DoLFIn: Distributions over Latent Features for Interpretability

Phong Le[*]
Independent Researcher
lephong.xyz@gmail.com

Willem Zuidema
University of Amsterdam
zuidema@uva.nl

Abstract

Interpreting the inner workings of neural models is a key step in ensuring the robustness and trustworthiness of the models, but work on neural network interpretability typically faces a trade-off: either the models are too constrained to be very useful, or the solutions found by the models are too complex to interpret. We propose a novel strategy for achieving interpretability that – in our experiments – avoids this trade-off. Our approach builds on the success of using probability as the central quantity, such as for instance within the attention mechanism. In our architecture, DoLFIn (Distributions over Latent Features for Interpretability), we do no determine beforehand what each feature represents, and features go altogether into an unordered set. Each feature has an associated probability ranging from 0 to 1, weighing its importance for further processing. We show that, unlike attention and saliency map approaches, this set-up makes it straight-forward to compute the probability with which an input component supports the decision the neural model makes. To demonstrate the usefulness of the approach, we apply DoLFIn to text classification, and show that DoLFIn not only provides interpretable solutions, but even slightly outperforms the classical CNN and BiLSTM text classifiers on the SST2 and AG-news datasets.

1 Introduction

Having insights into how a trained neural network solves a given task is important, in order to build more robust, trustworthy, and accurate models (Zhou et al., 2016; Gilpin et al., 2018; Poerner et al., 2018; Belinkov et al., 2017). However gaining insights is often challenging because of the large number of parameters and the nonlinear dependencies between components of the model. One approach to 'opening the blackbox' is to use saliency maps (Jacovi et al., 2018; Gupta and Schütze, 2018), to examine which input components (e.g., words) are taken into account. Nevertheless salience alone does not tell us how exactly salient components contribute to the decision of a model. For instance, a sentiment-analysis model might mark both "boring" and "wonderful" in "This film would be wonderful, if the beginning wasn't so boring" as salient, but saliency scores do not reveal the crucial interaction between all the words in the sentence that ultimately let it express a negative sentiment. Alternatively, one can analyse the flow of information through the network, by tracking (relevance) gradients backwards (Arras et al., 2017) or decomposing (forward) contributions to all intermediate quantities (Murdoch et al., 2018; Jumelet et al., 2019). These methods address some of the problems of saliency maps, but produce results that themselves require further interpretation. For instance, such a method might quantify the relative contribution of the 3rd word when processing the 7th word in the 2nd layer of a multilayer LSTM. That quantity, however, does not easily translate to an explanation for the final prediction that the model generates.

A third approach looks at the attention mechanism (Bahdanau et al., 2014), which regulates which components a model attends to. Visualising attended components is straight-forward (Xu et al., 2015; Abnar and Zuidema, 2020) as an attention weight is the probability that the corresponding component is taken into account for further computation. At a higher level, as in (Lei et al., 2016; Bastings et al., 2019),

[*]Now at Amazon, Cambridge, UK.

