on heuristics or extensive tuning. Our network uses the Gated Recurrent Unit (GRU) [3] as a principled way to combine the two sources of information, in which the weights are automatically determined for each short text. Similar to a human user who may continually update a search query for multiple rounds, our deep memory network also allows to expand the short text multiple times by using the reformulated short text as the next input. The final representation of short text is fed to a downstream learning task. In this paper, we take the task of short text classification as a demonstrative example. By optimizing the classification objective, the whole network is trained end-to-end by backpropagation.

We evaluate the proposed deep memory network using short texts of different genres, including titles of Wikipedia articles (general domain), titles of computer science publications (scientific domain), and tweets (social media domain). Experimental results show that it significantly outperforms classical text expansion methods and text classification methods that only use short text features.

To summarize, we make the following contributions:

- We propose a novel end-to-end solution for short text expansion. A deep memory network-based model is designed to gather useful information from relevant documents and integrate it with the original short text.
- We conduct extensive experiments on real-world short text data sets. Experimental results on short text classification show that our proposed deep memory networks significantly outperforms the classical query expansion methods and the methods which only use the features in short text.

Organization. The rest of this paper is organized as follows. Section 2 discusses the related work. Section 3 describes our end-to-end solution for short text expansion. Section 4 reports the experimental results, and we conclude the paper in Section 5.

2 RELATED WORK

Our work is related to three lines of research in literature: text representation, memory networks, and query/short text expansion.

2.1 Text Representation

Distributed representations of text have been proven to be very effective in various natural language processing tasks. These approaches can be roughly classified into two categories: unsupervised approaches (e.g., Skip-gram [20] and ParagraphVEC [16]) and supervised approaches (e.g., convolutional neural networks (CNN) [13], recurrent neural networks (RNN) [8], PTE [30], and FastText [12]). The representations learned by the unsupervised approaches are very general and can be applied to different tasks. However, their performance usually falls short on specific tasks since no supervision is leveraged when learning the representations. The supervised approaches have shown very promising results on different types of text corpus. For example, FastText [12] and PTE [30] achieve state-of-the-art results for text classification on long documents while on short documents, CNN [13] and RNN [8] achieve state-of-the-art results. All these approaches focus on learning text representations with the raw features, and no additional data is leveraged.

2.2 Memory Networks

Another line of related work are memory networks [9, 15, 23, 29], which use the attention and memory mechanism in deep learning models. For example, Sukhbaatar et al. proposed an end-to-end memory network [29] for question answering tasks. Given a question and contexts relevant to the question, the memory network employs a recurrent attention model over the contexts to iteratively identify relevant contexts for answering the question. Different variants of memory networks [15, 23] are proposed with different attention and memory updating mechanisms.

Compared to these works, the current paper differs in several aspects: (1) most of the work on memory network focuses on question answering while our work studies a very different application: short text expansion and classification; (2) in the setting of question answering, the number of contexts for each question is very limited while in our setting, for each short text, the entire collection of documents are used as potential relevant contexts. (3) the soft attention mechanism is usually used in these works while in our work we investigate both soft and hard attention mechanisms. In the experiments, we adapt the end-to-end memory networks to our task and compare it with our approach.

2.3 Short Text Expansion

Our approach uses the search results from a large collection to expand short texts. This general strategy has been applied in many data mining tasks, notably query expansion with relevance feedback [2, 5, 34], semantic relatedness analysis [7, 27], short text classification [10, 25], and question answering [4, 28]. The expanded text usually takes the form of interpolation between the original short text and the retrieved documents, which is then used in downstream tasks, such as retrieval and classification. Because the retrieved documents often contain noise, and the interpolation weights are often set by heuristics, the errors may accumulate in the pipeline and harm the performance of an end task. This problem is known as query drift [18] in query expansion.

Compared to previous work on short text expansion, we take an end-to-end approach to train the text expansion algorithm towards a clear learning objective. This turns short text expansion into an optimization problem and eliminates the need for extensive tuning of the interpolation weights [17].

3 DEEP MEMORY NETWORK FOR SHORT TEXT EXPANSION

To understand a piece of short text, the common practice of a Web user is to formulate the short text as a search query, and then seek for definition, examples, paraphrases, and contexts in the returned Web pages. In other words, the Web user is leveraging the huge collection of Web pages for short text understanding. Therefore, in this paper, we study how to leverage external documents for expanding the representations of short texts and understanding its meaning. We take short text classification as an example of such task. Our problem is formally defined as follows:

Definition 3.1. **(Problem Definition.)** *Given a collection of long documents C, we aim to learn a function f that expands a short text q into a richer representation q', i.e., $q' = f(q, C)$. Based on the richer*

