# Iterative Recursive Attention Model for Interpretable Sequence Classification

**Martin Tutek** and **Jan Šnajder**
Text Analysis and Knowledge Engineering Lab
Faculty of Electrical Engineering and Computing, University of Zagreb
Unska 3, 10000 Zagreb, Croatia
{martin.tutek,jan.snajder}@fer.hr

## Abstract

Natural language processing has greatly benefited from the introduction of the attention mechanism. However, standard attention models are of limited interpretability for tasks that involve a series of inference steps. We describe an iterative recursive attention model, which constructs incremental representations of input data through reusing results of previously computed queries. We train our model on sentiment classification datasets and demonstrate its capacity to identify and combine different aspects of the input in an easily interpretable manner, while obtaining performance close to the state of the art.

## 1 Introduction

The introduction of the attention mechanism (Bahdanau et al., 2014) offered a way to demystify the inference process of neural models. By assigning scalar weights to different elements of the input, we are able to visualize and potentially understand why the model made the decision it made, or discover a deficiency in the model by tracing down a relevant aspect of the input being overlooked by the model. Specifically in natural language processing (NLP), which abounds with variable-length word sequence classification tasks, attention alleviates the issue of learning long-term dependencies in recurrent neural networks (Bengio et al., 1994) by offering the model a glimpse into previously processed tokens.

Attention offers a good retrospective explanation of the classification decision by indicating what parts of the input contributed the most to the decision. However, in many cases the final decision is best interpreted as a result of a series of inference steps, each of which can potentially affect its polarity. A case in point is sentiment analysis, in which contrastive clauses and negations act as polarity switches of the overall sentence sentiment. In such cases, attention will only point to the part of the input sentence whose polarity matches that of the final decision. However, unfolding the inference process of a model into a series of interpretable steps would make the model more interpretable and allow one to identify its shortcomings.

As a step toward that goal, we propose an extension of the iterative attention mechanism (Sordoni et al., 2016), which we call the *iterative recursive attention model* (IRAM), where the result of an attentive query is nonlinearly transformed and then added to the set of vector representations of the input sequence. The nonlinear transformation, along with reusing the representations obtained in previous steps, allows the model to construct a recursive representation and process the input sequence bit by bit. The upshot is that we can inspect how the model weighs the different parts of the sentence and recursively combines them to give the final decision. We test the model on two sentiment analysis tasks and demonstrate its capacity to isolate different task-related aspects of the input, while reaching performance comparable with the state of the art.

## 2 Related Work

Attention (Bahdanau et al., 2014) and its variants (Luong et al., 2015) have initially been proposed for machine translation, but are now widely adopted in NLP. Attention has proven especially useful in tasks that involve long text sequences, such as summarization (Rush et al., 2015; See et al., 2017), question answering (Hermann et al., 2015; Xiong et al., 2016; Cui et al., 2017), and natural language inference (Rocktäschel et al., 2015; Yin et al., 2016; Parikh et al., 2016), as well as purely attentional machine translation (Vaswani et al., 2017; Gu et al., 2017).

Thus far, there has been a number of interesting and effective approaches for interpreting the in-

*Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 249–257
Brussels, Belgium, November 1, 2018. ©2018 Association for Computational Linguistics

ner workings of recurrent neural networks through methods such as representing them as finite automata (Weiss et al., 2017), extracting inference rules (Zanzotto and Ferrone, 2017), and analyzing saliency of inputs through first-order derivative information (Li et al., 2016; Arras et al., 2017).

Akin to the saliency analysis approaches, we opt not to condense the trained network into a finite set of rules. We differ from (Li et al., 2016; Arras et al., 2017) in that we attempt to decode the steps of the decision process of a recurrent network instead of demonstrating through saliency how the decision changes with respect to the inputs. In the context of sentiment analysis, the main benefit we see in representing the decision process of a recurrent network as a sequence of steps is that it offers a simple way to isolate sentiment-bearing phrases by observing how they get grouped in a single iteration. Secondly, we aim for improved interpretability of functional dependencies such as negation, where we demonstrate that our method first attends on the negated phrase, constructing an intermediate representation, which is then recursively transformed in the next iteration.

Sordoni et al. (2016) introduced the iterative attention mechanism for question answering, where attention alternates between the question and the document, and the query is updated in each step by a GRU cell (Cho et al., 2014). The model combines the weights obtained throughout the iterations to select the final answer, similar to the attention sum reader of Kadlec et al. (2016) and pointer networks of Vinyals et al. (2015).