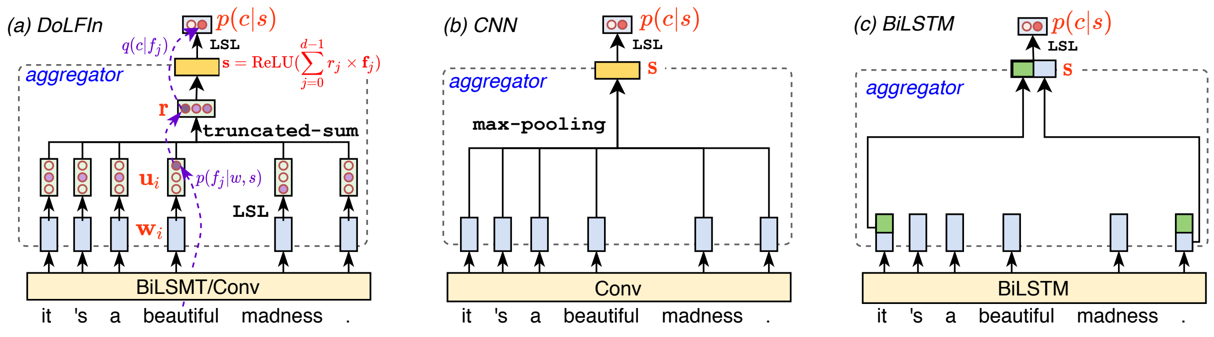


Figure 1: Text classification models. (a) DoLFIn with Bags of latent features (BoLF), using linear-softmax layers (LSL). The whole input text s is mapped to $\mathbf{r}$ which is a bag of latent features. $p(f_i|w,s)$ is the probability of mapping word w (in text s) to latent feature f_i, and $q(c|f_i)$ is the probability that f_i supports category c. (b) Traditional CNN proposed by (Kim, 2014). (c) BiLSTM.

one can compute rationales which are attended pieces of text. However, these approaches face some of the same problems as saliency maps; e.g., attention alone does not indicate how much an input component supports a category.

In this paper we focus, like the attention-based approach, on probability. We start from the observation that probability is a well studied mathematical tool, that has already been much used in building explainable and robust neural networks (although we also note that simply turning interpretation quantities from other approaches into probabilities by normalizing is not necessarily helpful, as these probabilities are not faithful to the true behaviour of the model). We propose DoLFIn (distributions over latent features for interpretability) based on probability; our architecture maps an input (e.g., a text) to a *bag of latent features* (BoLF, see Figure 1a and §3) so that interpretation quantities are probabilistic. In other words, the representation of an input is a vector whose elements range from 0 to 1, indicating to what extent a latent feature is in the bag. To do so, we employ linear-softmax layers (i.e., neural layers with softmax activation, LSL for short) to map input components to distributions over latent features, and a truncated sum to aggregate the resulting distributions. With this new type of representations, we can easily compute $q(c|w,s)$, the probability that input component w in context s supports category c, by decomposing the term to $p(f|w,s)$, the distribution over latent features, and $q(c|f)$, the probability that feature f supports category c. The former is given by an LSL and the latter can be estimated by the input-output statistics.

To illustrate the feasibility and benefits of this idea, we employ DoLFIn for text classification that, going beyond saliency maps and attention, can tell us the probability a word supports a category. We demonstrate that DoLFIn does not trade off interpretability against classification accuracy. We build DoLFIn-conv and DoLFIn-bilstm which are variants of the classical CNN proposed by (Kim, 2014) and BiLSTM text classifier (Figure 1b,c). Carrying out experiments on three popular datasets, TREC Question, SST2, and AG-news, we find that DoLFIn achieves slightly higher accuracy than CNN and BiLSTM on SST2 and AG-news. It is worth noting that, although used for text classification in this paper, DoLFIn is applicable to a wide range of classification tasks such as natural language inference and relation prediction.

2 Text Classification Baselines

Given a text of n words $s = (w_0, ..., w_{n-1})$ and a set of m categories $C = \{c_0, ..., c_{m-1}\}$, a probabilistic text classifier is $p(c|s)$, which assigns a probability to the prediction that c is the category of s. A traditional text classification architecture adopts the diagram

$$s \rightarrow \boxed{\text{encoder}} \xrightarrow{\mathbf{s}} \boxed{\text{classifier}} \rightarrow p(c|s)$$

where $\mathbf{s} \in \mathbb{R}^{d_s}$ is a vector representing text s. The classifier module is often a linear-softmax layer.

Shown in Figure 1b, a classical CNN text classifier (Kim, 2014) utilises a convolutional layer to map each word w_i (with its context) to a vector $\mathbf{w}_i \in \mathbb{R}^{d_w}$. It then uses max pooling over $\{\mathbf{w}_i\}_{i=0}^{n-1}$ to compute $\mathbf{s}$ for text s. A BiLSTM text classifier (Figure 1c) uses an BiLSTM to compute $\mathbf{w}_i$, and an aggregator concatenating the backward part of $\mathbf{w}_0$ and the forward part of $\mathbf{w}_{n-1}$ to form $\mathbf{s}$.

