Hierarchical Attention Networks for Document Classification

Zichao Yang[1], **Diyi Yang**[1], **Chris Dyer**[1], **Xiaodong He**[2], **Alex Smola**[1], **Eduard Hovy**[1]
[1]Carnegie Mellon University, [2]Microsoft Research, Redmond
{zichaoy, diyiy, cdyer, hovy}@cs.cmu.edu
xiaohe@microsoft.com alex@smola.org

Abstract

We propose a hierarchical attention network for document classification. Our model has two distinctive characteristics: (i) it has a hierarchical structure that mirrors the hierarchical structure of documents; (ii) it has two levels of attention mechanisms applied at the word- and sentence-level, enabling it to attend differentially to more and less important content when constructing the document representation. Experiments conducted on six large scale text classification tasks demonstrate that the proposed architecture outperform previous methods by a substantial margin. Visualization of the attention layers illustrates that the model selects qualitatively informative words and sentences.

1 Introduction

Text classification is one of the fundamental task in Natural Language Processing. The goal is to assign labels to text. It has broad applications including topic labeling (Wang and Manning, 2012), sentiment classification (Maas et al., 2011; Pang and Lee, 2008), and spam detection (Sahami et al., 1998). Traditional approaches of text classification represent documents with sparse lexical features, such as n-grams, and then use a linear model or kernel methods on this representation (Wang and Manning, 2012; Joachims, 1998). More recent approaches used deep learning, such as convolutional neural networks (Blunsom et al., 2014) and recurrent neural networks based on long short-term memory (LSTM) (Hochreiter and Schmidhuber, 1997) to learn text representations.

pork belly = delicious . || scallops? || I don't even like scallops, and these were a-m-a-z-i-n-g . || fun and tasty cocktails. || next time I in Phoenix, I will go back here. || Highly recommend.

Figure 1: A simple example review from Yelp 2013 that consists of five sentences, delimited by period, question mark. The first and third sentence delivers stronger meaning and inside, the word *delicious, a-m-a-z-i-n-g* contributes the most in defining sentiment of the two sentences.

Although neural-network–based approaches to text classification have been quite effective (Kim, 2014; Zhang et al., 2015; Johnson and Zhang, 2014; Tang et al., 2015), in this paper we test the hypothesis that better representations can be obtained by incorporating knowledge of document structure in the model architecture. The intuition underlying our model is that not all parts of a document are equally relevant for answering a query and that determining the relevant sections involves modeling the interactions of the words, not just their presence in isolation.

Our primary contribution is a new neural architecture (§2), the Hierarchical Attention Network (HAN) that is designed to capture two basic insights about document structure. First, since documents have a hierarchical structure (words form sentences, sentences form a document), we likewise construct a document representation by first building representations of sentences and then aggregating those into a document representation. Second, it is observed that different words and sentences in a documents are differentially informative. Moreover, the impor-

Proceedings of NAACL-HLT 2016, pages 1480–1489,
San Diego, California, June 12-17, 2016.

tance of words and sentences are highly context dependent, i.e. the same word or sentence may be differentially important in different context (§3.5). To include sensitivity to this fact, our model includes two levels of attention mechanisms (Bahdanau et al., 2014; Xu et al., 2015) — one at the word level and one at the sentence level — that let the model to pay more or less attention to individual words and sentences when constructing the representation of the document. To illustrate, consider the example in Fig. 1, which is a short Yelp review where the task is to predict the rating on a scale from 1–5. Intuitively, the first and third sentence have stronger information in assisting the prediction of the rating; within these sentences, the word `delicious`, `a-m-a-z-i-n-g` contributes more in implying the positive attitude contained in this review. Attention serves two benefits: not only does it often result in better performance, but it also provides insight into which words and sentences contribute to the classification decision which can be of value in applications and analysis (Shen et al., 2014; Gao et al., 2014).

The key difference to previous work is that our system uses *context* to discover *when* a sequence of tokens is relevant rather than simply filtering for (sequences of) tokens, taken out of context. To evaluate the performance of our model in comparison to other common classification architectures, we look at six data sets (§3). Our model outperforms previous approaches by a significant margin.

2 Hierarchical Attention Networks

The overall architecture of the Hierarchical Attention Network (HAN) is shown in Fig. 2. It consists of several parts: a word sequence encoder, a word-level attention layer, a sentence encoder and a sentence-level attention layer. We describe the details of different components in the following sections.

