A Framework for Task-specific Short Document Expansion

Ramakrishna B Bairi*
IITB-Monash Research Academy
IIT Bombay
Mumbai, India
bairi@cse.iitb.ac.in

Raghavendra Udupa
Microsoft Research India
Bangalore, India
raghavu@microsoft.com

Ganesh Ramakrishnan
Dept of CSE
IIT Bombay
ganesh@cse.iitb.ac.in

ABSTRACT

Collections that contain a large number of short texts are becoming increasingly common (*eg.*, tweets, reviews, *etc*).

Analytical tasks (such as classification, clustering, *etc.*) involving short texts could be challenging due to the lack of context and owing to their sparseness. An often encountered problem is low accuracy on the task. A standard technique used in the handling of short texts is expanding them before subjecting them to the task. However, existing works on short text expansion suffer from certain limitations: (i) they depend on domain knowledge to expand the text; (ii) they employ task-specific heuristics; and (iii) the expansion procedure is tightly coupled to the task. This makes it hard to adapt a procedure, designed for one task, into another. We present an expansion technique – TIDE (**T**ask-spec**I**fic short **D**ocument **E**xpansion) – that can be applied on several Machine Learning, NLP and Information Retrieval tasks on short texts (such as short text classification, clustering, entity disambiguation, and the like) without using task specific heuristics and domain-specific knowledge for expansion. At the same time, our technique is capable of learning to expand short texts in a task-specific way. That is, the same technique that is applied to expand a short text in two different tasks is able to learn to produce different expansions depending upon what expansion benefits the task's performance. To speed up the learning process, we also introduce a technique called *block learning*. Our experiments with classification and clustering tasks show that our framework improves upon several baselines according to the standard evaluation metrics which includes the accuracy and normalized mutual information (NMI).

1. INTRODUCTION

With the rapid growth of the Internet, Web users are generating an increasing number of short texts, including tweets, search snippets, product reviews and the like. Successfully processing them becomes increasingly important to many IR and Machine Learning applications. However, short texts are quite different from the traditional documents due to their being short and sparse. Hence, conventional machine learning and IR algorithms cannot apply to short texts directly. Owing to lack of context in the short text, it becomes challenging to build a representative feature vector to use it in a machine learning task such as classification, clustering, ranking or searching, to name a few.

Existing approaches [17, 5, 1, 20, 25] attempt to address challenges associated with analytic tasks on short texts by introducing more context through expansion techniques. Broadly, these approaches follow one of two techniques. The first one is to employ a search engine (for example, Google) or an IR system (for example, Lucene[1], Solr[2]) to use the short text as a search query and expand the short text using the top search results [1, 20, 25]. The other technique is to build a topic space and to project each short text into that space. These topics enrich the context of the short text [21, 2, 17, 5].

Although querying search engines using short texts can produce good expansions, it may not be an ideal solution for many applications, given its online nature and given restricted programmatic access to search engines. Though querying a local IR system is fast, we observe from experiments that the top result that is returned by the search engine or the IR system may not always be the best for the task.

Using a predefined topic space may not be a feasible solution because there may not be predefined topics/taxonomy for certain applications, domains and languages. A solution that is based on latent topic space is preferable because the latent topics can be generated from a large corpus that is available in the domain of an application. However, solutions that are based on topic space are restricted to certain types of applications. It is difficult to generalize these solutions in different types of applications. For example, though tasks such as classification of short texts can be efficiently solved by these approaches [17, 5], they may not be suitable for tasks on learning to rank short texts. Primarily, this is due to the bag-of-words model assumption that these approaches make while expanding the short texts, that is, the pseudo topic names that are appended to each short text as additional words help the classifiers to learn discriminative weights for each class, thus boosting the classification accu-

*Work done when the first author was interning with Microsoft Research India.

CIKM '16, October 24–28, 2016, Indianapolis, IN, USA.

 ISBN 978-1-4503-4073-1/16/10...$15.00

DOI: http://dx.doi.org/10.1145/2983323.2983811

[1]https://lucene.apache.org/

[2]http://lucene.apache.org/solr/

racy. However, if the task is the retrieval and ranking of short texts, the solution of adding pseudo topic names may not help. One of the main characteristics of our approach is that we do not make such assumptions, thus our approach generalizes well to other tasks.

A large body of research has been directed at using sources of structured semantic knowledge such as Wikipedia, DBPedia and WordNet for document expansion [2, 21, 10, 8, 9, 13]. On the other hand, distributional semantic approaches are based on the intuition that words appearing in similar contexts tend to have similar meanings. One such approach is word vectors—also referred to as word embeddings— which has recently seen a surge of interest since new ways of computing them efficiently [15, 16] have become available. Word embeddings provide a way to expand documents by attaching average or augmented word vectors of the terms in the document [24]. While all these approaches provide ways to expand short texts, the expansion itself is independent of the task that uses them; there is no guarantee that such an expansion would benefit the task.

To address these shortcomings we develop an approach that has the following characteristics: (i) It makes fewest assumptions about the task; the task should be able to consume the suggested expansions and produce a measurable performance metric (such as task accuracy), nothing else. (ii) It uses an expansion technique that generalizes well across different tasks. (iii) It expands the short texts selectively, that is, a short text is expanded only if such an expansion helps the task; otherwise, it is not expanded. Blindly expanding all the short texts - as done in earlier approaches - in fact degrades the task's performance. (iv) It is able to learn a model that produces the expansions specific to the task. However, the features used by the model generalizes well to all the tasks, thus making the approach applicable to many tasks. In section 2.3 we present different classes of feature functions which can be used across multiple tasks. Figure 1 depicts overall architecture of our approach/framework. As in many earlier approaches [17, 5] we use a corpus that contains relevant *long text* articles. In section 3.3, we comment more on choosing the right corpus. Using short text as a query, a language model or an IR system initially selects K long texts from the corpus as expansion candidates. Our observation shows that we can usually find the best long text for expansion within the top $10-20$ results. We then learn an expansion model that is specific to the task (by taking task's performance metric as an input) for selecting one of the K candidate long texts as the expansion of the short text. The goal of learning is to produce a mapping of the short texts to the long texts which best improves the task accuracy. Note that the framework only requires an accuracy value from the task and does not exploit any other characteristics of the task or domain. Hence the framework is applicable to a wide variety of tasks.

In particular, our framework learns a best mapping function $\mathcal{M} : \{\text{short\_text} \times \text{task}\} \rightarrow \{\text{long\_text}\}$ to maximize the task accuracy. We can present this function in an alternate form as

$$\mathcal{M}(\text{short\_text}) = \underset{\text{long\_text}}{\text{argmax}}\ \text{Sim}(\text{short\_text}, \text{long\_text};\ \text{task})$$

which is a mapping function that finds a maximally similar long text to a short text, given the task. The similarity is captured through a variety of functions that are designed from the IR, topic model and embedding techniques and generalize well to many tasks. The function $\mathcal{M}$ needs to be learned specific to the task. There are no additional training data for learning $\mathcal{M}$. We have designed a novel learning technique to learn $\mathcal{M}$ jointly with the task during the training phase of the task, thus making $\mathcal{M}$ task specific. In section 2.4, we elaborate our learning technique using a derivative-free optimization technique that is known as BOBYQA (Bound Optimization BY Quadratic Approximation) [18] along with our proposed *block learning* approach.