Training a text classifier is to minimise the cross-entropy loss

$$L(\theta) = -\frac{1}{|\mathcal{D}_{\text{train}}|} \sum_{(s,c)\in\mathcal{D}_{\text{train}}} \log p(c|s;\theta)$$

where θ is the parameter vector and $\mathcal{D}_{\text{train}}$ is a training set of s, c pairs.

3 DoLFIn with Bags of latent features (BoLF)

For simplicity, we introduce DoLFIn as a neural text classifier, depicted in Figure 1a, but DoLFIn should be applicable to several classification tasks.

The key part of DoLFIn, BoLF, is an aggregator consisting of linear-softmax layers (LSL) and a truncated sum. This aggregator firstly maps each $\mathbf{w}_i$ to a distribution $\mathbf{u}_i$ over d latent features using LSLs, i.e., $\mathbf{u}_i = \text{LSL}(\mathbf{w}_i)$, so that $\mathbf{u}_{i,j} = p(f_j|w_i, s)$. It then uses the truncated sum to compute

$$\mathbf{r} = \min(\mathbf{1}, \sum_{i=0}^{n-1} \mathbf{u}_i) \in \mathbb{R}^d$$

We apply the element-wise $\min$ operator so that the j-th entry of $\mathbf{r}$, i.e., $r_j \in [0, 1]$, can be considered as a soft indicator of the extent to which the j-th latent feature f_j contributes to the final output. Intuitively, we represent s by a *bag of latent features* $\mathbf{r}$. If r_j is close to 1, feature f_j likely appears in the bag.

Finally, we compute

$$\mathbf{s} = \text{ReLU}(\sum_{j=0}^{d-1} r_j \times \mathbf{f}_j)$$

to represent s, where each feature f_j is represented by a vector $\mathbf{f}_j \in \mathbb{R}^{d_s}$. The sum inside the brackets can be seen as a bag of the latent feature representations.

Interpretability

We now show how to analyse the impact of each input component w in context s to the classification decision of the model. To do so, we examine the probability $q(c|w, s)$, which can be seen as the support of w in context s for category c. We decompose this probability by (see the purple arrows in Figure 1):

$$q(c|w, s) = \sum_{j=0}^{d-1} q(c|f_j)p(f_j|w, s) \tag{1}$$

where $p(f_j|w, s)$ are given by the used LSLs as mentioned above. (Note that, because we aggregate $p(f_j|w, s)\ \forall j$ into $\mathbf{r}$, we do not need to take $\mathbf{r}$ into this equation.)

Because directly computing $q(c|f)$ is not trivial, we approximate it using the statistics of the model's input-output. Recall that the latent features from s's words are aggregated into $\mathbf{r}$, so that if r_j is close to 1, f_j is on and used to make the prediction. For simplicity, we assume that f_j is on when $r_j > \delta$ for $\delta \in [0, 1]$. Let S be a large set of unlabelled texts. Let $\text{count}_S(c, f_j)$ be the number of texts $s \in S$, whose $r_j > \delta$ and which are assigned to category c by the model. Then

$$q(c|f_j) \approx \frac{\text{count}_S(c, f_j)}{\sum_{c'} \text{count}_S(c', f_j)}$$

Intuitively, we take into account how many times feature f_j appears to yield the prediction c. Consequently, the closer $q(c|f_j)$ is to 1, the more likely f_j supports category c. Besides, if $q(.|f_j)$ is close to uniform, f_j is not helpful for classification. In our experiments, we set $\delta = 0.5$ and S the texts of dev sets.

4 Experiments

In our experimental evaluation we investigate whether DoLFIn can indeed produce interpretable solutions, without sacrificing accuracy. Our implementation is in Python with Pytorch (Paszke et al., 2019). The source code and data are available at `https://github.com/lephong/dolfin`. Extra information is provided in the appendix.