2.1 GRU-based sequence encoder

The GRU (Bahdanau et al., 2014) uses a gating mechanism to track the state of sequences without using separate memory cells. There are two types of gates: the reset gate r_t and the update gate z_t. They together control how information is updated to the

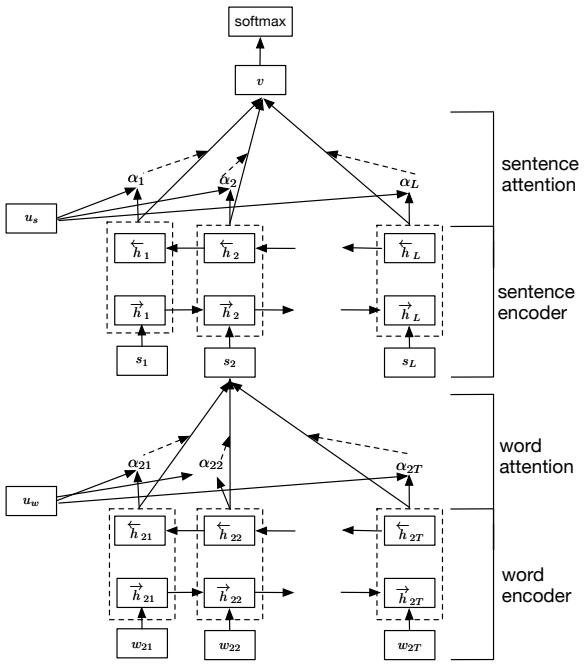

Figure 2: Hierarchical Attention Network.

state. At time t, the GRU computes the new state as

$$h_t = (1 - z_t) \odot h_{t-1} + z_t \odot \tilde{h}_t. \quad (1)$$

This is a linear interpolation between the previous state h_{t-1} and the current new state $\tilde{h}_t$ computed with new sequence information. The gate z_t decides how much past information is kept and how much new information is added. z_t is updated as:

$$z_t = \sigma(W_z x_t + U_z h_{t-1} + b_z), \quad (2)$$

where x_t is the sequence vector at time t. The candidate state $\tilde{h}_t$ is computed in a way similar to a traditional recurrent neural network (RNN):

$$\tilde{h}_t = \tanh(W_h x_t + r_t \odot (U_h h_{t-1}) + b_h), \quad (3)$$

Here r_t is the reset gate which controls how much the past state contributes to the candidate state. If r_t is zero, then it forgets the previous state. The reset gate is updated as follows:

$$r_t = \sigma(W_r x_t + U_r h_{t-1} + b_r) \quad (4)$$

2.2 Hierarchical Attention

We focus on document-level classification in this work. Assume that a document has L sentences

s_i and each sentence contains T_i words. w_{it} with $t \in [1, T]$ represents the words in the ith sentence. The proposed model projects the raw document into a vector representation, on which we build a classifier to perform document classification. In the following, we will present how we build the document level vector progressively from word vectors by using the hierarchical structure.

Word Encoder Given a sentence with words $w_{it}, t \in [0, T]$, we first embed the words to vectors through an embedding matrix W_e, $x_{ij} = W_e w_{ij}$. We use a bidirectional GRU (Bahdanau et al., 2014) to get annotations of words by summarizing information from both directions for words, and therefore incorporate the contextual information in the annotation. The bidirectional GRU contains the forward GRU $\overrightarrow{f}$ which reads the sentence s_i from w_{i1} to w_{iT} and a backward GRU $\overleftarrow{f}$ which reads from w_{iT} to w_{i1}:

$$\begin{aligned} x_{it} &= W_e w_{it}, t \in [1, T], \\ \overrightarrow{h}_{it} &= \overrightarrow{\mathrm{GRU}}(x_{it}), t \in [1, T], \\ \overleftarrow{h}_{it} &= \overleftarrow{\mathrm{GRU}}(x_{it}), t \in [T, 1]. \end{aligned}$$

We obtain an annotation for a given word w_{it} by concatenating the forward hidden state $\overrightarrow{h}_{it}$ and backward hidden state $\overleftarrow{h}_{it}$, i.e., $h_{it} = [\overrightarrow{h}_{it}, \overleftarrow{h}_{it}]$, which summarizes the information of the whole sentence centered around w_{it}.

Note that we directly use word embeddings. For a more complete model we could use a GRU to get word vectors directly from characters, similarly to (Ling et al., 2015). We omitted this for simplicity.

