545 Machine Learning, Fall 2011

Final Project Report

Experiments in Automatic Text Summarization Using Deep Neural Networks

---

Project Team:
Ben King
Rahul Jha
Tyler Johnson
Vaishnavi Sundararajan

Instructor:
Prof. Clayton Scott

December 16, 2011

Abstract

This project aims at applying neural network-based deep learning to the problem of extractive text summarization. Our work is inspired by the work of Collobert and Weston [Collobert et al., 2011], who created a unified deep learning architecture to learn several common NLP tasks. In this report, we give the motivation behind our work, describe our problem formulation and present some results.

1 Introduction

Deep learning is an emerging field of machine learning that has been applied to a wide variety of problems including vision, robotics and information retrieval. Recently, deep learning techniques have been applied to Natural Language Processing tasks with very encouraging results [Collobert and Weston, 2008], [Collobert et al., 2011]. Their work focusses on the lower level sequence labelling tasks in NLP such as Part of Speech Tagging, Named Entity Recognition and Semantic Role Labelling and is quite accurate for all of these tasks, which is unprecedented. This work inspired us to try the Deep learning experiment on a higher level NLP task. After looking at several options, we chose automatic text summarization.

Automatic text summarization is a well-established problem area in Natural Language Processing. The summarization community has produced a significant corpora of training and test data, especially in the Document Understanding Conferences (DUC) and SummBank (DUC 2003-2007, SummBank) and a number of evaluation measures (ROUGE, Relative Utility). An automatic summary is defined as: "a text that is produced from one or more texts, that conveys important information in the original text(s), and that is no longer than half of the original text(s) and usually signicantly less than that." [Radev et al., 2002]. Several approaches have been used for summarization, and we present a short survey of these in Section 2.

The rest of this report is organized as follows. Section 2 introduces the summarization problem and presents a short survey on existing techniques. It also gives a basic background of Deep Learning relevant to our work. We then present the motivation for our work in Section 3. Section 4 describes our problem formulation and experiment for applying deep learning to summarization. It also provides the details of the data we used. We present our results in Section 5 and a conclusion in Section 6. Finally, we provide a description of individual effort in Section 7.

2 Background

2.1 Summarization

Summarization emerged as an important field because of the problem of information overload: people frequently need to sift through several documents quickly to get to the important information. There are two kinds of summaries, each being useful in different contexts: **indicative** summaries suggest the content of the original document to help the user decide

whether he/she wants to read the complete document; **informative** summaries attempt to serve as a succinct replacement of the original document.

Based on experiment, summarization methods usually fall into two broad categories: **extractive** and **abstractive**. Extractive summarization focusses on identifying important sentences in a document and producing them verbatim. Abstractive summarization puts strong emphasis on the form, aiming to produce a grammatical summary and calls for advanced language generation techniques. Summarization can also be either **single-document** or **multi-document**, each requiring different strategies. In this work, we focus on generating single-document, extractive summaries.

Early research on summarization focussed on extracting salient sentences from text using features like word and phrase frequency [Luhn, 1958], position in the text [Baxendale, 1958] and key phrases [Edmundson, 1969]. In the 90's several publications appeared that employed statistical machine learning techniques for automatic text summarization. [Kupiec et al., 1995] described a Naive Bayes method to classify document sentences as worthy of extraction or not based on features in [Edmundson, 1969]. [Aone et al., 1999] also incorporated a naive-Bayes classifier, but with richer features. [Lin, 1999] used decision trees to model the problem of summarization.

More recently, [Conroy and O'leary, 2001] modeled the problem of extracting a sentence from a document using a hidden Markov model (HMM). [Osborne, 2002] decided to go past the feature independence assumption by using a maximum entropy log linear model and showed better results than Naive Bayes with a prior appended to both models. [Svore, 2007] proposed an algorithm based on neural nets and the use of third party datasets and reported good results outperforming the baseline with statistical significance.

Several other methods for summarization based on deeper linguistic features and text structure have been developed. Multi-document summarization has gained popularity in recent years as well. For a detailed survey of the area, please refer to [Das and Martins, 2007].