Datasets	#c	Train	Dev	Test	AvgL
TREC	6	5k	452	500	7.5
SST2	2	76.9k	872	1821	19.2
AG-news	4	110k	10k	7.6k	42.3

	Conv		BiLSTM	
	CNN	DoLFIn-conv	BiLSTM	DoLFIn-bilstm
TREC	**92.44** ±0.55	92.10 ±0.70	**92.44** ±0.59	91.72 ±0.76
SST2	85.03 ±0.43	**85.90** ±0.57	86.44 ±0.73	**87.23** ±0.66
AG-news	92.17 ±0.19	**92.59** ±0.17	93.26 ±0.11	**93.36** ±0.19

Table 1: *Left* - The statistics of TREC, SST2, and AG-news datasets. #c is the number of categories, AvgL is the average length (in words) of test texts. *Right* - Accuracy (%) of the four models on TREC, SST2, and AG-news. We show the mean and standard deviation across five runs.

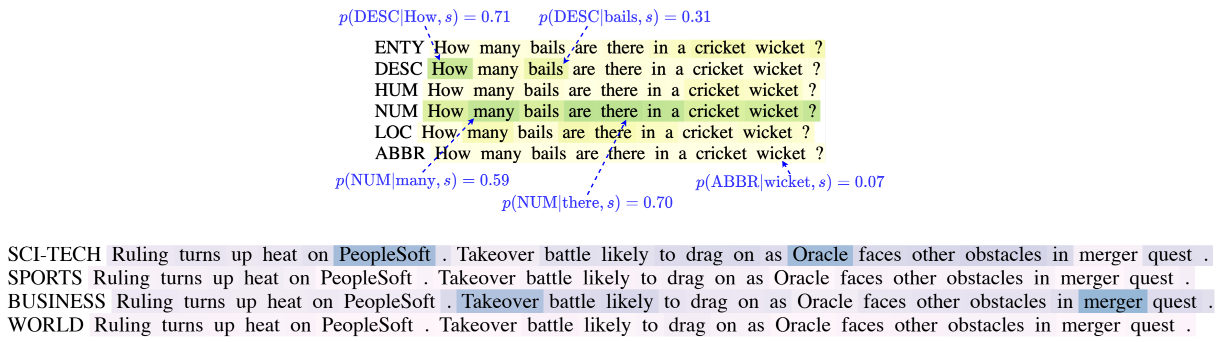

$p(\text{DESC}|\text{How}, s) = 0.71$
$p(\text{DESC}|\text{bails}, s) = 0.31$
ENTY How many bails are there in a cricket wicket ?
DESC How many bails are there in a cricket wicket ?
HUM How many bails are there in a cricket wicket ?
NUM How many bails are there in a cricket wicket ?
LOC How many bails are there in a cricket wicket ?
ABBR How many bails are there in a cricket wicket ?
$p(\text{NUM}|\text{many}, s) = 0.59$
$p(\text{NUM}|\text{there}, s) = 0.70$
$p(\text{ABBR}|\text{wicket}, s) = 0.07$
SCI-TECH Ruling turns up heat on PeopleSoft . Takeover battle likely to drag on as Oracle faces other obstacles in merger quest .
SPORTS Ruling turns up heat on PeopleSoft . Takeover battle likely to drag on as Oracle faces other obstacles in merger quest .
BUSINESS Ruling turns up heat on PeopleSoft . Takeover battle likely to drag on as Oracle faces other obstacles in merger quest .
WORLD Ruling turns up heat on PeopleSoft . Takeover battle likely to drag on as Oracle faces other obstacles in merger quest .

Figure 2: *Top* - A TREC question. Each word is highlighted according to $q(c|w, s)$. For example, "How" is highlighted more than "bails" in the second question because $q(\text{DESC}|\text{How}, s) > q(\text{DESC}|\text{bails}, s)$. The left-most token on each line is the label of category c (e.g. the second question is assigned to category DESC). *Bottom* - A text in AG-news.

Dataset We used three following text classification datasets, whose statistics are given in Table 1-left.

- TREC Question (Li and Roth, 2002) (TREC for short) is for classifying questions into six categories: ABBR (Abbreviation), DECS (Description), ENTY (Entity), HUM (Human), LOC (Location), and NUM (Number). The questions are generally short (7.5 words, on average), such as "Who are cartoondom 's Super Six ?"