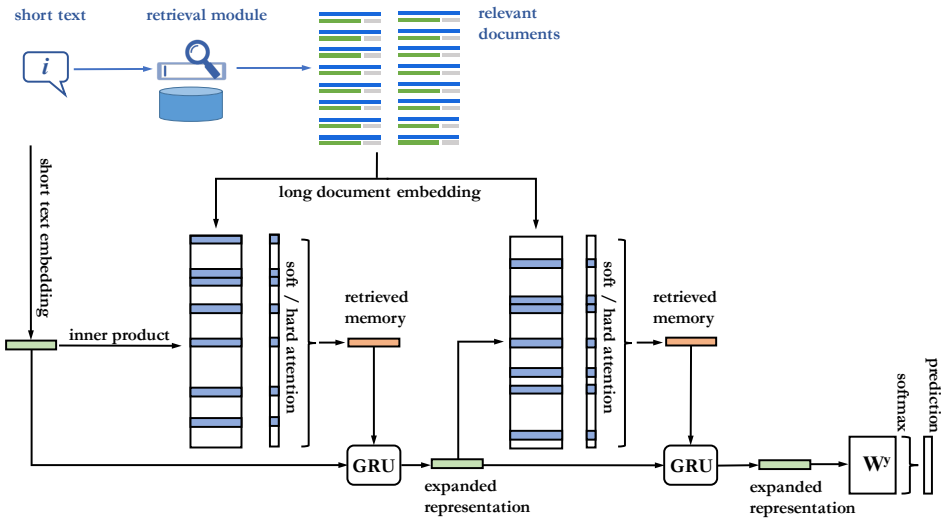


Figure 1: ExpaNet model structure (2 hops).

representation q', we can accurately classify the short text into one of the predefined categories $\mathcal{Y}$.

Leveraging long documents for short text expansion has been widely studied in information retrieval literature, where a query is expanded by leveraging the top-K documents returned by the retrieval systems, a method known as pseudo relevance feedback [34]. In such a method, top-K documents are usually retrieved with a search function, and then some terms are selected from the top-K documents using heuristic methods such as TFIDF weighting and probability weighting, which are added back to the original query. However, these methods are not accurate since the returned top-K documents and terms selected from the top-K documents can be noisy and cause topic drift.

In this section we introduce ExpaNet, an end-to-end solution based on deep memory network. Our approach shares similar intuition with pseudo-relevance feedback but is much more principled. It can be trained to automatically identify the relevant documents for the given query (short text) and filter out non-relevant ones.

ExpaNet integrates five different modules including retrieval module, short text representation module, long document representation module, expansion module, and classification module. Since the long document collection C can be huge, the *retrieval module* provides an efficient way to retrieve a small subset of potentially relevant documents C_q for the given short text q. The *short text representation module* maps the short text q into a continue representation $\vec{q}$. The *long document representation module* represents each document $d \in C_q$ with a continuous representation $\vec{d}$ and put it into the memory M. The *expansion module* expands the $\vec{q}$ into a new representation $\vec{q}'$ by leveraging the memory M using multiple hops. Finally, the *classification module* predicts the category with the expanded representation $\vec{q}'$ as input. All these modules are coupled together and trained through error backpropagation. Next, we introduce these different modules respectively.

3.1 Retrieval Module

In practice, the external long document collections C can be very large. For example, the entire Web contains billions of Web pages, and the entire Wikipedia contains millions of articles. For a short text q, only a few documents from the collection C would be relevant to the query. Therefore, we first use the original short text q as a query to search for a set of potentially relevant long documents C_q from an external large collection C. The documents will be used by the model as the "raw material" for text expansion. The goal of this step is to obtain relevant documents efficiently and with high recall. This process can be implemented efficiently with existing techniques such as an inverted index used in information retrieval, locality sensitive hashing for high-dimensional data points, and directly making use of APIs provided by existing search engines. To ensure a high recall, one can set the number of returned documents to be reasonably large, e.g., tens or hundreds of documents.

3.2 Short Text Representation Module

We represent each short text $q = w_1, \ldots, w_n$ as a d-dimensional vector $\vec{q}$ in a continuous space. Each word in the vocabulary is represented as a d-dimensional vector, and then the entire short text is represented as the average vector of words in the short text, i.e.,

$$\vec{q} = \frac{\sum_{i=1}^{n} \mathbf{A}_{w_n}}{n} \tag{1}$$

where $\mathbf{A} \in \mathbb{R}^{d \times V}$ is the word embedding matrix, V is the size of the vocabulary. There could be more sophisticated ways to encode a piece of text (such as with convolutional neural networks [13] or recurrent neural networks [24]). We choose the simple averaging approach as it was shown to work well in our previous work [30] and much easier to train.

3.3 Long Document Representation Module

Each long document is also represented as a d-dimensional vector space. Similarly, each document $d_i = w_1, \ldots, w_n$ are represented as the average vector of the words in the documents, i.e.,

$$\vec{d}_i = \frac{\sum_{i=1}^{n} \mathbf{B}_{w_n}}{n}, \tag{2}$$

where $\mathbf{B} \in \mathbb{R}^{d \times V}$ is word embedding matrix for long document representations.

3.4 Expansion Module

The expansion module is the core part of ExpaNet. The goal is to expand the continuous representation of input short text $\vec{q}$ by incorporating the information in the memory $M = \{\vec{d}_i\}_{i=1}^{K}$, where K is the number of documents in the memory. The expansion process can be divided into two different components: (1) given the query representation $\vec{q}$, what information should we read from the memory? (2) how to integrate the information from the memory with the original query representation $\vec{q}$?

3.4.1 Memory Reading. For memory reading component, we aim to identify the relevant documents to the given query $\vec{q}$. Here, we use the attention mechanism. Two types of attention mechanisms are used: soft attention [29] and hard attention [22].