2.2 Deep Learning

A traditional neural network has three layers - one input layer, one hidden layer and one output layer. **Deep neural networks** are modified neural networks with multiple hidden layers. Each of these hidden layers has a non-linear activation function, and learns features incrementally (for example, in image data, it learns edge detectors, then face detectors and so on), which allows us to compute more complex features than possible with a vanilla neural network. This is the key advantage of using deep networks for representation.

2.3 Training deep architectures

Till very recently, most research in the area of deep networks involved randomly initializing the weights of a deep network, and then training it with a labeled training set using a supervised learning objective (for example, by applying gradient descent). However, this did not work well, since data was not easily available, and there were various problems involving local optima (which is a much rarer occurrence for a vanilla neural network) and diffusion of

gradients (gradients become very small, and as a result of backpropagation, the derivative of the overall cost with respect to the weights in the earlier layers is very small).

The newer method that has seen some success is greedy layer-wise training. The layers of the network are trained one at a time, so that we first train a network with 1 hidden layer, and only then train a network with 2 hidden layers, and so on. This is often done in an unsupervised manner using **autoencoders**. The weights from training the layers individually are then used to initialize the weights in the overall deep network, and only then the entire architecture is trained together to optimize the labeled training set error. This method is presumed to better because there is no data paucity problem, and the problem of local optima is eliminated since weights are no longer initialized randomly.

2.4 Autoencoders

An autoencoder neural network is an unsupervised learning algorithm that applies backpropagation, setting the target values to be equal to the inputs. It tries, using backpropagation, to learn an approximation to the identity function, such that $h_{W,b}(x) = \hat{x} \approx x$. Since there are fewer hidden units than inputs, the network is forced to learn a compressed representation of the input. Even if the hidden layer had more units, we could impose sparsity constraints on the network and discover input structure. If the input were totally random, this would be very difficult, but if some features of the input are correlated, this simple autoencoder often ends up learning a low-dimensional representation very similar to PCAs.

2.4.1 Backpropagation

We use backpropagation by batch gradient descent in order to train our autoencoder network. The weights are represented by the matrix W and the bias values by the matrix b, for each layer. The goal is to minimize cost J wrt W and b.

In order to train the network, we start off with random values of $W^{(l)}$ and $b^{(l)}$ (random initialization is important to avoid all hidden units learning the same function). Now gradient descent is used to minimize the cost function, by calculating the activation functions $a_i^{(l)}$ for each node in each layer, using the partial derivatives of J wrt W and b, and updating their values accordingly.

2.4.2 Sparsity

We choose a **sparsity parameter** ρ, very close to zero, which is what we would like $\hat{\rho}_j$ to be, for all j values. The very close to zero constraint implies that many activations average to almost zero, giving us a sparser matrix than earlier. We modify the cost function to take into account this parameter ρ and try to minimize the cost function now.