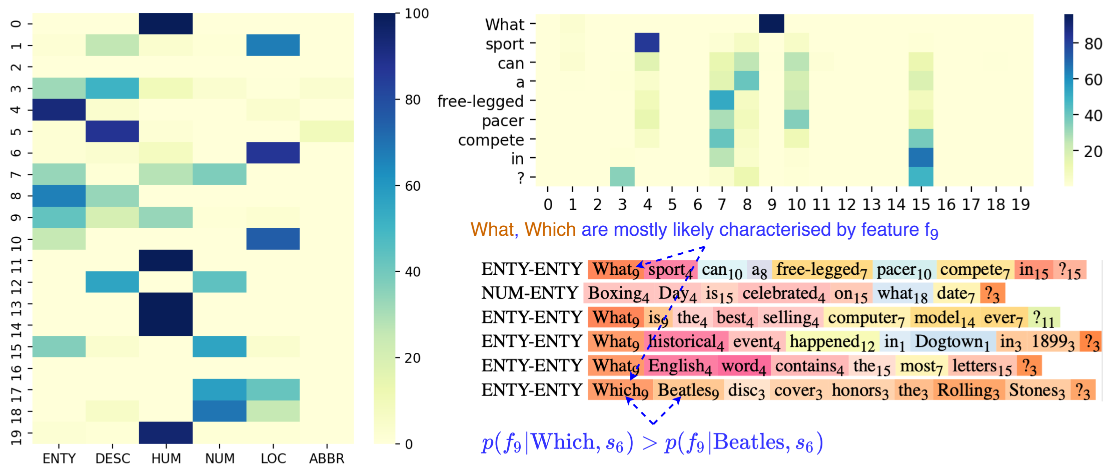
- SST2 (Socher et al., 2013) is for predicting the binary sentiment (positive/negative) of movie reviews. Different from the dev and test sets, the train set contains labelled phrases and sentences, rather than sentences alone.

- AG-news (Zhang et al., 2015) is a news topic classification dataset with four topics: WORLD, SPORTS, BUSINESS, and SCI-TECH. Among the three datasets, AG-news is the largest in terms of the number of texts and the average length.

Models We evaluated four models CNN, BiLSTM, and DoLFIn-conv/bilstm. Most of their hyper-parameters are identical to those used in (Kim, 2014) (see Appendix A). The number of latent features d is 20, 10, and 100 for DoLFIn when tested on TREC, SST2, and AG-news respectively. We used Glove word-embeddings (Pennington et al., 2014) and Adam optimizer (Kingma and Ba, 2014) with the default learning rate 0.001.

Results Table 1-right shows the results. Although DoLFIn performs worse than CNN and BiLSTM on TREC, it slightly outperforms them on SST2 and AG-news. These results suggest that using DoLFIn does not sacrifice the classification accuracy.

Interpretation To illustrate how to interpret DoLFIn, in Figure 2-top we visualise a TREC question in the dev set, where the weight for a word is $q(c|w, s)$, given by Eq. (1). If a word (in a context) supports the category in question, it will be highlighted. For instance, when considering category DESC, we can see that word "How" supports it strongly, "bails" slightly, and the other words do not. For NUM, "many" and "are there in" have high $q(\text{NUM}|w, s)$. DoLFIn correctly chose NUM. Figure 2-bottom shows a text in AG-news. DoLFIn reasonably focused on "PeopleSoft" and "Oracle" for SCI-TECH, and "Takeover"



