL on Interactive presentation sessions*. pp. 69-72.

Blei, D. M. 2012, “Probabilistic topic models,” *Communications of the ACM* (*55*:4), pp. 77-84.

Brown, P. F., Pietra, V. J. D., Pietra, S. A. D., & Mercer, R. L. (1993). The mathematics of statistical machine translation: Parameter estimation. *Computational linguistics*, *19*(2), 263-311.

Chahuara, P., Lampert, T., & Gançarski, P. 2016. “Retrieving and Ranking Similar Questions from Question-Answer Archives Using Topic Modelling and Topic Distribution Regression,” In *International Conference on Theory and Practice of Digital Libraries*. pp. 41-53.

Cho, K., Van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., & Bengio, Y. (2014). Learning phrase representations using RNN encoder-decoder for statistical machine translation. arXiv preprint arXiv:1406.1078.

Chung, J., Gulcehre, C., Cho, K., & Bengio, Y. (2014). Empirical evaluation of gated recurrent neural networks on sequence modeling. arXiv preprint arXiv:1412.3555.

Das, A., Shrivastava, M., & Chinnakotla, M. 2016. "Mirror on the wall: Finding similar questions with deep structured topic modeling" In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pp. 454-465.

Derivative. 2017, In *Merriam-Webster.com*. Retrieved April 30, 2017, from https://www.merriam-webster.com/dictionary/derivative

Fawcett, T. (2006) "An introduction to ROC analysis," *Pattern Recognition Letters* (27:8), pp. 861-874.

Fellegi, I. P., & Sunter, A. B. 1969, "A theory for record linkage," *Journal of the American Statistical Association* (*64*:328), pp. 1183-1210.

Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. Neural computation, 9(8), 1735-1780.

Jeon, J., Croft, W. B., & Lee, J. H. 2005. "Finding similar questions in large question and answer archives," In *Proceedings of the 14th ACM international conference on Information and knowledge management*, pp. 84-90.

Hawkins, D. M. (2004). The problem of overfitting. Journal of chemical information and computer sciences, 44(1), 1-12.

Hevner, A. R., March, S. T., Park, J., & Ram, S. 2004, "Design science in information systems research," *MIS quarterly* (*28*:1), pp. 75-105.

Ioffe, S., & Szegedy, C. (2015, June). Batch normalization: Accelerating deep network training by reducing internal covariate shift.

In International Conference on Machine Learning (pp. 448-456).

John, B. M., Goh, D. H. L., Chua, A. Y. K., & Wickramasinghe, N. 2016. "Graph-based Cluster Analysis to Identify Similar Questions: A Design Science Approach," *Journal of the Association for Information Systems* (*17*:9), September, pp.590-613.

Kalchbrenner, N., Grefenstette, E., & Blunsom, P. (2014). A convolutional neural network for modelling sentences. arXiv preprint arXiv:1404.2188.

Klambauer, G., Unterthiner, T., Mayr, A., & Hochreiter, S. (2017). Self-Normalizing Neural Networks. In advances in Neural Information Processing Systems (NIPS).

Kim, Y. 2014. "Convolutional neural networks for sentence classification," arXiv preprint arXiv: 1408.5882.

Kohavi, R. (1995, August). A study of cross-validation and bootstrap for accuracy estimation and model selection. In Ijcai (Vol. 14, No. 2, pp. 1137-1145).

Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In Advances in neural information processing systems (pp. 1097-1105).

Le, Q., & Mikolov, T. 2014. “Distributed representations of sentences and documents,” In *Proceedings of the 31st International Conference on Machine Learning (ICML-14)*, pp. 1188-1196.

LeCun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998). Gradient-based learning applied to document recognition. Proceedings of the IEEE, 86(11), 2278-2324.

Lee, J. T., Kim, S. B., Song, Y. I., & Rim, H. C. 2008, “Bridging lexical gaps between queries and questions on large online Q&A collections with compact translation models,” In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pp. 410-418

Lorist, M. M., Boksem, M. A., & Ridderinkhof, K. R. 2005, “Impaired cognitive control and reduced cingulate activity during mental fatigue,” *Cognitive Brain Research* (24:2), pp. 199-205.

Luong, M. T., Pham, H., & Manning, C. D. (2015). Effective approaches to attention-based neural machine translation. arXiv preprint arXiv:1508.04025.

