Softmax RNN for Short Text Classification

Ellery Wulczyn Clementine Jacoby

Department of Computer Science Stanford University ewulczyn@cs.stanford.edu mjacoby@stanford.edu

Abstract

We implement the Softmax Recursive Neural Network (SRNN), a deep learning algorithm for text classification. The model jointly learns vector representations of words, a method of merging word representations into document vector representations and a classifier over the document vector representations. Merging is done over a binary tree structure of the document. For a specific task, we show that the SRNN can learn low-dimensional, continuous representations of documents that are superior to the traditional bag-of-words representation for the purpose of classification. Furthermore, it produces high quality domain specific word vectors in the process.

1 Introduction

A binary RNN is a deep learning architecture, that repeatedly applies the same neural network at each node of a binary tree (See Figure 3). Due to its recursive nature, it elegantly handles inputs of variable length. This property in conjunction with the advent of continuous vector representations of words (one method of generating these word vectors s described in [1]) makes RNNs particularly attractive for learning structure from natural language. To date, RNNs have been applied with great success in parsing, paraphrase-detection and sentiment analysis [3–7]. In this paper, we investigate the use of RNN's for text classification.

y1 = tanh(W[x1;y2] + b)

y2 = tanh(W[x2;y1] + b)

y1 = tanh(W[x2;y1] + b)

x1 x2 x3 x4

Figure 1: Binary RNN Schematic

2 Model

We implement the Softmax Neural Network (SRNN), a deep learning algorithm for classification of short text documents. The model jointly learns continuous vector representations of words, a method of generating continuous vector representations of documents and a classifier over the documents.

Document vectors are generated by merging word vectors according to a binary tree structure of the document. A simple choice for the document tree structure, is a balanced binary tree, where the set of leaves corresponds to the words in the document.

The SRNN consist of three different types of computational units. The Look-Up unit operates on the leaves of the document tree. It maps from words to word strings to word vectors. The parameters associated with the Loop-Up unit are the word embedding matrix W, where each column corresponds to a word in the vocabulary. To be precise, the Look-Up unit applied to word w computes the element-wise tanh of the word embedding vector corresponding to w.

$$LU(w) = tanh(L_w)$$

The Merge unit operates on the children of nodes in the document tree. It computes a parent phrase representation by merging the left and right child phrase vector representations into one.

$$Merge(p_l, p_r) = tanh(Lp_l + Rp_r + b)$$

Finally the Softmax unit operates on the root. It takes in the full document vector and outputs probabilities over K document classes.

$$SM(d) = h$$
 where $h_i = exp(d^T\theta_i) / \sum_{j=1}^{K} exp(d^T\theta_j)$

Given a document tree T we can compute the associated document vector d recursively as follows:

RNN(T):

```
\label{eq:local_total_continuous} \begin{split} & \textbf{if T.} \text{isleaf then} \\ & & LU(T.word) \\ & \textbf{else} \\ & & Merge(RNN(T.left), RNN(T.right)) \\ & \textbf{end if} \end{split}
```

Where T.left is the left child of T, T.right is the right child of T and T.word is the word associated with T when T is a leaf node. To get the class probabilities, we compute SRNN(T) = SM(RNN(T))

3 Learning

Given a training set $D = \{(T^{(i)}, y^{(i)})\}_{i=1}^m$, we are interested in maximizing classification accuracy. As a proxy, we use the log-liklihood of the data given by:

$$l(\theta, W, L, R, b; D) = \sum_{i=1}^{m} 1_{y(i)}^{T} SRNN(T^{(i)})$$

where 1_y is the indicator vector, with all zeroes except for a 1 at position y. The following recursive back-propagation algorithm computes the gradients of l with respect to the parameters and a single training example (T, y).

```
\begin{aligned} &\textbf{BackProp}(\textbf{T},\textbf{y},\delta)\textbf{:}\\ &\textbf{if T.isLeaf then}\\ &L_{(T.word)}'+=\delta\\ &\textbf{else if T.isRoot then}\\ &\theta'+=RNN(T)(1_y-SRNN(T))^T\\ &\delta:=\theta_y-\theta SRNN(T)\\ &\delta:=\delta\circ(1-RNN(T)\circ RNN(T))\\ &BackProp(T.left,y,\delta)\\ &BackProp(T.right,y,\delta)\\ &\textbf{else}\\ &b'+=\delta\\ &L'+=\delta RNN(T.left)^T \end{aligned}
```

```
\begin{array}{l} R^{'}+=\delta RNN(T.right)^{T}\\ delta_{l}:=(L\delta)\circ(1-RNN(T.left)\circ RNN(T.left))\\ delta_{r}:=(R\delta)\circ(1-RNN(T.right)\circ RNN(T.right))\\ BackProp(T.left,y,\delta_{l})\\ BackProp(T.right,y,\delta_{r})\\ \textbf{end if} \end{array}
```

