APPLICABILITY OF DEEP NEURAL NETWORKS ON THE TASK OF DOCUMENT RETRIEVAL

M. Shoaib Malik[1, 2] and Dagmar Waltemath[1]

[1]Medical Informatics Laboratory, Institute for Community Medicine, University Medicine Greifswald, Germany
[2]Department of Computer Science, Air University, Islamabad, Pakistan

ABSTRACT

A Deep Neural Network (DNN) can be used to learn higher-level and more abstract representations of a particular input. DNNs have successfully been applied to analysis tasks including image processing, unsupervised feature learning, and natural language processing. DNNs furthermore can improve computing performance when compared to shallower networks, for example in pattern recognition tasks in machine learning. Recent usage of DNNs in search engines for the Web have impacted that technology in industrial scale applications. One example for such an application is deepgif - a search engine for Graphics Interchange Format (GIF) images that is based on a convolutional neural network and takes natural language text as query. In this study, we developed a tool and compared the performance of feed-forward neural networks and deep architectures of recurrent neural network using the case of document retrieval. This study first discusses two architectural setups used to build the models and then provide a detailed comparison of their performance. The goal is to identify the architecture that is most suited for the task of document retrieval.

KEYWORDS

Deep Neural Network, Machine Learning, Document Retrieval, Feed-Forward Neural Network, Recurrent Neural Network

1. INTRODUCTION

Textual documents are an integral part of our everyday life. With an increased amount of text, and thus documents, becoming available on the world wide web, the chance of finding the information that best meets a user's need is significantly decreasing (Skovajsova, 2010). The availability of cheap and effective storage media has resulted in an enormous rise in the size of textual databases which are widely used in traditional library science environments, in business applications (e.g., manuals, newsletters, and electronic data interchanges), and in scientific applications (e.g., electronic community systems and scientific databases) (Chen, 1995). This has required greater efforts in the retrieval of relevant information, especially from large scaled databases. Various methods have been proposed over the years to deal with the large amounts of data in textual document space. Models that are built on neural networks usually cluster or classify documents into groups with similar documents belonging to the same group. Many document retrieval systems are built on keyword searches. However, these systems do not consider the relations among the passages of text within a document (Treeratpituk and Callan, 2006) (Wei and Croft, 2006).

The task of document retrieval is to find documents of unstructured or semi-structured nature that satisfies the information need of a user from within a large collection. The goal of document retrieval systems is thus to retrieve documents that are relevant to the user's information need. Document retrieval systems must hence be able to retrieve desired information about a subject rather than to retrieve documents that look similar to a given query (Baeza-Yates and Ribeiro-Neto, 1999). To accomplish that task, document retrieval systems apply specific concepts to represent the query and documents, and to assign relevant documents to the query (Skovajsova, 2010). The issue of predicting relevance of a document to the user's information need is usually based on a ranking algorithm where documents appearing at the top of the list are considered to be more relevant than those at the bottom (Baeza-Yates and Ribeiro-Neto, 1999).

Many Web-based tools retrieve information through general-purpose search engines like Google, Bing and Yahoo or through specialized search engines such as PubMed (for biological and medical publications). While early search engines ranked their results based on content of the document, more modern search engines evaluate the semantics of the document. Search engines like Google and Yahoo consider page reputation as one of the major criteria of relevance ranking (Deepak and Deepika, 2012).

Several approaches to document retrieval have been proposed over the years including the boolean model (van Rijsbergen, 1979) (Baeza-Yates and Ribeiro-Neto, 1999) (Cordon et al., 2002) (Herrera-Viedma, 2001), vector space model (Manning et al., 2008) (Hotho et al., 2005) (Lan et al., 2005) (Scheir and Lindstaedt, 2006), document based language model (Ogilvie and Callan, 2003) (Wang et al., 2005), and the models built using neural networks (Cheung and Cannons, 2002).

The Boolean model is one of the first models of information retrieval for document extraction. It uses a term-document matrix where every cell in the Boolean matrix is filled with 0 or 1 based on whether a word appears in the document or not. The vectors for the terms in the query are put together using a Boolean operation such as AND, OR, NOT. For a collection with 1 million documents with each 1000 words, roughly 6GB are required to store such a collection (assuming an average of 6 bytes/word). For this reason, the documents are stored in an inverted index (Zobel and Moffart, 2006) with variable sized posting lists.

The vector space model is commonly used in information retrieval systems. It works on the Term Frequency- Inverse Document Frequency (TF-IDF) weights of the terms, which is a common weighting scheme in information retrieval. Opposed to results returned by Boolean model, which are not ranked in any presumed order of importance, the retrieved documents can easily be ranked in decreasing order of the query-document similarity for vector space models (Salton et al., 1983). The most common similarity metric used is the cosine similarity (Turney and Pantel, 2010) between the query and the document. A lot of variants have been proposed over the years for better retrieval output.

The document based language model is the most direct way to estimate a language model from a large collection and assumes an underlying multinomial model. It estimates the probability of each document in the collection generated the query independently. This approach is not very good at estimating novel terms. For this reason, smoothing (Chen and Goodman, 1999) is applied that compensates for data sparseness by stealing a little probability mass from the seen terms and adding it to the unseen terms.

Neural networks provide a convenient knowledge representation for document retrieval applications in which nodes typically represent objects such as user query, keywords or documents. Such models are usually a combination of a feed-forward and spreading activation neural network. The feed-forward model learns the keywords against the query whereas the spreading activation model learns the relevant documents against the input keywords as depicted in figure 1 (Skovajsova, 2010) (Chen, 1995) (Mokris and Skovajsova, 2005). Neural networks have been used in different context but mostly on sentiment analysis. Le and Mikolov (2014) presented a framework that learns continuous distributed vector representations for a paragraph from a document. This framework works in a similar manner as learning word embedding described in Mikolov et al. (2010). It maps every paragraph to a unique vector just like every word and predicts the next word in context using the concatenated sentence and current word representation. Similar concept is also described in Lin et al. (2015) for document modeling on sentiment analysis.

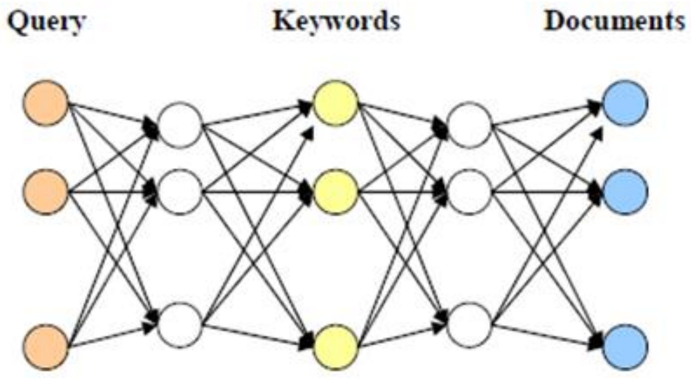


Figure 1: Cascade neural network model

The primary objective of this study is to develop and then evaluate the performance of deep learning based information retrieval systems, namely feed-forward based system and recurrent neural network based system. In particular, we will investigate feed-forward and recurrent neural networks for this task. Recent results have shown that recurrent models outperform feed forward model on the task of language modeling. This is mainly because the deep architecture of recurrent neural network allows to store context information in the hidden layer to better inform the current prediction. Our goal is to see if storing the context information for longer period of time is beneficial as well when building document model instead of a language model. Moreover, we also evaluate how change in the number of words in the query affects the retrieval output of models trained on both these architectures.

2. METHODS

This study investigates two different neural network architectures: the feed-forward (Bengio et al., 2003) and recurrent (Jain and L.R., 1999) neural network architectures.

2.1. Feed Forward Neural Network

The feed-forward neural network document model (Svozil et al., 1997) is an n-gram model where the posterior probability distribution of topic and document is computed for given n words. All collections of this study group documents into topics. Hence the output layer is factorized into a topic and a document layer. The size of the output topic layer is equal to the number of topics C and the size of the output document layer is equal to the number documents D in the collection. The feed-forward neural networks were introduced as an alternative to widely used back-off language models and have been reported to perform better in (Bengio et al., 2003), (Schwenk and Gauvain, 2004), (Gauvain et al., 2005), and (Emami and Jelinek, 2004) when used on the application of language modeling.

2.2. Recurrent Neural Network

When building a feed-forward document model, the network reads a subset of words (i.e., n-gram) at each time-step and then predicts the probability distribution of the topic and document that the input n-gram belongs to (Lin et al., 2015). The feed-forward neural network ignores the contextual information in texts and remains unsatisfactory for capturing the semantics of the words (Lai et al., 2015). It is an open question if capturing the semantics of the text by a deep recurrent neural network is of any value when learning topics along with documents in the output layer. Deep recurrent neural networks are created by stacking multiple hidden layers on top of each other, with the output sequence of one layer forming the input sequence for the next.

