CS224d Project Final Report

Elaina Chai
Department of Electrical Engineering
Stanford University
Stanford, CA 94305
echai@stanford.edu

Neil Gallagher
Department of Electrical Engineering
Stanford University
Stanford, CA 94305
neil.gallagher@stanford.edu

Abstract

We develop a Recurrent Neural Network (RNN) Language Model to extract sentences from Yelp Review Data for the purpose of automatic summarization. We compare these extracted sentences against user-generated tips in the Yelp Academic Dataset using ROUGE and BLEU metrics for summarization evaluation. The performance of a uni-directional RNN is compared against word-vector averaging.

1 Introduction

Many popular websites and apps, such as Yelp, Amazon, and Lulu, allow consumers to leave reviews for various goods and services. The point of reviews is to inform the reader about the subject of the review, most likely to make some decision about that subject. Oftentimes it is the case that there are hundreds or thousands of reviews available for a single entity, making it too burdensome for one person to read all reviews for that entity. This presents a problem for the reader of those reviews, who will likely want to gain as much relevant information as possible before making some decision about the subject of the reviews. In such cases, it is likely that much of the information contained in the reviews is repeated many times or is not useful or relevant to the reader. One solution to the reader's problem is to condense the information contained in all of the user generated reviews into a single summary.

Automatic summarization is currently an active field of research in natural language processing (NLP). The goal of summarization is to produce a document based on one or more original documents that is significantly shorter in length than the original, while still giving a comprehensive representation of the information contained in the original. There are two general methods for summarizing a document. In the first, the summary is made up of words, phrases, and/or sentences that are extracted from the original document. The other option is to generate novel text that summarizes the information in the original document using some model of the language in which the summery is to be written.

In this paper, we attempt the task of extracting sentences from Yelp reviews for the purpose of automatic summarization. This task can be broken down into three parts: generating a vector representation for each sentence we encounter, using those vectors to choose sentences to extract, and evaluating the performance of our extraction method. Each of these steps is described in detail below.

2 Background

2.1 Yelp Dataset

We are using the data set provided by the "Yelp Dataset Challenge":
`http://www.yelp.com/dataset_challenge` This data set is provided for free by Yelp

for their DataSet Challenge. It contains 1.6M reviews, over 500K tips from 366K users for 61K businesses.

2.2 Word Vector Representations

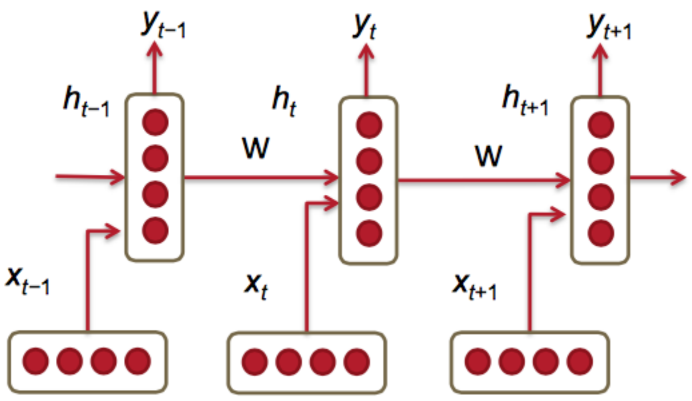
A common tool in natural language processing is the representation of words as vectors in a high dimensional space. Various ways of generating these word vectors to capture semantic and syntactic relationships between words have been proposed. For many purposes, the most effective of these representation techniques is called GloVe, or *Global Vectors for Word Representation* [8]. If the word vectors are being used as an input to a deep learning system, it may be desirable to update the word vectors as a parameter in the network via backpropagation. In this case, one may initialize the vectors representations using an algorithm such as GloVe or generate a set of word vectors by randomly initializing the vector representations.

2.3 Recurrent Neural Network Language Model

For our task, we need to develop a language model that computes the sentence vector from a sequence of words. For this task, we chose to use a basic uni-directional Recurrent Neural Network [4] [5].

Because we are predicting a probability of a sequence of words, a recurrent neural network, is a natural choice for a relatively simple model for predicting word sequences. The model takes as input a word vector, computes a single hidden layer, and does final classification using softmax regression.