Word Attention Not all words contribute equally to the representation of the sentence meaning. Hence, we introduce attention mechanism to extract such words that are important to the meaning of the sentence and aggregate the representation of those informative words to form a sentence vector. Specifically,

$$u_{it} = \tanh(W_w h_{it} + b_w) \tag{5}$$

$$\alpha_{it} = \frac{\exp(u_{it}^\top u_w)}{\sum_t \exp(u_{it}^\top u_w)} \tag{6}$$

$$s_i = \sum_t \alpha_{it} h_{it}. \tag{7}$$

That is, we first feed the word annotation h_{it} through a one-layer MLP to get u_{it} as a hidden representation of h_{it}, then we measure the importance of the word as the similarity of u_{it} with a word level context vector u_w and get a normalized importance weight α_{it} through a softmax function. After that, we compute the sentence vector s_i (we abuse the notation here) as a weighted sum of the word annotations based on the weights. The context vector u_w can be seen as a high level representation of a fixed query "what is the informative word" over the words like that used in memory networks (Sukhbaatar et al., 2015; Kumar et al., 2015). The word context vector u_w is randomly initialized and jointly learned during the training process.

Sentence Encoder Given the sentence vectors s_i, we can get a document vector in a similar way. We use a bidirectional GRU to encode the sentences:

$$\begin{aligned} \overrightarrow{h}_i &= \overrightarrow{\mathrm{GRU}}(s_i), i \in [1, L], \\ \overleftarrow{h}_i &= \overleftarrow{\mathrm{GRU}}(s_i), t \in [L, 1]. \end{aligned}$$

We concatenate $\overrightarrow{h}_i$ and $\overleftarrow{h}_j$ to get an annotation of sentence i, i.e., $h_i = [\overrightarrow{h}_i, \overleftarrow{h}_i]$. h_i summarizes the neighbor sentences around sentence i but still focus on sentence i.

Sentence Attention To reward sentences that are clues to correctly classify a document, we again use attention mechanism and introduce a sentence level context vector u_s and use the vector to measure the importance of the sentences. This yields

$$u_i = \tanh(W_s h_i + b_s), \tag{8}$$

$$\alpha_i = \frac{\exp(u_i^\top u_s)}{\sum_i \exp(u_i^\top u_s)}, \tag{9}$$

$$v = \sum_i \alpha_i h_i, \tag{10}$$

where v is the document vector that summarizes all the information of sentences in a document. Similarly, the sentence level context vector can be randomly initialized and jointly learned during the training process.

2.3 Document Classification

The document vector v is a high level representation of the document and can be used as features for doc-

ument classification:

$$p = \text{softmax}(W_c v + b_c). \quad (11)$$

We use the negative log likelihood of the correct labels as training loss:

$$L = -\sum_d \log p_{dj}, \quad (12)$$

where j is the label of document d.

3 Experiments

3.1 Data sets

We evaluate the effectiveness of our model on six large scale document classification data sets. These data sets can be categorized into two types of document classification tasks: sentiment estimation and topic classification. The statistics of the data sets are summarized in Table 1. We use 80% of the data for training, 10% for validation, and the remaining 10% for test, unless stated otherwise.

Yelp reviews are obtained from the Yelp Dataset Challenge in 2013, 2014 and 2015 (Tang et al., 2015). There are five levels of ratings from 1 to 5 (higher is better).

IMDB reviews are obtained from (Diao et al., 2014). The ratings range from 1 to 10.

Yahoo answers are obtained from (Zhang et al., 2015). This is a topic classification task with 10 classes: Society & Culture, Science & Mathematics, Health, Education & Reference, Computers & Internet, Sports, Business & Finance, Entertainment & Music, Family & Relationships and Politics & Government. The document we use includes question titles, question contexts and best answers. There are 140,000 training samples and 5000 testing samples. The original data set does not provide validation samples. We randomly select 10% of the training samples as validation.

Amazon reviews are obtained from (Zhang et al., 2015). The ratings are from 1 to 5. 3,000,000 reviews are used for training and 650,000 reviews for testing. Similarly, we use 10% of the training samples as validation.

3.2 Baselines

We compare HAN with several baseline methods, including traditional approaches such as linear methods, SVMs and paragraph embeddings using neural networks, LSTMs, word-based CNN, character-based CNN, and Conv-GRNN, LSTM-GRNN. These baseline methods and results are reported in (Zhang et al., 2015; Tang et al., 2015).

3.2.1 Linear methods

Linear methods (Zhang et al., 2015) use the constructed statistics as features. A linear classifier based on multinomial logistic regression is used to classify the documents using the features.

BOW and BOW+TFIDF The 50,000 most frequent words from the training set are selected and the count of each word is used features. Bow+TFIDF, as implied by the name, uses the TFIDF of counts as features.

n-grams and n-grams+TFIDF used the most frequent 500,000 n-grams (up to 5-grams).

Bag-of-means The average word2vec embedding (Mikolov et al., 2013) is used as feature set.

