Pruning and Sparsemax Methods
for Hierarchical Attention Networks

João G. Ribeiro¹,², Frederico S. Felisberto¹, and Isabel C. Neto¹

¹Department of Computer Science, Instituto Superior Técnico, University of Lisbon
²INESC-ID

Abstract

This paper introduces and evaluates two novel Hierarchical Attention Network models [Yang et al., 2016] - i) Hierarchical Pruned Attention Networks, which remove the irrelevant words and sentences from the classification process in order to reduce potential noise in the document classification accuracy and ii) Hierarchical Sparsemax Attention Networks, which replace the Softmax function used in the attention mechanism with the Sparsemax [Martins and Astudillo, 2016], capable of better handling importance distributions where a lot of words or sentences have very low probabilities. Our empirical evaluation on the IMDB Review for sentiment analysis datasets shows both approaches to be able to match the results obtained by the current state-of-the-art (without, however, any significant benefits). All our source code is made available at https://github.com/jmribeiro/dsl-project.

1 Introduction

Recent years have witnessed a significant number of successes in the sub-field of natural language processing, document classification [Yang et al., 2016; Kowsari et al., 2019, 2018; Chen et al., 2018; Jiang et al., 2018].

With the introduction of Hierarchical Attention Networks (HANs) [Yang et al., 2016, the classification of documents while identifying which words and sentences are or not relevant for the task turned out to be a crucial issue in the field. These models don’t only take into account a document’s hierarchical structure (i.e., documents contain sentences and sentences contain words), but also introduce an attention mechanism which assigns an relevance value to the words and sentences, telling how relevant they are for the task at hand. [Yang et al., 2016] evaluate their approach on six different datasets (Yelp 2013, 2014 and 2015, IMDB review, Yahoo Answer and Amazon Review [Yang et al., 2016]) and are able to obtain state-of-the-art document classification accuracies, outperforming all previous works.

However, such models always assume these components (words and sentences) to be at least somewhat relevant for classification (when some components may only be introducing noise into the discrimination process itself). In this paper we therefore follow [Yang et al., 2016] by going one step further and asking a new hypothesis - what if words and sentences, which the attention mechanism assigns a very low value (e.g., stop words), instead of being assigned low values, could be instead completely removed from the classification process?

To tackle this issue, this paper introduces and evaluates two new methods. The first one, named the Hierarchical Pruned Attention Networks aims to ignore words and sentences which may not be very relevant for the classification, and the second one, named the Hierarchical Sparsemax Attention Networks replaces the Softmax function in the attention mechanism in order to better handle a lot of irrelevant words and sentences. We perform an empirical evaluation (N=5, confidence level of 99%),
comparing both models against the original Hierarchical Attention Networks. Our evaluation on the IMDB review dataset for sentiment analysis [Maas et al., 2011] shows possible to match the current state-of-the-art, however without any clear benefits of choosing one method over the others.

2 Outline

This rest of this paper is organized as following:

- Section 3 surveys the relevant literature.
- Section 4 details the hierarchical attention networks architecture, which sets the foundation for our two novel models.
- Section 5 highlights our main contributions and describes the two main architectures introduced.
- Section 6 describes the performed empirical evaluation procedure.
- Section 7 reports and discusses the obtained results.
- Section 8 concludes the paper and lists future lines of work.

3 Related Work

Available literature in the NLP sub-field of document classification can be surveyed in the light of the document classification accuracy [Kowsari et al., 2019].

When the field of natural language processing was benefiting from the usage of linear models combined with tools, such as bag-of-words and n-grams, Zhang et al. [2015] first introduce LSTMs and CNNs for text classification. These models, still having a rather low degree of complexity, are shown by the authors to outperform traditional models in three datasets (Yelp’15, Yahoo Answers and Amazon).

Following their line of work, Tang et al. [2015] later on introduce gated recurrent neural networks for document classification. Their models outperform the current state-of-the-art at the time, in the Yelp’15 dataset and are also evaluated on three additional datasets - the Yelp’13, Yelp’14 and IMDB datasets [Tang et al., 2015].

Finally, following Tang et al. [2015], Yang et al. [2016] introduced Hierarchical Attention Networks (HANs). Unlike most document classifiers, HANs assume a hierarchical structure to a document (i.e., a document is a sequence of sentences, and sentences are sequences of words). Instead of processing a document as a simple sequence of words, HANs exploit this hierarchical structure while weighting the relevance of each word/sentence using an attention mechanism (hence the name).