Soft attention: Soft attention is widely used in existing memory networks. We use the same mechanism as [29]. The relevance between the query $\vec{q}$ and each document $\vec{d}_i$ is calculated as their inner product, and a softmax function is used to define the attention probability over each document i in the memory, i.e.,

$$a_i = \text{Softmax}\left(\vec{q}^{\top}\vec{d}_i\right), \tag{3}$$

where $\text{Softmax}(z_i) = e^{z_i}/\sum_j e^{z_j}$. In this way, the a_i's define a probability distribution over the long documents in memory M, and the information read from the document is defined as:

$$\vec{o} = \sum_{i=1}^{K} a_i \vec{d}_i \ . \tag{4}$$

Hard attention: Instead of looking at each document with some probability, a human searcher often picks a document that seems relevant and focus on it. Therefore, we also investigate using hard attention here [22], which is achieved by randomly sampling a document from the probability distribution $\vec{a} = (a_1, \ldots, a_K)$ defined in the soft attention, i.e.,

$$\vec{p} \sim \text{multinomial}(\vec{a}), \tag{5}$$

where $\vec{p}$ is a one-hot vector. Then the information read from the memory is defined as:

$$\vec{o} = \sum_{i=1}^{K} p_i \vec{d}_i. \tag{6}$$

However, as mentioned in [22], training a hard attention model is very hard, which has a high variance of the gradients (e.g., the REINFORCE [32] algorithm), and complicated variance reduction methods [33] must be used. In this paper, we use a recent technique, the Gumbel-Softmax [11], for backpropagating through samples, which has a low gradient variance. Specifically, each sample is drawn according to the following distribution:

$$p_i = \text{Softmax}\left(\frac{\vec{q}^{\top}\vec{d}_i + g_i}{\tau}\right), \tag{7}$$

where g_i follows the Gumbel(0,1) distribution, and τ is the temperature hyperparameter (τ is set as 2.0 in our experiments). For more details about the Gumbel-softmax distribution, readers can refer to [11].

3.4.2 Short Text Expansion. With the *memory reading* component, the model is able to retrieve relevant information, i.e., $\vec{o}$, from the memory. Then how should we reformulate the short text? It is natural to integrate information from the original representation $\vec{q}$, and information retrieved from the memory $\vec{o}$. Indeed, a typical method for query expansion in information retrieval is to interpolate between the original query and the expanded document [26, 34], where a scalar parameter is used to trade off between the two information and empirically tuned, which is based on heuristics. Here, we use a principled method to integrate the two sources of information. We use a gating mechanism, the Gated Recurrent Unit (GRU) [3], to combine the information, which is able to automatically determine the weight of the two sources of information. Specifically, the two sources of information are integrated as follows:

$$\vec{z} = \sigma\left(\mathbf{W}^{(z)}\vec{q} + \mathbf{U}^{(z)}\vec{o}\right) \ ; \tag{8}$$

$$\vec{r} = \sigma\left(\mathbf{W}^{(r)}\vec{q} + \mathbf{U}^{(r)}\vec{o}\right) \ ; \tag{9}$$

$$\vec{o}' = \tanh\left(\mathbf{W}\vec{q} + \vec{r} \circ \mathbf{U}\vec{o}\right) \ ; \tag{10}$$

$$\vec{q}' = (\mathbf{1} - \vec{z}) \circ \vec{q} + \vec{z} \circ \vec{o}' \ , \tag{11}$$

where $\circ$ denotes elementwise multiplication and $\sigma(x) = 1/(1 + \exp(x))$, $\tanh(x) = (1 - \exp(-2x))/(1 + \exp(-2x))$ are both elementwise operations. $\vec{o}'$ is the new information from the memory, which is determined by both sources of information $\vec{q}$ and $\vec{o}$. $\vec{z}$ is the weighting vector between the original information $\vec{q}$ and the new information $\vec{o}'$. The output $\vec{q}'$ is the expanded representation of the input short text q.

3.4.3 Iterative Expansion via Multiple-hops. When a Web user inputs a query and reads the search results, the user may reformulate the query. This process can continue several times until the user understands the query. Our algorithm also tries to simulate this process, which is achieved through the recurrent attention mechanism using multiple hops in the *short text expansion* component. More specifically, when a expanded representation $\vec{q}'$ is output by the *short text expansion* component, the representation $\vec{q}'$ is treated as an initial query to the module. This process can be repeated several times, and the final output representation is used as the representation of the original query q.

In practice, when the query is updated, one may ask that the set of relevant documents should be re-retrieved from the entire collections. However, as mentioned previously, the initial set of retrieved documents have a very high recall, which means that the relevant documents to the new query are very likely to belong the initial retrieved set. Only the weights between the query and

the documents need to be updated, which is taken care of by the attention mechanism.

3.5 Classification Module

As in classical methods for query expansion, we keep the original short text representation $\vec{q}$ and represent the final short text representation as a concatenation of $\vec{q}$ and the expanded representation $\vec{q}'$, i.e., $\vec{q}_{\text{final}} = [\vec{q}, \vec{q}']$, which is then used to predict the category of the short text. A fully connected layer is first applied to the short text representation and then followed by a Softmax transformation, yielding a distribution over the categories, i.e.,

$$p(y|\vec{q}_{\text{final}}) = \text{Softmax}(\mathbf{W}^y \vec{q}_{\text{final}}), \tag{12}$$

where $\mathbf{W}^y \in \mathbb{R}^{|\mathcal{Y}| \times 2d}$ is the parameter for fully connected layer.