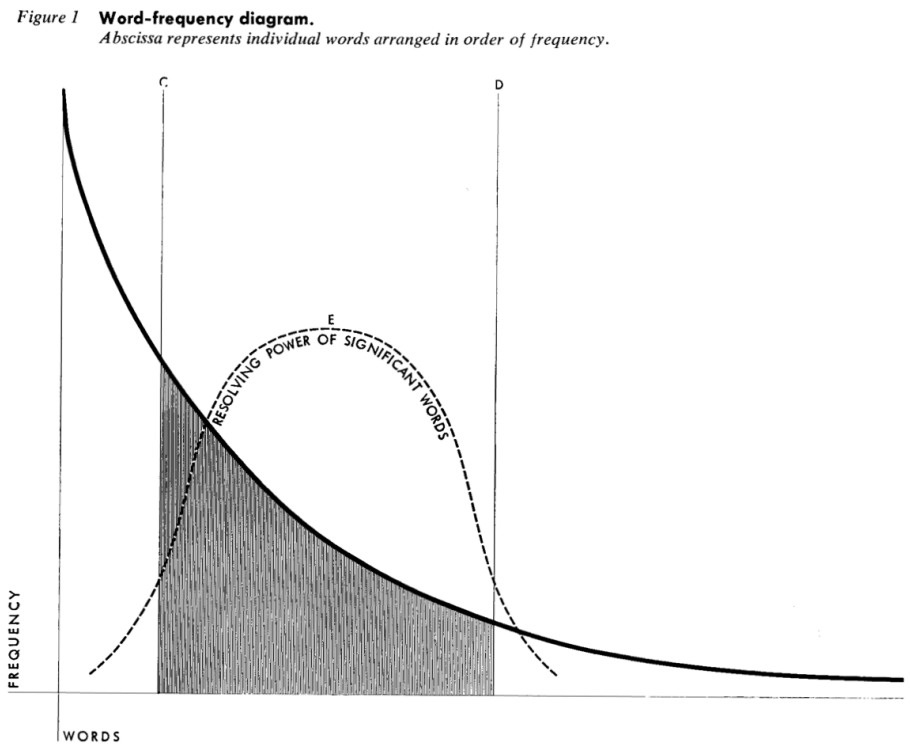

Figure 1: Courtesy [Luhn, 1958]

3 Motivation & Problem Formulation

It is an established fact that the high frequency words in a document are a good indicator of the salience of a sentence for inclusion in its summary [Luhn, 1958]. However, the highest frequency words include irrelevant words like the articles (a, an, the) and other such stop words. The frequency distribution of the words in any document in fact follows a power law and there is both a lower cut-off and an upper cut-off on the frequency of words that should be used to determine the salience of sentences for a document as shown in Figure 1.

We computed the frequencies of the words on both our unlabelled training data, labelled training data and test data and then plotted their average frequency for each sentence. The distribution of words follows a power law as we expected. Figure 2 shows the graphs for the average frequency of words in a sentence for 250 of the highest frequency words in a document.

This motivated us to encode our sentence in terms of a feature vector representing the frequency of the 250 highest frequency words in the document. We hoped that the autoencoder would learn a condensed representation of these features. For the supervised learning part, we added sentence length and sentence position as features as well and then trained our neural network to learn the utility of a sentence for the summary given these feature vectors. We describe experiments and results are described in the next section.

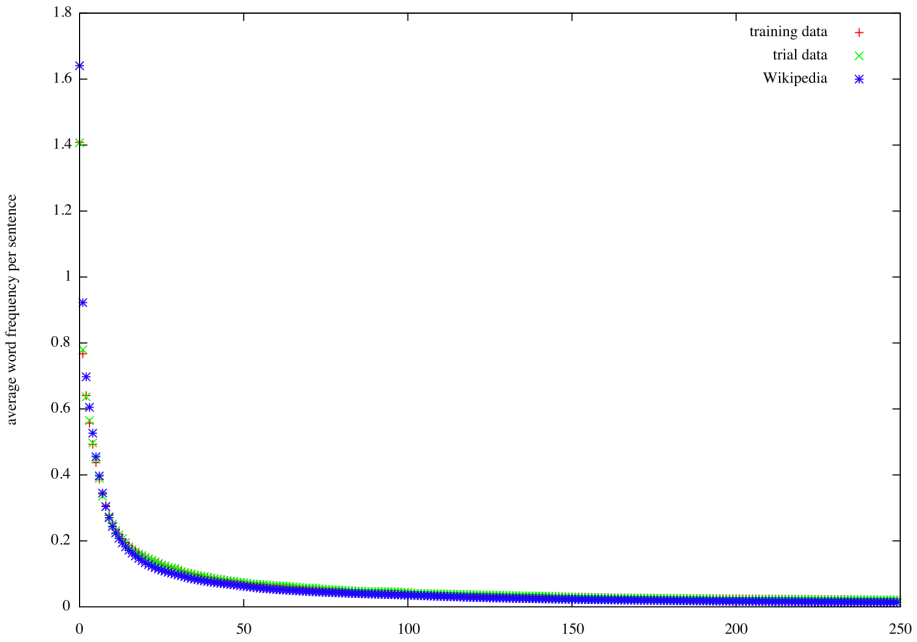

Figure 2: Average sentence frequencies for the top 250 highest frequency words

3.1 Problem Formulation

We first trained an autoencoder on unlabelled sentences to learn a condensed feature representation. We then trained another neural network using supervised learning to classify sentence salience for summarization based on these features. We theorized that The question we sought to answer was whether using the autoencoder helped in increasing the accuracy of the neural network or not. We also experimented with changing the feature representations in other ways.