Figure 1: A Uni-directional RNN with 3 Time Steps Shown [5]



2.4 Metric

For evaluating summarization results, we will be using two metrics: ROUGE-1 and BLEU-1 (BLEU with unigrams). For both metrics, our reference data will be the actual tips for the business. The test data will be the sentences extracted from the review data for the same business.

2.4.1 ROUGE-1, or ROUGE-N with Unigrams

ROUGE-N, Recall-Oriented Understudy for Gisting Evaluation [1], is a recall based metric for measuring the performance of summarization NLP tasks. Given a test summary and a reference summary, ROUGE-N counts the number of matching N-grams, and divides by the total number of N-grams in the reference summary:

$$\text{ROUGE-N} = \frac{\sum\limits_{S \in \text{ReferenceSummaries}} \sum\limits_{\text{gram}_n \in S} \text{Count}_{\text{match}}(\text{gram}_n)}{\sum\limits_{S \in \text{ReferenceSummaries}} \sum\limits_{\text{gram}_n \in S} \text{Count}(\text{gram}_n)} \tag{1}$$

In our case we will be using unigrams, i.e. $N = 1$. In the ROUGE-N metric, unigrams were found to correlate well with human generated summaries[2].

In the case of multiple references, the resulting score would be the pairwise maximum of the extracted tips and all tips for the business.

2.4.2 BLEU with Unigrams

BLEU, or Bilingual Evaluation Understudy [7], is a precision based metric used primarily for machine translation NLP tasks. The metric is as follows; for a candidate or test summary, and a reference summary, let us define the following variables:

- $m_w(i)$ = number of occurrences of *ith* n-gram in the test summary
- $m_{ref}(j)$ = number of occurrences of *jth* n-gram in the reference summary
- $m_{max}(i) = min(m_w(i), m_{ref}(i))$. If the *ith* n-gram does not appear in the reference summary, this value is 0.
- w_t = total of all n-grams in test summary

Then the BLEU metric is given by the following formula:

$$\text{BLEU} = \frac{\sum\limits_{i \in \text{n-grams}} m_{max}(i)}{w_t} \tag{2}$$

In [7] it was noted that unigrams satisfied *adequacy* and had the advantage that they were simpler. Longer n-grams were better for measuring *fluency*. Since we are only evaluating a summarization task that uses extraction, i.e. we are no interested in measuring fluency, we chose to use unigrams in the evaluation.

For the case of multiple references, as noted [7], $m_{ref}(j)$ is set to the maximum value across all reference summaries for the *jth* unigram.

3 Approach

Our approach to this task can be broken down into three components: sentence vector generation, sentence extraction, and evaluation. Sentence vectors were generated to represent sentences from business reviews in a 50-dimensional space. Those vectors were then given to an extraction heuristic, which was used to choose three sentences to extract as 'tips' for each business. The three tips were then passed on to a couple of evaluation metrics which compare the extracted sentences to user provided tips for that business from the Yelp dataset.

3.1 Sentence Vector Generation

The goal of sentence vector generation is to produce a vector which accurately represents the information within a sentence in the same way that we see good word vector representations do for individual words [8].

3.1.1 RNN Language Model

One of the simplest deep learning models that can be used to generate a vector representation for a sentence is a simple, uni-directional, recurrent neural network (RNN). We used and altered version of the RNN model provided in assignment 2 [11]. The equations required to use the RNN as a language model are similar to those given in the assignment. For $t = 1, ..., n-1$:

$$h^{(t)} = \text{relu}(Hh^{(t-1)} + Lx^{(t)}) \tag{3}$$

$$\hat{y}^{(t)} = \text{softmax}(Uh^{(t)}) \tag{4}$$

$$\hat{y}^{(t)} = \text{softmax}(Uh^{(t)}) \tag{5}$$

where $h^{(0)}$ is initialized as a vector of zeros of the same length as each word vector, and $Lx^{(t)}$ is the product of L, which contains each word vector as a separate column, with the one-hot vector $x^{(t)}$. The model is trained by minimizing the average cross-entropy loss for all words in a sequence.

The final hidden vector $h^{(n)}$ is used as a vector representation of the sentence for the summarization step.