ENTY-ENTY What$_9$ sport$_4$ can$_{10}$ a$_8$ free-legged$_7$ pacer$_{10}$ compete$_7$ in$_{15}$?$_{15}$
NUM-ENTY Boxing$_4$ Day$_4$ is$_{15}$ celebrated$_4$ on$_{15}$ what$_{18}$ date$_7$?$_3$
ENTY-ENTY What$_9$ is$_9$ the$_4$ best$_4$ selling$_4$ computer$_7$ model$_{14}$ ever$_7$?$_{11}$
ENTY-ENTY What$_9$ historical$_4$ event$_4$ happened$_{12}$ in$_1$ Dogtown$_1$ in$_3$ 1899$_3$?$_3$
ENTY-ENTY What$_9$ English$_4$ word$_4$ contains$_4$ the$_{15}$ most$_7$ letters$_{15}$?$_3$
ENTY-ENTY Which$_9$ Beatles$_9$ disc$_3$ cover$_3$ honors$_3$ the$_3$ Rolling$_3$ Stones$_3$?$_3$

Figure 3: *Left* - Heatmap of $q(c|f)$ with 20 latent features for TREC. The (j+1)-th row indicates $q(c|f_j)$ (%). For example, feature f_9 supports categories ENTY, DESC, and HUM. *Right-top* - Heatmap of $q(f|w, s)$ for A TREC question. "What" in this sentence is mostly likely characterised by feature f_9, whereas "a" can be f_7, f_8, f_{10}, or f_{15}. *Right-bottom* - TREC questions predicted as ENTY. Each word w has a subscript $j = \arg\max_k p(f_k|w, s)$, and is highlighted according to $p(f_j|w, s)$. The left-most token on each line is the gold label-the predicted label.

and "merger" for BUSINESS. It then chose BUSINESS, whereas the correct topic is SCI-TECH. (This is a difficult case even to humans.)

Another way to interpret the behaviour of DoLFIn is to examine $q(c|f)$ and $p(f|w, s)$, especially the meanings of latent features. Figure 3-left shows a heatmap visualising $q(c|f)$ for TREC. We can see that features f_2 and f_{16} are not helpful because they support all categories almost uniformly. Feature f_4 strongly supports ENTY whereas f_0, f_{11}, f_{19} support HUM. Feature f_9 prefers ENTY but it can also be used for DESC and HUM. Interestingly, there are no features strongly supporting ABBR. DoLFIn seems to rely on the absence of all features when predicting ABBR. Figure 3-right-top shows a heatmap of $p(f|w, s)$.

Figure 3-right-bottom shows TREC questions whose words with their most probable latent features are highlighted.[1] For instance, the first "What" is mapped to feature f_9 and $q(f_9|\text{What}, s)$ is high (see the first row in Figure 3-right-top). In general, DoLFIn uses f_9 for words "What", "Which" that are often for ENTY, but sometimes for HUM (e.g., "What is the most popular last name ?"), and DESC ("What is Java ?") (see Figure 3-left). If DoLFIn can utilise the next words to choose ENTY, it will assign f_4 to them (e.g., DoLFIn knows that the term "What *sports*$_4$" of the first question in Figure 3-right-bottom is for asking about an entity). Otherwise, it will again use f_9 (e.g., DoLFIn still can not decide if "What *is*$_9$" of the third question is for asking about an entity or a description).

5 Conclusion

We have proposed a new architecture DoLFIN based on probability for building explainable models. DoLFIn represents input by a *bag of latent features* using linear-softmax layers to map input components to distributions over latent features, and a truncated sum to aggregate these resulting distributions. We showed that, different from attention and saliency maps, it is straight-forward to compute how much an input component supports a category. Demonstrating our idea, we applied DoLFIn to text classification. Compared with the classical CNN and BiLSTM text classifiers, DoLFIn achieved comparable accuracies, but much better interpretability.

Acknowledgement

We would like to thank anonymous reviewers and Thy Tran for their suggestions and comments.

[1]This example is cherry-picked for a simple analysis. Deeply analysing the meaning of latent features is left for future work.

References

Samira Abnar and Willem Zuidema. 2020. Quantifying attention flow in transformers. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4190–4197, Online, July. Association for Computational Linguistics.

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, Copenhagen, Denmark, September. Association for Computational Linguistics.

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

Jasmijn Bastings, Wilker Aziz, and Ivan Titov. 2019. Interpretable neural predictions with differentiable binary variables. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2963–2977, Florence, Italy, July. Association for Computational Linguistics.