Yang et al. [2016] then evaluate their model on six different datasets (Yelp 2013, 2014 and 2015), IMDB, Yahoo Answer, and Amazon [Yang et al., 2016]) and are able to obtain state-of-the-art document classification accuracies, outperforming all previous works.

Discussion

Taking all the relevant literature into account, we follow Yang et al. [2016] and ask the following question - what if classification accuracy could be improved by either removing all words/sentences with low attention weights or using a function other than the original Softmax? Recently, Martins and Astudillo [2016] introduced the Sparsemax function which is capable of better handling distributions with a lot of near-zero probabilities. When there are a lot of words/sentences which may not be relevant to the classification task at all (such as stop words, assuming such components to be always somewhat relevant for the classification task may introduce noise into the process. We therefore extend the available literature by introducing and evaluating two novel models which tackle this issue.
Hierarchical Attention Networks (HANs) are models which discriminate a document into its most likely class. An overview of their architecture can be visualized in figure 1.

Figure 1: The Hierarchical Attention Network architecture (from [Yang et al., 2016]).

Given as input a flat, padded document $d$ describing a sequence of word tokens/indices from a given vocabulary, with shape $(S, L)$ (where $S$ is the number of sentences and $L$ the number of words per sentence), the classification process is detailed as following:

1. Each sentence of $L$ words is first passed through a embedding layer, obtaining a word embeddings tensor $w$ with shape $(S, L, E)$, where $E$ represents the dimension of a single embedding vector. As in the original work, we resort to a pre-loaded weight vector with dimension of 200 embeddings, however, unlike the original work, we do not pre-train this vectors ourselves (loading a pre-trained GloVe vector).

2. Word embeddings $w$ are then passed through a bidirectional GRU, computing the hidden word scores $h = [h^>, h^<]$.

3. Hidden word scores $h$ are then passed through a single-layer feedforward network, obtaining hidden word representations $u = tanh(W_u h + b_u)$.

4. Hidden word representations are then passed through an attention mechanism, which using a trainable word level context vector $u_w$, compute the attention weights $\alpha = \text{Softmax}(u^\top u_w)$.

5. Attention weights $\alpha$ and hidden word scores $h$ are multiplied, obtaining the sentence vectors $s = \alpha \odot h$.

6. Sentence vectors $s$ are then passed through a bidirectional GRU, computing the hidden sentence scores $h_s = [h_s^>, h_s^<]$.

7. Hidden sentence scores $h_s$ are then passed through a single-layer feedforward network, obtaining hidden sentence representations $u_s = tanh(W_s h_s + b_s)$.

---

1https://nlp.stanford.edu/projects/glove/
8. Hidden sentence representations are then passed through an attention mechanism, which using a trainable sentence level context vector $u_s$, compute the attention weights $\alpha = \text{Softmax}(u_s^\top u_s)$.

9. Attention weights $\alpha$ and hidden sentence scores $h_s$ are then multiplied, obtaining the document features $v = \alpha \odot h_s$.

10. Document features $v$ are finally passed through an affine transformation, obtaining the class logit scores $z = W_v v + b_v$.

11. Finally, $z$ can be used to compute the probabilities of each class $p = \text{softmax}(z)$. Cross entropy loss is then used to compute the gradients for all parameters.

5 Proposed Models

We introduce two novel model architectures to tackle the issue of unimportant words and sentences in the original attention mechanisms [Yang et al. 2016]. The first one, which we named Hierarchical Pruned Attention Networks (HPAN), completely ignores components with low attention weights, and the second, named Hierarchical Sparsemax Attention Networks (HSAN), replaces the original Softmax function with the Sparsemax function, in order to better handle cases when there are a lot of irrelevant components (in which case simply removing them using the first approach may not be helpful).

5.1 Hierarchical Pruned Attention Networks (HPAN)

However, given that all components are taken into account, components with very low attention values (such as stop words), may only be introducing noise into the classification process. To mitigate this issue we modify both attention mechanisms to completely ignore words and sentences with an attention weight below a given threshold (which we call $\alpha_{\text{min}}$).

Like with the original HANs, the HPANs, given as input a flat, padded document $d$ describing a sequence of word tokens/indices from a given vocabulary, with shape $(S, L)$ (where $S$ is the number of sentences and $L$ the number of words per sentence), follow the detailed classification process:

1. Each sentence of $L$ words is first passed through a embedding layer, obtaining a word embeddings tensor $w$ with shape $(S, L, E)$, where $E$ represents the dimension of a single embedding vector. As in the original work, we resort to a pre-loaded weight vector with dimension of 200 embeddings (GloVe).