3.6 Training

In this paper, we take the example of short text classification as the goal of short text expansion. Therefore, the ultimate goal is to accurately predict the category of the short text, and the cross entropy loss function is used. Specifically, given a training data set (q_i, y_i) and a document collection C, we aim to minimize the loss:

$$\sum_{i=1}^{n} \sum_{y \in \mathcal{Y}} \mathbf{1}_{\{y=y_i\}} \log p\,(y|q_i, C)\,, \tag{13}$$

where $p\,(y|q_i, C)$ is the probability of class y given short text q_i and long document collection C, predicted by the network. The whole network is trained by backpropagation including the word embeddings $\mathbf{A}$, $\mathbf{B}$, weights of fully connected network $\mathbf{W}^y$ for classification, and parameters $\mathbf{W}^{(z)}, \mathbf{U}^{(z)}, \mathbf{W}^{(r)}, \mathbf{U}^{(r)}, \mathbf{W}, \mathbf{U}$ in the GRU.

4 EXPERIMENTS

In the experiments, we compare our algorithm (ExpaNet) to state-of-the-art methods for short text classification and classical methods of query expansion. On three real world data sets, ExpaNet shows superior performance. We also analyze the effect of retrieval collection choice, parameter sensitivity, and attention distribution.

4.1 Data Sets

We test our algorithm on three different genres of short texts. Basic statistics of these data sets are summarized in Table 1 and 2.

Wikipedia. Titles of Wikipedia articles represent short texts in the general domain. The length of Wikipedia titles is on average 3.12 words, similar to that of search queries [1]. We take a recent snapshot of English Wikipedia[1] to construct this data set. We use 15 categories in the main topic classifications of Wikipedia[2] as our labels: "Arts", "Games", "Geography", "Health", "History", "Industry", "Law", "Life", "Mathematics", "Matter", "Nature", "People", "Religion", "Science and Technology", and "Society". We assign a title to its closest category in terms of geodesic distance (shortest path length) in the graph of Wikipedia categories. To generate multiclass classification data set, we include only titles with a unique category, i.e. only that category has the shortest geodesic distance to the article. To construct a collection of semantically related long documents, we use the the abstract of all Wikipedia articles.

Dblp. Titles of computer science literature represent short texts in formal communication. We choose 6 diverse research fields for classification, including "Database", "Artificial Intelligence", "Hardware", "Systems", "Programming Languages", and "Theory". For each field, we select representative conferences and collect the titles of papers published in these conferences as labeled data. To construct a collection of semantically related long documents, we use the abstracts of all papers in DBLP bibliography database. [3]

Twitter. The 140-character microblog data represent informal short texts widely used in social media. We use a large corpus of tweets for positive/negative sentiment classification.[4] We randomly sampled 1,200,000 tweets and split them into training and test sets. Because tweets is special genre of text, it is non-trivial to obtain semantically related long documents. Since the data set itself is reasonably large, we use the training set as the document collection.

We use the Apache Lucene library [5] to construct full-text index for each document collection. This allows efficient document retrieval using short texts as queries. For each short text, we associate top K relevant documents as its *memory*. We use Dirichlet smoothing language modeling as the retrieval function [35].

Table 1: Statistics of short text data sets

	Wikipedia	Dblp	Twitter
Train	18,000	61,479	800,000
Test	12,000	20,000	400,000
Vocabulary	25,550	22,686	535,997
Avg. doc. length	3.12	9.48	13.69
# of classes	15	6	2

Table 2: Statistics of long document collections

	Wikipedia abstract	Dblp abstract
# of docs	4,747,988	480,558
Vocabulary	3,768,403	310,178
Avg. doc. length	98.37	138.68

We use standard classification performance metrics to evaluate different algorithms: micro-averaged F1 and macro-averaged F1.

4.2 Experimental Setup

4.2.1 Compared Methods. We compare three different types of methods: (1) existing typical approaches for text representations, which only use the information in the original short text; (2) classical relevance feedback methods for query expansion in information retrieval, which leverages external documents to improve short text representation; (3) end-to-end short text expansion solution based on memory networks, which also leverage external documents and are trained in an end-to-end fashion.

[1] https://dumps.wikimedia.org/enwiki/20161120
[2] https://en.wikipedia.org/wiki/Category:Main_topic_classifications
[3] http://aminer.org/lab-datasets/citation/DBLP-citation-Jan8.tar.bz2
[4] http://thinknook.com/twitter-sentiment-analysis-training-corpus-dataset-2012-09-22/
[5] https://lucene.apache.org

Text representation approaches can be grouped as follows.

Unsupervised methods: (1) The classical "bag-of-words" represenation (**BOW**). Each document is a $|V|$-dimensional vector, where each dimension is the TFIDF representation a word. (2) **Skip-gram**: the state-of-the-art word embedding model [21]; (3) **LINE**: the large-scale information network embedding model [31]. It is used to learn unsupervised word embeddings from word co-occurrence networks and word-document networks. We take average of word embeddings to produce a document embedding.

Supervised methods: (1) **PTE**: the predictive text embedding model [30]. We use the PTE model to learn supervised word embeddings from word co-occurrence networks, word-document networks, and word-label networks. We take average of word embeddings to produce a document embedding. (2) **CNN**: the supervised text embedding model based on convolutional neural networks [13]. (3) **RNN**: the supervised text embedding model based on recurrent neural networks with bidirectional GRUs. (4) **FastText**: a supervised text embedding model showing comparable performance to more sophisticated deep learning models [12].

Query expansion methods: For classical query expansion algorithms, we consider Rocchio's method [26]. Both short text and long documents are represented as sparse "bag-of-words" vectors with TFIDF weighting. We treat the long documents $\vec{d}_i$ as pseudo relevant documents, and expand the short text $\vec{q}$ by interpolating between $\vec{q}$ and the average of $\vec{d}_i$'s:

$$\vec{q'} = (1-\lambda)\,\vec{q} + \frac{\lambda}{K}\sum_{i=1}^{K}\vec{d}_i\,, \tag{14}$$

where $\lambda \in [0,1]$ is the interpolation weight. λ is tuned for each data set on a validation set. We call this method $\mathbf{BOW}_{RF}$.