We believe there is much to gain from the iterative attention mechanism by eliminating the direct link between the intermediate representations and the output, allowing the model to construct its own sequential representation of the input. Our model only connects the last attention step to the output, removing the need for intermediate steps to contain all the information relevant for the final decision. Apart from (Sordoni et al., 2016), related work closest to ours consists of concepts of multi-head attention (Lin et al., 2017; Vaswani et al., 2017), in which all queries are generated at once, pairwise attention (Cui et al., 2017; Xiong et al., 2016), where attention is applied to multiple inputs but is not applied iteratively and hierarchical iterative attention (Yang et al., 2016), where the authors first use a intra-sentence attention mechanism and then combine the intermediate representations with inter-sentence attention. In contrast to their work, we do not predetermine the level on which the attention is applied – in each iteration the mechanism can focus on any element of the input sequence.

## 3 Model

Throughout the experiments, we will use two variants of our model: (1) the vanilla model and (2) the full model. The vanilla model contains the bare minimum of components needed for the attention mechanism to function as intended. The purpose of the vanilla model is to eliminate any additional confounders for the performance and showcase the interpretability of the model. For the full model, we extend the vanilla model with additional deep learning components commonly employed in state-of-the-art models, to showcase the performance of the model when given capacity akin to competing models.

In both versions of the model, data is processed in three phases: (1) *encoding phase*, which contextualizes the word representations; (2) *attention phase*, which uses iterative recursive attention to isolate and combine the different parts of the input; and (3) *classification phase*, where the learned representation is fed as an input to a classifier.

The vanilla and full model differ only in the encoding phase, while our proposed attention mechanism is employed only in the second phase. We begin with a detailed account of the proposed attention mechanism and its regularization, and continue with a description of the remaining components, highlighting the differences between the vanilla and full models.

### 3.1 Iterative Recursive Attention

Fig. 1 shows the architecture of the iterative attention mechanism. The mechanism uses a recurrent network, dubbed the *controller*, to refine the attention query throughout $T$ iterations.

Inputs to the mechanism are an initial query $\hat{x}$ and a set of hidden states $H = [h_i, \ldots, h_N]$ constituting the input sequence, both obtained from the encoding step.

As the controller, we use a gated recurrent unit (GRU) (Cho et al., 2014) cell. The input to the controller is the transformed result of the previous query, while the hidden state is the previous query.

For the attention mechanism we use bilinear attention (Luong et al., 2015):

$$\vec{a} = softmax(\vec{q}WH) \tag{1}$$

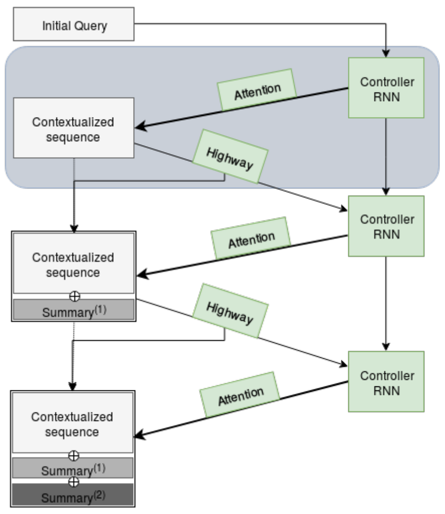


Figure 1: The iterative recursive attention model (IRAM). Green-colored components share their parameters with components of the same type. Highlighted in gray is one iteration of IRAM.

where $\vec{q}$ is the current query vector, $W$ a parameter of size $\mathbb{R}^{d_q \times d_h}$, while $d_q$ and $d_h$ are the dimensionalities of the query and the hidden state, respectively.

The attention weights are then used to compute the input summary in timestep $t$ as a linear combination of the hidden states:

$$\hat{s}^{(t)} = \sum_{i}^{N} a_i^{(t)} h_i \tag{2}$$

As we intend to use $\hat{s}^{(t)}$ in the next iteration of the attention mechanism, we need to allow the network the capacity to discern between the new additions and original inputs. To this end, we use a highway network (Srivastava et al., 2015), which gives the model the option to pass subsets of the summary as-is or transform them with a nonlinearity. If the summary is not transformed with a nonlinearity, it ends up being merely a linear combination of the hidden states, and we gain no information from adding it to the sequence.