Our contributions can be summarized as follows:

- A framework for task-specific document expansion that can be adapted by many machine learning tasks that deal with the short texts
- The introduction of various classes of task agnostic feature functions (IR, topic model and embedding based features)
- A technique for learning a model over task agnostic features, using the task's data through the task itself, thus learning to expand a short text in a task-specific way.
- A *block learning* technique for learning feature weights in blocks

2. LEARNING TASK-SPECIFIC DOCUMENT EXPANSION

2.1 Problem Formulation

Let $\mathcal{D}_s = \{s_i\}_{i=1}^n$ be a set of n short texts, where s_i is the representation of i^{th} short text. For example, s_i can be a TF-IDF vector representation of the short text, or it can simply be a bag of words. The exact form of s_i is task-specific.

Let $\mathcal{T}(\mathcal{D}_s)$ be a task that operates on the short texts $\mathcal{D}_s$. For example, $\mathcal{T}$ can be a classification task and $\mathcal{D}_s$ is a set of short texts for training and testing the classifier. The underlying assumption is that task $\mathcal{T}$ can be run on the data $\mathcal{D}_s$ and produce a measurable result that indicates the goodness of the task, such as the accuracy of the task. In the classification task example that is mentioned above, we can measure the accuracy of the classifier that is trained on the training set and validated on the test set.

Other examples of the tasks include Clustering of short texts, Named Entity Recognition in short texts and Categorization of short texts using Wikipedia, to name a few.

Let $\mathcal{E}(\mathcal{T}(\mathcal{D}_s))$ be the empirical error in performing task $\mathcal{T}$. For example, in the classification task, $\mathcal{E}$ could be a "misclassification rate"; in the clustering task, it could be "degree of clustering disagreement". Furthermore, we assume that the error can be scaled to the [0,1] interval so that we can simply compute the accuracy of the task $\mathcal{T}$ as $1-\mathcal{E}(\cdot)$.

A standard technique to improve the accuracy of the task $\mathcal{T}$ is to expand the short texts $\mathcal{D}_s$ into *bigger* texts. Let $\mathcal{D}_e = \{e_i\}_{i=1}^n$ be the expanded short texts, where e_i is a representation for the expanded i^{th} short text. For example., e_i can be a TF-IDF vector that represent the expanded text.

Our aim is to present a technique that expands the short texts $\mathcal{D}_s$ into long texts $\mathcal{D}_e$ such that, performing the task $\mathcal{T}$ on these expanded texts best improves the task accuracy

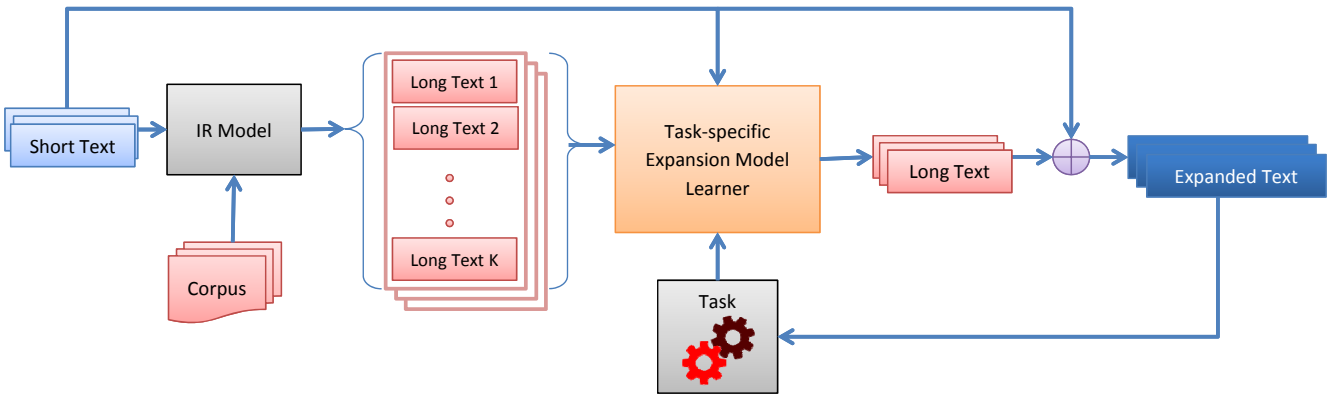

Figure 1: Architecture of expansion model

(this equivalently reduces the task error rate.) Formally, by applying our technique, it is possible to achieve the following with a maximum difference between RHS and LHS.

$$\mathcal{E}\left(\mathcal{T}\left(\mathcal{D}_e\right)\right) \leq \mathcal{E}\left(\mathcal{T}\left(\mathcal{D}_s\right)\right) \tag{1}$$

To perform the expansion, we assume the existence of a right universal corpus $\mathcal{C}$ of articles. Such a corpus has to contain a large amount of articles in the domain of short texts, which covers the topics and concepts present in the short texts. These articles provide contexts to the short texts and help in expanding them. We refer to these articles as *long* texts. Our goal is to identify the best possible *long* text $l_i \in \mathcal{C}$ for every short text s_i. This long text l_i is then used to expand the short text s_i. We use the operator $\oplus$ to represent the expansion process of obtaining an expanded text representation of a short text from a long text. That is, $e_i = s_i \oplus l_i$. The exact process of this expansion depends on the task. For example, $\oplus$ may be a union of s_i and e_i, or it may be a weighted average of TF vectors. Mapping short texts $\{s_i\}_{i=1}^n$ to long texts $\{l_i\}_{i=1}^n$ is done in such a way that the task using the expanded short texts will have maximum improvement in accuracy. This makes the short text mapping/expansion task-specific. That means that the same short text may map to different long texts in different tasks, depending upon which mapping makes the task better. However, our framework for expansion remains the same making it task agnostic.