4 Experiments

4.1 Data Used

For training the auto-encoder we used data from Wikipedia articles. The corpus consisted of about 25, 000 sentences from Wikipedia. We used DUC (Document Understanding Conference) 2001 data for the training and evaluation of the supervised neural network. For training, we used the DUC 2001 training data which contains 30 clusters of 10 newswire/paper documents. For each document, a 100 word summary is provided. We evaluated our trained neural network on 2 clusters of DUC 2001 test data consisting of about 60 documents. For each of the documents, we had 3 model summaries to evaluate against. For comparison, we had one baseline summary and 10 peer summaries (from participating teams in DUC 2001).

4.2 Experiment Setup

As features, we use the 250 most common words appearing in a document. The premise, as discussed earlier, is that the presence of these words in a particular sentence characterize the importance of that sentence towards accurately expressing the crux of the ideas presented in the document, thereby allowing us to make a correct choice towards including that sentence in our summary of the document or not.

4.3 Autoencoder training

For training the autoencoder, we followed a process similar to that used in [Ng et al., 2010]. In [Ng et al., 2010], Ng et al set up the experiment such that they give 10000 random 8x8 patches (reshaped as 64x1 vectors) from any one of 10 images (randomly picked) as input to the autoencoder and use backpropagation to train it, after adding constraints for regularization and sparsity. The autoencoder in [Ng et al., 2010] is a 3-layer neural network, with 64 nodes in the input and output layers, and 25 nodes in the hidden layer. It uses the sigmoid activation function for both the weights of the hidden layer and the output layer. We also used a 3-layer autoencoder and used backpropagation to train it, except our autoencoder had 250 nodes in the input and output layers, and 50 nodes in the hidden layer. Each input to the autoencoder represented a sentence in the document, and was a 250x1 vector of the frequencies in that sentence of the 250 most common words in that particular document.

4.4 Neural Network training

We used another 3-layer neural network to perform supervised learning and evaluate the performance of our setup. This network takes 252 inputs - the 250 feature values, the sentence position, and the sentence length. It had the same number of input and hidden nodes, and used the *tanh* activation function. The learning rates for the weights of the hidden and output layers were 0.0005 and 0.00005 respectively. This neural network gave as output a real-valued utility score for the sentence - the higher the utility score, the better the sentence is at accurately summarizing the document it belongs to.

We trained the neural network for summarization by first passing the DUC 2001 training data through the autoencoder and then training the neural network on the learned features for summarization. It tries to predict the utility of each sentence against a human generated summary by counting the number of overlapping unigrams. We finally evaluated the neural network by passing the test data through the autoencoder and then estimating the utility of each sentence using our trained neural network. The top sentences within the 100 word limit were then selected to create a summary.

5 Results

Implementing the autoencoder as specified in [Ng et al., 2010] had given us an output where the autoencoder managed to learn a set of edge indicators from the images. Each node in

System	Minimum	Maximum	Average
Baseline	0.4284753125	0.504350625	0.4680203125
test_summaries_autoencoded_with_stopwords	0.3242725	0.3640134375	0.3450025
test_summaries_with_stopwords	0.338948125	0.3807453125	0.360644375
test_summaries_autoencoded_without_stopwords	0.322035	0.356385	0.33894125
test_summaries_without_stopwords	0.3432571875	0.3876378125	0.365981875
test_summaries_no_word_features	0.32534375	0.367369375	0.3455765625
Peer 1	0.43093375	0.4869109375	0.459663125
Peer 2	0.4169121875	0.466843125	0.444074375
Peer 3	0.4290784375	0.483383125	0.4564465625
Peer 4	0.430373125	0.48346375	0.4564340625
Peer 5	0.3873	0.4446290625	0.415354375
Peer 6	0.411035625	0.4686409375	0.44121125
Peer 7	0.3722734375	0.441584375	0.4072540625
Peer 8	0.3881190625	0.45004	0.4222859375
Peer 9	0.4071978125	0.4678815625	0.439105625
Peer 10	0.37901375	0.435126875	0.40756875

Table 1: Rouge Evaluation Results

the hidden layer learnt one type of edge indicator. We believed that our autoencoder would learn a similar set of features combinations from the input features as well. In particular, we wanted it to give higher weights to the set of features that are within the threshold limits of what [Luhn, 1958] reported (Figure 1). We saw the expected results upon plotting the maximum weight that every input node received across all hidden nodes as shown in Figure 5. Also, since the autoencoder tries to learn the identity function, we expected to see a similar distribution of weights on the hidden nodes for the input and output nodes. We confirmed this intuition by plotting these weights as shown in Figures 4 and 5.