3.2.2 SVMs

SVMs-based methods are reported in (Tang et al., 2015), including **SVM+Unigrams, Bigrams, Text Features, AverageSG, SSWE**. In detail, **Unigrams** and **Bigrams** uses bag-of-unigrams and bag-of-bigrams as features respectively.

Text Features are constructed according to (Kiritchenko et al., 2014), including word and character n-grams, sentiment lexicon features etc.

AverageSG constructs 200-dimensional word vectors using word2vec and the average word embeddings of each document are used.

SSWE uses sentiment specific word embeddings according to (Tang et al., 2014).

3.2.3 Neural Network methods

The neural network based methods are reported in (Tang et al., 2015) and (Zhang et al., 2015).

CNN-word Word based CNN models like that of (Kim, 2014) are used.

CNN-char Character level CNN models are reported in (Zhang et al., 2015).

Data set	classes	documents	average #s	max #s	average #w	max #w	vocabulary
Yelp 2013	5	335,018	8.9	151	151.6	1184	211,245
Yelp 2014	5	1,125,457	9.2	151	156.9	1199	476,191
Yelp 2015	5	1,569,264	9.0	151	151.9	1199	612,636
IMDB review	10	348,415	14.0	148	325.6	2802	115,831
Yahoo Answer	10	1,450,000	6.4	515	108.4	4002	1,554,607
Amazon review	5	3,650,000	4.9	99	91.9	596	1,919,336

Table 1: Data statistics: #s denotes the number of sentences (average and maximum per document), #w denotes the number of words (average and maximum per document).

LSTM takes the whole document as a single sequence and the average of the hidden states of all words is used as feature for classification.

Conv-GRNN and LSTM-GRNN were proposed by (Tang et al., 2015). They also explore the hierarchical structure: a CNN or LSTM provides a sentence vector, and then a gated recurrent neural network (GRNN) combines the sentence vectors from a document level vector representation for classification.

3.3 Model configuration and training

We split documents into sentences and tokenize each sentence using Stanford's CoreNLP (Manning et al., 2014). We only retain words appearing more than 5 times in building the vocabulary and replace the words that appear 5 times with a special UNK token. We obtain the word embedding by training an unsupervised word2vec (Mikolov et al., 2013) model on the training and validation splits and then use the word embedding to initialize W_e.

The hyper parameters of the models are tuned on the validation set. In our experiments, we set the word embedding dimension to be 200 and the GRU dimension to be 50. In this case a combination of forward and backward GRU gives us 100 dimensions for word/sentence annotation. The word/sentence context vectors also have a dimension of 100, initialized at random.

For training, we use a mini-batch size of 64 and documents of similar length (in terms of the number of sentences in the documents) are organized to be a batch. We find that length-adjustment can accelerate training by three times. We use stochastic gradient descent to train all models with momentum of 0.9. We pick the best learning rate using grid search on the validation set.

3.4 Results and analysis

The experimental results on all data sets are shown in Table 2. We refer to our models as **HN-{AVE, MAX, ATT}**. Here HN stands for Hierarchical Network, AVE indicates averaging, MAX indicates max-pooling, and ATT indicates our proposed hierarchical attention model. Results show that HN-ATT gives the best performance across all data sets.

The improvement is regardless of data sizes. For smaller data sets such as Yelp 2013 and IMDB, our model outperforms the previous best baseline methods by 3.1% and 4.1% respectively. This finding is consistent across other larger data sets. Our model outperforms previous best models by 3.2%, 3.4%, 4.6% and 6.0% on Yelp 2014, Yelp 2015, Yahoo Answers and Amazon Reviews. The improvement also occurs regardless of the type of task: sentiment classification, which includes Yelp 2013-2014, IMDB, Amazon Reviews and topic classification for Yahoo Answers.

From Table 2 we can see that neural network based methods that do *not* explore hierarchical document structure, such as LSTM, CNN-word, CNN-char have little advantage over traditional methods for large scale (in terms of document size) text classification. E.g. SVM+TextFeatures gives performance 59.8, 61.8, 62.4, 40.5 for Yelp 2013, 2014, 2015 and IMDB respectively, while CNN-word has accuracy 59.7, 61.0, 61.5, 37.6 respectively.