In particular, we present a technique in this paper to discover a mapping of $l_i = \mathcal{M}\left(s_i\right)$ to a long text $l_i \in \mathcal{C}$ for every short text $s_i \in \mathcal{D}_s$, such that Equation 1 is satisfied with maximum margin. Formally, we solve the following optimization problem.

$$\begin{aligned} \mathcal{M} = & \underset{\mathcal{M}}{\operatorname{argmax}}\ \gamma \\ & \text{s.t. } \mathcal{E}\left(\mathcal{T}\left(\mathcal{D}_s \oplus \mathcal{M}\left(\mathcal{D}_s\right)\right)\right) = \mathcal{E}\left(\mathcal{T}\left(\mathcal{D}_s\right)\right) + \gamma \\ & \gamma \geq 0 \end{aligned} \tag{2}$$

where $\mathcal{D}_e = \{e_i\}_{i=1}^n = \mathcal{D}_s \oplus \mathcal{M}\left(\mathcal{D}_s\right) = \{s_i \oplus \mathcal{M}\left(s_i\right)\}_{i=1}^n$

An optimum mapping $\mathcal{M}$ increases the difference between $\mathcal{E}\left(\mathcal{T}\left(\mathcal{D}_s \oplus \mathcal{M}\left(\mathcal{D}_s\right)\right)\right)$ and $\mathcal{E}\left(\mathcal{T}\left(\mathcal{D}_s\right)\right)$, thus maximizing the margin γ. Solving this optimization problem turns out to be hard. In the following Sections we present a technique for the solving of the optimization problem mentioned above approximately. Our experiments on benchmark datasets show that our approach is very effective practically and can produce results that are close to optimal.

2.2 Learning framework

Finding optimal mapping to solve Equation 2 is a combinatorial problem. There are $|\mathcal{C}|^{|\mathcal{D}_s|}$ possible mappings of short texts $s_i \in \mathcal{D}_s$ to long texts $l_i \in \mathcal{C}$. However, not all of these mappings are meaningful. For example, mapping a short text on "soccer" to an article on "photosynthesis" makes very little sense. To be a semantically correct expansion, the short text and its mapped long text have to be topically similar. Many IR techniques [14] have been successful in retrieving and ranking documents for short queries. They adapt language modeling techniques [12] to map short text queries to long text articles. This makes the language modeling approach a possible approximation to solve Eq 2.

In the language modeling approach to information retrieval, a multinomial model $p\left(w|l_i\right)$ over terms is estimated for each document l_i in the collection $\mathcal{C}$ to be indexed and searched. This model is used to assign a likelihood to a short text $s_i = (w_1, w_2, ..., w_m)$. In the simplest case, each short text term is assumed to be independent of the other short text terms, so that the short text likelihood is given by $p\left(s_i|l_j\right) = \prod_{k=1}^m p\left(w_k|l_j\right)$. After the specification of a document prior $p(l_j)$, the a posteriori probability of a document is given by: $p\left(l_j|s_i\right) \propto p\left(s_i|l_j\right) p\left(l_j\right)$, and is used to rank the documents in collection $\mathcal{C}$.

The language model approach has the following limitations in mapping a short text s_i to a long text l_i: (i) it always produces the same mapping irrespective of the task; (ii) it does not take the task accuracy into account while mapping; and (iii) our observations show that the top ranked result of language model need not always be the best mapping.

In order to overcome these limitations, we propose a two-step process for mapping. First, we use language model to retrieve the top K candidate for the mapping of long texts $\left\{l_i^k \in \mathcal{C}\right\}_{k=1}^K$ for every short text s_i. Second, we define a mapping $\mathcal{M}_w$ to map a short text s_i to the best long text l_i from the K candidate long texts, such that using l_i to expand s_i best improves the accuracy of the task. $\mathcal{M}_w$ is computed by solving the following linear model:

$$l_i = \mathcal{M}_w\left(s_i\right) = \underset{l \in \left\{l_i^k \in \mathcal{C}\right\}_{k=1}^K}{\operatorname{argamax}} \sum_f w_f \phi_f\left(s_i, l\right)$$

where, ϕ_f is a feature function that measures the similarity

between a short and a long text and w_f is the weight of that feature function in the mixture of features. A bunch of feature functions designed from the proven methods of IR, topic model and embedding techniques are explained in Section 2.3. These feature functions are then combined by learning weights $W = [w_f]_{f=1}^{f=(\text{number of features})}$ for them in a task-specific manner, which is explained in Section 2.4.

2.3 Classes of Feature Functions for Mapping

In the below sections, we define a variety of feature functions that measure the similarity of a short text to a long text. We group them into three classes: (i) IR based, (ii) Topic Model based, (iii) Embedding based.

In the following sections we use the notations s and l to represent short and long texts, respectively, and $|l|$ to denote the length of l. We drop the subscript i for brevity.

2.3.1 IR Based Features

IR-based features compute the relevance of a short to a long text using standard document similarity functions such as BM25, Consine, and Bigram. These functions have been successfully adapted by the information retrieval community in orer to retrieve documents using short queries [14]. Accordingly, we define features such as $\phi_{bm25}(s,l) = \text{BM25}(s,l)$,$\phi_{cosine}(s,l) = \text{COSINE}(s,l)$, and $\phi_{bigram}(s,l) = \text{BIGRAM}(s,l)$ to compute the similarity between a short and a long text.

2.3.2 Topic Model-based Features

Topic models such as LDA [3] have been successful in discovering hidden topic distributions in a text corpus. Earlier works [19] have shown that matching texts based on the similarity models with hidden topics yield better results. Based on these findings, we define a set of topic model-related functions to compute the similarity between a short and a long text. To discover the topics, we run LDA on the articles in the corpus $\mathcal{C}$. Let T_i be the i^{th}topic discovered by the topic model and V_i be the set of top words in T_i. Let $\text{tf}(w;l)$ be the term frequency of term w in long text l . We define the following topic model based feature functions:

Topic Score is defined as $\phi_{T_i}(s,l) = \frac{1}{|l|}\sum_{w\in V_i} tf(w;l)$. It measures the relevance of a long text l to the topic T_i. Although this feature is not a function of short text s, it helps to measure the relevance of a long text l in the topic space of the corpus, which by construction includes the topic space of the short texts. For each topic T_i, one such similarity function is created..

Differential Topic Score is defined as $\phi_{dt_i}(s,l) = \frac{1}{|l|}\sum_{w\in V_i\setminus s} tf(w;l)$. It measures the similarity of a long text l to the topic words without considering the words in the short text s. The intuition here is that, we want to eliminate the part of the score that comes from matching words between short text s and long text l. Since IR-based feature functions already capture that, we want to get the score that solely comes from matching the long text to the topic T_i. Note that for every topic T_i, we define one feature function ϕ_{dt_i}..

Note that for the topic model based functions to be effective, it is very important to have a corpus with a topic distribution that represents the topic distribution of the short texts. More discussion on this is deferred to Section 3.3

2.3.3 Embedding-based features

Word embeddings [15, 16] are vector representations of terms, and are computed from unlabelled data, that represent terms in a semantic space in which the proximity of vectors can be interpreted as a semantic similarity. Word embeddings have been shown to produce good results in many works [11, 6] in the comparing of semantic similarities between terms, sentences, paragraphs, and documents. Inspired by these results, we define a bunch of feature functions that are based on the publicly available, pre-trained word vectors that are known as word2vec [15]. In the following sections, we use $v(w)$ to represent a low dimension vector representation of word w in the word2vec setting.