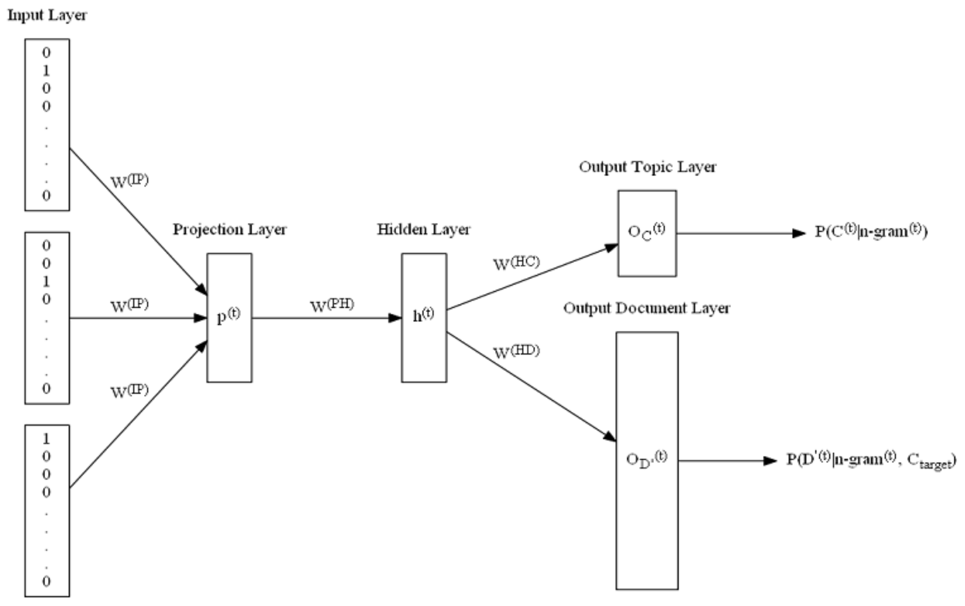


Figure 2: Architecture of feed-forward neural network

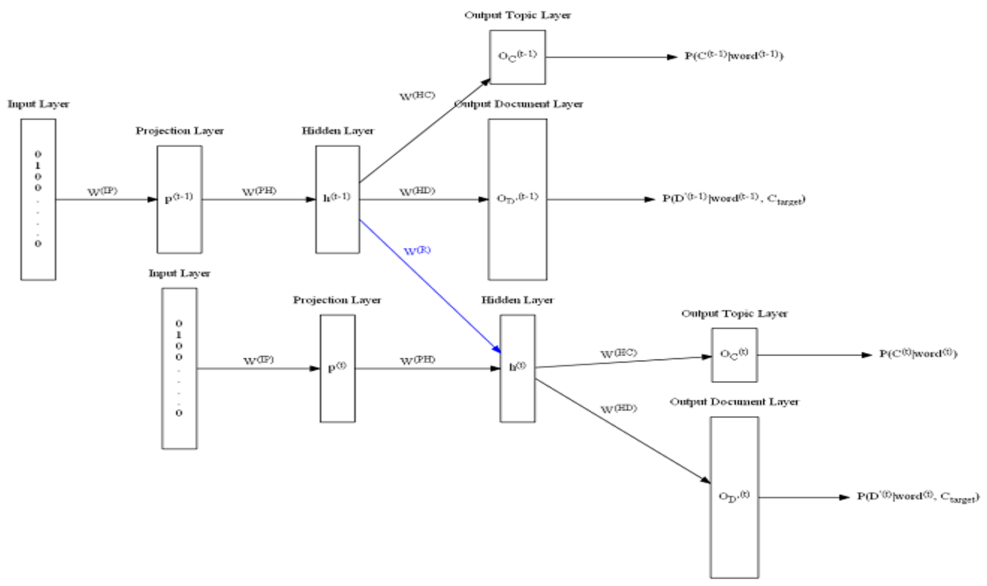


Figure 3: Architecture of recurrent neural network

3. IMPLEMENTATION DETAILS

The Neural Network Document Retrieval (NNDR) toolkit, which implements feed-forward neural network and deep architecture of recurrent neural network. is implemented using CUDA C/C++ (Nickolls et al., 2008) and Java (Arnold et al., 2000). This chapter discusses the translation of functions in respective forward pass and learning algorithms to CUDA pseudocode along with some of the considerations that were considered during the implementation of the toolkit in order to train the networks.

3.1. CUDA

CUDA is a parallel computing platform and application programming interface model created by NVIDIA for applications running on Graphics Processing Unit (GPU). Nowadays, hundreds of industry-leading scientific computing applications are already GPU-accelerated making use of multiple cores of GPU and fast arithmetic operation capability with greater floating-point performance from CUDA. From the programmer's perspective, the CUDA architecture is divided into threads, blocks and grids. A grid is organized as a 2D array of blocks and a block is organized as 3D array of threads. The high performance comes from the concurrent execution of multiple threads resulting in reduced latency.

3.2. Data Parallelism

With data parallelism, we take our documents and we process a subset of these documents in a batch. This means that we use the same model for each mini-batch but feed it with different document. The two dimensionalities of the CUDA grid can be taken advantage of in order to achieve data parallelism with each mini-batch or document being processed in the separate row of the grid. Data parallelism uses the same weights in forward pass of the network to give out topic and document probabilities as output for each mini-batch. However, in the backward pass, the weight gradients need to be synchronized from all the mini-batches and then averaged. If we do not process documents in a batch and rather process each document individually, such an approach would undo learning that it did with document D-1, D-2, etc. and would not converge to the optimal parameters. It would just hop back and forth because it does not consider all the topics at once. To avoid this, documents are processed in batch and the batch size is set to the number of topics and only one document is processed from each topic. It is made sure that the number of words processed in batch is equal for all documents. In the next iteration, the next document from same topic is chosen and similar steps described in preceding text are performed.

3.3. Text Processing

The raw text in the collections is pre-processed to reduce the problem's dimensionality and to ensure the completeness, consistency, and interpretability of the data. The following steps are performed.

1. Substitute TAB, NEWLINE and RETURN characters by SPACE.
2. Turn all letters to lowercase.
3. Substitute multiple SPACES by a single SPACE.
4. Remove the 524 SMART stopwords.
5. The title of each document is simply added in the beginning of the document's text.
6. Apply Porter's Stemmer (Porter, 1980) to the remaining words to reduce the words to their morphological root, so that the number of different terms in the documents is reduced.

7. Split the training dataset into training and validation dataset using heldout technique. The validation data is used to control the learning rate.

3.4. Vocabular Truncation

Just like hidden and output layer, the computation done between input and projection layer is also very computationally expensive - even more so than hidden and output layer in some cases, which limits its application to real world problems. Taking an example of language modeling, most of the models trained in today's research are trained on millions of words. It would take impractically long to train these models with very large vocabulary. We took the same measures as most of the researchers working in the field of natural language processing do - merge all infrequent words into a special <unk> class that represents the probability of all rarely seen words in the collection. Although this approach improves the speed of the training, it suffers from a loss of accuracy (Mikolov et al., 2010).

3.5. Out of Vocabulary Words

Out of Vocabulary words are unknown words that appear in the test data but not in the training data. Since it is practically impossible to train a system on all words that exist in natural language, some steps must be taken to deal with this problem. Although, most common approach used in textual applications of natural language processing for models trained on neural networks is to assign an <unk> tag to some of the most infrequent words in the collection, we went an extra mile. From the piece of text that was selected to be trained from a document, we replaced a word at a random location with an <unk> tag. This way, we tried to make sure that the models that were trained were also able to generalize well for the words that are not seen in the training data.

3.6. Variable Learning Rate

A standard refinement to gradient descent is to use a variable learning rate that is updated after each training epoch. We used perplexity as the evaluation criterion when training the models. Perplexity is a measure of the average branching factor of the topics and documents when predicting them from the input n-grams or words. The learning rate is varied according to changes in validation perplexities across epochs. If the previous learning rate decreased the validation perplexities across epochs, then the learning rate is left unchanged. If the previous learning rate did not decrease the validation perplexities across epochs, the learning rate is reduced to half after each subsequent epoch (Blackwood, 2005). The default value of initial learning rate for models trained on feed-forward neural network is 0.1 and it is 0.2 for models trained on recurrent neural network.

The document retrieval system was trained and evaluated on 3 collections. Every collection consists of a set of pre-classified documents where every document in the collection belongs to a particular topic. The datasets are the 20-Newsgroup collection, the Reuters-21578 collection, and the Cade collection. All collections were obtained from (Cardoso-Cachopo, 2007).

Table 1: Document distribution and vocabulary sizes of all the collections.