The final input summary is thus obtained as $s^{(t)} = \mathit{Highway}(\hat{s}^{(t)})$ and added to the set of hidden states $H = \{h_i, \dots, h_n, s^{(1)}, \dots, s^{(t)}\}$.

## 3.2 Attention Regularization

Ideally, we want the model to focus on different task-related aspects of the input in each iteration. However, the model is in no way incentivized to learn to propagate information through the summaries and can in principle focus on the same segment in each step.

To prevent this from happening – and push the model to focus on different aspects of the input in every step – we regularize it by minimizing the pairwise dot products between all iterations of attention:

$$L_{attn} = \frac{\gamma}{2T} \sum_{i \neq j} [AA^T]_{ij} \tag{3}$$

where $\gamma$ is a hyperparameter determining the regularization strength and $A \in \mathbb{R}^{T \times N+T-1}$ is a matrix containing the attention weights generated in $T$ steps over $N$ inputs by the iterative attention mechanism. The matrix has $N + T - 1$ columns to account for attention over $T - 1$ added summaries, as the summary generated in the last iteration cannot be attended over. In each row $t$, the matrix has $T - t - 1$ trailing zeroes, corresponding to summaries that are not yet available in iteration $t$.

Concretely, the attention weight vector in row $t$ of the matrix $A$ consists of:

$$A_t = [\overbrace{a_1, \dots, a_N}^{\text{Input sequence}}, \overbrace{a_{N+1}, \dots, a_{N+t-1}}^{\text{Summaries in } t^- < t}, 0, \dots] \tag{4}$$

resulting in each element $i, j$ of the regularization matrix $AA^T$ storing the dot product between attention weights in iterations $i, j$. The regularization expression is a sum over all off-diagonal elements. The diagonal elements are dot products of attention weights in the same iteration so we ignore them. We scale by $\frac{1}{2}$ to account for the symmetrical elements in $A^T A$ and by $\frac{1}{T}$ to account for the number of dot product comparisons.

We note that, while this regularization penalty does encourage the model to focus on different elements of the input sequence, there is still a trivial way for the model to minimize the penalty without learning a meaningful behavior. Since the attention weight over the summary in iteration $t$ is zero in all iterations $t^- < t$, the model can simply attend over any elements of the input sequence in the first iteration, and afterwards propagate the information forward by fully attending only over the summary generated in the previous iteration. We will illustrate this behavior with concrete examples in Section 4.

## 3.3 Vanilla Encoder

For training, the inputs of the encoding phase are a sequence of words $x = [w_1, \dots, w_N]$ and a class

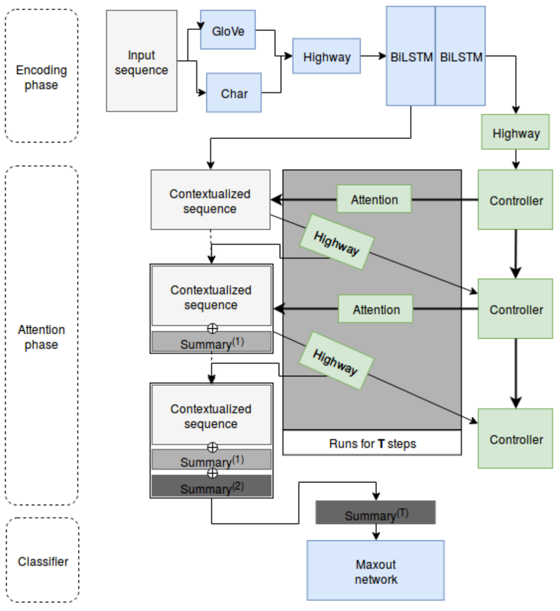


Figure 2: The full version of the iterative recursive attention model (IRAM). Green-colored components share their parameters with components of the same type; blue-colored components each have their own parameters.

label $y$. The encoder of the vanilla model maps the word indices to dense vector representations using pretrained GloVe vectors (Pennington et al., 2014). The sequence of word vectors is then fed as input to a bidirectional long-short term memory (BiLSTM) network (Hochreiter and Schmidhuber, 1997). The outputs of the BiLSTM are used as the input sequence to the iterative attention step, while the cell state in the last timestep is used as the initial query.