2. Word embeddings $w$ are then passed through a bidirectional GRU, computing the hidden word scores $h = [\overrightarrow{h}, \overleftarrow{h}]$.

3. Hidden word scores $h$ are then passed through a single-layer feedforward network, obtaining hidden word representations $u = \tanh(W_w h + b_w)$.

4. Hidden word representations are then passed through an attention mechanism, which using a trainable word level context vector $u_w$, compute the attention weights $\alpha = \text{Softmax}(u^\top u_w)$.

5. Attention weights $\alpha$ below a minimum attention threshold $\alpha_{\text{min}}$ are set to zero and then normalized so that the remaining attention weights still sum up to 1.0.

6. New attention weights $\alpha$ and hidden word scores $h$ are then multiplied, obtaining the sentence vectors $s = \alpha \odot h$.

7. Sentence vectors $s$ are then passed through a bidirectional GRU, computing the hidden sentence scores $h_s = [\overrightarrow{h_s}, \overleftarrow{h_s}]$.

8. Hidden sentence scores $h_s$ are then passed through a single-layer feedforward network, obtaining hidden sentence representations $u_s = \tanh(W_s h_s + b_s)$.

9. Hidden sentence representations are then passed through an attention mechanism, which using a trainable sentence level context vector $u_s$, compute the attention weights $\alpha = \text{Softmax}(u_s^\top u_s)$.

10. Attention weights $\alpha$ below a minimum attention threshold $\alpha_{\text{min}}$ are set to zero and then normalized so that the remaining attention weights still sum up to 1.0.
11. New attention weights $\alpha$ and hidden sentence scores $h_s$ are then again multiplied, obtaining the document features $v = \alpha \odot h_s$

12. Document features $v$ are finally passed through an affine transformation, obtaining the class logit scores $z = W_v v + b_v$

13. Finally, $z$ can be used to compute the probabilities of each class $p = \text{softmax}(z)$. Cross entropy loss is then used to compute the gradients for all parameters.

5.2 Hierarchical Sparsemax Attention Networks (HSAN)

In the original HANs Yang et al. [2016], attention weights $\alpha$ are computed using the Softmax function:

$$\text{softmax}(z) := \frac{\exp(z)}{\sum_{z'} \exp(z')}$$  \hspace{1cm} (1)

However, if there are a lot of irrelevant words and sentences, completely removing them using the pruning mechanism described above may not be beneficial. We therefore follow Martins and Astudillo [2016] work by using their introduced Sparsemax function.

The Sparsemax function is proven to better handle distributions with a lot of near-zero probabilities by returning the "Euclidean projection of the input vector $z$ onto the probability simplex" [Martins and Astudillo 2016], i.e.:

$$\text{sparsemax}(z) := \arg \min_{p \in \Delta^{k+1}} ||p - z||^2$$  \hspace{1cm} (2)

The combination of such function with the attention mechanism only seemed logical, and therefore, to mitigate this issue, we explore a second model, named the Hierarchical Sparsemax Attention Network (HSAN), which replaces the Softmax function used in both attention mechanisms with the Sparsemax function.

Once again, given as input a flat, padded document $d$ describing a sequence of word tokens/indices from a given vocabulary, with shape $(S, L)$ (where $S$ is the number of sentences and $L$ the number of words per sentence), the document classification process for the HSANs is the following:

1. Each sentence of $L$ words is first passed through a embedding layer, obtaining a word embeddings tensor $w$ with shape $(S, L, E)$, where $E$ represents the dimension of a single embedding vector. As in the original work, we resort to a pre-loaded weight vector with dimension of 200 embeddings (GloVe).

2. Word embeddings $w$ are then passed through a bidirectional GRU, computing the hidden word scores $h = [h^\rightarrow, h^\leftarrow]$.

3. Hidden word scores $h$ are then passed through a single-layer feedforward network, obtaining hidden word representations $u = \tanh(W_w h + b_w)$.

4. Hidden word representations are then passed through an attention mechanism, which using a trainable word level context vector $u_w$, compute the attention weights $\alpha = \text{sparsemax}(u^\top u_w)$

5. Attention weights $\alpha$ and hidden word scores $h$ are multiplied, obtaining the sentence vectors $s = \alpha \odot h$

6. Sentence vectors $s$ are then passed through a bidirectional GRU, computing the hidden sentence scores $h_s = [h_s^\rightarrow, h_s^\leftarrow]$.

7. Hidden sentence scores $h_s$ are then passed through a single-layer feedforward network, obtaining hidden sentence representations $u_s = \tanh(W_s h_s + b_s)$.