We ran the DUC 2001 test data through our auto-encoder and then through our trained neural network. We then evaluated the ROUGE scores for the summary against 3 model summaries generated by humans. We also ran the baseline summaries and summaries from 10 peers (from the DUC conference) through the same pipeline. The results are presented in Table 1.

6 Discussion

For our project, we framed the problem of document summarization as a regression problem, taking individual sentences from a document as input vectors and estimating a relevance value as output. This approach requires embedding natural language sentences as mathematical vectors, and to do so, we used a method based on the frequency of each word in the document. We hoped that we could apply concepts from deep learning to combine these

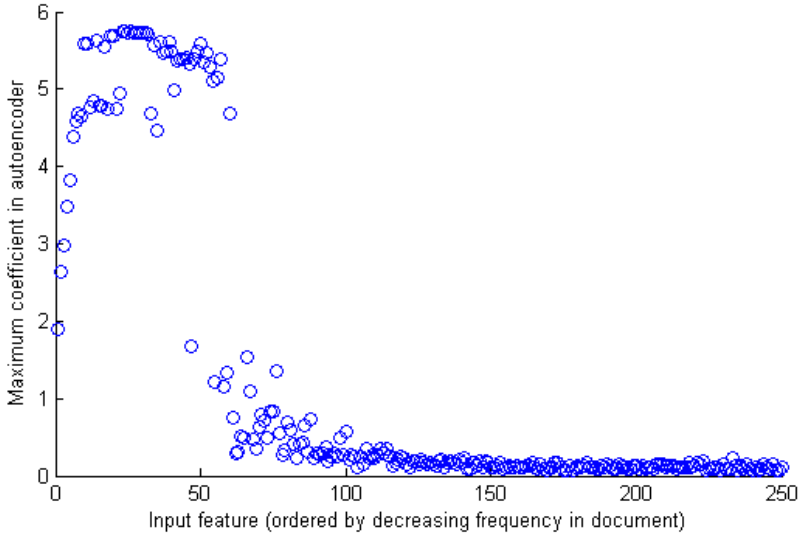

Figure 3: Maximum weight learned for each input node across all hidden nodes

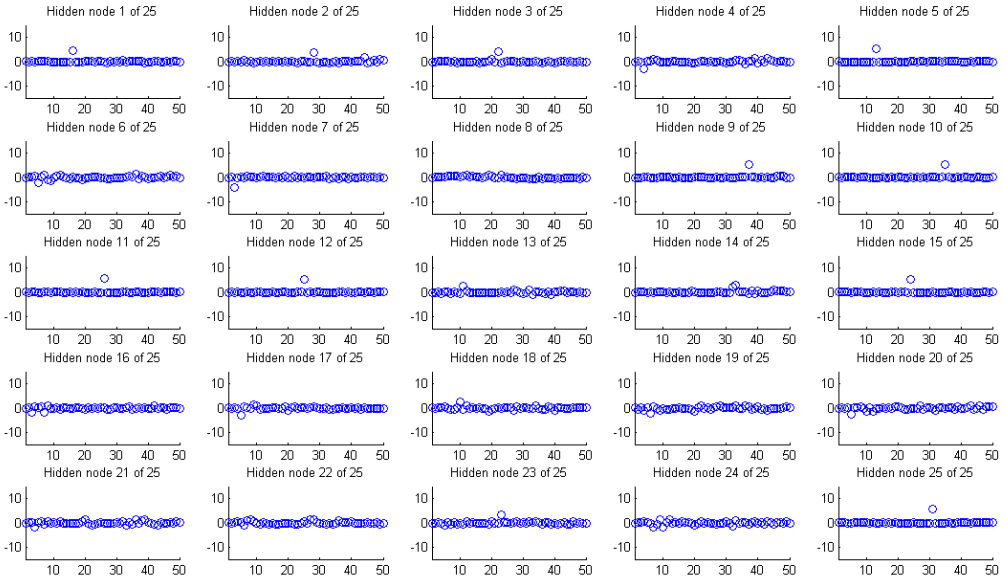

Figure 4: Weights for the hidden nodes corresponding to the first 50 input nodes.