## 3.4 Full Encoder

There are three key differences between the full encoder and the vanilla encoder. The full encoder uses (1) character n-gram embeddings, (2) an additional highway network, whose task is to fine-tune the word embeddings, and (3) an additional layer of BiLSTM, followed by a highway layer to construct the initial query. For extensions (1) and (2), we took inspiration from McCann et al. (2017), who also use both components. However, unlike McCann et al. (2017), who used a ReLU feedforward network to fine-tune the embeddings for the task, we use a highway network, which we found performs better.

The pretrained character n-gram vectors obtained from (Hashimoto et al., 2016) are first averaged over all character n-grams for a given word and then concatenated to the GloVe embedding. Further on, before feeding the sequence of word embeddings to a recurrent model, we use a two-layer highway network (Srivastava et al., 2015) to fine-tune the embeddings for the task, which is especially beneficial when the input vectors are kept fixed.

To contextualize the input sequence and produce an initial attention query, we use a bidirectional long-short term memory (BiLSTM) network. We split the network conceptually into two parts: the lower $l_{ctx}$ layers are used to transform the input sequence of word embeddings into a sequence of contextualized word representations, while the upper $l_{query}$ layers are used to read and comprehend the now-transformed sequence and capture its relevant aspects into a single vector. The rationale for the split is that recurrent networks are often required to tackle two tasks at once: contextualize the input and comprehend the whole sequence. Intuitively, the split should incite a division of labor between the two parts of the network: contextualization network only has to memorize the local information specific to each word (e.g., verb tense, noun gender) in order to transform its representation, while comprehension network needs to model aspects of meaning pertaining to the entire sequence (e.g., the overall sentiment of the sentence, locations of sentiment bearing phrases).

We use a single $(l_{ctx} + l_{query})$-layered BiLSTM, where we use the output of the $l_{ctx}$-th layer, while we use the cell state from the last layer as the sequence representation $\hat{x}$.

Lastly, since the weights of the BiLSTM network are suited toward processing the input sequence rather than preparing the query vector, we add an additional highway layer designed to fine-tune the sentence representation into the initial query.

## 3.5 Classifier

As input to the classifier, we use the summary vector obtained from the last step of iterative attention $s^{(T)}$. This way we force the network to propagate information through the attention steps, and also because the intermediate summaries do not contribute directly toward the classification and hence need not have the same polarity. The last summary vector is fed into a maxout network (Goodfellow et al., 2013) to obtain the class-conditional probabilities.

Fig. 2 shows the full version of the iterative attention mechanism with all of the aforementioned components.

## 4 Experiments

### 4.1 Datasets

We test IRAM on two sentiment classification datasets. The first is the Stanford Sentiment Treebank (SST) (Socher et al., 2013), a dataset derived from movie reviews on Rotten Tomatoes and containing 11,855 sentences labeled into five classes at the sentence level and at the level of each node in the constituency parse tree. The binary version with the neutral class removed contains 56,400 instances, while the fine-grained version with scores ranging from 1 (very negative) to 5 (very positive) contains 94,200 text-sentiment pairs. The second dataset is the Internet Movie Database (IMDb) (Maas et al., 2011), containing 22,500 multi-sentence reviews extracted from positive and negative reviews. We truncate each sentence from this dataset to a maximum length of 200 tokens.

Firstly, we demonstrate and analyze how each component in the vanilla model contributes to the performance and interpretability. We then analyze the full model and evaluate it on the aforementioned datasets.

### 4.2 Experimental Setup

Unless stated otherwise, all weights are initialized from a Gaussian distribution with zero mean and standard deviation of $0.01$. We use the Adam optimizer (Kingma and Ba, 2014) with the AmsGrad modification (Reddi et al., 2018) and $\alpha = 0.0003$. We clip the global norm of the gradients to $1.0$ and set weight decay to $0.00003$.

We use 300-dimensional GloVe word embeddings trained on the Common Crawl corpus and 100-dimensional character embeddings. We follow the recommendation of Mu et al. (2017) and standardize the embeddings. Dropout of $0.1$ is applied to the word embedding matrix.

For both datasets, we set $l_{ctx} = 2$ and $l_{query} = 1$. The highway network for fine-tuning the input embeddings has two layers, while the ones fine-tuning the query and the summary have a single layer. All highway networks' gate biases are initialized to 1, as recommended by Srivastava et al. (2015), as well as the biases of the LSTM forget gates.