8. Hidden sentence representations are then passed through an attention mechanism, which using a trainable sentence level context vector $u_s$, compute the attention weights $\alpha = \text{sparsemax}(u_s^\top u_s)$

9. Attention weights $\alpha$ and hidden sentence scores $h_s$ are then multiplied, obtaining the document features $v = \alpha \odot h_s$
10. Document features $v$ are finally passed through an affine transformation, obtaining the class logit scores $z = W_v v + b_v$.

11. Finally, $z$ can be used to compute the probabilities of each class $p = \text{softmax}(z)$. Cross entropy loss is then used to compute the gradients for all parameters.

6 Evaluation

We compare both our models (HPAN and HSNA) against the original HAN.

Table 1 displays all used hyperparameters, alongside their descriptions and their values.

| Hyperparameter                     | Our value | Yang et al. [2016]'s value |
|-----------------------------------|-----------|---------------------------|
| Word2vec embeddings size          | 200       | 200                       |
| GRU layers                        | 1         | 1                         |
| GRU layers hidden sizes           | 50        | 50                        |
| Dropout                           | 0.1       | unreported                |
| Training epochs                   | 3         | unreported                |
| Optimizer                         | adam      | sgd                       |
| Learning rate                     | 0.001     | unreported*               |
| Batch size                        | 64        | 64                        |
| HPAN min. attention threshold     | 0.05      | n.a.                      |

* The authors pick the best learning rate from a grid search, however, do not report the value. [Yang et al., 2016].

6.1 Datasets

Unfortunately, due to long loading and training times for most of the datasets, we decided to perform our evaluation for all models in the IMDB Review dataset for sentiment analysis.

Even though we still provide all the necessary code to train any model on any of the original datasets (Yelp'15, Yahoo Answers, Amazon and IMDB), we chose this dataset due to its relatively low sample size (which allows both faster loading and training) and its simplicity.

The IMDB review dataset, introduced by [Maas et al., 2011] contains 50'000 movie review and polarizes their classifications to either positive and negative (unlike the original IMDB dataset which kept 10 classification labels). There’s an even number of positive and negative reviews. However, the authors [Maas et al., 2011] only considered ‘highly polarized reviews’, or in other words, classifications are considered positive a movie review with a classification of 7 or higher, and negative a movie review with a classification of 4 or lower (both out of 10). Movie reviews with classifications of 5 and 6 were not included. The dataset was then evenly divided into training and test sets, both containing 25’000 movie reviews, and the 25’000 training reviews were then split into 70% training and 30% validation.

We load the IMDB review dataset and preprocess it using the official Pytorch torchtext module using the spacy tokenizer.

6.2 Experimental Procedure

Given a total of three models - i) HAN, ii) HPAN and iii) HSAN and a single dataset - IMDB review for sentiment analysis - the evaluation procedure for a single experiment for a model (N=1) was the following:

https://ai.stanford.edu/amaas/data/sentiment/
https://pytorch.org/text/_modules/torchtext/datasets/imdb.html
https://spacy.io/api/tokenizer
1. Train a model on the IMDB dataset for three epochs on the training set.
2. Evaluate, at each epoch, the classification accuracy on the validation set.
3. Evaluate, after all epochs, the classification accuracy on test dataset.

Finally, to compare the results obtained by our models, we use the following metric: 

\[ M \]: Document classification accuracy (%) on obtained by a trained model on the test dataset, after three training epochs.

### 7 Results

All three models (HAN, HPAN and HSAN) were trained and evaluated according to metric \( M \) (classification accuracy (%) on test dataset, after three training epochs).

Once again, due to time constraints and available computational resources, we were able to perform a total of 5 (independent) experiments (N=5) for the HAN and HSAN models and 3 experiments (N=3) for the HPAN model (due to gradient computation issue, further discussed in section 8).

Table 2 reports the obtained document classification accuracies on test sets for all models and figure 2 reports our obtained results with a confidence level of 99%.

Table 2: Obtained document classification accuracies in the IMDB review test set, for all three models, after three training epochs.

| Model | \( \mu \)     | \( \sigma \) | N  |
|-------|--------------|--------------|----|
| HAN   | 87.08\%     | 0.004        | 5  |
| HPAN  | 87.01\%     | 0.011        | 3  |
| HSAN  | 85.64\%     | 0.011        | 5  |

Taking all experiments into account, we can make three conclusions.