Another straightforward query expansion approach is to concatenate the short text and its long documents together, and use existing methods to learn text representation for this pseudo-document. Specifically, we use Skip-gram, LINE, and PTE to learn word embeddings, and generate the pseudo-document vector by averaging word embeddings. This gives us three variants: $\mathbf{Skip\text{-}gram}_{RF}$, $\mathbf{LINE}_{RF}$, and $\mathbf{PTE}_{RF}$, corresponding to three text representation methods.

Memory network-based methods: we adapt the original memory network [29] to our problem setting (**MemNet**). We treat the short text as the question, the documents retrieved by our retrieval module as memories, and the target category as the answer. Finally, we include two variants of our algorithm: the short text expansion memory network with soft attention (**ExpaNet-S**) and hard attention (**ExpaNet-H**).

We consider two settings of retrieved documents: (1) **short text memory**: for each short text in training and test set, the memories are short texts pre-retrieved from the training set. (2) **long document memory**: for each short text in training and test set, the memories are long documents pre-retrieved from an external document collection. As mentioned in Section 4.1, we only consider short text memory for the Twitter data set.

4.2.2 Parameter Settings. To prepare relevant documents for each short text, we retrieve top 20 results from the document collection returned by Dirichlet smoothing language modeling ($\mu = 2000$, default in Lucene). Considering Web search engines typically show 10 results per page, 20 results is reasonably large for a high recall of relevant documents. In rare cases where the retrieval function does not return enough results, we randomly duplicate the returned results to make 20 documents.

To train text classification models, we use one-vs-rest multiclass support vector machines with linear kernel implemented in the LibLinear package [6] with regularization weight $c = 1$. For text embedding methods, we set embedding dimension to be 100. To represent a piece of text as a sequence of words, we use the first 15 words for short text and the first 100 words for long documents; zero-padding is used when the text is not long enough. End-to-end learning algorithms are trained using Adam stochastic gradient descent algorithm [14], in mini-batches of size 32. The initial learning rate is empirically set to be 10^{-3} for CNN and RNN, and 10^{-2} for FastText and memory network-based methods. Word embedding matrices are initialized with numbers drawn from $\mathcal{N}(0, 0.1^2)$.

For each of the three memory network-based methods, the number of hops is tuned on hold-out data sets ($\#hops \in \{1,2,3,4\}$).

4.3 Overall Performance

We compare three different types of approaches: methods with only short text features, short text expansion using classical query expansion methods, and end-to-end short text expansion methods based on memory networks. For the latter two types of methods, besides leveraging external long documents as memory, we also investigate the effect of treating the training short documents as memory. For all the methods, the performance are averaged over five runs with random parameter initialization. Table 3 summarizes the performance of all compared methods. Statistical significance of the results are provided by comparing our methods to the original memory networks [29], a strong baseline method. In general, our methods significantly outperforms the original memory networks and baseline methods only using original features of short texts and classical query expansion methods.

For the methods with only short text features, the unsupervised approaches BOW, Skipgram,and LINE do not perform well since no supervision is used to learn the representations. For the supervised approaches, the PTE and FastText, which ignore the order of the words, perform comparably as CNN and RNN on topic classification tasks (Dblp and Wikipedia). However, the performance is significantly inferior to CNN and RNN on the task of sentiment classification on the Twitter data set, for which the order of the words is very important.

For query expansion methods, additional information from relevant documents tends to improve classification performance. The classical "bag-of-words" representation performs very well. Long document memory provide richer information than short document memory, and is more effective in general. The performance gain is most salient when the text is extremely short and the relevant documents are long (Wikipedia).

Compared to the classical query expansion methods, the performance of memory network-based methods are significantly better since the expansion process of classical methods are heuristic, which may introduce noise into the original representation while the memory network based solution provide an end-to-end solution. Our model with either soft or hard attention mechanism outperforms the original memory networks since the GRU unit is used to

Table 3: Classification performance of compared methods

Settings	Methods	Wikipedia		Dblp		Twitter	
		Micro-F1	Macro-F1	Micro-F1	Macro-F1	Micro-F1	Macro-F1
short text only	BOW	42.63	42.37	75.28	71.59	75.27	75.27
	Skip-gram	26.75	25.90	73.08	68.92	73.02	73.00
	LINE	34.93	32.84	73.98	69.02	73.19	73.18
	PTE	38.78	38.58	76.45	72.74	73.80	73.80
	CNN	41.70	41.68	77.38	74.35	77.83	77.23
	RNN	42.40	42.10	77.84	74.83	77.45	76.72
	FastText	43.33	42.79	77.15	74.05	74.20	74.07
short text memory	BOW_{RF}	43.01	42.89	77.45	74.16	76.21	76.19
	$Skip\text{-}gram_{RF}$	34.57	32.84	75.68	71.84	73.14	73.11
	$LINE_{RF}$	35.05	33.19	75.92	72.14	73.20	73.20
	PTE_{RF}	39.54	39.17	77.37	73.84	73.93	73.91
	MemNet	42.20	41.74	77.94	74.75	77.76	77.44
	ExpaNet-S	43.67^{***}	43.60^{***}	79.05^{***}	76.09^{***}	78.55	78.24
	ExpaNet-H	42.37	42.31^{**}	78.94^{***}	75.97^{***}	79.25^{***}	78.91^{***}
long doc. memory	BOW_{RF}	47.13	47.12	78.26	75.25	-	-
	$Skip\text{-}gram_{RF}$	46.66	45.54	75.55	71.93	-	-
	$LINE_{RF}$	46.52	45.36	75.75	72.19	-	-
	PTE_{RF}	48.15	47.43	78.31	75.26	-	-
	MemNet	47.66	47.57	79.16	75.91	-	-
	ExpaNet-S	50.85^{***}	50.69^{***}	80.32^{***}	77.60^{***}	-	-
	ExpaNet-H	50.68^{***}	50.50^{***}	80.12^{***}	77.35^{***}	-	-