3.1.2 Word Vector Averaging

One of the simplest non-deep learning models that can be used to generate a vector representation for a sentence is word vector averaging. Sentence vectors are generated this way by taking the average of the vectors representing each word contained in the sentence. For a sentence containing n words, the sentence vector s is generated by:

$$s = \frac{\sum_{t=1}^{n} Lx^{(t)}}{n} \tag{6}$$

where L and $x^{(t)}$ are defined as before.

3.2 Summarization

We chose to use an extractive technique, rather than a generative one, to perform summarization of Yelp business reviews. Unlike generative techniques, extraction does not require a sophisticated language model to generate new sentences during summarization, and is therefore more straightforward to implement.

In addition, using an extractive technique allows us to more easily account for non-standard expressions, such as slang (LOL), words with emphasis (all caps are WAAYYYYY overused), and emoticons :) . It is debatable as to whether such expressions should be included in a summary, as they seem to communicate more information about the reviewer and their sentiment towards the business being reviewed, rather than objective information that would be valuable in a summary. Nonetheless, allowing for non-standard expression proved to be valuable in this case, because we chose to use user provided tips from the Yelp mobile app as the human-generated summaries against which to compare our results. By inspection, both the reviews set and user provided tips contained non-standard expressions of each type mentioned above.

One downside of choosing extractive techniques is that sentences found in reviews are often more specific than is desired for a summary. For instance, one review contained the sentence ”Once again I am working in the downtown area and there are quite a few Mexican restaurants near me but I will always drive down to South Phoenix to grab food from here.”, which contains extraneous information about the reviewer and could ideally be condensed to something like ”This is better than many other Mexican restaurants in Phoenix”.

3.2.1 Extraction by Density Estimate

The objective we chose to maximize when choosing which sentences to extract was a simple, custom heuristic for the frequency that the information contained in a sentence was repeated in among all reviews for a business. Extracting sentences containing information that is commonly repeated among all reviews seemed like a good choice for two reasons. Firstly, a good summarization of several reviews should contain the information that is common between all of the reviews. Second, it seems likely that such information is also likely to be contained in user generated tips used in our evaluation metric, thus increasing the relevance of that metric.

The heuristic used chooses which sentences to extract by minimizing a score criterion that is approximately inversely proportional to the density of sentence vectors around a the given sentence vector. When choosing multiple sentences per business, a penalty term is added to score to discourage selection of sentences that are near in distance to each other. The penalty term is equal to the squared sum of cosine similarities between the current sentence vector and all those that have been previously extracted, times the maximum score for all sentences already extracted. Before the penalty is added, the score is calculated by summing the euclidean distance from the sentence vector to the k nearest neighbors:

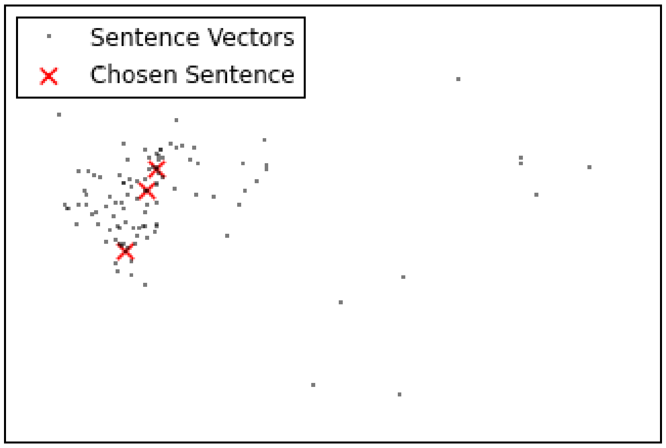
$$\text{score} = \sum_{i=1}^{k} dist(s, s_i) + \text{penalty} \tag{7}$$

where s_i indicates the i^{th} nearest neighbor to sentence vector s.

$$\text{penalty}(s) = max(scores) \sum_{s_e \in \text{Extracted Sentences}} \left(\frac{s \cdot s_e}{\|s\|\|s_e\|}\right)^2 \tag{8}$$

$$k = n_{reviews}/\alpha + 1 \tag{9}$$

Figure 2: Example of Extracted Sentence Vectors in 2-Dimensional Space



3.3 Evaluation Metric using ROUGE and BLEU

The models were trained using Stochastic Gradient Descent, but the loss function tries to maximize the probability of a sequence of words appearing in a test sentence. For sentence extraction, we needed a way to measure the similarity between our extracted tips and the set of real user-generated tips for a given business.