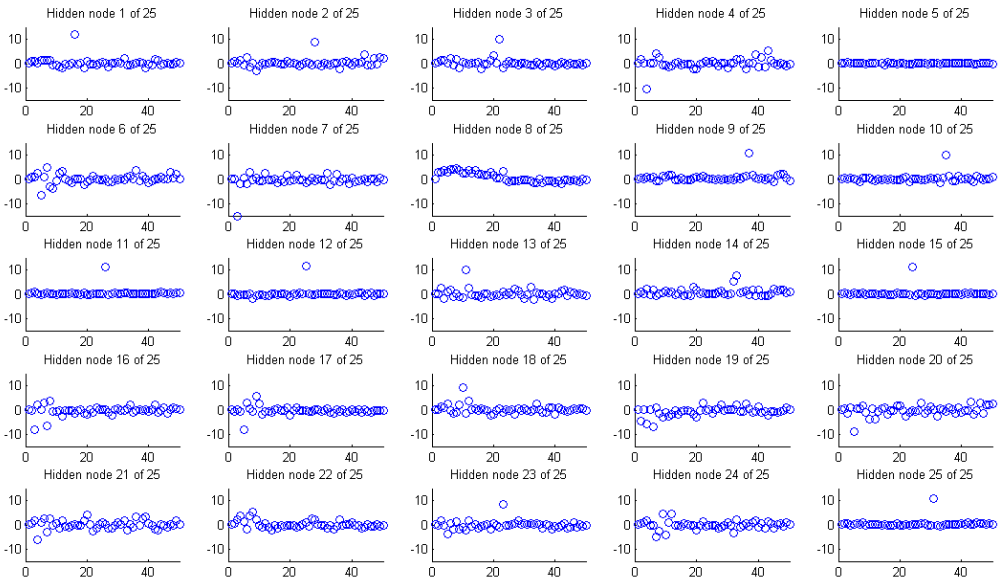

Figure 5: Weights for the hidden nodes corresponding to the first 50 output nodes.

low-level features into higher-level features using unsupervised learning. The idea was that this could improve performance of the algorithm as a whole.

Unfortunately, the autoencoder method used for unsupervised learning led to no overall increases in performance. In fact, performance slightly decreased with the introduction of the autoencoder. To explain this, we examine the relationship of our lowest-level features with the results of the autoencoder.

Autoencoders work well when there is strong statistical correlation between input features. In images, for example, the values of each pixel typically have some relationship to the values of neighboring pixels, and autoencoders can be trained to identify edges of various orientations in image patches.

Our input feature vectors, however, were considerably different. Our input feature vectors were simply vectors of length 250 with each feature corresponding to the number of occurrences of a particular word in a sentence. As a result, most values of the input vectors were zero, and the values that were not zero had minimal correlation with one other.

Because of this, we found that the autoencoder did not combine these features to form new features. Instead, the autoencoder filtered the features, essentially throwing away the majority of information regarding less common words and representing only the frequencies of words that were most common in the document. Because of the underlying power-law distribution of these words, one can show that this method of dimension reduction indeed preserves the majority of information. Even so, this is not the purpose that was envisioned for the autoencoder. Rather than creating higher-level features, the autoencoder simply removes many of the features and even introduces noise to the features it does preserve.

As a result, it is now clear that the autoencoder would not increase the performance of our algorithm. In fact, we could achieve the same result by using less features. While [Collobert and Weston, 2008] has previously shown improvements in natural language processing using deep learning, it seems that our lowest-level features are not compatible with this methodology.

In [Collobert and Weston, 2008], the authors embed sentences into mathematical features in a much more sophisticated way. By training an unsupervised learning algorithm on the entire Wikipedia site for more than a week, they were able to embed words into vectors in a way that simulated relationships between words. They were also able to embed sentences into vectors such that information about the position of words was preserved. In this manner, the input features in this paper were very low-level yet also informative, making them very suitable for deep learning.

In our case, our lowest level features were already significantly pre-engineered. By only examining word frequency, our features lost considerable information about word meaning and position within sentences. As a result, our features were not very low-level and not very informative, making them undesirable for deep learning.