First, both our models were able to perform similarly as the current state-of-the-art (HAN). The second conclusion is that neither pruning words and sentences nor replacing the Softmax function with the Sparsemax function in the attention mechanism showed great advantage.
Both these conclusions are only true regarding the fact that the experiments were conducted on the IMDB Review dataset for sentiment analysis. This binary/polarized dataset, even tough it provided a first good test bed to evaluate our proposed models, may not allow deeper conclusions due to its simplicity. The only way to extrapolate these conclusions is to repeat the process by evaluating all three models using more complex datasets (such as the ones used in the original work [Yang et al., 2016]).

Finally, the HSAN hinted a slightly worst performance after training for the same number of epochs and using the same hyperparameters as the other models. There wasn’t, however, a very significant difference to make this conclusion a very strong one.

8 Conclusion & Future Work

In this paper we tackled the issue of irrelevant words and sentences by modifying the Hierarchical Attention Networks attention mechanism [Yang et al., 2016]. We introduced and evaluated two novel models, namely the Hierarchical Pruned Attention Networks and the Hierarchical Sparsemax Attention Networks. We compared both models against the original Hierarchical Attention Networks and, regarding the IMDB Review dataset for sentiment analysis, our empirical evaluation (N=5, confidence level of 99%) showed there was no significant performance gains (in terms of document classification accuracy).

Unfortunately, the evaluation of these on a single, polarized dataset may not be enough to conclude that there are no advantages/disadvantages between the three methods. Logically, our next line of work will be to continue this study and perform a new, empirical evaluation, on three additional datasets – Yelp’15, Amazon and Yahoo Answers – which all label each document with one of K classes (instead of simply two, like the IMDB review for sentiment analysis). During this study, we will also set out to perform a fine-tuning for all hyperparameters (including the minimum attention threshold for the HPAN).

Even though our HPAN model was able to train properly, it had the requirement of being restricted to a cpu device only, rendering its experiments very slow when in comparison with the other two models. This was caused by an issue in PyTorch’s autograd module when running on a gpu device (and also happened sometimes when using cpu), which neither of us were able to solve - when setting to zero all attention weights below a minimum threshold, the backpropagation of the gradients yielded nan.

Finally, it will also be interesting to study which are the effects, in terms of percentage of pruned words/sentences, when given a certain minimum attention threshold to the HPAN. Since there are two individual attention mechanisms (one for words and one for sentences), it can also prove interesting to explore the effects of pruning on each individual level (words/sentences). The same process can be done with the Sparsemax function, which is currently used on both levels.

References

Zichao Yang, Diyi Yang, Chris Dyer, Xiaodong He, Alex Smola, and Eduard Hovy. Hierarchical attention networks for document classification. In Proceedings of the 2016 conference of the North American chapter of the association for computational linguistics: human language technologies, pages 1480–1489, 2016.

Andre Martins and Ramon Astudillo. From softmax to sparsemax: A sparse model of attention and multi-label classification. In International Conference on Machine Learning, pages 1614–1623, 2016.

Kamran Kowsari, Kiana Jafari Meimandi, Mojtaba Heidarysafa, Sanjana Mendu, Laura Barnes, and Donald Brown. Text classification algorithms: A survey. Information, 10(4):150, 2019.

Kamran Kowsari, Mojtaba Heidarysafa, Donald E Brown, Kiana Jafari Meimandi, and Laura E Barnes. Rmdl: Random multimodel deep learning for classification. In Proceedings of the 2nd International Conference on Information System and Data Mining, pages 19–28. ACM, 2018.

Junyi Chen, Shankai Yan, and Ka-Chun Wong. Verbal aggression detection on twitter comments: Convolutional neural network for short-text sentiment analysis. Neural Computing and Applications, pages 1–10, 2018.
Mingyang Jiang, Yanchun Liang, Xiaoyue Feng, Xiaojing Fan, Zhili Pei, Yu Xue, and Renchu Guan. Text classification based on deep belief network and softmax regression. *Neural Computing and Applications*, 29(1):61–70, 2018.

Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts. Learning word vectors for sentiment analysis. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 142–150, Portland, Oregon, USA, June 2011. Association for Computational Linguistics. URL [http://www.aclweb.org/anthology/P11-1015](http://www.aclweb.org/anthology/P11-1015).

Xiang Zhang, Junbo Zhao, and Yann LeCun. Character-level convolutional networks for text classification. In *Advances in neural information processing systems*, pages 649–657, 2015.

Duyu Tang, Bing Qin, and Ting Liu. Document modeling with gated recurrent neural network for sentiment classification. In *Proceedings of the 2015 conference on empirical methods in natural language processing*, pages 1422–1432, 2015.