Word2Vec Score is defined as.

$$\phi_{w2v}(s,l) = \frac{1}{|s||l|}\sum_{w\in s}\sum_{w'\in l}\left(\text{tf}(w;s)\times\text{tf}(w;l)\times\max\left\{0, v^T(w)\bullet v\left(w'\right)\right\}\right)$$

It measures the term-level similarity between a short text s and a long text l by incorporating semantic similarity that is measured by the distance between the word vectors for those terms.

Word2Vec Topic Score is defined as.

$$\phi_{w2vt_i}(s,l) = \frac{1}{|l||V_i|}\sum_{w\in l}\sum_{w'\in V_i}\left(\text{tf}(w;l)\times\max\left\{0, v^T(w)\bullet v\left(w'\right)\right\}\right)$$

This function is an extension of the "Topic Score" function to the semantic space. Here we measure the relevance of a long text l to the topic T_i by comparing the term-level semantic similarities that are measured by the distance between the word vectors for those terms.

Word2Vec Differential Topic Score is defined as.

$$\phi_{w2vdt_i}(s,l) = \frac{1}{|l||V_i\setminus s|}\sum_{w\in l}\sum_{w'\in V_i\setminus s}\left(\text{tf}(w;l)\times\max\left\{0, v^T(w)\bullet v\left(w'\right)\right\}\right)$$

This is an extension of the "Differential Topic Score" function to the semantic space. It measures the semantic similarity of a long text l to the topic T_i through the word vectors, without considering the terms in the short text s.

2.3.4 Selective Expansion via the Bias Feature

Our observation shows that not all short texts need to be expanded to improve the accuracy of the task. Forcing expansion on all short texts sometimes reduces the task accuracy. In order to enforce selective expansion, we introduce a *Bias* feature. The bias feature is always set to -1

$$\phi_{bias}(s,l) = -1$$

A short text is expanded only if

$$\sum_f w_f\phi_f(s,l) + w_{bias}\phi_{bias}(s,l) \geq 0$$

That means, similarity score between short and long text has to be above some threshold to force mapping/expansion

$$\sum_f w_f \phi_f(s,l) \geq w_{bias} \quad (3)$$

2.4 Task-specific Learning of Mixture Weights w_f

Feature weights w_f are learned for a given task such that expanding short texts using the mappings obtained from w_f best improves the task accuracy. Let $\mathcal{L}(\mathcal{T}(\mathcal{D}_e))$be the loss function that is defined for task $\mathcal{T}$. It measures the empirical loss of the task on the expanded texts $\mathcal{D}_e$. By varying w_f, the mapping of short texts to long texts changes, which in turn changes the expansions $\mathcal{D}_s$, affecting the loss function. Our aim is to learn w_f such that the loss $\mathcal{L}$ is minimized for the task $\mathcal{T}$. Formally, we solve the following optimization problem:

$$\begin{aligned} &\underset{\mathbf{w}}{\text{argmin}} \mathcal{L}(\mathcal{T}(\mathcal{D}_e)) \\ =&\underset{\mathbf{w}}{\text{argmin}} \mathcal{L}\left(\mathcal{T}\left(\{s_i \oplus \mathcal{M}_w(s_i)\}_{i=1}^n\right)\right) \end{aligned} \quad (4)$$

The form of loss function $\mathcal{L}$ is task dependent and unknown to us. It can be convex or non-convex; linear or non-linear; or can be extremely complex and non differentiable (for example, an output of a deep neural network.) Since our framework has to be task agnostic, we cannot make any assumptions about the form of loss function. Hence we use a derivative-free optimization technique that is known as BOBYQA [18]

BOBYQA solves bound constrained optimization problems without using derivatives of the objective function, which makes it a derivative-free algorithm. The algorithm solves the problem using a trust region method that forms quadratic approximation models of the objective function by interpolation. One new point is computed on each iteration, usually by solving a trust region subproblem subject to the bound constraints, or alternatively, by choosing a point to replace an interpolation point so as to promote good linear independence in the interpolation conditions. BOBYQA constructs the quadratic approximation models by the least Frobenius norm updating technique.

For a non-convex loss function $\mathcal{L}$, BOBYQA finds a solution that is at the local minimum. One of the standard techniques used to overcome the local minimum problem is to adapt random restarts. That is, we start with a random assignment of the weights w_f and run the BOBYQA procedure with that assignment as the starting value. By repeating this procedure multiple times with a different random initialization each time, and picking a solution that produces the least value for $\mathcal{L}$, we can possibly avoid a local minimum and achieve better solutions.

2.5 Alternate Minimization for Task-specific Learning

In this section we throw some insights into the learning that takes place with BOBYQA. Solving Equation 4 with BOBYQA involves evaluating the task $\mathcal{T}$ on the given set of expanded documents. In particular, evaluation of $\mathcal{L}$ involves fitting of the task parameters to the data first and then comparing the task results with the ground truth. It is important to distinguish the task-specific parameters from our model parameters. Let Θ be the parameters of the task (for example, in a classification task, Θ is word/feature weights, and in a clustering task Θ is cluster membership.) and W be our model parameters for the expanding of the short texts. The key idea in using BOBYQA is to learn Θ and W jointly, such that optimal task results are achieved through the optimal mapping of the short texts to the long texts. The procedure for this joint learning is outlined in Algorithm 1.

Algorithm 1 Joint learning of our model parameters W and task parameters Θ

Input: Corpus $\mathcal{C}$, Short Texts Training Set $\mathcal{D}^{(\text{train})}$, Short Texts Development Set $\mathcal{D}^{(\text{dev})}$, Task $\mathcal{T}$
Output: Model parameters W

1: Randomly Initialize $W = [w_f]_{f=1}^{f=(\text{num features})}$
2: **while** not converged **do**
3: Find K candidate mapping long texts for every short text in $\mathcal{D}^{(\text{train})}$ and $\mathcal{D}^{(\text{dev})}$

$$\mathcal{M}_w(s_i) = \underset{l \in \left\{l_i^k \in \mathcal{C}\right\}_{k=1}^K}{\text{argamax}} \sum_f w_f \phi_f(s_i, l)$$

4: Learn task parameters Θ

$$\Theta^* = \underset{\Theta}{\text{argmin}} \mathcal{R}(\Theta) + C \sum_{s_i \in \mathcal{D}^{(\text{train})}} \mathcal{L}_{\mathcal{T}}(e_i; \Theta, W)$$

where $e_i = s_i \oplus \mathcal{M}_w(s_i)$ is the feature vector of the expanded text

5: Quadratically approximate $\sum_{s_i \in \mathcal{D}^{(\text{dev})}} \mathcal{L}(e_i; \Theta^*, W)$ to $\mathcal{L}_Q\left(W; \mathcal{D}^{(\text{dev})}, \Theta^*\right)$. This approximation happens in BOBYQA.