Collection	**No. of Training Documents**	**No. of Validation Documents**	**No. of Test Documents**	**No. of Topics**	**Vocabulary Size**
20-Newsgroups	8951	2231	7528	20	13926
Reuters-21578	4901	1200	2568	52	8862
Cade	13597	3393	13661	12	18947

Every document in the collections is available as a single running text. Since the documents are not available as sentences, the number of words trained from each document in batch is equal to the number of words W in the shortest document. From the rest of the documents except the shortest document, a random chunk of text is taken whose length is equal to W.

The 20-Newsgroups collection is a set of newsgroup documents, which are nearly evenly partitioned across 20 different newsgroups. The collection has become a popular dataset for experiments in text applications of machine learning techniques, such as text classification and text clustering.

The Reuters-21578 collection is also one of the widely used collections in text classification. All the documents contained in this collection appeared on the Reuters newswire in 1987 and were manually classified by personnel from Reuters Ltd. and Carnegie Group, Inc. in 1987.

The documents in the Cade collection correspond to a subset of web pages extracted from the Cade Web Directory, which points to Brazilian web pages classified by human experts.

4. EVALUATION METRICS

A document retrieval system assigns higher relevance to documents that are more similar to the query. We compared the performance of different models that we developed and trained. This comparison can be carried out either by looking at the ranked retrieval results, or by adopting a performance measure as an indicator to derive the perfection in prediction, in particular as a function of the number of topics and topic imbalance.

As far as the collections used for this study are concerned, every test document is available as a single running text and hence a substring of document is chosen as a query to the networks. The substring is selected from the beginning of the document as we found the words in beginning of the document to be most informative regarding the document and the topic. When evaluating the model trained on a particular architecture for a particular collection, for each input word or n-gram, the topic with maximum probability from output topic layer is recorded. Then, the probabilities of all the documents from that topic are computed in output document layer and are added to the output vector whose length is equal to the number of training documents in the collection. In other words, we estimate the probability that each document generated the query.

4.1. Mean Average Precision

Precision (Powers, 2007) is the fraction of retrieved documents that are relevant to the query. A document is considered retrieved if its probability in the output vector is greater than 0. In recent years, other measures have become more common, one of which is Mean Average Precision (MAP) (Manning et al., 2008), which provides a single-figure measure of quality across recall levels and has been shown to have especially good discrimination and stability. For a single query, average precision (Manning et al., 2008) is the average of the precision values obtained for the set of top k documents. For our system, the number of relevant documents is equal to the number of documents belonging to target class Ctarget. For evaluation purposes, we evaluate only the top 20 documents that are retrieved (at most) and hence the number of relevant documents is the lower bound on 20 and number of documents belonging to Ctarget (i.e., n=min(20,|Ctarget|)). At the end when all queries from the query set Q have been run, mean average precision can be calculated as the mean of the average precision scores for each query.

4.2. Matthews Correlation Coefficient

The Matthews correlation coefficient (MCC) (Matthews, 1975) is used in machine learning as a measure of the quality of classifications. MCC formulation was originally reported for binary classification that works on a 2 x 2 contingency table taking into account the true positives (TP), false positives (FP), false negatives (FN), and true negatives (TN) and is generally regarded as a balanced measure which can be used even for imbalanced collection. The MCC is in essence a correlation coefficient between the observed and predicted classifications returning a single value in range [-1, +1] where +1 is perfect classification with all zeros in the contingency table except the diagonal, -1 is the extreme mis-classification case with all zeros in the diagonal of the contingency table, and 0 corresponds to prediction at random. A totally random prediction can occur when for all the queries, the model retrieves documents of one particular topic only or when for queries belonging a particular topic, the number of predictions is equal for all the topics.

Since all the collections used for this study are divided into more than 2 topics, the definition of MCC reported for the multi-class case (Jurman et al., 2012) was used. For multi-class case, the MCC formulation works on a N x N contingency table C where N is the number of topics/classes in the collection. For evaluation purposes, the number of true positives for a particular topic is equal to the number of successful retrievals for that topic. If a retrieval is unsuccessful for a query belonging to a topic i, the entry ij in the contingency table is incremented by 1 where j is the topic that the highest ranked document in the output vector belongs to.

4.3. Mean Rank

Ranking (Baeza-Yates and Ribeiro-Neto, 1999) of the documents can be computed by sorting the documents in decreasing order of probability from output vector. The rank is the highest index of document from the target topic Ctarget in the output vector.

5. EXPERIMENTS AND RESULTS

This section discusses the results obtained by running experiments on the collections in order to evaluate the retrieval outputs of both the architectures. To evaluate the performance of the feed-forward neural network, 1-gram, 3-gram, and 5-gram models were built. A 1-gram model takes one word in the input whereas 3-gram and 5-gram models take a sequence of 3 and 5 words in the input respectively. After each network run, the window in the document slides by 1 and the next sequence of 1, 3, or 5 words is read from the document. For recurrent neural network, models were built using bptt parameter of 1, 3, and 5 for each collection to see how storing the context information affects the retrieval output. Recurrent models were trained using an initial learning rate of 0.2 whereas feed-forward models were trained using an initial learning rate of 0.1. To avoid over-fitting of the training documents, the regularization parameter used for models of both architectures was fixed at 10-6. It shall be noted that for models built on feed-forward neural network, the projection layer size is 200 for each word in input n-gram.

Table 2: Network setups used to build the models.

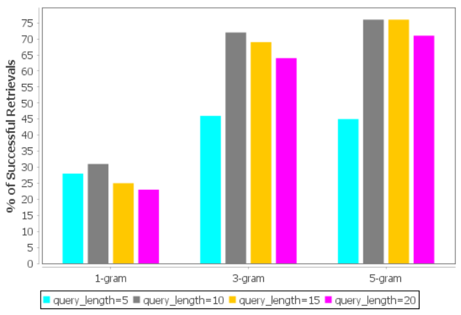
	Collection					
	20-Newsgroup		Reuters-21578		Cade	
	Projection Layer Size	Hidden Layer Size	Projection Layer Size	Hidden Layer Size	Projection Layer Size	Hidden Layer Size
FFNN	200	800	200	600	200	800
RNN	200	800	200	600	200	800

5.1. Retrieval Effectiveness for FFNN

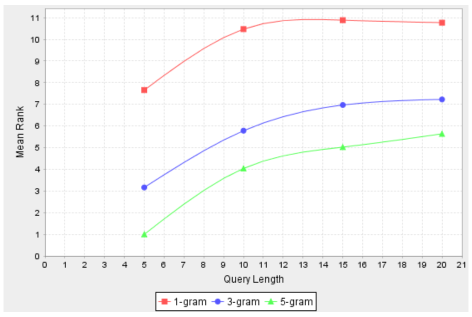
Figure 4, 5, and 6 show the percentage of successful retrievals along with the results returned by the evaluation metrics for the respective collections trained on feed-forward model. A retrieval is considered to be successful if at least one of the relevant documents is retrieved for a given query. It is seen that the mean rank and the mean average precision for the Reuters-21578 and 20-Newsgroup collection is significantly better than the Cade collection which is expected as the number of documents and vocabulary size of the Reuters-21578 and 20-Newsgroup collection is lower than that of Cade collection. A larger vocabulary and greater number of documents in the Cade collection makes it a little hard for the network to capture the n-gram to topic and n-gram to document relationship.

For higher order n-grams, the models trained on all the collections performed better than lower order n-grams, which had been expected. The larger the n-gram on which we train the model, the more coherent the training documents. For 1-gram model, there is no coherent relation between words whereas the 5-gram model has some local word-to-word coherence which is reflected in the results. For 5-gram model that was trained for each collection, it is seen that with an increase in the length of query from 5 words to 10 words, the retrieval performance is significantly decreased. However, for the 20-Newsgroup collection, MAP becomes better for query length of 15 and 20. The reason for best performance in case of shortest query of 5 words for a 5-gram model is that average precision for a query will be 1 whenever a correct prediction is made in the output topic layer for the input 5-gram. However, for queries where incorrect prediction is made in the output topic layer, the average precision will be 0. Under such circumstances, the percentage of successful retrievals gives the same estimate as the MAP score which is evident from the results shown for respective collections.

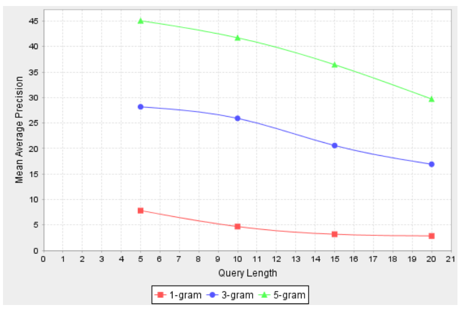
For the Reuters-21578 and Cade collection, we were able to achieve maximum retrieval performance for shortest query length of 5. For longer queries, performance of the Reuters-21578 collection became poorer while that of the Cade collection remained constant. Figure 7 shows the skewness of training data for all collections. Since the training dataset for the Reuters-21578 collection is very skewed, the inability of models trained on these two collections to perform better for longer queries can be explained with the help of an example model trained on a skewed collection.