$^{**}(^{***})$ means the result is significant according to Student's T-test at level 0.05 (0.01) compared to MemNet.

integrate the information from the original query and information read from memory. The performances of soft and hard attention are on par with each other. On two data sets (Wikipedia and Dblp), soft attention performs slightly better than hard attention. On Twitter, hard attention performs slightly better than soft attention.

4.4 Comparison of Expansion Using General v.s. Specific Domain Long Documents

We often have access to abundant long documents in general domain, such as Wikipedia and the World Wide Web, but less so for specific domains. These long documents may not exactly match the domain of a short text classification task, but some of them may still provide relevant information as long as there is some overlap in semantics and genre. To test this hypothesis, we take titles of Dblp and search for relevant documents in Wikipedia as the memory for classification. Intuitively, we are testing if "reading computer science literature could be as useful as reading relevant articles in Wikipedia". The classification performance is shown in Table 4.

Overall, we observe a performance drop after using Wikipedia abstracts instead of Dblp abstracts as memory. The performance drop is more salient when we use unsupervised text representation, because long documents in general domain necessarily introduce noise. Supervised representation learning methods can mitigate noise by fine-tuning text representation towards the task. Memory networks-based methods have the smallest performance gap by providing an end-to-end solution, among which our algorithm (ExpaNet) performs the best.

4.5 Parameter Sensitivity

4.5.1 Effect of number of hops. In our model, each hop selectively incorporates information from memory and refines the representation of short text from previous hop. Naturally we raise the hypothesis that more hops will further refine short text representation and lead to increased performance. To test the hypothesis, we train the memory networks with different number of hops (0,1,2,3,4) and observe the macro-averaged F1 score on the three data sets. To obtain statistical confidence, we run each configuration 5 times and compute the mean and standard deviation. Figure 2 shows the performance curve as the number of hops increases.

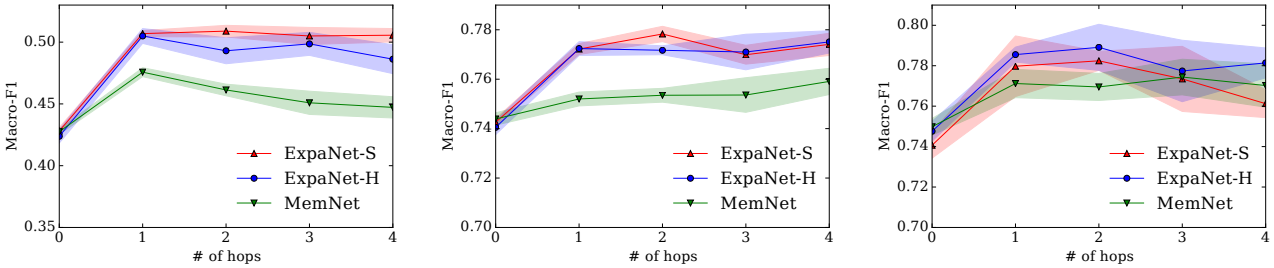
We see that as the number of hops increases, classification performance will increase and then quickly saturate. A large gain can be observed from #hops=0 to 1, which introduces the most information, and a slight gain from #hops=1 to 2. The performance becomes somewhat unstable with even more hops, which may be due to over-fitting. Ideally, the number of hops is different for different short text: some need more refinements as they are too short, while others need less. We leave how to *learn* the best number of hops for expanding each short text as our future work.

4.5.2 Effect of memory size. In the *retrieval module*, we retrieve a set of a relevant documents, which are put into the memory. In this part, we study the effect of memory size with respect to classification performance. We take the ExpaNet-soft with 1 hop as an example and vary the number of memory cells K. All the results are averaged over five runs with random initialization.

Table 4: Performance comparison of expansion using long documents from general domain vs. specific domain

