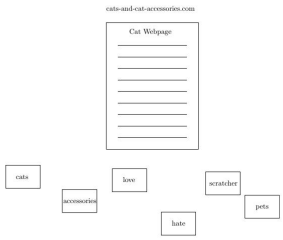
The Unordered Transformer For Web-page Classification

Ramy El Garawany

CS 230. https://youtu.be/7bclg8WqjqU

The Problem

Suppose there exists a web-page about cats.



Suppose that, at a large scale, it is not feasible to get the entire contents of the web-page. So instead, you get a bag of unordered words. Some are from the URL, some are from site metadata, and some are from the page content itself.

The task is to **a)** figure out if this web-page is commercial in nature and if so to **b)** determine what the web-page is about, from a list of approximately 770 possible categories.

The Embedding

Each input word is actually a 100-dimensional vector looked up from an embedding.

Figure 1:Embedding Vector

The embedding matrix contains approximately 13 million words, across many different languages. This embedding is continuously and incrementally updated, as new words enter our lexicon.

model that groups semantically similar words together by learning on a huge corpus.

The Baseline

The current baseline model is a simple 3-layer feed-forward network. It uses the bag-of- words approach [2], and simply averages all the embedding values together.

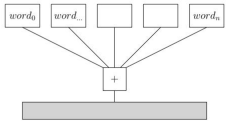


Figure 2:Bag-Of-Words

Order Matters!

Related Work

My initial plan was to use the model described in [3]. The authors observed that, even when dealing with unordered sets, the order that you present the inputs to the network matters! (For example, in some situations, running over your inputs in reverse might improve performance!).

They propose a model that instead learns an order-independent representation over the inputs. It works by using attention (very similarly to the final model this paper used) over the inputs, where the scoring function does not depend on input order.

An LSTM without any inputs or outputs is used to create this representation. At each step, the LSTM's hidden state is used to compute a score over all the inputs (i.e., attention). The inputs are then linearly combined, and that value, along with the LSTM's hidden state, is passed along to the next time-step.

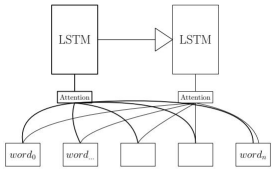


Figure 3:The Order Matters model.

Problems

However, there were some issues with this model. First, regularization is hard to get right. Simple dropout used on the cell's hidden state will prevent the model from being able to learn over long sequences [5].

The paper suggested a variant of dropout called DropConnect [4], that instead applies dropout on the *weights* of the hidden-to-hidden connections. The paper also suggests further tricks (such as using Averaged SGD).

Finally, this model was incredibly slow to train, and more importantly, to serve. Running sequential operations over long input chains does not parallelize well. So I unfortunately had to find another solution!

The Transformer

The Transformer is a state-of-the-art[1] model that uses a concept called "self-attention" in order to learn dependencies between inputs.

The attention model is quite simple. First, each input word is projected (using separate matrices) to vectors called Queries (Q), Keys (K), and Values (V).

Figure 4:Computing Q, K, and V.

Then, a dot product between each query Q and every other key K is performed to compute a score per (Q, K) pair. Finally, that score is used to linearly combine the values V.

Figure 5:Output of each self-attention sub-layer.

The real Transformer model also includes positional-encoding. This is necessary, because as described above, the model has no way of learning the relative order-related dependencies between words. However, for this case, that is a feature, not a bug.

Results

Table 1:Transformer and Baseline Comparisons

	Macro F-Score	P@5	R@5	Micro F-Score
Transformer Model	78.5	58.8	89.8	88.1
Baseline Model	77.5	57.3	87.2	85.81

While the model outperforms the baseline, the Transformer is much more complex. It is also more expensive to serve. Therefore, the cost-benefit analysis might not go in the favor of this model.

I did not have enough time to really fine-tune the hyper-parameters. A more rigorous exploration of the hyper-parameter space might yield more favorable results.

Furthermore, I would like to do an analysis of mis-classified problems to see if data augmentation (e.g. by attempting to extract the original text) would help.

References

[1] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[2] Zellig S Harris. Distributional structure. *Word*, 10(2-3):146–162, 1954.

[3] Oriol Vinyals, Samy Bengio, and Manjunath Kudlur. Order matters: Sequence to sequence for sets. *arXiv preprint arXiv:1511.06391*, 2015.

[4] Li Wan, Matthew Zeiler, Sixin Zhang, Yann Le Cun, and Rob Fergus. Regularization of neural networks using dropconnect. In *International Conference on Machine Learning*, pages 1058–1066, 2013.

[5] Wojciech Zaremba, Ilya Sutskever, and Oriol Vinyals. Recurrent neural network regularization. *arXiv preprint arXiv:1409.2329*, 2014.

Almost. [1] stacked a bidirectional Transformer and was able to beat it.