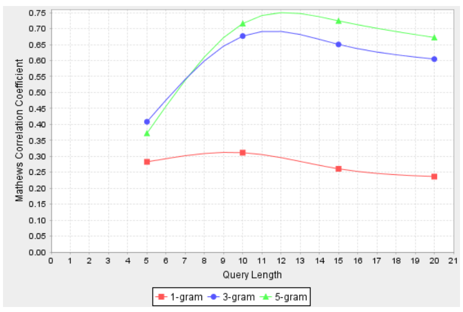
(a)



(b)

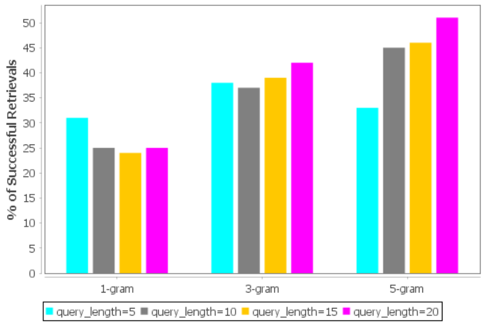


(c)

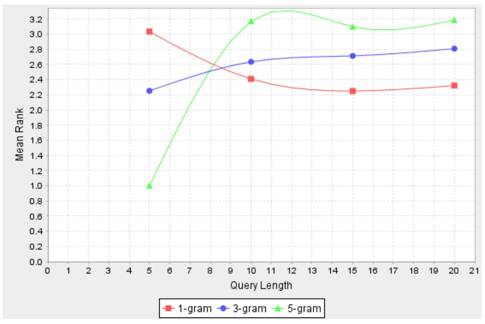


(d)

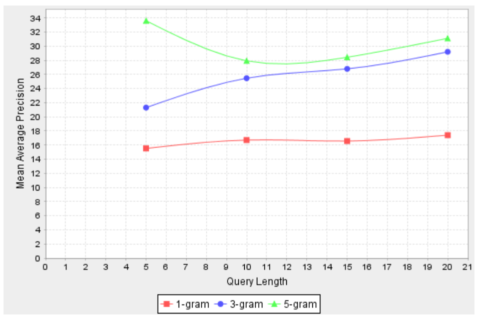
Figure 4: (a) percentage of successful retrievals, (b) mean rank, (c) mean average precision, and (d) matthews correlation of the Reuters-21578 collection trained on feed-forward neural network



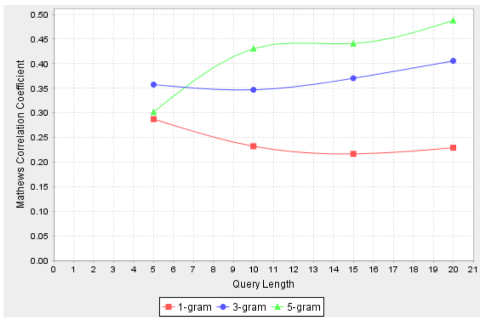
(a)



(b)

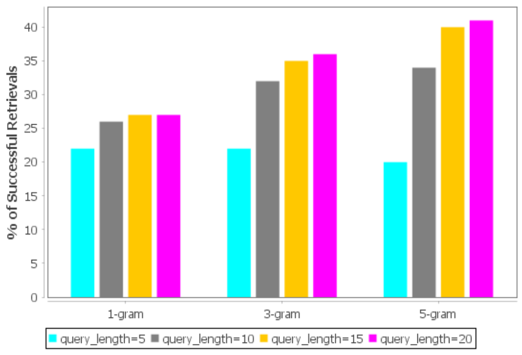


(c)

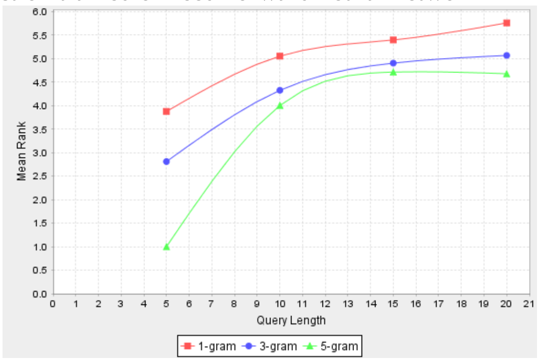


(d)

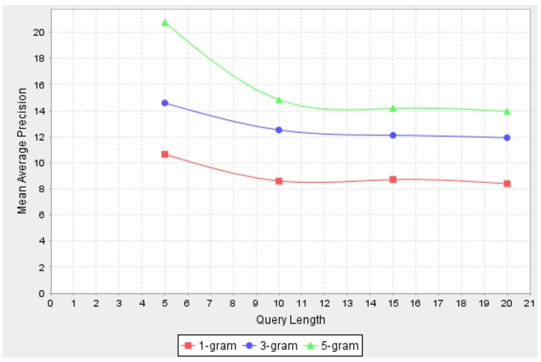
Figure 5: (a) percentage of successful retrievals, (b) mean rank, (c) mean average precision, and (d) matthews correlation of the Reuters-21578 collection trained on feed-forward neural network



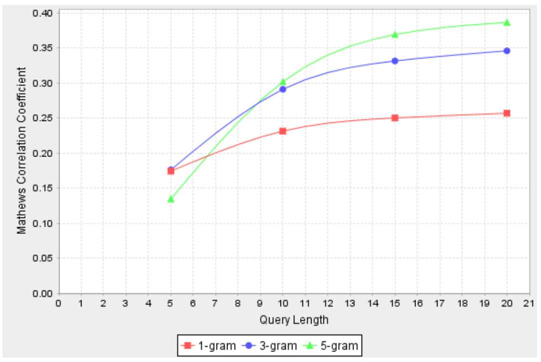
(a)



(b)



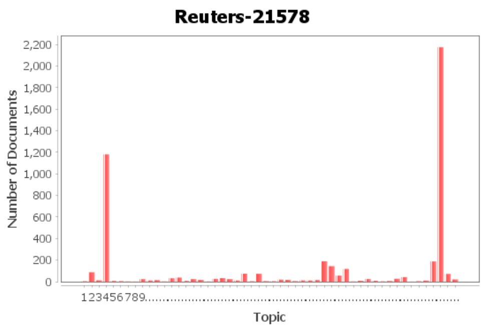
(c)



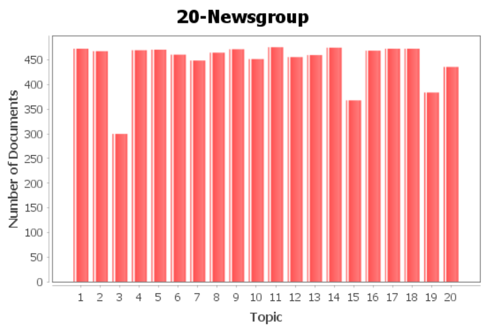
(d)

Figure 6: (a) percentage of successful retrievals, (b) mean rank, (c) mean average precision, and (d) matthews correlation of the Cade collection trained on feed-forward neural network

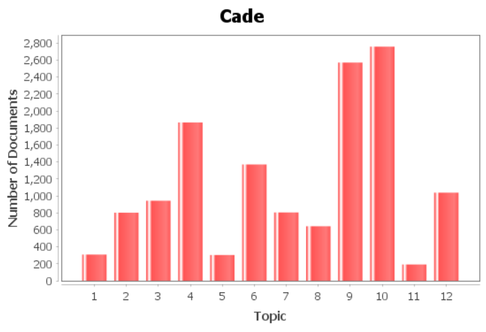
Figure 8 shows snapshot of network runs for a 1-gram model with the state of the output vector after the softmax layers are computed for both topics and documents for each input word. Let us suppose we have a test document “university of saarland germany”, and the collection consists of 7 documents, divided into 3 topics: d0, d1, d2 belong to t0; d3, d4, d5 belong to t1; d6 belongs to t2. Further suppose that the test document belongs to t1 (i.e. Ctarget = t1). For evaluation purposes, let us consider the first 5 documents (at most) that are retrieved. Let us say our model predicts the first two words of the test document belonging to t0 with a probability of 0.6 and 0.5, receptively. The state of the output vector after sorting and averaging is d0 = 0:4; d1 = 0:3; d2 = 0:3. It can be seen that Ctarget is never maximized by the model and hence no document from that topic is retrieved and as of yet the average precision for that query is 0. Let us increase the query length to 3 and suppose that for the next input word saarland, Ctarget is maximized with probability of 0.4 and we are able to retrieve the relevant documents, d3, d4 and d5, for this input word. Now, the state of the output vector after sorting and averaging is d0 = 0:27; d1 = 0:2; d2 = 0:2; d4 = 0:13; d5 = 0:1; d3 = 0:08 and rank is 4. At this stage, the average precision for the query is 0.22. Obviously if Ctarget is not or seldom maximized with a shorter queries, the chances of the highest ranked document from Ctarget to be higher in the output vector for longer queries also decrease which is why rank is poor for longer queries. If we now increase the query length to 4, provide the next word in the test document (i.e., germany) to the model and assume that this time the model predicts this word belonging to topic t2. Since there is only one document belonging to t2, the model will predict the probability of document d6 to be 1.0. The state of the output vector after sorting and averaging at this stage is d6 = 0:25; d0 = 0:2; d1 = 0:15; d2 = 0:15; d4 = 0:1; d5 = 0:08; d3 = 0:06 and the average precision for this query is decreased to 0.07.