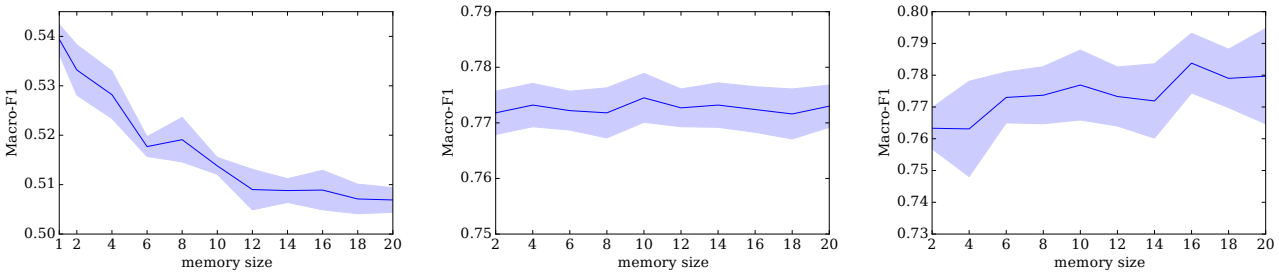
Methods	Dblp memory		Wikipedia memory		Performance gap	
	Micro-F1	Macro-F1	Micro-F1	Macro-F1	Micro-F1	Macro-F1
BOW_{RF}	78.26	75.25	71.00	67.55	-7.26	-7.70
Skip-$gram_{RF}$	75.55	71.93	68.63	64.73	-6.92	-7.20
$LINE_{RF}$	75.75	72.19	68.95	65.15	-6.80	-7.04
PTE_{RF}	78.31	75.26	75.78	72.35	-2.53	-2.91
MemNet	79.16	75.91	77.18	74.04	-1.98	-1.87
ExpaNet-S	80.32^{***}	77.60^{***}	78.34^{***}	75.52^{***}	-1.98	-2.08
ExpaNet-H	80.12^{***}	77.35^{***}	78.04^{**}	75.15^{**}	-2.08	-2.20

$^{**}(^{***})$ means the result is significant according to Student's T-test at level 0.05 (0.01) compared to MemNet.



(a) Wikipedia (b) Dblp (c) Twitter

Figure 2: Performance w.r.t. # of hops. When #hops=0, MemNet, ExpaNet-S, and ExpaNet-H are the same model since none of them use the memory. Color regions correspond to ±1 standard deviation around the mean.



(a) Wikipedia (b) Dblp (c) Twitter

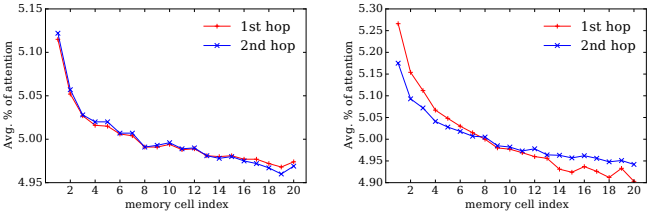
Figure 3: Classification performance w.r.t. memory size (ExpaNet-S, #hops = 1). Color regions correspond to ±1 standard deviation around the mean.

We observe that on both Dblp and Twitter data sets, the performance are not sensitive to the memory size. However, on Wikipedia data set, smaller memory size leads to slightly better performance. The reason is that on Wikipedia data set, there is only one or very few relevant documents about a Wikipedia title, which is retrieved by the retrieval module. Adding more documents into the memory introduces more noise. However, this may not hold in real-world data: Web search usually returns many relevant documents.

4.6 Attention Interpretation

The attention mechanism in the ExpaNet essentially computes similarity between a short text and each document in memory. We are interested in the attention mechanism learned by our algorithm. We are interested in the following questions: which memory cells does the model learn to pay more attention to? With more number of hops, how does the attention change? Note that when the long documents are loaded into memory, their retrieval ranks are reserved: the highest ranked document is placed in Cell 1, the lowest in Cell 20. The memory networks, on the other hand, treat the memory as "a bag of cells" without considering the order.

In Figure 4, we plot the attention distribution over 20 memory cells in a soft attention memory network. The distribution is estimated by averaging the attentions on test data set. Hard attention memory networks have similar attention distribution but with higher variance because of its stochastic nature. The attention distribution agrees with our prior knowledge of retrieval relevance.



(a) Wikipedia (b) Dblp

Figure 4: Attention distribution over memory cells. The plot of Twitter data set is similar to Dblp hence omitted.

Memory cells from left to right hold documents with decreasing relevance scores. After training, the algorithm learns to pay more attention to cells on the left, which agrees with the relevance ranking. With more hops, attention on Dblp and Twitter tends to distribute uniformly across memory cells, indicating the expansion process can identify more relevant documents from the memory with more hops.

5 CONCLUSION

In this paper, we proposed an end-to-end deep memory network approach for short text expansion with a large corpus of long documents. Inspired by the human search strategy, the memory network learns to select relevant documents with attention mechanism, combine short text and expanded documents with a gating mechanism, and is trained end-to-end with short text classification as the objective. Extensive experiments on several real-world data sets show that our model significantly outperforms classical query expansion methods and methods without using external data. In the future, we plan to study how to automatically infer the optimal number of expansion hops for each short text.

Acknowledgements. We thank the anonymous reviewers for their constructive comments. This work is supported by the National Institutes of Health under grant NLM 2R01LM010681-05 and the National Science Foundation under grant IIS-1054199.