Exploring the hierarchical structure only, as in HN-AVE, HN-MAX, can significantly improve over LSTM, CNN-word and CNN-char. For example, our HN-AVE outperforms CNN-word by 7.3%, 8.8%, 8.5%, 10.2% than CNN-word on Yelp 2013, 2014, 2015 and IMDB respectively. Our model HN-ATT that further utilizes attention mechanism

	Methods	Yelp'13	Yelp'14	Yelp'15	IMDB	Yahoo Answer	Amazon
Zhang et al., 2015	BoW	-	-	58.0	-	68.9	54.4
	BoW TFIDF	-	-	59.9	-	71.0	55.3
	ngrams	-	-	56.3	-	68.5	54.3
	ngrams TFIDF	-	-	54.8	-	68.5	52.4
	Bag-of-means	-	-	52.5	-	60.5	44.1
Tang et al., 2015	Majority	35.6	36.1	36.9	17.9	-	-
	SVM + Unigrams	58.9	60.0	61.1	39.9	-	-
	SVM + Bigrams	57.6	61.6	62.4	40.9	-	-
	SVM + TextFeatures	59.8	61.8	62.4	40.5	-	-
	SVM + AverageSG	54.3	55.7	56.8	31.9	-	-
	SVM + SSWE	53.5	54.3	55.4	26.2	-	-
Zhang et al., 2015	LSTM	-	-	58.2	-	70.8	59.4
	CNN-char	-	-	62.0	-	71.2	59.6
	CNN-word	-	-	60.5	-	71.2	57.6
Tang et al., 2015	Paragraph Vector	57.7	59.2	60.5	34.1	-	-
	CNN-word	59.7	61.0	61.5	37.6	-	-
	Conv-GRNN	63.7	65.5	66.0	42.5	-	-
	LSTM-GRNN	65.1	67.1	67.6	45.3	-	-
This paper	HN-AVE	67.0	69.3	69.9	47.8	75.2	62.9
	HN-MAX	66.9	69.3	70.1	48.2	75.2	62.9
	HN-ATT	**68.2**	**70.5**	**71.0**	**49.4**	**75.8**	**63.6**

Table 2: Document Classification, in percentage

combined with hierarchical structure improves over previous models (LSTM-GRNN) by 3.1%, 3.4%, 3.5% and 4.1% respectively. More interestingly, in the experiments, HN-AVE is equivalent to using non-informative global word/sentence context vectors (e.g., if they are all-zero vectors, then the attention weights in Eq. 6 and 9 become uniform weights). Compared to HN-AVE, the HN-ATT model gives superior performance across the board. This clearly demonstrates the effectiveness of the proposed global word and sentence importance vectors for the HAN.

3.5 Context dependent attention weights

If words were inherently important or not important, models without attention mechanism might work well since the model could automatically assign low weights to irrelevant words and vice versa. However, the importance of words is highly context dependent. For example, the word `good` may appear in a review that has the lowest rating either because users are only happy with part of the product/service or because they use it in a negation, such as `not good`. To verify that our model can capture context dependent word importance, we plot the distribution of the attention weights of the words `good` and `bad` from the test split of Yelp 2013 data set as shown in Figure 3(a) and Figure 4(a). We can see that the distribution has a attention weight assigned to a word from 0 to 1. This indicates that our model captures diverse context and assign context-dependent weight to the words.

For further illustration, we plot the distribution when conditioned on the ratings of the review. Subfigures 3(b)-(f) in Figure 3 and Figure 4 correspond to the rating 1-5 respectively. In particular, Figure 3(b) shows that the weight of `good` concentrates on the low end in the reviews with rating 1. As the rating increases, so does the weight distribution. This means that the word `good` plays a more important role for reviews with higher ratings. We can observe the converse trend in Figure 4 for the word `bad`. This confirms that our model can capture the context-dependent word importance.

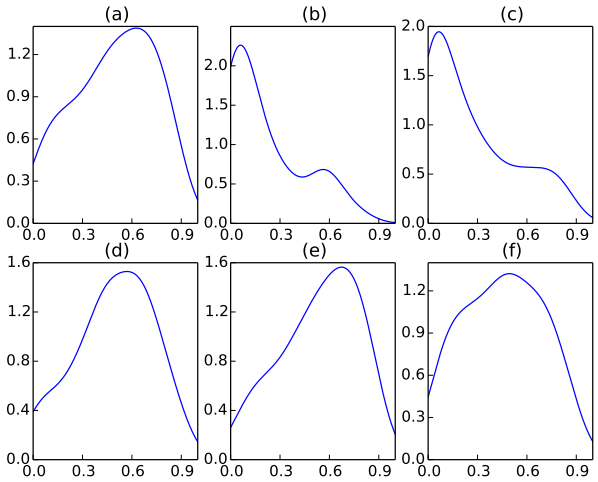

Figure 3: Attention weight distribution of `good`. (a) — aggregate distribution on the test split; (b)-(f) stratified for reviews with ratings 1-5 respectively. We can see that the weight distribution shifts to *higher* end as the rating goes higher.