The maxout network uses two 200-dimensional layers with a pool size of $4$.

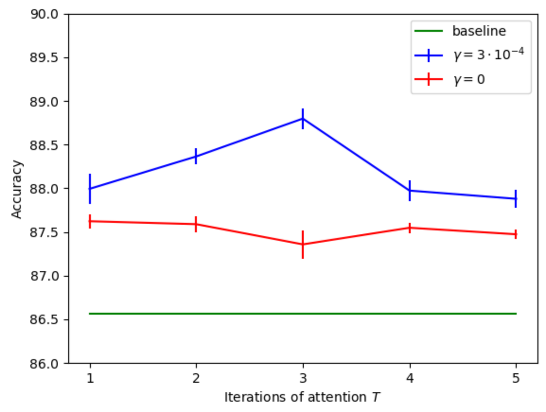


Figure 3: The effect of regularization $\gamma$ across different values of $T$

Throughout our experiments, we have experimented with selecting the batch size from $\{32, 64\}$, dropout for the recurrent layers and the maxout classifier from $\{0.1, 0.2, 0.3, 0.4\}$, and the LSTM hidden state size from $\{400, 500, 1000\}$. The word and character n-gram vectors are kept fixed for SST but are learned for the IMDb dataset. These parameters are optimized using cross-validation, and the best configuration is ran on the test set. As IMDb has no official validation set, we randomly select $10\%$ of the dataset and use it for all of the experiments. The values of other hyperparameters were selected through inexhaustive search.

### 4.3 Analysis of the Vanilla Model

The vanilla model defined in Section 3.3 has two main confounding variables: strength (and presence) of attention regularization ($\gamma$) and the number of iterations of the iterative recursive attention mechanism ($T$). We also would like to examine the difference in performance of the vanilla IRAM compared to some baseline sequence classifier. To this end, we implement a baseline model without the attention mechanism – a maxout classifier over the last hidden state of the encoder BiLSTM. To keep the running time of the experiments feasible, in this section we use only the binary SST dataset.

**Effect of regularization.** For each experiment in this round, we run every model three times with different random seeds and report the average results along with the standard deviations across the experiments. In Fig. 3 we present the comparison between the performance of the vanilla model with and without regularization. A more telling sign of

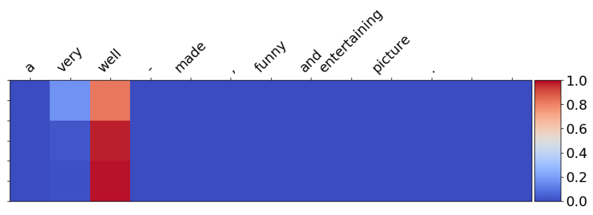


Figure 4: Attention sample for $\gamma = 0$ and $T = 3$

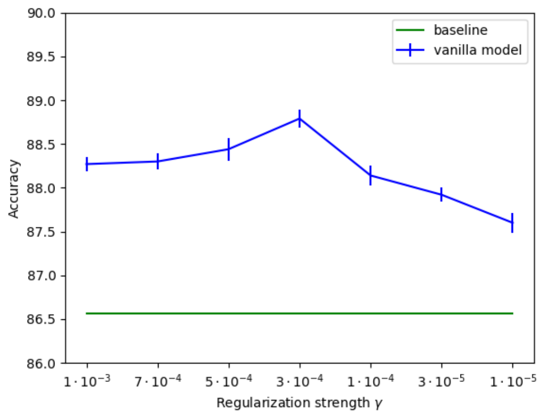


Figure 5: Classification accuracy for different values of $\gamma$

the different behavior between the models can be seen through inspecting attention weights.

In Fig. 4 we can see that the attention mechanism, when not regularized, fails to use its capacity and simply attends over the same element in each timestep. The last two columns, which contain the summaries from the first two steps of the iterative attention mechanism, have an attention weight of 0, which means that the model does not pass any information through the summaries nor refine the query. This behavior initially prompted us to add the regularization penalty term.

Through inexhaustive search we isolated a critical range of values for $\gamma$, for which we perform a detailed analysis of performance. For this experiment, we fix $T = 3$ as it has exhibited better performance for the vanilla model.