Yonatan Belinkov, Lluís Màrquez, Hassan Sajjad, Nadir Durrani, Fahim Dalvi, and James Glass. 2017. Evaluating layers of representation in neural machine translation on part-of-speech and semantic tagging tasks. In *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1–10, Taipei, Taiwan, November. Asian Federation of Natural Language Processing.

Leilani H. Gilpin, David Bau, Ben Z Yuan, Ayesha Bajwa, Michael Specter, and Lalana Kagal. 2018. Explaining explanations: An overview of interpretability of machine learning. In *2018 IEEE 5th International Conference on data science and advanced analytics (DSAA)*, pages 80–89. IEEE.

Pankaj Gupta and Hinrich Schütze. 2018. LISA: Explaining recurrent neural network judgments via layer-wIse semantic accumulation and example to pattern transformation. In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 154–164, Brussels, Belgium, November. Association for Computational Linguistics.

Alon Jacovi, Oren Sar Shalom, and Yoav Goldberg. 2018. Understanding convolutional neural networks for text classification. In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 56–65, Brussels, Belgium, November. Association for Computational Linguistics.

Jaap Jumelet, Willem Zuidema, and Dieuwke Hupkes. 2019. Analysing neural language models: Contextual decomposition reveals default reasoning in number and gender assignment. In *Proceedings of the 23rd Conference on Computational Natural Language Learning (CoNLL)*, pages 1–11, Hong Kong, China, November. Association for Computational Linguistics.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1746–1751, Doha, Qatar, October. Association for Computational Linguistics.

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

Tao Lei, Regina Barzilay, and Tommi Jaakkola. 2016. Rationalizing neural predictions. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 107–117, Austin, Texas, November. Association for Computational Linguistics.

Xin Li and Dan Roth. 2002. Learning question classifiers. In *COLING 2002: The 19th International Conference on Computational Linguistics*.

W James Murdoch, Peter J Liu, and Bin Yu. 2018. Beyond word importance: Contextual decomposition to extract interactions from lstms. *ICLR*.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. 2019. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems*, pages 8024–8035.

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, Doha, Qatar, October. Association for Computational Linguistics.

Nina Poerner, Hinrich Schütze, and Benjamin Roth. 2018. Evaluating neural network explanation methods using hybrid documents and morphosyntactic agreement. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 340–350, Melbourne, Australia, July. Association for Computational Linguistics.

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, Seattle, Washington, USA, October. Association for Computational Linguistics.

Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhudinov, Rich Zemel, and Yoshua Bengio. 2015. Show, attend and tell: Neural image caption generation with visual attention. In *International conference on machine learning*, pages 2048–2057.

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

Bolei Zhou, Aditya Khosla, Agata Lapedriza, Aude Oliva, and Antonio Torralba. 2016. Learning deep features for discriminative localization. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2921–2929.

A Experiment Setting

A.1 Dataset

The links to the used three dataset are given below.
• TREC Question, `https://cogcomp.seas.upenn.edu/Data/QA/QC/`
• SST2, `https://nlp.stanford.edu/sentiment/`
• AG-news, `https://github.com/mhjabreel/CharCnn_Keras/tree/master/data/`
The only pre-processing step we applied is tokenization.

A.2 Hyper-parameters

The hyper-parameters of the four models are shown in Table 2. For DoLFIn, we set the number of latent features d to 20, 10, and 100 when testing it on TREC, SST2, and AG-news respectively. We used Glove word-embeddings downloaded from `http://nlp.stanford.edu/data/glove.840B.300d.zip` (Pennington et al., 2014).

Following (Kim, 2014), we applied a dropout layer to text representation s, with dropout rate 0.5.

word embedding dimensions d_w	300 (GloVE)
convolutional filter sizes	(3,4,5)
convolutional filter number	100
bilstm hidden dimensions	100
text vector dimenions d_s	100 (CNN)
	100 (DoLFIn)
	200 (BiLSTM)
minibatch size	50
patience (for early stopping)	10
optimiser	Adam
learning rate	0.001

Table 2: Hyper-parameters