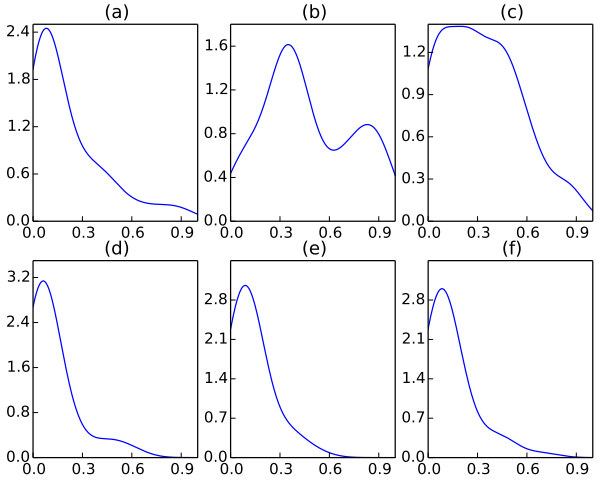

Figure 4: Attention weight distribution of the word `bad`. The setup is as above: (a) contains the aggregate distribution, while (b)-(f) contain stratifications to reviews with ratings 1-5 respectively. Contrary to before, the word `bad` is considered important for poor ratings and less so for good ones.

3.6 Visualization of attention

In order to validate that our model is able to select informative sentences and words in a document, we visualize the hierarchical attention layers in Figures 5 and 6 for several documents from the Yelp 2013 and Yahoo Answers data sets.

Every line is a sentence (sometimes sentences spill over several lines due to their length). Red denotes the sentence weight and blue denotes the word weight. Due to the hierarchical structure, we normalize the word weight by the sentence weight to make sure that only important words in important sentences are emphasized. For visualization purposes we display $\sqrt{p_s}p_w$. The $\sqrt{p_s}$ term displays the important words in unimportant sentences to ensure that they are not totally invisible.

Figure 5 shows that our model can select the words carrying strong sentiment like `delicious`, `amazing`, `terrible` and their corresponding sentences. Sentences containing many words like `cocktails`, `pasta`, `entree` are disregarded. Note that our model can not only select words carrying strong sentiment, it can also deal with complex across-sentence context. For example, there are sentences like `i don't even like scallops` in the first document of Fig. 5, if looking purely at the single sentence, we may think this is negative comment. However, our model looks at the context of this sentence and figures out this is a positive review and chooses to ignore this sentence.

Our hierarchical attention mechanism also works well for topic classification in the Yahoo Answer data set. For example, for the left document in Figure 6 with label 1, which denotes Science and Mathematics, our model accurately localizes the words `zebra`, `strips`, `camouflage`, `predator` and their corresponding sentences. For the right document with label 4, which denotes Computers and Internet, our model focuses on `web`, `searches`, `browsers` and their corresponding sentences. Note that this happens in a *multiclass* setting, that is, detection happens before the selection of the topic!

4 Related Work

Kim (2014) use neural networks for text classification. The architecture is a direct application of CNNs, as used in computer vision (LeCun et al., 1998), albeit with NLP interpretations. Johnson and Zhang (2014) explores the case of directly using a high-dimensional one hot vector as input. They find that it performs well. Unlike word level modelings, Zhang et al. (2015) apply a character-level CNN for text classification and achieve competitive results. Socher et al. (2013) use recursive neural networks for text classification. Tai et al. (2015)