With considerably time, a method for embedding sentences inspired by [Collobert and Weston, 2008] may allow for better results.

7 Individual Effort

Ben King was responsible for processing both the unlabeled data (from Wikipedia) and the labeled data (DUC newswire), tabulating the most frequent words in each document, calculating the ultility scores for DUC sentences, and formatting the data in an easy-to-process way. He also created, trained, tested, and built summaries with the top-level neural network.

Vaishnavi Sundararajan was responsible for looking up past research on deep learning (including the Stanford UFLDL tutorial in [Ng et al., 2010]) and helped with the framework to integrate deep learning with summarization. She also implemented the autoencoder in the UFLDL tutorial and wrote some parts of the report.

Tyler Johnson implemented and verified the final backpropagation algorithm for the autoencoder. He also performed the unsupervised training on Wikipedia articles, transferring the results to labeled DUC sentences to prepare them for the top-level neural net. He assisted with general group tasks, such as designing the algorithm framework and assembling reports, as well.

Rahul was responsible for making the project idea concrete and did the initial literature survey for both summarization and deep learning. He obtained the training and evaluation data and assisted Ben in cleaning it up it to train the neural network. He also designed and wrote the scripts for evaluations.

Acknowledgment

We would like to thank Prof. Dragomir Radev and Prof. Honglak Lee for their input on summarization and deep learning respectively.

References

[Aone et al., 1999] Aone, C., Okurowski, M. E., Gorlinsky, J., and Larsen, B. (1999). A Trainable Summarizer with Knowledge Acquired from Robust NLP Techniques. In Mani, I. and Maybury, M. T., editors, *Advances in Automated Text Summarization*. The MIT Press.

[Baxendale, 1958] Baxendale, P. B. (1958). Machine-made index for technical literature: an experiment. *IBM J. Res. Dev.*, 2:354–361.

[Collobert and Weston, 2008] Collobert, R. and Weston, J. (2008). A unified architecture for natural language processing: Deep neural networks with multitask learning.

[Collobert et al., 2011] Collobert, R., Weston, J., Bottou, L., Karlen, M., Kavukcuoglu, K., and Kuksa, P. P. (2011). Natural language processing (almost) from scratch. *CoRR*, abs/1103.0398. informal publication.

[Conroy and O'leary, 2001] Conroy, J. M. and O'leary, D. P. (2001). Text summarization via hidden markov models. In *Proceedings of the 24th annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '01, pages 406–407, New York, NY, USA. ACM.

[Das and Martins, 2007] Das, D. and Martins, A. F. T. (2007). A survey on automatic text summarization.

[Edmundson, 1969] Edmundson, H. P. (1969). New methods in automatic extracting. *J. ACM*, 16:264–285.

[Kupiec et al., 1995] Kupiec, J., Pedersen, J., and Chen, F. (1995). A trainable document summarizer. In *Proceedings of the 18th annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '95, pages 68–73, New York, NY, USA. ACM.

[Lin, 1999] Lin, C.-Y. (1999). Training a selection function for extraction. In *Proceedings of the eighth international conference on Information and knowledge management*, CIKM '99, pages 55–62, New York, NY, USA. ACM.

[Luhn, 1958] Luhn, H. P. (1958). The automatic creation of literature abstracts. *IBM J. Res. Dev.*, 2:159–165.

[Ng et al., 2010] Ng, A., Ngiam, J., Foo, C. Y., Mai, Y., and Suen, C. (2010). Ufldl tutorial. *http://ufldl.stanford.edu/wiki/index.php/UFLDL_Tutorial*.

[Osborne, 2002] Osborne, M. (2002). Using maximum entropy for sentence extraction. In *Proceedings of the ACL-02 Workshop on Automatic Summarization - Volume 4*, AS '02, pages 1–8, Stroudsburg, PA, USA. Association for Computational Linguistics.

[Radev et al., 2002] Radev, D. R., Hovy, E., and McKeown, K. (2002). Introduction to the special issue on summarization. *Comput. Linguist.*, 28:399–408.

[Svore, 2007] Svore, K. M. (2007). Enhancing single-document summarization by combining ranknet and third-party sources.