For this reason, we chose to use the ROUGE and BLEU metrics.

ROUGE is a recall-based metric. In a binary context, a recall-based metric is a sensitivity score, i.e. it tries to create a score by looking at the ratio between identified true positives and all data that should been labeled as positive [10]. In this context, ROUGE tries to score the extracted tip by looking at how many unique n-grams in a extracted tip are also present in the set of reference tips.

For BLEU, on the other hand, is a precision based metric. In a binary context, a precision based metric a confidence score, i.e. it creates a score by looking at the ratio between true positives and results of the algorithm (i.e. the sum of true and false positives) [10]. In this context, BLEU tries to score the extracted tip by looking at how many n-grams in the extracted tip also appear in the reference, and divide this by the total number of n-grams in the extracted tip.

4 Experiment

4.1 Training

4.1.1 Data

The Yelp data is split into review data and tips data. We train the language model on the tip data, but the extracted tip is taken from the review data.

The data is provided in a set of JSON files. We do pre-processing in the form of tokenization of the reviews and tips. For this task, we use tools in the Stanford CoreNLP [3], provided by the Stanford NLP group and a python wrapper [6].

Our initial vocabulary list was taken from the Wikipedia Crawl Corpus on the GloVe website [9]. Using the top 5000 words, we created a training, test and development set using all tips of which at least 70% of their tokens were present in the vocabulary list. We did this as a crude way of filtering the data for reviews and tips that contained too much slang, misspelt words or were in a foreign language.

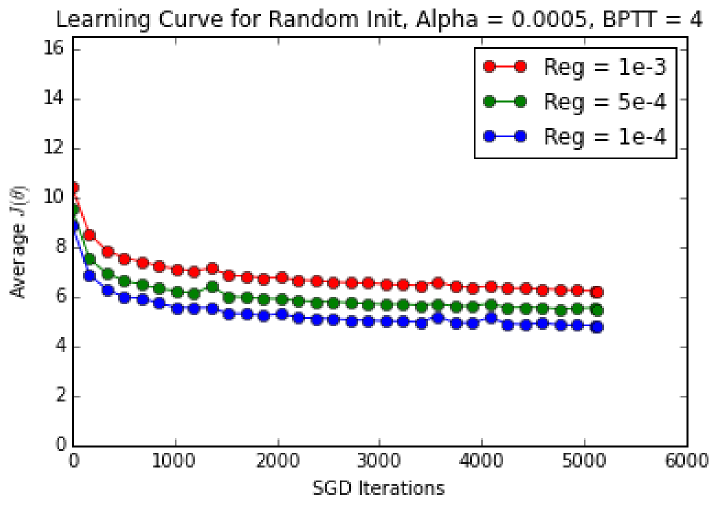
4.1.2 Models

We use 50-dimensional word-vectors. To initialize, we initialize using one of two ways:

- Random Initialization
- Initialization using pre-trained GloVe vectors [9]

We then use Stochastic Gradient Descent to further train the language model using each initialization in our RNN. We then train our models varying the SGD step size, back-prop time steps, and regularization size.

The following is an example of one of the training curves from this training:



4.2 Results

To extract the test tips for evaluation, we use the method described in Section 3.2.1.

The *reference* tips are the real tips for the a given business. Using the formula described in the Section 2.4, we calculate a ROUGE and BLEU Score for each extracted tip and choose the best performing tip for ROUGE and BLEU respectively. We then average this score across the set of businesses. In this case, our set of businesses is the 10 businesses with highest number of tips (for the largest possible reference set per business).

Using the models described in the 'Training' Section, we applied the ROUGE and BLEU metrics. We found that for ROUGE-1, Word-Vector Averaging performed best. For the BLEU metric, the RNN model performed the best:

Table 1: Results using ROUGE and BLEU Metrics

	ROUGE-1		BLEU-1	
	WV-Averaging	RNN	WV-Averaging	RNN
GloVe	0.511	0.446	0.668	0.672
Random	0.562	0.477	0.638	0.656

A sample of the extracted tip and its closest neighbors are given below for the RNN model:

Extracted Tip: ”Our drinks arrived quick and our server Melissa had honest and accurate suggestions.”