(Mikolov et al 2013a) Mikolov, T., Chen, K., Corrado, G., & Dean, J. 2013. "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., & Dean, J. (2013). Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems* (pp. 3111-3119).

Mu, J., Bhat, S., & Viswanath, P. (2017). Representing Sentences as Low-Rank Subspaces. arXiv preprint arXiv:1704.05358.

Pennington, J., Socher, R., & Manning, C. (2014). Glove: Global vectors for word representation. In Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP) (pp. 1532-1543).

Perkins, J. 2010. *Python text processing with NLTK 2.0 cookbook*. Packt Publishing Ltd.

Poria, S., Cambria, E., Hazarika, D., & Vij, P. (2016). A deeper look into sarcastic tweets using deep convolutional neural networks. arXiv preprint arXiv:1610.08815.

Powers, D.M. 2011. Evaluation: from precision, recall, and F-measure to ROC, informedness, markedness, and correlation.

Refaeilzadeh, P., Tang, L., & Liu, H. (2009). Cross-validation. In Encyclopedia of database systems (pp. 532-538). Springer US.

Ruder, S., Ghaffari, P., & Breslin, J. G. (2016). INSIGHT-1 at SemEval-2016 Task 5: Deep Learning for Multilingual Aspect-based Sentiment Analysis. arXiv preprint arXiv:1609.02748.

Salton, G., & Buckley, C. (1988). Term-weighting approaches in automatic text retrieval. *Information processing & management*, *24*(5), 513-523.

Selvaraju, R. R., Cogswell, M., Das, A., Vedantam, R., Parikh, D., & Batra, D. (2016). Grad-CAM: Visual Explanations from Deep Networks via Gradient-based Localization. See https://arxiv. org/abs/1610.02391 v3.

Shmueli, G., Patel, N. R., & Bruce, P. C. 2016. *Data Mining for Business Analytics: Concepts, Techniques, and Applications with XLMiner*. John Wiley & Sons.

Stack Exchange. 2017. "All sites." https://StackExchange.com/sites?view=list#traffic. Accessed April 30 2017.

Surowiecki, J. 2005. *The wisdom of crowds*, Anchor.

Sutskever, I., Vinyals, O., & Le, Q. V. (2014). Sequence to sequence learning with neural networks. In Advances in neural information processing systems (pp. 3104-3112).

Xue, X., Jeon, J., & Croft, W. B. 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. pp. 475-482.

Yih, S. W. T., He, X., & Meek, C. (2014). Semantic parsing for single-relation question answering.

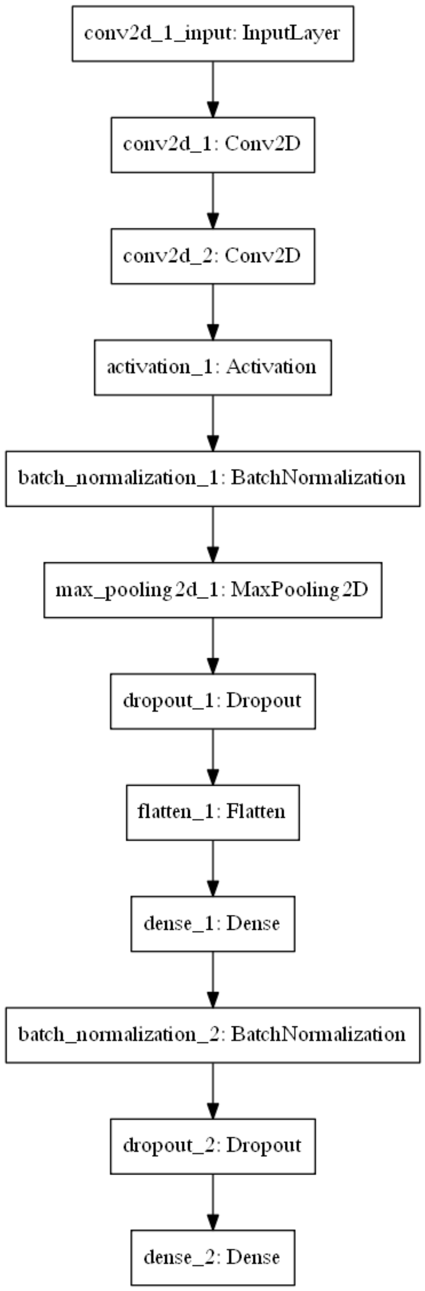
Young, T., Hazarika, D., Poria, S., & Cambria, E. (2017). Recent trends in deep learning based natural language processing. arXiv preprint arXiv:1708.02709.