6: Learn our model parameters W by minimizing the quadratic function $\mathcal{L}_Q$

$$W^* = \underset{W}{\text{argmin}} \frac{1}{\left|\mathcal{D}^{(\text{dev})}\right|} \mathcal{L}_Q\left(W; \mathcal{D}^{(\text{dev})}, \Theta^*\right)$$

7: **end while**

Considering a loss regularized framework for task $\mathcal{T}$ and the model parameters W of our framework, the optimal task parameters Θ^* are learned from the data (expanded texts) by optimizing the objective shown in Step 4. Here $\mathcal{R}$ is the regularizer, and $\mathcal{L}_{\mathcal{T}}$ is the task-specific loss function (for example, hinge loss in SVM classification task). Note, $\mathcal{L}_{\mathcal{T}}$ is different from the loss function $\mathcal{L}$ that our framework uses to learn W. In the next step (Step 5), BOBYQA approximates the loss function $\mathcal{L}$ to a quadratic function $\mathcal{L}_Q$, using the interpolation technique. During this process, BOBYQA evaluates the loss function $\mathcal{L}$ at multiple points (W) using the task parameters Θ^*. In Step 6, the quadratic function $\mathcal{L}_Q$ is optimized to find the optimum parameters W^*. The updated weights W then produce new mappings/expansions for the short texts. The procedure repeats until no further update happens to W within the tolerance limit set in BOBYQA.

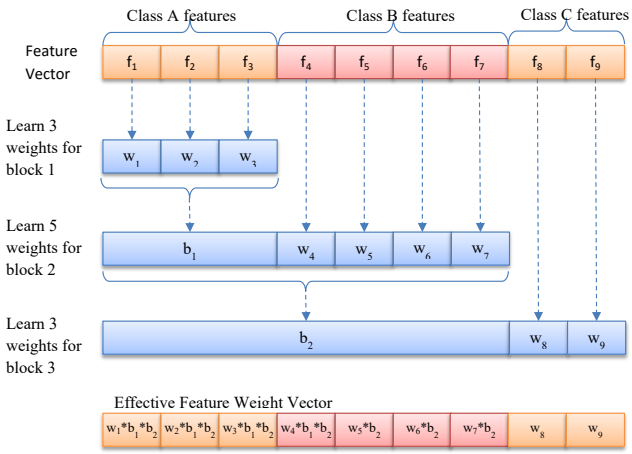

Figure 2: Block-learning approach.

2.6 Practical BOBYQA: The Block Learning Approach

Minimizing the loss function $\mathcal{L}$ using BOBYQA results in multiple evaluations of the function $\mathcal{L}$ during quadratic approximation and the trust region growing/shrinking steps by BOBYQA. Each evaluation of $\mathcal{L}$ calls for the execution of task $\mathcal{T}$. For certain tasks, this evaluation can be time/resource intensive. For example, for a classification task, computing the classification loss for a given dataset involves training the classifier in the training split and then evaluating it on the test split. If the dataset size is very large, it may take a good amount of time to train the classifier. Hence, it becomes important to reduce the number of loss function evaluations.

The number of loss function evaluations depends upon the number of variables in the optimization problem, that is, the dimension of the weight vector W. To improve the learning (of weights w_f) time of the algorithm, we need to reduce the number of task evaluations, without reducing the number of features. In the following section we introduce a novel technique called "Block Learning" to achieve this.

In block learning, we divide our entire set of features into groups of smaller number of features called "blocks" and learn the weights one block at a time. Each block is a set of features of a particular class. For example, IR features constitute the 1st block, topic model features constitute the 2nd block and so on. The learning starts with block-1 in which the BOBYQA finds optimum weights for the features in block-1. Then we move on to block-2, in which BOBYQA learns weights for the features in block-2. At this stage, we treat block-1 as an additional feature and learn one weight for it. Next, we learn weights for features in block-3 using BOBYQA. Now, block-1 and block-2 together are treated as an additional feature and one weight is learned for it. The effective weights of each feature is the product of weights learned during its block and the additional feature weights learned during the subsequent blocks. This is depicted in Figure 2. The weight for feature f1 is $w_1 \times b_1 \times b_2$, where w_1 is the feature weight learned by BOBYQA during the first block; b_1 is the weight learned for the entire block-1 by treating block-1 as an additional feature; and b_2 is the weight learned for the entire block-1 and block-2 by treating block-1 and block-2 together as an additional feature. Similarly, weights for the other features are calculated.

The main advantage of block learning is that, at a time, BOBYQA has to optimize only those variables in a block along with one more variable for previous blocks. This reduces the number of evaluations of loss function $\mathcal{L}$ that BOBYQA makes, speeding up the learning process. Though block learning is not equivalent to learning all weights together, empirically, we observe that the task accuracy with expansion using block learned weights is at par with the accuracy from the expansion that uses weights learned with all features together. However, the total number of BOBYQA evaluations are significantly reduced with block learning, thus reducing the learning time.

3. EXPERIMENTS AND EVALUATIONS

In order to evaluate our approach, we compare our work with several baselines and earlier works in the literature, which describe the handling of short texts. We demonstrate the effectiveness of our approach on two different ML tasks: classification and clustering. Our experimental results show that the proposed approach produces results that are comparable to the state-of-the-art techniques and that it is generic enough for application to many ML tasks. We have named our approach as TIDE (**T**ask-spec**I**fic short **D**ocument **E**xpansion), which has been used throughout these experiments and evaluations.

3.1 Short Text Tasks

In this paper, we evaluate classification and clustering tasks for short texts. Though our technique can be applied to other types of ML tasks, we find that classification and clustering tasks on short texts have a good presence in the literature, which gives us an opportunity to use the standard data sets and compare against the baselines and earlier works. In these tasks, we represent short and long texts as TF-IDF vectors. To expand a short text s_i using a long text l_i, we smooth the TF-IDF vector of the short text using that of the long text in the following manner: $e_i = \alpha s_i + (1-\alpha)\, l_i$ The $0 \leq \alpha \leq 1$ is a smoothing parameter that controls how much importance has to be given to short text and long texts. When $\alpha = 1$, only the short text used; when $\alpha = 0$, only the long text is used. The value of α is learned as part of the optimization in Equation 4. That is, we optimize the following to learn $\mathbf{w}$ and α:

$$\begin{aligned} &\underset{\mathbf{w},\alpha}{\text{argmin}} \mathcal{L}\left(\mathcal{T}\left(\mathcal{D}_e\right)\right) \\ =&\underset{\mathbf{w},\alpha}{\text{argmin}} \mathcal{L}\left(\mathcal{T}\left(\left\{\alpha s_i + (1-\alpha)\, \mathcal{M}_w\left(s_i\right)\right\}_{i=1}^{n}\right)\right) \end{aligned}$$

Note that short text is expanded only if similarity score is above a threshold as explained in Equation 3. Using BOBYQA and the block learning machinery that is explained in Section 2.4, the optimization problem above is solved for $\mathbf{w}$ and α.