We can compute the gradients of l with respect to the parameters and the full training set as follows:

```
ComputeGradients(D): b^{'}, L^{'}, R^{'}, L^{'}, W^{'}, \theta^{'} = 0 for (T, y) in D do BackProp(T, y, null) end for
```

4 Implementation Notes

In our implementation, we choose not to apply tanh in the Look-Up Unit and add a bias term to the Softmax Unit. We also add an L_2 penalty to the log-likilihood for regularization. We left this out of the pseudo-code above for sake of clarity. To optimize the weights given the gradients, we use the LBFGS optimizer from the Stanford's Core-NLP library. Large gains in efficiency are gained by caching the activations of the SRNN at every node for a given set of weights and reusing them in the gradient computations. To prevent the softmax unit from suffering from overflow, we subtract the largest $d^T\theta_i$ from all the $d^T\theta_i$ before evaluating the softmax function. We initialize parameters W, L, R, b from [3] and initialize θ randomly. As a guide for checking the correctness of our derivations for the above back-propagation algorithm, we performed a gradient-check by computing the numerical gradient.

5 Previous Work

5.1 Document Representation

A common document representation for classification is the "bag-of-words" model, where each document has a feature for each word in the vocabulary that indicates the presence, count or tf-idf score of the word in the document. An issue with this representation is that documents with semantically or syntactically similar words do not have similar representations. As a simple example, consider three (very short) documents. The first consists of the words "ship" and "have". The second consists of "boat" and "has". And the third consists of "ball" and "score". Although "boat" and "ship" are synonyms and "have" and "has" are different cases of the same verb, the first document is no more similar to the second document than it is to the third document under the bag of words model as measured by say cosine similarity of the document vectors.

The last five years have seen an upsurge in research on embedding words in a vector space such that semantically and syntactically similar words are "close", as measured by say the cosine similarity. We would like to be able to embed entire documents in a vector space such the first document is closer to the second document than the third document

Socher developed an unsupervised RNN for merging word vectors into a sentence vector according to the parse structure of the sentence in his work on paraphrase detection [3]. Words where merged into a sentence vector with the goal of being able to reconstruct the individual words that went into the sentence vector. The SRNN merges words in a document into a sentence vector, with the goal of being able to determine which class the document belongs to.

5.2 Algorithms

Linear classifiers such as SVM or Softmax using the "bag-of-words" document representation often do very well. Non-Recursive neural networks using the bag-of-words representation have been attempted, but in practice don't do as well as the simpler linear models [2]. As far as we know, there is no literature on using RNN's for text classification.

6 Experiments

6.1 Data

Our data comes from the Hewlet Foundation's competition on automated essay scoring. We took essays written in response to 4 different prompts and learn a classifier over sentences, where the goal is to determine which prompt generated the sentence. (The choice of the data is a bit of a historical accident, since we began the project with the goal of essay scoring, but found the SRNN more exciting). We chose not to classify entire essays, since preliminary analysis showed this to be a trivial task. Training and testing was done over sets of sentences taken from disjoint sets of essays and with even class distributions. The training set contained 50k sentences, the test set contained 10k sentences. What follows are the four prompts that define the classes for our task. We include the prompts in their entirety since, they are important for a later section on interpreting what the SRNN learns.

Prompt 1: Effects of Computers

More and more people use computers, but not everyone agrees that this benefits society. Those who support advances in technology believe that computers have a positive effect on people. They teach hand-eye coordination, give people the ability to learn about faraway places and people, and even allow people to talk online with other people. Others have different ideas. Some experts are concerned that people are spending too much time on their computers and less time exercising, enjoying nature, and interacting with family and friends. Write a letter to your local newspaper in which you state your opinion on the effects computers have on people. Persuade the readers to agree with you.

Prompt 2: Censorship in the Libraries

Write a persuasive essay to a newspaper reflecting your views on censorship in libraries. Do you believe that certain materials, such as books, music, movies, magazines, etc., should be removed from the shelves if they are found offensive? Support your position with convincing arguments from your own experience, observations, and/or reading.

Prompt 3: Patience

Write about patience. Being patient means that you are understanding and tolerant. A patient person experience difficulties without complaining. Do only one of the following: write a story about a time when you were patient OR write a story about a time when someone you know was patient OR write a story in your own way about patience.

Prompt 4: Benefits of Laughter

We all understand the benefits of laughter. For example, someone once said, Laughter is the shortest distance between two people. Many other people believe that laughter is an important part of any relationship. Tell a true story in which laughter was one element or part.