(a)



(b)

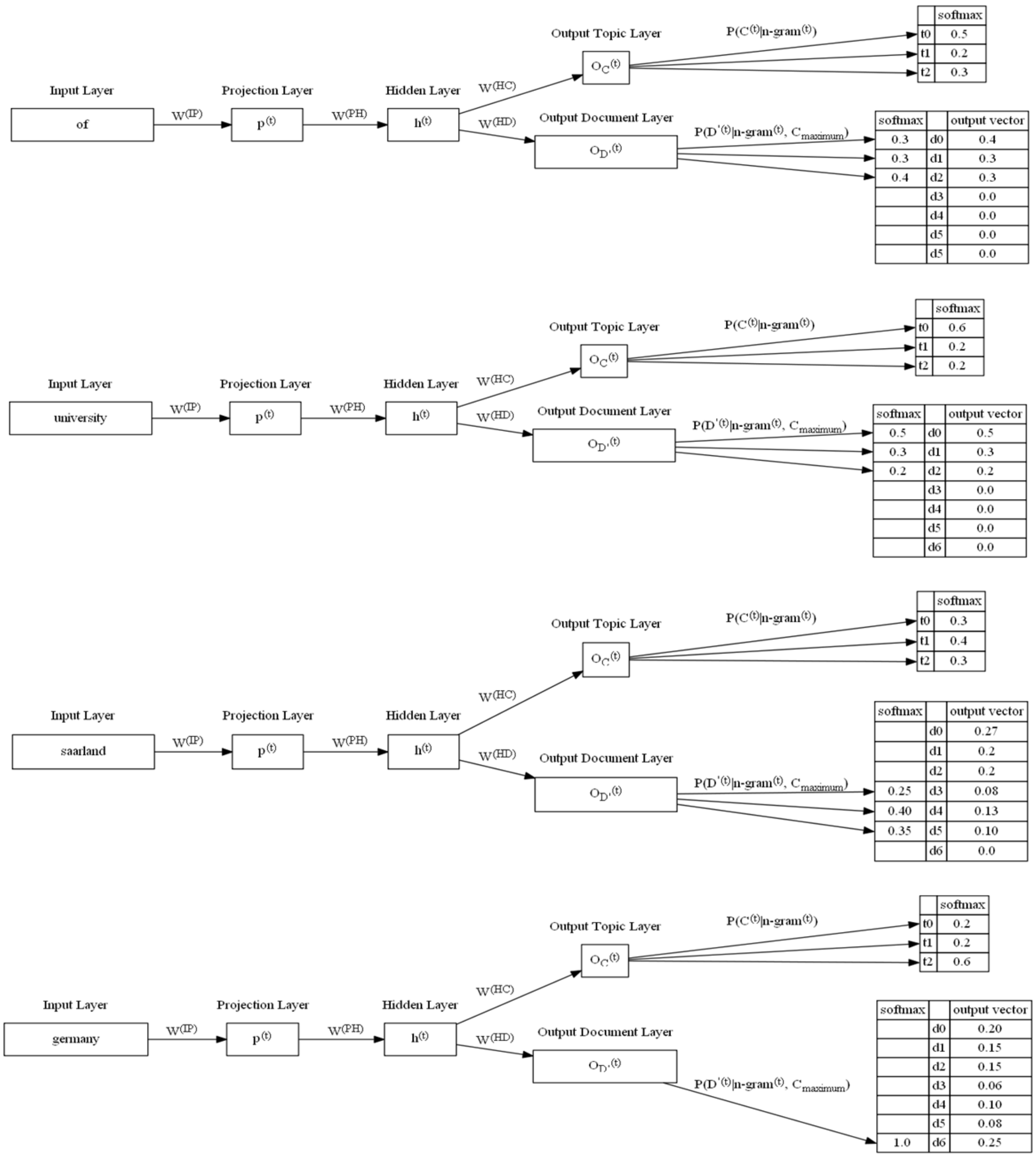


(c)

Figure 7: Skewness of the training data for (a) Reuters-21578, (b) 20-Newsgroup, and (c) Cade collection

We clearly see how a wrong prediction in the output topic layer can result in a drastic decrease in the performance on a model trained on a very skewed dataset. From our example, the document d6 has the highest probability in the output vector and consequently has the highest rank and has pushed the relevant documents d3, d4, and d5 further down in the output vector. Because of this, the chances of these relevant documents to be among the top few documents to be evaluated becomes very low. Thus, the performance of the system trained on skewed datasets gets poorer with an increase in the query length. For the Reuters-21578 collection, the number of successful retrievals increases when the number of words in the query is altered from 5 words to 10 words. After that the number of successful retrievals exhibit a continuous decrease for longer queries which is also depicted in figure 9. For topic 4 (i.e., acq) of that collection, which has 1181 documents out of a total of 4901 in the training dataset, increasing the length of the query has resulted in lesser number of successful retrievals when increasing the query length from 10 to 20 words. Conversely for query length of 15 or 20 words for the 20-Newsgroup collection, the increase in the length of the query has resulted in more relevant documents to be among the top 20 documents that are evaluated as compared to 10 words in the query. A non-skewed collection like 20-Newsgroup is less prone to the wrong predictions when increasing the length of query. For almost all 20 topics of the 20-Newsgroup collection, increasing the length of the query has resulted in more successful retrievals as shown in figure 10. For topics that show slight decrease in number of successful retrievals with increase in query length, wrong predictions for few of the n-grams in the query does not greatly affect the overall performance of the model for that query because of non-skewness of the dataset.

A striking observation can be made when looking at the MAP and MCC metrics. For all the collections trained on forward neural network, there is an inverse relationship between the MAP and the MCC value with respect to change in the length of query. For the Reuters-21578 and Cade collection, very high MCC values are recorded for longer queries along with low MAP scores. The inverse relationship between the MCC values and the MAP scores indicate that for the models trained on feed-forward neural network for the Reuters-21578 and Cade collection, the confidence in the relatively better retrieval output is very low for shorter queries. The high MCC values for low precision scores also suggest that for most of the queries, the models were able to maximize the topic Ctarget at least once and were able retrieve the relevant documents but most probably these documents were low in the output vector because of the skewness of the training dataset. For the 20-Newsgroup collection, different results are observed. For this collection, high MCC values are recorded for longer queries along with high precision scores suggesting that the confidence in relatively better retrieval output for longer queries is greater as compared to shorter queries.



	softmax
t0	0.5
t1	0.2
t2	0.3

softmax		output vector
0.3	d0	0.4
0.3	d1	0.3
0.4	d2	0.3
	d3	0.0
	d4	0.0
	d5	0.0
	d5	0.0

	softmax
t0	0.6
t1	0.2
t2	0.2

softmax		output vector
0.5	d0	0.5
0.3	d1	0.3
0.2	d2	0.2
	d3	0.0
	d4	0.0
	d5	0.0
	d6	0.0

	softmax
t0	0.3
t1	0.4
t2	0.3

softmax		output vector
	d0	0.27
	d1	0.2
	d2	0.2
0.25	d3	0.08
0.40	d4	0.13
0.35	d5	0.10
	d6	0.0

	softmax
t0	0.2
t1	0.2
t2	0.6

softmax		output vector
	d0	0.20
	d1	0.15
	d2	0.15
	d3	0.06
	d4	0.10
	d5	0.08
1.0	d6	0.25

Figure 8: Example to demonstrate the effect on evaluation metrics with alteration in query length for collections trained on feed-forward model