- Neighbor 1: ”The Bartender was visually a bit flustered as there were three unsavory looking and sounding women seemingly trying his patience with their drinks and menu ordering.”
- Neighbor 2: ”My husband ordered the 7 oz sirloin with riblets , seasonal vegatables and mashed potatoes.
- Neighbor 3: ”try parmesan shrimp or their pasta bowls.”

A sample of the extracted tip and its closest neighbors are given below for the equivalent word vector averaging model:

Extracted Tip: ”We do not go places looking for an all smiles server , but this guy did not smile and when he saw we were not enjoying our meal and my husband's plate was still full , he did not even ask if the food was okay or if there was something he could do.”

- Neighbor 1: ”We don't like to complain and just were not in the mood to even let him or the manager know.”
- Neighbor 2: ”We had a large group so I can understand it taking a little longer than normal but our server took forever to ask us what we wanted to drink (we were a little later than the rest of the party who already had a drink).”
- Neighbor 3: ”usually i go there just by myself but sever always pays a lot of attention to me.”

5 Conclusion

By inspection, we saw both methods we used to generate sentence vectors did not do a good job of representing the information contained within those sentences. We know this because the nearest neighbors to chosen sentences did not seem to have any more information in common with them than other randomly chosen sentences. This indicates that neither word averaging, nor unidirectional RNN models were a good choice for the summarization task that we were trying to solve. A better, albeit more difficult to implement, choice for this project might have been a tree LSTM network model [12].

One interesting thing that we do see is that different model types showed a pattern of performing better or worse on one of our two evaluation metrics. Specifically, pre-training our word vector representations using GloVe and using a unidirectional RNN language model to generate sentence vectors performed better on BLEU, which is a precision-based metric. Randomly initializing word vectors and word vector averaging tended to perform better on ROUGE, which is recall-based. We speculate that the reason for this is that word vector averaging and random initialization increases the expected length of the extracted sentences, which would increase performance on a recall-based metric and decrease it on a precision-based metric.

References

[1] Chin-Yew Lin. ROUGE: A Package for Automatic Evaluation of Summaries.

[2] Chin-Yew Lin and Eduard Hovy. Automatic evaluation of summaries using N-gram co-occurrence statistics. *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology - NAACL '03*, 2003(June):71–78, 2003.

[3] Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J. Bethard, and David McClosky. The Stanford CoreNLP natural language processing toolkit. In *Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 55–60, 2014.

[4] Tomáš Mikolov, Martin Karafiát, Lukáš Burget, and Sanjeev Khudanpur. Recurrent neural network based language model.

[5] Socher Mohammadi, Mundra. Recurrent neural networks – for language modeling and other tasks. `http://cs224d.stanford.edu/lecture_notes/LectureNotes4.pdf`. Accessed 2015-06-07.

[6] Brendan O'Conner. Stanford corenlp pywrapper. `https://github.com/brendano/stanford_corenlp_pywrapper`, 2015. Accessed 2015-05-11.

[7] Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. BLEU: a Method for Automatic Evaluation of Machine Translation.

[8] Jeffrey Pennington, Richard Socher, and C Manning. Glove: Global vectors for word representation. *Emnlp2014.Org*.

[9] Manning Pennington, Socher. Glove: Global vectors for word representation. `http://nlp.stanford.edu/projects/glove`. Accessed 2015-05-11.

[10] David M W Powers. Evaluation : From Precision , Recall and F-Factor to ROC , Informedness , Markedness & Correlation. *Journal of Machine Learning Technologies*, 2(December):37–63, 2007.

[11] Peng Qi, Francois Chaubard, Ian Tenney, Namrata Anand, Rohit Mundra, and Milad Mohammadi. CS 224d : Assignment # 2, 2015.

[12] Kai Sheng Tai, Richard Socher, and Christopher D. Manning. Improved Semantic Representations From Tree-Structured Long Short-Term Memory Networks. page 10, February 2015.