Zhou, G., He, T., Zhao, J., & Hu, P. 2015. “Learning Continuous Word Embedding with Metadata for Question Retrieval in Community Question Answering,” In *the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing*, pp.250-259.

5 Appendices

Appendix 1

: Visualization of model (using Keras in Python)



Appendix 2

: Grid Search Results

1) Scenario 1

Trial	Average Accuracy	Batch Size	Dropout Rate	Epochs	Number of Filters
1	0.917738	30	0.2	100	10
2	0.939447	30	0.2	100	20
3	0.956748	30	0.2	200	10
4	0.965650	30	0.2	200	20
5	0.947132	30	0.3	100	10
6	0.946754	30	0.3	100	20
7	0.965776	30	0.3	200	10
8	0.965273	30	0.3	200	20
9	0.957126	30	0.4	100	10
10	0.954733	30	0.4	100	20
11	0.954229	30	0.4	200	10
12	0.964853	30	0.4	200	20
13	0.918157	40	0.2	100	10
14	0.902410	40	0.2	100	20
15	0.935122	40	0.2	200	10

16	0.963761	40	0.2	200	20
17	0.950071	40	0.3	100	10
18	0.936088	40	0.3	100	20
19	0.946544	40	0.3	200	10
20	0.960779	40	0.3	200	20
21	0.917947	40	0.4	100	10
22	0.963887	40	0.4	100	20
23	0.963593	40	0.4	200	10
24	0.970354	40	0.4	200	20
AVG	0.948507				

2) Scenario 2

Trial	Average Accuracy	Batch Size	Dropout Rate	Epochs	Number of Filters
1	0.948998	30	0.3	100	10
2	0.954877	30	0.3	100	20
3	0.970909	30	0.3	200	10
4	0.963275	30	0.3	200	20
5	0.971062	30	0.4	100	10
6	0.970070	30	0.4	100	20

7	0.974269	30	0.4	200	10
8	0.973658	30	0.4	200	20
9	0.969153	40	0.3	100	10
10	0.967932	40	0.3	100	20
11	0.966787	40	0.3	200	10
12	0.973123	40	0.3	200	20
13	0.966022	40	0.4	100	10
14	0.972513	40	0.4	100	20
15	0.968848	40	0.4	200	10
16	0.969917	40	0.4	200	20
AVG	0.967588				

3) Scenario 3

Trial	Average Accuracy	Batch Size	Dropout Rate	Epochs	Number of Filters
1	0.936113	30	0.3	100	10
2	0.990186	30	0.3	100	20
3	0.990768	30	0.3	200	10
4	0.990601	30	0.3	200	20
5	0.990435	30	0.4	100	10

6	0.987940	30	0.4	100	20
7	0.989105	30	0.4	200	10
8	0.990186	30	0.4	200	20
9	0.990685	40	0.3	100	10
10	0.991600	40	0.3	100	20
11	0.990768	40	0.3	200	10
12	0.990518	40	0.3	200	20
13	0.990768	40	0.4	100	10
14	0.991267	40	0.4	100	20
15	0.990435	40	0.4	200	10
16	0.989021	40	0.4	200	20
AVG	0.9869				

Appendix 3

: Random Search results

1) Scenario 2

Trial	Average Precision	Batch Size	Dropout Rate	Epochs	Number of Filters
1	0.908705	39	0.5	57	13
2	0.918524	35	0.3	143	16

3	0.911144	41	0.3	92	12
4	0.914986	37	0.3	60	13
5	0.913085	44	0.5	52	16
6	0.924158	41	0.3	172	12
7	0.926931	33	0.4	149	15
8	0.915296	47	0.4	59	17
9	0.882247	46	0.3	79	19
10	0.919945	44	0.3	92	15
AVG	0.913502				

2) Scenario 3

Trial	Average Precision	Batch Size	Dropout Rate	Epochs	Number of Filters
1	0.390429	47	0.5	95	18
2	0.474817	41	0.5	95	18
3	0.464673	44	0.3	173	16
4	0.458598	39	0.5	130	10
5	0.463360	38	0.5	99	13
6	0.451081	47	0.3	182	17
7	0.391716	48	0.5	87	10

8	0.411866	48	0.3	157	10
9	0.417540	36	0.3	117	13
10	0.375774	35	0.4	75	12
AVG	0.429985				