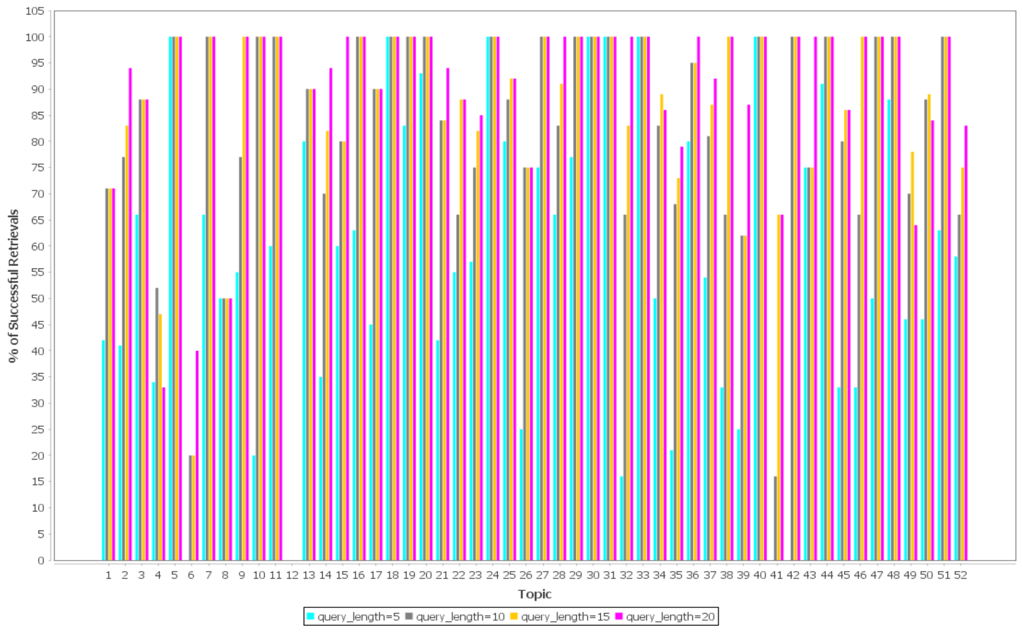


Figure 9: Percentage of successful retrievals for each topic of Reuters-21578 collection with increase in length of query for 5-gram model

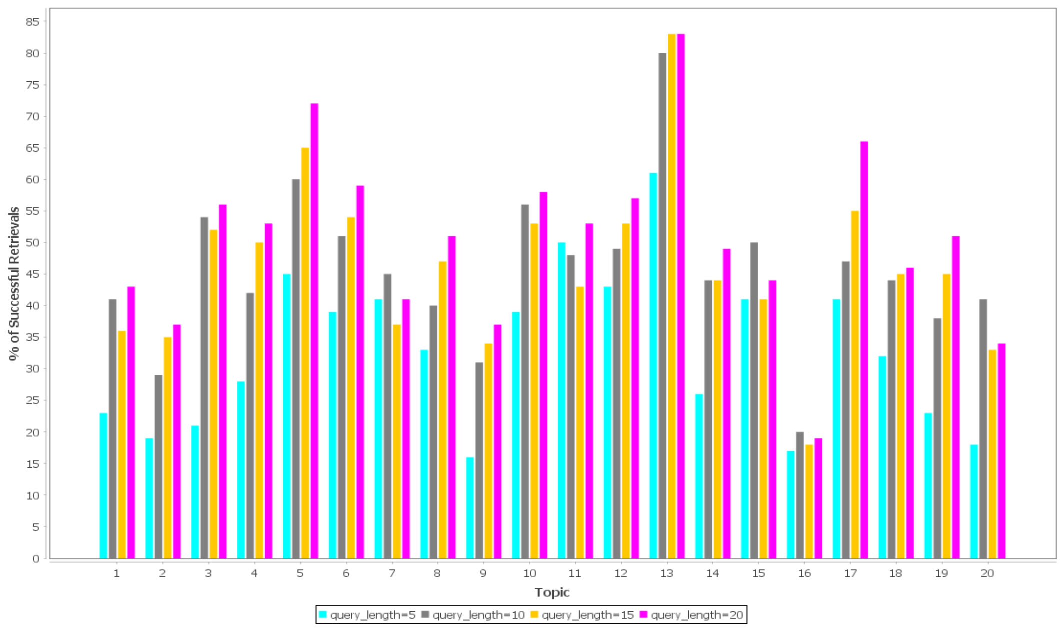


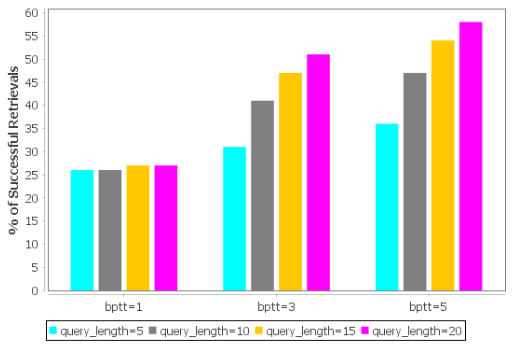
Figure 10: Percentage of successful retrievals for each topic of 20-Newsgroup collection with increase in length of query for 5-gram model

5.2. Retrieval Effectiveness for RNN

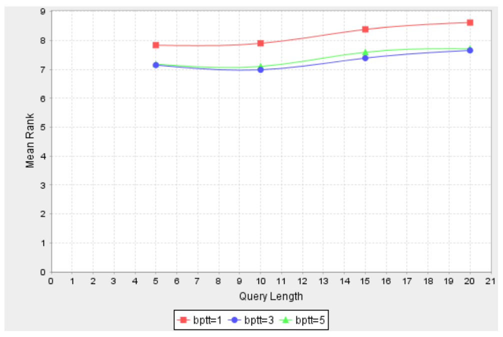
In the previous passage, we described the effect on retrieval output when changing the number of words in query for models trained on feed-forward neural network. On the other hand, when we ran the queries on the models trained on recurrent neural network, we saw slightly different results for Reuters-21578. In contrast to results obtained from the feed-forward neural network model trained on this collection, we saw an improvement in retrieval output when we altered the length of query from 5 words to 20 words for higher order bptt parameter. The improvement in performance reflects the ability of recurrent neural networks to capture the context in the passage of text. The effectiveness of recurrent neural networks can only be seen by allowing the network to see relatively greater number of words in input query as compared to feed-forward network.

The most significant observation made when evaluating retrieval output for recurrent models trained on all the collections was the shift in performance with an increase in the bptt parameter. For the Reuters-21578 collection, the network was able to learn and generalize well across sequences of words in a query for a bptt parameter of 3 and 5. For the Cade collection, storing a context of up to 3 words gave the best performance whereas for 20-Newsgroup collection, the performance actually degraded by storing any context while training. This degradation in performance can be explained by the fact that many times local context does not provide the most useful predictive clues, which instead are provided by long distance dependencies for which long short-term memory neural networks (Hochreiter and Schmidhuber, 1997) are used. Although recurrent neural networks are able to connect past information in order to better inform about the current prediction, this is not the case every time. In cases where the gap between the relevant information and the place that information is needed is small, recurrent neural network can learn to use that past information. But there are also cases where a larger context is needed in order to predict the current word and where the gap between the relevant information in context and the point where that information is needed increases. Long-term memory based neural networks are a special kind of recurrent networks, that successfully cater to the problem of remembering information for longer periods of time and work well where there is an increased gap between the relevant information and the point where it is required.

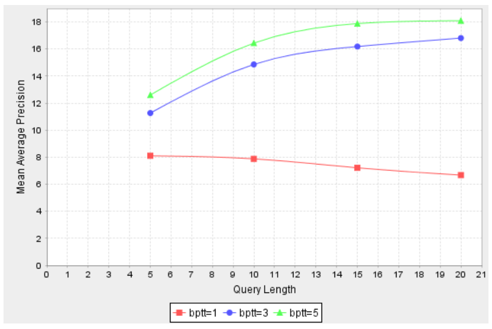
It is seen that the recurrent models trained on only the Reuters-21578 collection, having a vocabulary size of 8862 words, performed better when we increase the number of words in the query. For the 20-Newsgroup and Cade collection, we observed degradation in performance for longer queries. A larger vocabulary of 20-Newsgroup and Cade collection has more tendency to have long distance dependencies in the text because of which the models were not able to generalize well for longer queries. One of the reasons of using the truncated back propagation through time algorithm is that the algorithm suffers from vanishing gradient problem. Whenever the gradient of the error function of the neural network is propagated back through time, it gets scaled by a certain factor which is either greater than one or smaller than one. As a result, the gradient either blows up or decays exponentially over time. Thus, the gradient either dominates the next weight adaptation step or effectively gets lost.