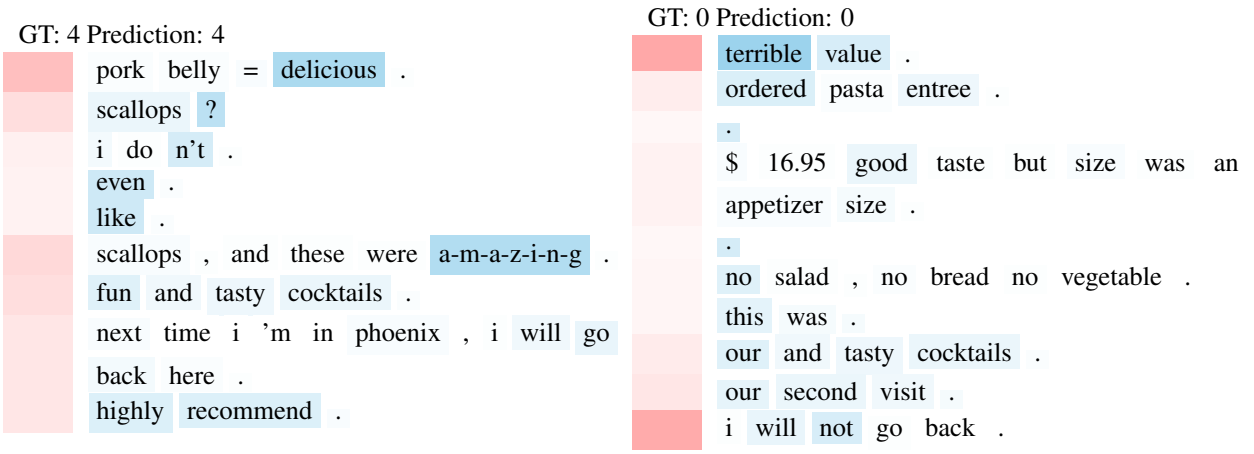

Figure 5: Documents from Yelp 2013. Label 4 means star 5, label 0 means star 1.

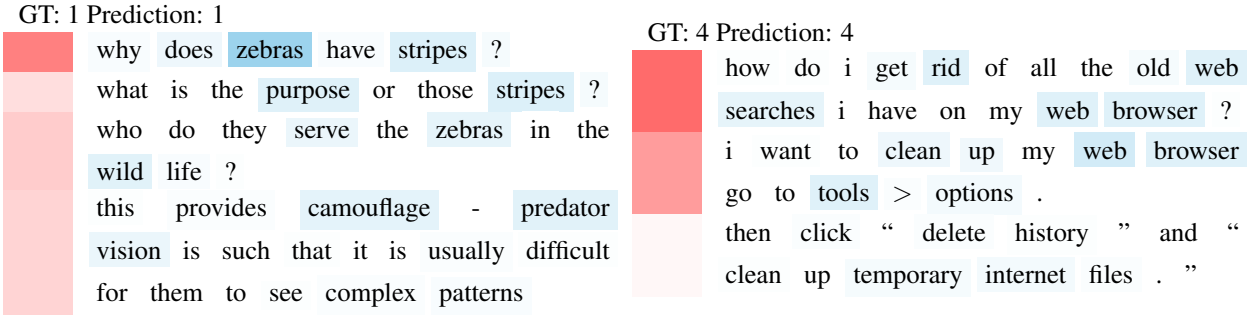

Figure 6: Documents from Yahoo Answers. Label 1 denotes Science and Mathematics and label 4 denotes Computers and Internet.

explore the structure of a sentence and use a tree-structured LSTMs for classification. There are also some works that combine LSTM and CNN structure to for sentence classification (Lai et al., 2015; Zhou et al., 2015). Tang et al. (2015) use hierarchical structure in sentiment classification. They first use a CNN or LSTM to get a sentence vector and then a bi-directional gated recurrent neural network to compose the sentence vectors to get a document vectors. There are some other works that use hierarchical structure in sequence generation (Li et al., 2015) and language modeling (Lin et al., 2015).

The attention mechanism was proposed by (Bahdanau et al., 2014) in machine translation. The encoder decoder framework is used and an attention mechanism is used to select the reference words in original language for words in foreign language before translation. Xu et al. (2015) uses the attention mechanism in image caption generation to select the relevant image regions when generating words in the captions. Further uses of the attention mechanism include parsing (Vinyals et al., 2014), natural language question answering (Sukhbaatar et al., 2015; Kumar et al., 2015; Hermann et al., 2015), and image question answering (Yang et al., 2015). Unlike these works, we explore a *hierarchical* attention mechanism (to the best of our knowledge this is the first such instance).

5 Conclusion

In this paper, we proposed hierarchical attention networks (HAN) for classifying documents. As a convenient side-effect we obtained better visualization using the highly informative components of a document. Our model progressively builds a document vector by aggregating important words into sentence vectors and then aggregating important sentences vectors to document vectors. Experimental results demonstrate that our model performs significantly better than previous methods. Visualization of these attention layers illustrates that our model is effective in picking out important words and sentences.

Acknowledgments This work was supported by Microsoft Research.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

Phil Blunsom, Edward Grefenstette, Nal Kalchbrenner, et al. 2014. A convolutional neural network for modelling sentences. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*. Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics.

Qiming Diao, Minghui Qiu, Chao-Yuan Wu, Alexander J Smola, Jing Jiang, and Chong Wang. 2014. Jointly modeling aspects, ratings and sentiments for movie recommendation (jmars). In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 193–202. ACM.

Jianfeng Gao, Patrick Pantel, Michael Gamon, Xiaodong He, Li Deng, and Yelong Shen. 2014. Modeling interestingness with deep neural networks. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*.

Karl Moritz Hermann, Tomáš Kočiskỳ, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching machines to read and comprehend. *arXiv preprint arXiv:1506.03340*.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Thorsten Joachims. 1998. *Text categorization with support vector machines: Learning with many relevant features*. Springer.