REFERENCES

[1] Michael Bendersky and W Bruce Croft. 2009. Analysis of long queries in a large scale search log. In *Proceedings of 2009 Workshop on Web Search Click Data*. 8–14.
[2] Chris Buckley, Gerard Salton, James Allan, and Amit Singhal. 1995. Automatic query expansion using SMART: TREC 3. *NIST Special Publication* (1995), 69–69.
[3] Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using RNN encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078* (2014).
[4] Susan Dumais, Michele Banko, Eric Brill, Jimmy Lin, and Andrew Ng. 2002. Web question answering: Is more always better?. In *Proceedings of 25th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 291–298.
[5] Miles Efron, Peter Organisciak, and Katrina Fenlon. 2012. Improving retrieval of short texts through document expansion. In *Proceedings of 35th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 911–920.
[6] Rong-En Fan, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-Rui Wang, and Chih-Jen Lin. 2008. LIBLINEAR: A library for large linear classification. *Journal of Machine Learning Research* 9, Aug (2008), 1871–1874.
[7] Evgeniy Gabrilovich and Shaul Markovitch. 2007. Computing semantic relatedness using wikipedia-based explicit semantic analysis.. In *IJcAI*, Vol. 7. 1606–1611.
[8] Alex Graves. 2012. Supervised sequence labeling. In *Supervised Sequence Labeling with Recurrent Neural Networks*. Springer, 5–13.
[9] Alex Graves, Greg Wayne, and Ivo Danihelka. 2014. Neural turing machines. *arXiv preprint arXiv:1410.5401* (2014).
[10] Xia Hu, Nan Sun, Chao Zhang, and Tat-Seng Chua. 2009. Exploiting internal and external semantics for the clustering of short texts using world knowledge. In *Proceedings of 18th ACM Conference on Information and Knowledge Management*. 919–928.
[11] Eric Jang, Shixiang Gu, and Ben Poole. 2016. Categorical Reparameterization with Gumbel-Softmax. *arXiv preprint arXiv:1611.01144* (2016).
[12] Armand Joulin, Edouard Grave, Piotr Bojanowski, and Tomas Mikolov. 2016. Bag of tricks for efficient text classification. *arXiv preprint arXiv:1607.01759* (2016).
[13] Yoon Kim. 2014. Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882* (2014).
[14] Diederik Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014).
[15] Ankit Kumar, Ozan Irsoy, Jonathan Su, James Bradbury, Robert English, Brian Pierce, Peter Ondruska, Ishaan Gulrajani, and Richard Socher. 2015. Ask me anything: Dynamic memory networks for natural language processing. *arXiv preprint arXiv:1506.07285* (2015).
[16] Quoc V Le and Tomas Mikolov. 2014. Distributed Representations of Sentences and Documents.. In *ICML*, Vol. 14. 1188–1196.
[17] Carol Lundquist, David A Grossman, and Ophir Frieder. 1997. Improving relevance feedback in the vector space model. In *Proceedings of 6th International Conference on Information and Knowledge Management*. ACM, 16–23.
[18] Craig Macdonald and Iadh Ounis. 2007. Expertise drift and query expansion in expert search. In *Proceedings of 16th ACM Conference on Information and Knowledge Management*. 341–350.
[19] Michele Merler, Carolina Galleguillos, and Serge Belongie. Recognizing groceries in situ using in vitro training data. In *CVPR 2007*.
[20] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781* (????).
[21] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems*. 3111–3119.
[22] Volodymyr Mnih, Nicolas Heess, Alex Graves, and Koray Kavukcuoglu. 2014. Recurrent models of visual attention. In *Advances in Neural Information Processing Systems*. 2204–2212.
[23] Tsendsuren Munkhdalai and Hong Yu. 2016. Neural Semantic Encoders. *arXiv preprint arXiv:1607.04315* (2016).
[24] Hamid Palangi, Li Deng, Yelong Shen, Jianfeng Gao, Xiaodong He, Jianshu Chen, Xinying Song, and Rabab Ward. 2016. Deep sentence embedding using long short-term memory networks: Analysis and application to information retrieval. *IEEE/ACM Transactions on Audio, Speech and Language Processing (TASLP)* 24, 4 (2016), 694–707.
[25] Xuan-Hieu Phan, Le-Minh Nguyen, and Susumu Horiguchi. 2008. Learning to classify short and sparse text & web with hidden topics from large-scale data collections. In *Proceedings of 17th International Conference on World Wide Web*. ACM, 91–100.
[26] Joseph John Rocchio. 1971. Relevance feedback in information retrieval. (1971).
[27] Mehran Sahami and Timothy D Heilman. 2006. A web-based kernel function for measuring the similarity of short text snippets. In *Proceedings of 15th International Conference on World Wide Web*. ACM, 377–386.
[28] Nico Schlaefer, Jennifer Chu-Carroll, Eric Nyberg, James Fan, Wlodek Zadrozny, and David Ferrucci. 2011. Statistical source expansion for question answering. In *Proceedings of 20th ACM International Conference on Information and Knowledge Management*. 345–354.
[29] Sainbayar Sukhbaatar, Jason Weston, Rob Fergus, and others. 2015. End-to-end memory networks. In *Advances in Neural Information Processing Systems*. 2440–2448.
[30] Jian Tang, Meng Qu, and Qiaozhu Mei. 2015. PTE: Predictive text embedding through large-scale heterogeneous text networks. In *Proceedings of 21st ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 1165–1174.
[31] Jian Tang, Meng Qu, Mingzhe Wang, Ming Zhang, Jun Yan, and Qiaozhu Mei. 2015. Line: Large-scale information network embedding. In *Proceedings of 24th International Conference on World Wide Web*. ACM, 1067–1077.
[32] Ronald J Williams. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning* 8, 3-4 (1992), 229–256.
[33] Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhutdinov, Richard S Zemel, and Yoshua Bengio. 2015. Show, Attend and Tell: Neural Image Caption Generation with Visual Attention.. In *ICML*, Vol. 14. 77–81.
[34] Chengxiang Zhai and John Lafferty. 2001. Model-based feedback in the language modeling approach to information retrieval. In *Proceedings of 10th International Conference on Information and Knowledge Management*. ACM, 403–410.
[35] Chengxiang Zhai and John Lafferty. 2001. A study of smoothing methods for language models applied to ad hoc information retrieval. In *Proceedings of 24th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 334–342.