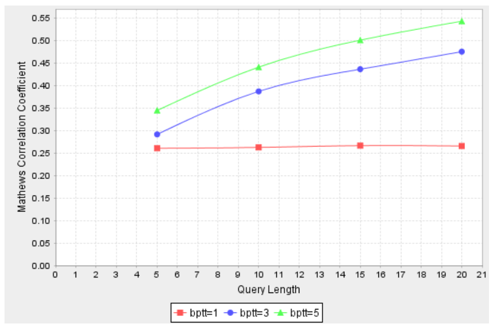
(a)



(b)

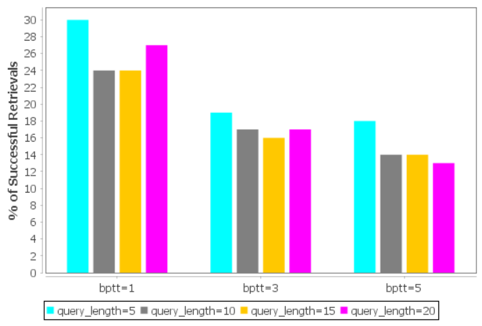


(c)

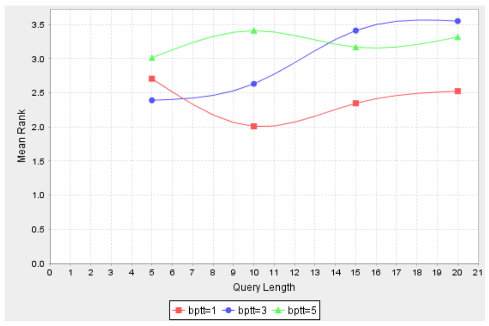


(d)

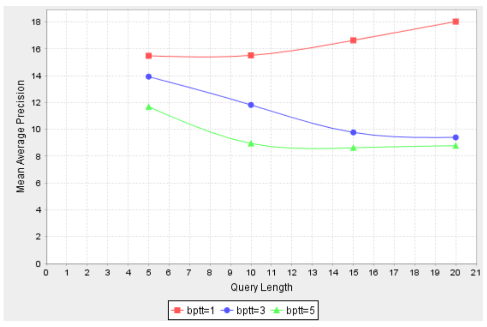
Figure 11: (a) percentage of successful retrievals, (b) mean rank, (c) mean average precision, and (d) matthews correlation of the Reuters-21578 collection trained on recurrent neural network



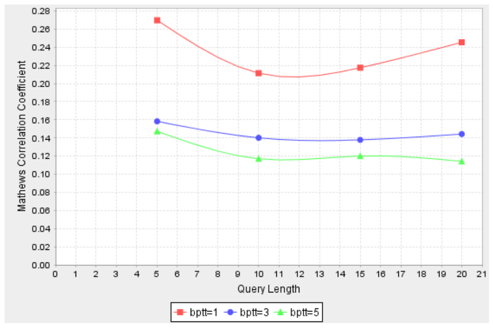
(a)



(b)

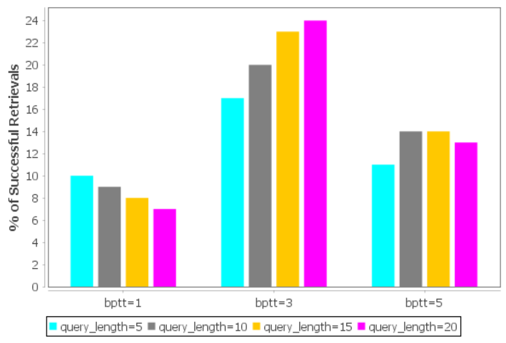


(c)

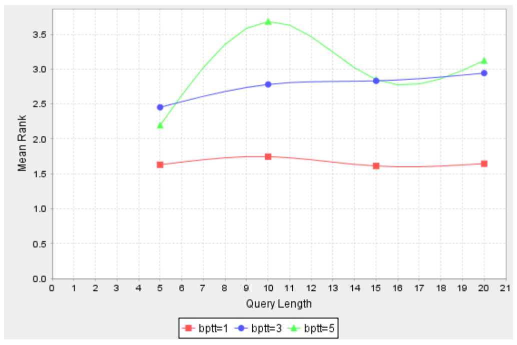


(d)

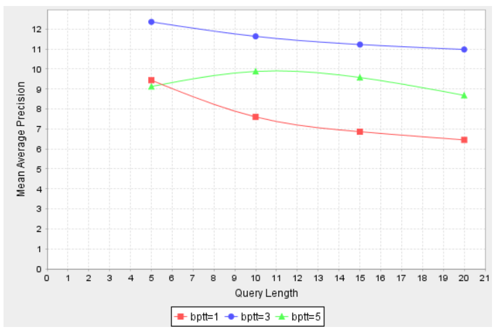
Figure 12: (a) percentage of successful retrievals, (b) mean rank, (c) mean average precision, and (d) matthews correlation of the Reuters-21578 collection trained on recurrent neural network

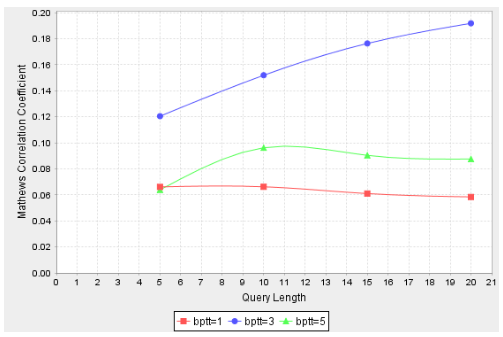


(a)



(b)





(c) (d)

Figure 13: (a) percentage of successful retrievals, (b) mean rank, (c) mean average precision, and (d) matthews correlation of the Cade collection trained on recurrent neural network

5.3. Best Performance

In theory, the recurrent neural networks consider the long-term dependencies when modeling text in natural language. In practice, however, learning long term dependencies with gradient descent on a document modeling application when the number of words are small is a difficult task. For Reuters-21578 and Cade collection, average number of words trained in batch from a document were 10. For 20-Newsgroup collection, that number was 20. The work on recurrent neural networks described in (Mikolov et al., 2010) does not address this problem. That work focuses more to democratize the use of recurrent neural networks for the application of language modeling by making them relatively fast to train in comparison to old techniques.

Table 3 shows a comparison of the mean average precision scores for models trained on both feed-forward and recurrent neural network for all the collections. The comparison is shown for a query length of 5 since it is very unlikely for a user to query a search engine with 10 or more words. The best performance on a collection is highlighted in bold and it can be seen that a 5-gram feed-forward model gives the best performance on all the collections. The fact that long term dependencies are still difficult to learn in case of document modeling would argue in favor of using n-gram sequences as an input to the neural network.

The main advantages of using a recurrent neural network over feed-forward neural network would be the greater representational power of recurrent neural networks and their ability to perform intelligent smoothing by considering syntactic and semantic features but we have seen in this study that it does not lead to very good results on the application of document modeling where we train very few words from a document. Although n-gram based feed-forward neural network does not solve the problem of n-gram context that expresses the semantic character of text but by comparison this approach works better in determining the topic of the document from input n-gram. The representation by n-grams assumes that high probability is assigned for those input words that co-occur and low probability is assigned for those input words that do not co-occur without caring too much about where they appear in the document. The architectural setup of feedforward neural network is relatively simple and modeling systems for document retrieval task is also fairly easy. For this reason, their application is easily verified and are much more suitable for text document retrieval for predefined document set structures.

Table 3: MAP scores for query length of 5.

	FFNN			RNN		
	1-gram	3-gram	**5-gram**	bptt=1	bptt-3	bptt=5
20-Newsgroup	15.52	21.30	**33.59**	15.47	13.92	11.68
Reuters-21578	7.82	28.15	**45.05**	8.10	11.26	12.59
Cade	10.65	14.58	**20.77**	9.43	12.36	9.12

6. CONCLUSION

We described the implementation of feed-forward and recurrent neural networks and have reported the performance of both networks with respect to retrieval output on altering the length of the query. We also showed how a wrong prediction in the output topic layer can lead to significant decrease in performance in case of a skewed dataset. As a first step, we showed how altering the length of the query affects the retrieval output of feed-forward and recurrent models. Moreover, a comprehensive analysis of feed-forward and recurrent neural network architectures was provided. We saw that backpropagation through time algorithm does not improve the performance of retrieval as compared to standard backpropagation on the application of document retrieval mainly due to small number of words trained from the documents in batch.

The results show that the retrieval system returns best MAP scores for a 5-gram feed-forward model. If our target topic is maximized with lesser number of words in a query, the chances of irrelevant documents to be in the top 20 documents gets minimized since there is less opportunity for the model to maximize a non-target topic. More relevant documents seen in the output vector also increase the chances of highest ranked document from the target topic to be higher in the output vector as well. Although the number of successful retrievals is low for shorter queries, but whenever a target topic is maximized for a given 5-gram, most of the documents from that topic appear to be among the top 20 documents. This also explains why MAP scores are better when the model is presented with fewer number of words in query. In other words, the decrease in performance with longer queries can also be explained with the fact that our design assumes of user foreseeing the exact words and phrases belonging to that topic and only to that topic. Consequently, longer word phrases lead to smaller chances of its subset belonging to the same topic.

Moreover, we showed that using higher order bptt parameter to store context information does improve the performance of the retrieval system in some cases but in other cases local context is unable to provide effective clues in prediction, which instead are provided by long distance context for which long short-term memory neural networks are used. We saw that learning context information with backpropagation through time algorithm in case of recurrent neural network does not outperform the standard backpropagation algorithm of feed-forward neural network since it is difficult to capture those dependencies with relatively fewer words from document as input in the batch.