**Effect of the number of iterations.** Apart from comparing the effect of the existence of regularization, in Fig. 3 we can also observe the effect of the number of timesteps $T$. Increasing $T$ beyond 3 has a diminishing effect on classification performance, something which we find to be consistent for the IMDB dataset as well.

We attribute this decrease in performance to the fact that SST is relatively simple, containing at most two contrastive aspects in each sentence, making any additional steps unnecessary. While the model could in theory exploit the pass-through mechanism, we believe that this operation adds some noise to the final representations and in turn affects performance slightly.

| SST | |
|---|---|
| NSE (Munkhdalai and Yu, 2017) | 89.7 |
| **IRAM** | 90.1 |
| BCN + CoVe (McCann et al., 2017) | 90.3 |
| bmLSTM (Radford et al., 2017) | **91.8** |
| SST-5 | |
| **IRAM** | 53.7 |
| BCN + CoVe (McCann et al., 2017) | 53.7 |
| BCN + ELMo (Peters et al., 2018) | **54.7** |
| IMDb | |
| **IRAM** | 91.2 |
| TRNN (Dieng et al., 2016) | 93.8 |
| oh-LSTM (Johnson and Zhang, 2016) | **94.1** |
| Virtual (Miyato et al., 2016) | **94.1** |

Table 1: Classification accuracy on the test sets

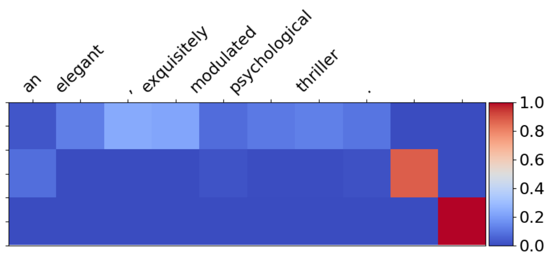
| **Removed component** | **Accuracy** |
|---|---|
| Full model | **90.1** |
| Vanilla model | 88.7 |
| – char n-grams | 89.3 |
| – query fine-tune | 89.8 |
| – embedding fine-tune | 89.3 |

Table 2: Effect of removing components on performance

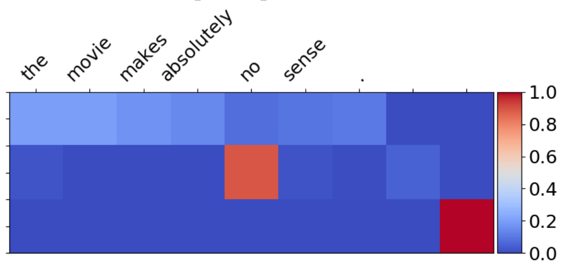
## 4.4 Analysis of the Full Model

We now evaluate the full model. Table 1 shows the accuracy scores of our best models (for $T = 3$, $\gamma = 0.0003$) and other state-of-the-art models on the test portions of the SST and IMDb datasets. Our model performs competitively with the best results on SST and SST-5 datasets. It is important to note that our model does not use transfer learning apart from the pretrained word vectors, which is not the case for the competing models.

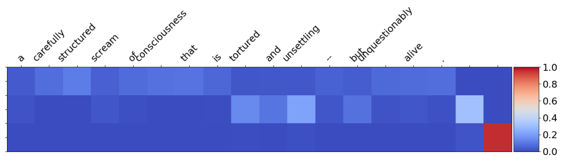
**Ablation study of encoder components.** As mentioned in Section 3.4, through adding various components to the model we introduced a number of confounders. In order to determine the effect of each of the added components on the overall score, we evaluate the performance of the full model on the binary SST dataset with the remaining hyperparameters fixed and one of the components removed in isolation.



(a) Simple unipolar sentence



(b) Sentence with a negation



(c) Contrastive multipolar sentence

Figure 6: Visualization of attention across sentence words (horizontal) and $T$=3 time steps (vertical). The last $T$-1 columns contain the attention weights over the result of the previous attentive query.

### 4.5 Visualizing Attention

To gain an intuition about the working of IRAM, we visually analyzed its attention mechanism on a number of sentences from our dataset. We limit ourselves to examples from the test set of the SST dataset as the length of examples is manageable for visualization. We isolate three specific cases where the attention mechanism demonstrates interesting results: (1) simple unipolar sentences, (2) sentences with negations, and (3) multipolar sentences.