3.2 Datasets

We report our experimental results on several benchmark datasets used in the literature for classification and clustering tasks. We give an overview of the datasets, below.

3.2.1 Reuters21578

Reuters21578 dataset is a collection of news articles. Each article has a short title and a long description of the news. All articles are classified into various topics. We take the

articles classified as "Corn" or "Wheat" and consider their titles as short texts. The task here is to classify the short titles into the Corn or Wheat class. The choice of these two classes is due to their high inter-class confusion while classifying short titles. We use the long descriptions (without the title in it) from the articles to form the universal corpus $\mathcal{C}$. The corpus is then indexed using Lucene software. Using Lucene as an IR system, we retrieve the top K candidate long texts by using the title as a short text query. The weights w_f are learned in order to choose a mapping long text for every short text such that expanding the short text by using the long text improves the classification accuracy.

In this dataset, we know the *true* long text for a short text (title), which is the long description of that news article. Using this true mapping we can compute a best classifier using the *true* long texts of the short texts. This helps us to compare our technique against the *true* expansion.

3.2.2 News Dataset from TagMyNews

TagMyNews[3] datasets is a collection of datasets of short text fragments that are used for the evaluation of the topic-based text classifier. One of the dataset from this collection is the News dataset. This is a dataset of ˜32K English news that is extracted from RSS feeds of popular newspaper websites (nyt.com, usatoday.com, reuters.com). Each news snippet has a title and a very short (one or two lines) description of the news. Every news snippet is classified into one of the following categories: Sport, Business, U.S., Health, Sci&Tech, World and Entertainment. We use the titles and the short descriptions as the short texts for the classification task.

3.2.3 Web-Snippets

Web-Snippets [17] dataset has around 12K short web search snippets that are classified into seven classes. Out of this, 10K short texts are used to train the classifier and 2K for testing.

3.2.4 ODPTweets

ODPtweets[4] is a large-scale Twitter dataset with nearly 25 million tweets that are categorized in the structure of the Open Directory Project (ODP). This dataset was used for the tweets classification task in WWW work [26]. The categorization of tweets is inferred from the links that they point to. From the ODP style category structure that is associated with each tweet, we extracted the top-level ODP category as the category of the tweet. For example, for one of the tweets, the ODP category structure associated is "Computer/Software/Programming/Java". The top-level category is "Computer" in this case, which we associate with the tweet as the true category. There are 15 top-level categories in this dataset (Computer, Health, etc.). For each of these categories, we extracted around 600 tweets using Twitter API. We then discarded tweets that contained only junk characters or less than three words or non English tweets. This gave us as a collection with around 500 tweets in each category.

3.2.5 StackOverflowQuestions

We use the challenge data published in kaggle.com[5] [23] that contains questions that are posted by the users on stackoverflow.com[6]. This dataset consists of 3,370,528 questions posted on stackoverflow.com from July 31, 2012 to August 14, 2012. In our experiments, we randomly select 20,000 question titles as short texts from 20 different tags, as done in [23].

3.3 Corpus

Choosing the right universal corpus is very important. First, the universal corpus, as its name implies, must be large and rich enough to cover a lot of words, concepts, and topics that are relevant to the task problem. Second, this corpus should be consistent with the training and future unseen data that the task will deal with. This means that the nature of universal data (for example, patterns, statistics, and their co-occurrence of them) should be observed by humans to determine whether or not the potential topics analyzed from this data can help in making the task more robust. For example, the universal corpus has to help make the classifier more discriminative.

Today, Wikipedia is known as the richest online encyclopedia written collaboratively by a large number of contributors around the world. A huge number of documents are available in various languages and are placed in a organized structure which inspires the WWW, IR, and NLP research communities to use it as a large corpus [7].

Another data collection that can be used as a universal corpus in the medical domain is Ohsumed/MEDLINE. Unlike Wikipedia, Ohsumed only includes medical abstracts. This corpus can be used for tasks in the medical domain.

We use Wikipedia articles as a universal corpus $\mathcal{C}$ of long texts for experiments with TagMyNews, Web-snippets, ODP-Tweets and StackOverflowQuestioins datasets. We use the Lucene software to index these articles and retrieve the top K candidate long texts using short texts as queries. The weights w_f are learned such that expanding the short texts using the long texts (Wikipedia articles) improves the classification and clustering accuracy.

3.4 Robustness Analysis of Our Approach

One of the important characteristics of our approach is its robustness to the noise in the corpus. Unlike other approaches [1, 20, 25] our method does not force expansion on all the short texts. Expansion is done only if that helps improving the performance of the task. When a wrong corpus (one from a different domain) or a noisy corpus is used, it may not provide the right long texts for the expansion and, hence, expansion may not boost the performance of the task. In our experiments, we demonstrate the robustness of our method against the noisy corpus from the following three scenarios: (i) a wrong corpus (ii) a corpus that is partially relevant or has noise and (iii) a corpus that contains the short texts themselves. In the third scenario, we show that our method is capable of avoiding the selection of short texts from the corpus for expansion, whereas, other IR-based expansion techniques result in the selection of short texts themselves from the corpus for the expansion, which does not help the task to improve the performance.

3.5 Evaluation Methodology

All the ML tasks are carried out by expanding the short texts and by measuring the improvement in the task's per-

[3]http://acube.di.unipi.it/tmn-dataset/
[4]http://www.zubiaga.org/datasets/odptweets/
[5]http://www.kaggle.com/
[6]http://www.stackoverflow.com/

Expt#	Dataset	Short Text	Corpus
1	Reuters21578	News Title	Articles from entire Reuters21578 collection
2	Web-Snippets	Snippet	Wikipedia articles
3	TagMyNews	Title + RSS feed	Wikipedia articles
4	ODPTweets	Tweet	Wikipedia articles
5	TagMyNews	Title + RSS feed	Wikipedia articles + all short texts from the dataset
6	TagMyNews	Title + RSS feed	Wikipedia articles + Ohsumed
7	TagMyNews	Title + RSS feed	Ohsumed

Table 1: Experiments for the classification task using different datasets and corpora

Expt#	Accuracy (%)						% Short Texts Not Expanded
	Short Text Only	Lucene First	Word2Vec	TIDE	Comparing Technique	Actual	
1	85.8	87.2	67.5	91.6	-	92.1	14
2	62.1	76.8	52.9	84.2	82.2 (Phan [17]) 85.3 (Chen [5])	-	8
3	71.3	73.7	56.3	81.3	80 (Vitale [22])	-	11
4	39.4	41.3	21.2	47.8	-	-	0
5	71.3	71.8	56.3	81.1	-	-	0
6	71.3	73.1	56.3	81	-	-	10
7	71.3	62.3	56.3	70.1	-	-	92

Table 2: Accuracy comparison of classification task (TIDE is our approach)

formance. We compare our expansion technique against various baselines and previous works. The two baselines we compare against are (i) IR system-based expansion and (ii) Word2Vec-based expansion

In an IR system based expansion, the long text articles in the corpus are initially indexed. Using the short text as a query, the top ranked result from the IR system is used to expand the short text. In our experiments, we used Lucene as the IR system.