Rie Johnson and Tong Zhang. 2014. Effective use of word order for text categorization with convolutional neural networks. *arXiv preprint arXiv:1412.1058*.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882*.

Svetlana Kiritchenko, Xiaodan Zhu, and Saif M Mohammad. 2014. Sentiment analysis of short informal texts. *Journal of Artificial Intelligence Research*, pages 723–762.

Ankit Kumar, Ozan Irsoy, Jonathan Su, James Bradbury, Robert English, Brian Pierce, Peter Ondruska, Ishaan Gulrajani, and Richard Socher. 2015. Ask me anything: Dynamic memory networks for natural language processing. *arXiv preprint arXiv:1506.07285*.

Siwei Lai, Liheng Xu, Kang Liu, and Jun Zhao. 2015. Recurrent convolutional neural networks for text classification. In *Twenty-Ninth AAAI Conference on Artificial Intelligence*.

Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. 1998. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324.

Jiwei Li, Minh-Thang Luong, and Dan Jurafsky. 2015. A hierarchical neural autoencoder for paragraphs and documents. *arXiv preprint arXiv:1506.01057*.

Rui Lin, Shujie Liu, Muyun Yang, Mu Li, Ming Zhou, and Sheng Li. 2015. Hierarchical recurrent neural network for document modeling. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 899–907.

Wang Ling, Tiago Luís, Luís Marujo, Ramón Fernandez Astudillo, Silvio Amir, Chris Dyer, Alan W Black, and Isabel Trancoso. 2015. Finding function in form: Compositional character models for open vocabulary word representation. *arXiv preprint arXiv:1508.02096*.

Andrew L Maas, Raymond E Daly, Peter T Pham, Dan Huang, Andrew Y Ng, and Christopher Potts. 2011. Learning word vectors for sentiment analysis. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, pages 142–150. Association for Computational Linguistics.

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

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119.

Bo Pang and Lillian Lee. 2008. Opinion mining and sentiment analysis. *Foundations and trends in information retrieval*, 2(1-2):1–135.

Mehran Sahami, Susan Dumais, David Heckerman, and Eric Horvitz. 1998. A bayesian approach to filtering junk e-mail. In *Learning for Text Categorization: Papers from the 1998 workshop*, volume 62, pages 98–105.

Yelong Shen, Xiaodong He, Jianfeng Gao, Li Deng, and Gregoire Mesnil. 2014. A latent semantic model with convolutional-pooling structure for information retrieval. In *Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management*, pages 101–110. ACM.

Richard Socher, Alex Perelygin, Jean Y. Wu, Jason Chuang, Christopher D. Manning, Andrew Y. Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proc. EMNLP*.

Sainbayar Sukhbaatar, Arthur Szlam, Jason Weston, and Rob Fergus. 2015. End-to-end memory networks. *arXiv preprint arXiv:1503.08895*.

Kai Sheng Tai, Richard Socher, and Christopher D. Manning. 2015. Improved semantic representations from tree-structured long short-term memory networks. In *Proc. ACL*.

Duyu Tang, Furu Wei, Nan Yang, Ming Zhou, Ting Liu, and Bing Qin. 2014. Learning sentiment-specific word embedding for twitter sentiment classification. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*, volume 1, pages 1555–1565.

Duyu Tang, Bing Qin, and Ting Liu. 2015. 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.

Oriol Vinyals, Lukasz Kaiser, Terry Koo, Slav Petrov, Ilya Sutskever, and Geoffrey Hinton. 2014. Grammar as a foreign language. *arXiv preprint arXiv:1412.7449*.

Sida Wang and Christopher D Manning. 2012. Baselines and bigrams: Simple, good sentiment and topic classification. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Short Papers-Volume 2*, pages 90–94. Association for Computational Linguistics.

Kelvin Xu, Jimmy Ba, Ryan Kiros, Aaron Courville, Ruslan Salakhutdinov, Richard Zemel, and Yoshua Bengio. 2015. Show, attend and tell: Neural image caption generation with visual attention. *arXiv preprint arXiv:1502.03044*.

Zichao Yang, Xiaodong He, Jianfeng Gao, Li Deng, and Alex Smola. 2015. Stacked attention networks for image question answering. *arXiv preprint arXiv:1511.02274*.

Xiang Zhang, Junbo Zhao, and Yann LeCun. 2015. Character-level convolutional networks for text classification. *arXiv preprint arXiv:1509.01626*.

Chunting Zhou, Chonglin Sun, Zhiyuan Liu, and Francis Lau. 2015. A c-lstm neural network for text classification. *arXiv preprint arXiv:1511.08630*.