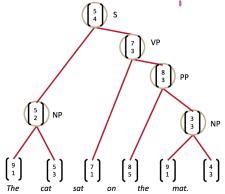
6.2 Document Tree Structures

Since each of our documents is actually a sentence, we had the ability to use the binarized parse tree of each sentence as the document tree. The hope is that using a hierarchical structure to perform merging based on grammar is advantageous. A potential issue is that if the tree is left branching, words that occur later in the sentence, like ".", have exponentially more influence than earlier words, like the subject of the sentence.

The competing option is to use a balanced binary parse tree, which has the property that all words contribute roughly equally to the sentence vector. It is also easy to throw out words that are not verbs, nouns or adjectives, which may not be relvant for the classification task. Finally, you can permute the leaves of a balanced binary tree to generate more training data, if you have issues with overfitting, although we did not do this.

Surprisingly, we found that the choice of document tree representation has no bearing on classification accuracy. In other words, the natural tree structure over the sentence, the parse tree is not better than a balanced tree. Word order is totally unimportant and you can leave in words that

Figure 2: Computing a Sentence Vector According to Binary Parse Tree



occur with equal probability in all classes (like "the" or "."). You could call the model robust or note that it fails to make use of much of the structure in human language. In the end, the balanced binary tree, with permuted leaves, is is not unlike a bag of words. The only difference is in how the words in the "bag" are put together to form the document. This robustness, makes the SRNN a suitable candidate for "out-of-the-box" text classifiers.

6.3 Classification Results

Figure 4 shows the learning curves for the SRNN using balanced trees as a document structure. We also include two benchmarks. For the first benchmark bag, we ran a standard softmax classifier with a bag-of-words document representation, using the RTextTools package. Here the term-document matrix contains the count of word j in document i position (i,j). We pre-process the data by stripping whitespace, removing punctuation, converting to lowercase, stemming, and removing exceedingly rare words. For the second benchmark paraphrase, we take all parameters from [3] except for θ and only train the Softmax Unit. As a reminder in [3], Socher trains an unsupervised, recursive auto-encoder for sentence vectors.

The fact that the SRNN, rapidly surpasses the paraphrase baseline, proves that modifying the word-embedding and the merge matrices for a specific task is highly useful. After about 50 iterations, test accuracy reaches its minimum at 19.1%. This is 5.08% lower than the bag baseline, showing that the SRNN is able to learn relatively small, continuous document representations that are better for text-classification than the potentially high-dimensional bag-of-words representation. Note that with more tuning or a different algorithm, it might be possible to do better than our baseline or even the SRNN with a bag of words document representation. The goal of this paper is not a do as well as possible on this fake task, but to explore and analyze an algorithm. The baselines serve mainly as a sanity check for whether the model is learning something useful. Finally, we note that the SRNN is very powerful, achieving perfect accuracy on the training set given enough iterations. The learning curves suggest that with heavier regularization, it might be possible to achieve better results. However, we tried varying the regularization weight, but found little no to change in results.

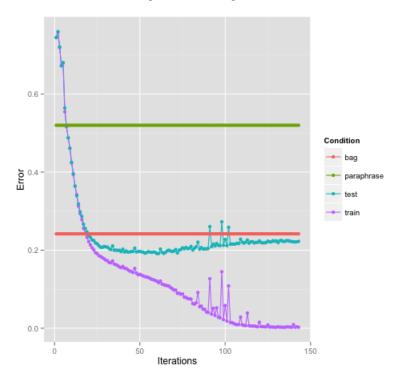
7 Qualitative Analysis

7.1 Word Vectors

To better understand our model, we are interested in what modifications the SRNN makes on the word embedding matrix W. We formulate two hypotheses:

- 1. Words with the most discriminatory power will change the most.
- 2. Words with very different meanings and hence relatively far apart in the word embedding space before training, will move towards each other if they are relevant to the same prompt.

Figure 3: Learning Curves



As an investigation of the first hypothesis, we look at words that changed the most before and after training. We use euclidean distance of the difference between a word vector before and after training to measure change. Table 1, shows the 40 words that the SRNN changed the most. Each word is assigned to the question in which the relative frequency of the word in the set of responses is highest. We note the words that changed the most tend to be words that appeared in the question prompt, which heavily influences what students write in their response and is useful for classification.

Prompt	Words
1	computers, computer, Computers, internet, website, websites, online, Dear, coordina-
	tion, exercise, obesity, technology
2	book, books, Books, magazines, movies, offended, offensive, censorship, Censorship,
	library, libraries, shelves, shelf, drugs, materials, material
3	patients, patient, patience, impatient, wait, waited, hour, ", I
4	laughing, laughter, laughed, laugh, relationship, funny

Table 1: Word Vectors Modified most heavily by the SRNN

The few words that did not appear in the question prompt are "patients" "Dear", "I" and """. "Patients" is a popular misspelling for "patience", which occurs in Prompt 3. Prompt 1 asks students to write a letter, hence it seems natural that the existence of the word "Dear" in a sentence is a strong indicator of prompt 1. Finally, Prompt 3 asks the students to relate a story about themselves or someone they now and so we would expect frequent use of the words "I" and quotation marks for dialog.