7. FUTURE WORK

In future, it will be interesting to investigate how our approach compares with keyword-based models presented in (Skovajsova, 2010) and with recent research on vector based and Boolean models on the task of document retrieval. It will be interesting to train the models on a two sub-system network with non-factorized output layer, the architecture of which is somewhat similar to the cascade neural network presented in (Skovajsova, 2010). The inputs to the networks are n-gram(s) or word(s), and our implementation learns the input to topic and input to document relationship in a single pass with the error backpropagating from both the output layers to the hidden layer. An alternative to this approach would be to just learn input to topic relationship in the first subsystem for all the documents in the collection and then later learn the input to document relationship in the second subsystem.

REFERENCES

1. Arnold, K., Gosling, J., and Holmes, D. (2000). The Java Programming Language. Addison-Wesley Longman Publishing Co., Inc. Boston, MA, USA.
2. Baeza-Yates and Ribeiro-Neto (1999). Modern Information Retrieval. Addison Wesley/ ACM Press.
3. Bengio, Y., Ducharme, R., Vincent, P., and Jauvin, C. (2003). A Neural Probabilistic Language Model. Journal of Machine Learning Research, 3:1137-1155.
4. Blackwood, G. W. (2005). Neural Network-based Language Model for Conversational Telephone Speech Recognition. PhD thesis, St. Catherine's College.
5. Bullinaria, J. A. (2015). Neural Computation.
6. Cardoso-Cachopo, A. (2007). Improving Methods for Single-label Text Categorization. PhD thesis, Instituto Superior Tecnico, Universidade Tecnica de Lisboa.
7. Chen, H. (1995). Machine Learning for Information Retrieval: Neural Networks, Symbolic Learning, and Genetic Algorithms. Journal Of The Americal Society For Information Science, 46(3):194-216.
8. Chen, S. F. and Goodman, J. (1999). An Empirical Study of Smoothing Techniques for Language Modeling. Computer Speech and Language, 13:359-364.
9. Cheung, V. and Cannons, K. (2002). An Introduction to Neural Networks.
10. Cordon, O., Moya, F., and Zarco, C. (2002). A New Evolutionary Algorithm Combining
11. Simulated Annealing and Genetic Programming for Relevance Feedback in Fuzzy Information Retrieval Systems. Soft Computing - A Fusion of Foundations, Methodologies and Applications, 6(5):308-319.
12. Deepak, G. and Deepika, S. (2012). Information Retrieval on the Web and its Evaluation. International Journal of Computer Applications, 40(3):70-78.
13. Dunne, R. A. and A., C. N. (1997). On the pairing of the softmax activation and crossentropy penalty functions and the derivation of the softmax activation function. In 8th Australian Conference on Neural Networks, Melbourne, Australia, pages 181-185.
14. Emami, A. and Jelinek, F. (2004). Exact Training of a Neural Syntactic Language Model. In ICASSP, pages 245-248.
15. Gauvain, J., Adda, G., Adda-Decker, M., Allauzen, M., Gendner, V., Lamel, L., and H., S. (2005). Where Are We In Transcribing BN French? In Eurospeech, pages 1665-1668.
16. Goodman (2001). A Bit of Progress in Language Modeling Extended Version. Machine Learning and Applied Statistics Group.
17. Herrera-Viedma (2001). Modelling the Retrieval Process for an Information Retrieval System using an Ordinal Fuzzy Linguistic Approach. Journal of the American Society for Information Science and Technology, 52(6):460-475.
18. Hochreiter, S. and Schmidhuber, J. (1997). Long Short-Term Memory. Neural Computation, 9(8):1735-1780.
19. Hotho, A., Nurnberger, A., and Paas, G. (2005). Brief Survey of Text Mining. LDV-forum, 20(1):19-62.
20. Jain, L. C. and L.R., M. (1999). Recurrent Neural Networks: Design and Applications. CRC Press, Inc. Boca Raton, FL, USA.
21. Jurman, G., Riccadonna, S., and Furlanello, C. (2012). A Comparison of MCC and CEN Error Measures in Multi-Class Prediction. PLoS ONE, 7(8).
22. Lai, S., Xu, L., Liu, K., and Zhao, J. (2015). Recurrent Convolutional Neural Networks for Text Classification. In Proceeding AAAI'15 Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence, pages 2267-2273.
23. Lan, M., Tan, C., Low, H., and Sung, S. (2005). A Comprehensive Comparative Study on Term Weighting Schemes for Text Categorization with Support Vector Machines. In Special interest Tracks and Posters of the 14th international Conference on World Wide Web, Chiba, Japan, pages 1032-1033.

24. Le, Q. and Mikolov, T. (2014). Distributed Representations of Sentences and Documents. Proceedings of the 31st International Conference on Machine Learning, Beijing, China, 32:1188-1196.
25. Lin, R., Liu, S., Yang, M., Li, M., Zhou, M., and Li, S. (2015). Hierarchical Recurrent Neural Network for Document Modeling. Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, Lisbon, Portugal, pages 899-907.
26. Manning, C. D., Raghavan, P., and Schütze, H. (2008). Introduction to Information Retrieval. Cambridge University Press.
27. Matthews, B. W. (1975). Comparison of the predicted and observed secondary structure of T4 phage lysozyme. Biochimica et Biophysica Acta (BBA) - Protein Structure, 405(2):442-451.
28. Mikolov, T., Karafiat, M., Burget, L., Cernocky, J. H., and Khudanpur, S. (2010). Recurrent neural network based language model. INTERSPEECH, 11th Annual Conference of the International Speech Communication Association, Makuhari, Chiba, Japan.
29. Mokris, I. and Skovajsova, L. (2005). Neural Network Model Of System For Information Retrieval From Text Documents In Slovak Language. Acta Electrotechnica et Informatica, 5(3).
30. Nickolls, J., Buck, I., Garland, M., and Skadron, K. (2008). Scalable Parallel Programming with CUDA. ACM Queue, 6(2):40-53.
31. Ogilvie, P. and Callan, J. (2003). Language Models and Structured Document Retrieval. Proceedings of the First Workshop of the Initiative for the Evaluation of XML Retrieval (INEX), Delos.
32. Porter (1980). An algorithm for suffix stripping. Program, 14(2):130-137.
33. Powers, D. (2007). Evaluation: From Precision, Recall and FFactor to ROC, Informedness, Markedness & Correlation. Human Communication Science SummerFest.
34. Rumelhart, D., Hinton, G. E., and R.J., W. (1986). Learning representations by backpropagating errors. Nature, 323:533-536.
35. Salton, G., Edward, A. F., and Wu, H. (1983). Extended Boolean Information Retrieval. Communications of the ACM, 26(11):1022-1036.
36. Scheir, P. and Lindstaedt, S. N. (2006). A Network Model Approach to Document Retrieval taking into account Domain Knowledge. Lernen-Wissendeckung-Adaptivitat, pages 154-158.
37. Schwenk, H. and Gauvain, J. (2004). Neural Network Language Models for Conversational Speech Recognition. In ICSLP, pages 1215-1218.
38. Skovajsova, L. (2010). Text Document Retrieval by Feed-forward Neural Networks. Information Sciences and Technologies Bulletin of the ACM Slovakia, 2(2):70-78.
39. Svozil, D., Kvasnicka, V., and Pospichal, J. (1997). Introduction to multilayer feed-forward neural networks. Chemometrics and Intelligent Laboratory Systems, pages 43-62.
40. Treeratpituk, P. and Callan, J. (2006). An Experimental Study on Automatically Labeling Hierarchical Clusters using Statistical Features. Annual ACM Conference on Research and Development in Information Retrieval, pages 707-708.
41. Turney, P. and Pantel, P. (2010). From Frequency to Meaning: Vector Space Models of Semantics. Journal of Artificial Intelligence Research, 37(1):141-188.
42. van Rijsbergen (1979). Information Retrieval (2nd edition).
43. Wang, S., Schuurmans, D., Peng, F., and Zhao, Y. (2005). Combining Statistical Language Models via the Latent Maximum Entropy Principle. Machine Learning, 60(1-3):229-250.
44. Wei, X. and Croft, B. (2006). LDA-based Document Models for ad-hoc Retrieval. Proceedings of the 29th Annual International ACM SIGIR Conference on Research and
45. Development in Information Retrieval, Seattle, Washington, USA, pages 178-185.
46. Werbos (1990). Backpropagation Through Time: What It Does and How to Do It. Proceedings of the IEEE, 78(10):1550-1560.
47. Zobel, J. and Moffart, A. (2006). Inverted Files for Text Search Engines. ACM Computing Surveys (CSUR), 38(2).