In Word2Vec based expansion, a word vector for every word in the short text is obtained using the word2vec tool. The average word vector is then computed from all these word vectors and appended to the short text to produce a long/expanded text.

3.5.1 Methodology for Classification Task

There are various criteria that can determine the effectiveness of a classification task; however, precision, recall, and accuracy are most often used. We choose accuracy (macro accuracy in case of multi-class classification) to measure the performance of the task, though other criteria may equally be used. In fact, it does not matter which criteria we choose because the goal of our experiments is to demonstrate an improvement in the task's performance using our technique and not to judge the task itself.

The datasets described in Section 3.2 are divided into train and test splits according to the previous works using those datasets. We use 25% of the data from the test split as the development set and the remaining as the test set for evaluation. During the training phase, the model parameters (of both our framework and classier) are optimized by training the classifier on the expanded short texts from the training set and measuring the accuracy on the expanded short texts from the development set. In the testing phase, expanded short texts from the test set are classified and accuracy is measured.

We report the results for the task using the SVM classification algorithm, however, we observed a similar result when using other classification algorithms.

3.5.2 Methodology for Clustering Task

We test our algorithm on the StackOverflowQuestions dataset. The clustering performance is evaluated by comparing the clustering results of short texts with the tags/labels provided by the text corpus. The accuracy [4] and NMI metrics[14] are used to evaluate the clusters.

Since the focus our investigation is to demonstrate the improvement in the clustering accuracy of short texts through our expansion technique rather than the clustering method itself, we used the standard k-means algorithm whose implementation is readily available in many of the ML packages (which, is also the algorithm used in the work we compare against.) In other words, we believe our technique can be used with any other clustering method.

3.6 Results and Discussion

3.6.1 Classification Accuracy

To investigate the accuracy improvement of a short text task using our approach, we have designed several experiments, as shown in Table 1. For each of these experiments Table 1 shows the dataset, the short texts, and the universal corpus used. Experiments 1-4 show expansion using the right corpus and 5-7 show the robustness of our approach against the incorrect or noisy corpus.

Table 2 shows classification accuracies of our method against other baselines. In the case of Reuters21578 (Experiment 1) we use Reuters21578 articles as the corpus. That means, the true expansion of the short text (title) is present in the corpus. We consider the body of a news article as the true expansion of the news title. This experiment lets us investigate how close our approach can perform to the true (oracular) expansion. Interestingly, we observe that our method achieves 91.6% accuracy which is close to the true expansion accuracy of 92.1%.

Baselines/Comparing techniques	Accuracy(%)	NMI(%)
Short Text Only	26.3	30.2
Lucene First	38.8	40.1
Word2Vec	11.4	13.6
Jiaming Xu et.al [23]	51.1	49
TIDE	50.8	52.4

Table 3: Accuracy and NMI comparison of clustering task (TIDE is our approach)

In Experiment 5, the corpus contains exact short texts from the dataset as short articles. The IR method retrieves these short texts as top results due to the high matching score and, hence, does not help the expansion. Experiment 6 has the corpus with the noise: Wikipedia articles mixed with Ohsumed articles. We observe that the Wikipedia articles provide the right candidates for expansion, whereas Ohsumed articles are irrelevant (noise) for the news snippets in the TagMyNews dataset. The IR method does a good job of selecting only relevant candidates for the expansion from the corpus and discard the noisy candidates. In addition, our model assists in the selecting of expansion texts, which improves the task accuracy. Experiment 7 has an irrelevant corpus for the expansion. As we can see from the results, Lucene First (IR method) is forced to choose a best-matching candidate for the expansion. Whereas, forcing this expansion reduces the task accuracy. However, our model's selective expansion strategy makes the short texts not expand in such scenarios. As shown in Table 2, the percentage of short texts not expanded is up to 92% in this case.

Table 2 also shows comparison of classification accuracies reported in short text classification techniques from the literature with our method. In most of the cases, our method performs at par with other techniques or beats them. In comparison to [5], we perform slightly worse. We believe this is due to the differences between the corpus used in our method and [5]. The seed words that are used for crawling the Wikipedia and the process of building the corpus are not clear from [5]. We used all the Wikipedia articles in our experiments; however, we believe that the accuracy of our method would improve if we build a focused crawler to generate more relevant corpus.

3.6.2 *Clustering Accuracy and NMI*

The results of the clustering task are shown in Table 3. We observe that our method beats all the baselines and performs at par with [23]. The Lucene First and Word2Vec-based expansions do not consider the clustering accuracy and NMI during expansion, our method, however does. This leads to better task performance when compared to these baselines.

Note that the framework and features used for the clustering task is same as that of the classification task. Hence, we state that our framework is task agnostic. However, it learns to expand the short texts for the classification and the clustering tasks in a way that improves the task's performance. This makes our framework powerful for adaption by many IR/ML/NLP tasks that deal with short texts.

3.6.3 *Effect of Block Learning*

In the next set of experiments, we investigate the effect of the block learning mechanism. For the classification task, we run the experiments with and without block learning in order to investigate (i) the amount of time that block learning saves and (ii) the effect on classification accuracy. Figure 3 shows that in some cases there are 30% lesser task evaluations with block learning, with a marginal drop in the classification accuracy. Note that we report the saving in training time w.r.t. the number of task evaluations, because the absolute time for a task depends upon various factors such as the size of the dataset, the task training/validation methodology, and the like. Interestingly, in the case of TagMyNews, we observe a slight increase in the classification accuracy with block learning. We believe that this is because of our model settles for a local minimum in a high-dimensional feature space when all the features are used together as one block (no block learning). Whereas, when block learning is employed, the reduction in the feature space dimension helps to find better solutions.

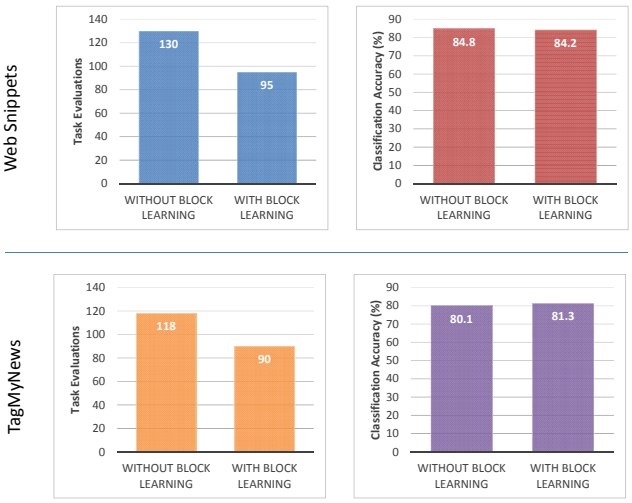

Figure 3: Block Learning: Reduction in the number of task evaluations and impact on accuracy

3.6.4 *Feature Ablation*

In this section we investigate the usefulness of different classes of feature functions through feature ablation tests. We start with only IR features and then incorporate topic model and embedding features one by one. We then compare how the learning of expansion improves the classification task accuracy with the addition of these features. Figure 4 shows the improvement in classification accuracies as we add different classes of features. IR based features alone are able to achieve a gain of around 5% in accuracy over Lucene first, followed by another $4-5\%$ gain through topic model-based features. While IR features can be computed very easily, topic model features require a one time computation of the topic distribution of the corpus from LDA or similar mechanisms. There is a marginal improvement when word2vec features are used. In these experiments, we use pre-built word vectors (of 300 dimension) published by Google. Experiments using word vectors trained from the corpus and of different dimensions will be part of our future work.