To investigate hypothesis 2, we look at nearest neighbors of select words before and after training. Since we initialize word vectors from [3], the "Before" vectors are excaltly the vectors from [3]. Tables 2-5 show the nearest neighbors for a prominent word for each prompt in order of increasing distance.

Before	courage, frustration, resentment, instructions, honesty, hostility, unhappiness
After	patient, patients, wait, waited, impatient, line, waiting, hour, doctor, weeks, lines,
	calm, patiently, hours

Table 2: Nearest Neighbors for "patience"

Before	injuries, hatred, complications, headaches, beatings, candles, cheers, memories, dis-
	may, destruction
After	laugh, laughing, relationship, laughed, funny, laughs, together, smile, closer, humor,
	met, feeling, jokes, hapiness

Table 3: Nearest Neighbors for "laughter"

Before	software, digital, chemical, mobile, entertainement, quality, plastic, furniture, car, ve-
	hicle
After	computers, online, technology, internet, Computers, coordination, exercise, websites,
	nature, chat, family, email, website, sites, screen, outside

Table 4: Nearest Neighbors for "computer"

Before	relief, corruption, troop, propaganda, self, personality, slander, segregation
After	Censorship, magazines, movies, materials, material. media, shelves, offended, drugs

Table 5: Nearest Neighbors for "censorship"

Before training, the nearest neighbors of "patience" are generally words that describe the qualities of a person, but have nothing to do with the meaning of "patience". After training the SRNN, grammatical variants of patience, common mispellings, antonyms, semantically related words such as "waiting" or "time", and words indicative of situations that require patience like "lines" and "weeks" all appear as nearest neighbors.

For "laughter", there appears to be no rhyme or reason to the nearest neighbors, except that they tend to be plural nouns. After training, we again get many grammatical variants such as "laugh", "laughing", "laughs" and "laughed" and highly related words like "humor", "joke" and "funny". In line with our second hypothesis, we also see "together" and "relationship", words in themselves unrelated to laughter, but made relevant by the fourth prompt which suggests laughter is a way building relationships.

Highly related nouns such as "digital" and software" appear as the nearest neighbors for "computer" even before training. After training we also get highly relevant words such as "screen", "online", "internet", "websites". But even more markedly than for "laughter" a priori unrelated words such as "nature" and "family" enter as nearest neighbors. Again, this is because of the question prompt, which suggests computers detract from family and nature.

For "censorship" nearest neighbors move from politically flavored nouns, to items than might be censored such as "books" and "magazines" and reasons for censorship such as "drugs" and "sex".

This analysis suggests that word embeddings learned purely based on context and cooccurrence statistics often fail to place synonyms and grammatical variants close together in the embedding space. For words relevant to the classification task, the SRNN is able to correct for this. Moreover it is a powerful tool for learning highly domain specific word embeddings, by pushing words that are related under a specific context, like a question prompt, together

8 Future Work

It will be exciting to apply the SRNN to classification tasks involving larger documents than just documents with one sentence. Given the success of discarding parse trees and simply using balanced binary trees, we think that this will be similarly successful. Attempting to reach state of the art on

some classic text classification task, will show if the SRNN is a competitive alternative to current text classification technologies.

References

- [1] Ronan Collobert and Jason Weston. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of the 25th international conference on Machine learning*, pages 160–167. ACM, 2008.
- [2] Lawrence McAfee. Document classification using deep belief nets. In http://nlp.stanford.edu/courses/cs224n/2008/reports/10.pdf.
- [3] Richard Socher, Eric H Huang, Jeffrey Pennin, Christopher D Manning, and Andrew Ng. Dynamic pooling and unfolding recursive autoencoders for paraphrase detection. In *Advances in Neural Information Processing Systems*, pages 801–809, 2011.
- [4] Richard Socher, Brody Huval, Christopher D Manning, and Andrew Y Ng. Semantic compositionality through recursive matrix-vector spaces. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 1201–1211. Association for Computational Linguistics, 2012.
- [5] Richard Socher, Cliff C Lin, Andrew Ng, and Chris Manning. Parsing natural scenes and natural language with recursive neural networks. In *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, pages 129–136, 2011.
- [6] Richard Socher, Christopher D Manning, and Andrew Y Ng. Learning continuous phrase representations and syntactic parsing with recursive neural networks. *Proceedings of the NIPS-2010 Deep Learning and Unsupervised Feature Learning Workshop*, 2010.
- [7] Richard Socher, Alex Perelygin, Jean Y Wu, Jason Chuang, Christopher D Manning, Andrew Y Ng, and Christopher Potts Potts. Recursive deep models for semantic compositionality over a sentiment treebank. In EMNLP, 2013.