The least interesting case is the unipolar, as the attention mechanism often does not need multiple iterations. Fig. 6a shows the attention mechanism simply propagating information, since sentiment classification is straightforward and does not require multiple attention steps. This can be seen from most of the attention weight in the second and third steps being on the columns corresponding to the summaries.

The more interesting cases are sentences involving negations and modifiers. Fig. 6b shows the handling of negation: attention is initially on all words except on the negator. In the second step, the mechanism combines the output of the first step with the negation. We interpret this as flipping the sentiment – the model cannot rely solely on recognizing a negative word, and has to account for what that word negates through a functional dependence. These examples highlight one of the drawbacks of recurrent networks which we aim to alleviate. In case a standard attention mechanism is applied to a sentence containing a negator, the hidden representation of the negator has to scale or negate the intensity of an expression. Our model has the capacity to process such sequences iteratively, first constructing the representation of an expression, which is then adjusted by the nonlinear transformation and simpler to combine with the negator in the next step.

Lastly, Fig. 6c shows a contrastive multipolar sentence, where the model in the first step focuses on positive words, and then combines the negative words (*tortured*, *unsettling*) with the results of the first step. In such cases, the model succeeds to isolate the contrasting aspects of the sentence and attends to them in different iterations of the model, alleviating the burden of simultaneously representing the positive and negative aspects. After both contrastive representations have been formed, the model has the capacity to *weigh* them one against other and compute the final representation.

## 5 Conclusion

The proposed iterative recursive attention model (IRAM) has the capacity to construct representations of the input sequence in a recursive fashion, making inference more interpretable. We demonstrated that the model can learn to focus on various task-relevant parts of the input, and can propagate the information in a meaningful way to handle the more difficult cases. On the sentiment analysis task, the model performs comparable to the state of the art. Our next goals will be to try to use the iterative attention mechanism to extract tree-like sentence structures akin to constituency parse trees, evaluate the model on more complex datasets as well as extend the model to support an adaptive number of iterative steps.

## Acknowledgment

This research has been supported by the European Regional Development Fund under the grant KK.01.1.1.01.0009 (DATACROSS).

## References

Leila Arras, Grégoire Montavon, Klaus-Robert Müller, and Wojciech Samek. 2017. Explaining recurrent neural network predictions in sentiment analysis. In *Proceedings of the 8th Workshop on Computational Approaches to Subjectivity, Sentiment and Social Media Analysis*, pages 159–168.

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

Yoshua Bengio, Patrice Simard, and Paolo Frasconi. 1994. Learning long-term dependencies with gradient descent is difficult. *IEEE transactions on neural networks*, 5(2):157–166.

Kyunghyun Cho, Bart van Merriënboer, Dzmitry Bahdanau, and Yoshua Bengio. 2014. On the properties of neural machine translation: Encoder–decoder approaches. *Syntax, Semantics and Structure in Statistical Translation*, page 103.

Yiming Cui, Zhipeng Chen, Si Wei, Shijin Wang, Ting Liu, and Guoping Hu. 2017. Attention-over-attention neural networks for reading comprehension. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 593–602.

Adji B Dieng, Chong Wang, Jianfeng Gao, and John Paisley. 2016. TopicRNN: A recurrent neural network with long-range semantic dependency. In *ICLR 2016*.

Ian J Goodfellow, David Warde-Farley, Mehdi Mirza, Aaron Courville, and Yoshua Bengio. 2013. Maxout networks. *arXiv preprint arXiv:1302.4389*.

Jiatao Gu, James Bradbury, Caiming Xiong, Victor OK Li, and Richard Socher. 2017. Non-autoregressive neural machine translation. *arXiv preprint arXiv:1711.02281*.

Kazuma Hashimoto, Caiming Xiong, Yoshimasa Tsuruoka, and Richard Socher. 2016. A joint many-task model: Growing a neural network for multiple nlp tasks. *arXiv preprint arXiv:1611.01587*.

Karl Moritz Hermann, Tomas Kocisky, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching machines to read and comprehend. In *Advances in Neural Information Processing Systems*, pages 1693–1701.

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

Rie Johnson and Tong Zhang. 2016. Supervised and semi-supervised text categorization using lstm for region embeddings. In *International Conference on Machine Learning*, pages 526–534.