3.6.5 *Random Restart Results*

To overcome the problem of the local minimum with our approach, we adapt a standard technique of random restarts. In this section we investigate the effect of random restarts on

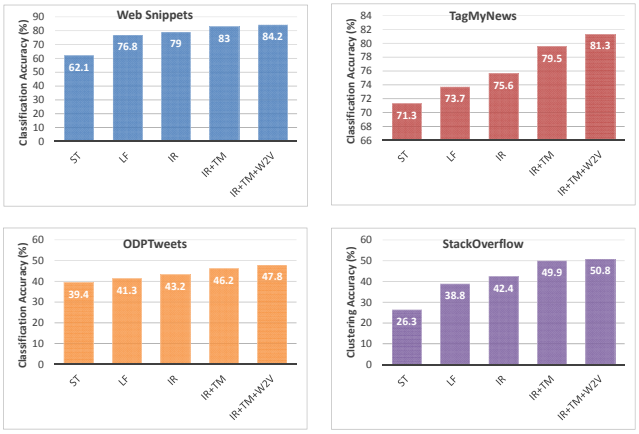

Figure 4: Feature Ablation Results

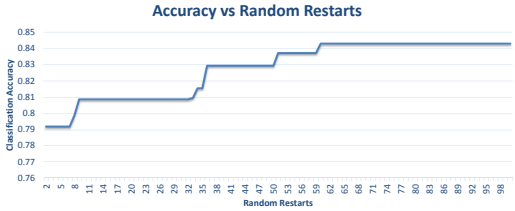

Figure 5: Classification Accuracy improvement with Random Restarts in experiments using Web-snippets dataset (Experiment 2)

the classification accuracy of the task by plotting the best classification accuracy that is observed so far against the number of random restarts. Figure 5 shows the improvement in the accuracy over multiple random restarts. We observe that in about $65-75$ random restarts, we reach the maximum task improvement that can be achieved by our method.

4. CONCLUSION

In this work we presented a technique for learning to expand short texts in a task-specific way. The expansion is such that the task accuracy best improves when expanded texts are used. We do not make any assumptions regarding the tasks except that the task can be evaluated with the expanded texts. Hence, our technique can be adapted to expand short texts for any task. To learn task-specific expansion, we presented several classes of mapping feature functions: IR, topic model and embedding-based. Using a derivative-free optimization technique known as BOBYQA, we presented the efficient learning of feature weights in a block learning fashion. As part of our future work, we plan to investigate performance improvement in other types of task by using our framework.

5. REFERENCES

[1] Measuring semantic similarity between words using web search engines. WWW, 2007.

[2] S. Banerjee, K. Ramanathan, and A. Gupta. Clustering short texts using wikipedia. SIGIR, 2007.

[3] D. M. Blei, A. Y. Ng, and M. I. Jordan. Latent dirichlet allocation. 2003.

[4] D. Cai, X. He, and J. Han. Document clustering using locality preserving indexing. *KDE*, 2005.

[5] M. Chen, X. Jin, and D. Shen. Short text classification improved by learning multi-granularity topics. IJCAI, 2011.

[6] A. M. Dai, C. Olah, and Q. V. Le. Document embedding with paragraph vectors. *arXiv preprint arXiv:1507.07998*, 2015.

[7] L. Denoyer and P. Gallinari. The wikipedia xml corpus. ACM SIGIR Forum, 2006.

[8] S. Fern and M. Stevenson. A semantic similarity approach to paraphrase detection. CLUK, 2008.

[9] R. Ferreira, R. D. Lins, F. Freitas, S. J. Simske, and M. Riss. A new sentence similarity assessment measure based on a three-layer sentence representation. DocEng, 2014.

[10] A. Huang, D. Milne, E. Frank, and I. H. Witten. Clustering documents using a wikipedia-based concept representation. PAKDD, 2009.

[11] T. Kenter and M. de Rijke. Short text similarity with word embeddings. CIKM, 2015.

[12] J. Lafferty and C. Zhai. Document language models, query models, and risk minimization for information retrieval. ACM, 2001.

[13] M. C. Lintean and V. Rus. Measuring semantic similarity in short texts through greedy pairing and word semantics. FLAIRS, 2012.

[14] C. D. Manning, P. Raghavan, H. Schütze, et al. *Introduction to information retrieval*. Cambridge university press Cambridge, 2008.

[15] T. Mikolov, K. Chen, G. Corrado, and J. Dean. Efficient estimation of word representations in vector space. 2013.

[16] J. Pennington, R. Socher, and C. D. Manning. Glove: Global vectors for word representation. EMNLP, 2014.

[17] X.-H. Phan, L.-M. Nguyen, and S. Horiguchi. Learning to classify short and sparse text & web with hidden topics from large-scale data collections. WWW, 2008.

[18] M. J. Powell. The bobyqa algorithm for bound constrained optimization without derivatives. *Cambridge NA Report NA2009/06, University of Cambridge, Cambridge*, 2009.

[19] V. Rus, N. Niraula, and R. Banjade. Similarity measures based on latent dirichlet allocation. CICLing. 2013.

[20] M. Sahami and T. D. Heilman. A web-based kernel function for measuring the similarity of short text snippets. WWW, 2006.

[21] P. Schonhofen. Identifying document topics using the wikipedia category network. WI, 2006.

[22] D. Vitale, P. Ferragina, and U. Scaiella. Classification of short texts by deploying topical annotations. ECIR, 2012.

[23] J. Xu, P. Wang, G. Tian, B. Xu, J. Zhao, F. Wang, and H. Hao. Short text clustering via convolutional neural networks. NAACL-HLT, 2015.

[24] S. Yagcioglu, E. Erdem, A. Erdem, and R. Cakıcı. A distributed representation based query expansion approach for image captioning. 2015.

[25] W.-T. Yih and C. Meek. Improving similarity measures for short segments of text. AAAI, 2007.

[26] A. Zubiaga and H. Ji. Harnessing web page directories for large-scale classification of tweets. WWW, 2013.