Rudolf Kadlec, Martin Schmid, Ondrej Bajgar, and Jan Kleindienst. 2016. Text understanding with the attention sum reader network. *arXiv preprint arXiv:1603.01547*.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Jiwei Li, Xinlei Chen, Eduard Hovy, and Dan Jurafsky. 2016. Visualizing and understanding neural models in nlp. In *Proceedings of NAACL-HLT*, pages 681–691.

Zhouhan Lin, Minwei Feng, Cicero Nogueira dos Santos, Mo Yu, Bing Xiang, Bowen Zhou, and Yoshua Bengio. 2017. A structured self-attentive sentence embedding. *arXiv preprint arXiv:1703.03130*.

Thang Luong, Hieu Pham, and Christopher D Manning. 2015. Effective approaches to attention-based neural machine translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1412–1421.

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.

Bryan McCann, James Bradbury, Caiming Xiong, and Richard Socher. 2017. Learned in translation: Contextualized word vectors. In *Advances in Neural Information Processing Systems*, pages 6297–6308.

Takeru Miyato, Andrew M Dai, and Ian Goodfellow. 2016. Adversarial training methods for semi-supervised text classification. *arXiv preprint arXiv:1605.07725*.

Jiaqi Mu, Suma Bhat, and Pramod Viswanath. 2017. All-but-the-top: simple and effective postprocessing for word representations. *arXiv preprint arXiv:1702.01417*.

Tsendsuren Munkhdalai and Hong Yu. 2017. Neural semantic encoders. In *Proceedings of the conference. Association for Computational Linguistics. Meeting*, volume 1, page 397. NIH Public Access.

Ankur P Parikh, Oscar Täckström, Dipanjan Das, and Jakob Uszkoreit. 2016. A decomposable attention model for natural language inference. *arXiv preprint arXiv:1606.01933*.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. GloVe: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.

Matthew E Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. *arXiv preprint arXiv:1802.05365*.

Alec Radford, Rafal Jozefowicz, and Ilya Sutskever. 2017. Learning to generate reviews and discovering sentiment. *arXiv preprint arXiv:1704.01444*.

Sashank J Reddi, Satyen Kale, and Sanjiv Kumar. 2018. On the convergence of Adam and beyond. In *International Conference on Learning Representations*.

Tim Rocktäschel, Edward Grefenstette, Karl Moritz Hermann, Tomáš Kočiský, and Phil Blunsom. 2015. Reasoning about entailment with neural attention. *arXiv preprint arXiv:1509.06664*.

Alexander M Rush, Sumit Chopra, and Jason Weston. 2015. A neural attention model for abstractive sentence summarization. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 379–389.

Abigail See, Peter J Liu, and Christopher D Manning. 2017. Get to the point: Summarization with pointer-generator networks. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 1073–1083.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D Manning, Andrew Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 conference on empirical methods in natural language processing*, pages 1631–1642.

Alessandro Sordoni, Philip Bachman, Adam Trischler, and Yoshua Bengio. 2016. Iterative alternating neural attention for machine reading. *arXiv preprint arXiv:1606.02245*.

Rupesh Kumar Srivastava, Klaus Greff, and Jürgen Schmidhuber. 2015. Highway networks. *arXiv preprint arXiv:1505.00387*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, pages 6000–6010.

Oriol Vinyals, Meire Fortunato, and Navdeep Jaitly. 2015. Pointer networks. In *Advances in Neural Information Processing Systems*, pages 2692–2700.

Gail Weiss, Yoav Goldberg, and Eran Yahav. 2017. Extracting automata from recurrent neural networks using queries and counterexamples. *arXiv preprint arXiv:1711.09576*.

Caiming Xiong, Victor Zhong, and Richard Socher. 2016. Dynamic coattention networks for question answering. *arXiv preprint arXiv:1611.01604*.

Zichao Yang, Diyi Yang, Chris Dyer, Xiaodong He, Alex Smola, and Eduard Hovy. 2016. 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.

Wenpeng Yin, Hinrich Schütze, Bing Xiang, and Bowen Zhou. 2016. ABCNN: Attention-based convolutional neural network for modeling sentence pairs. *Transactions of the Association of Computational Linguistics*, 4(1):259–272.

Fabio Massimo Zanzotto and Lorenzo Ferrone. 2017. Can we explain natural language inference decisions taken with neural networks? inference rules in distributed representations. In *Neural Networks (IJCNN), 2017 International Joint Conference on*, pages 3680–3687. IEEE.