The Unordered Transformer For Web-page Classification
Ramy El Garawary
CS 2010 https://www.cs.uic.edu/~tlikformal/Zs

The Problem
Suppose there exists a web page about cats.

... Figure 1: A web page...

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 figure out if this web page is commercial in nature and if so to 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 look up from an embedding.

<table>
<thead>
<tr>
<th>w1</th>
<th>w2</th>
<th>w3</th>
<th>w4</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

The embedding matrix contains approximately 10 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 large 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.

<table>
<thead>
<tr>
<th>b1</th>
<th>b2</th>
<th>b3</th>
<th>b4</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Order Matters!

Related Work
My initial plan was to see 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 (So, attended). The inputs are then linearly combined, and that value, along with the LSTM’s hidden state, is passed along to the next time step.

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 (6), that instead applies dropout on the weights of the hidden-to-hidden connections. The paper also suggests further tricks such as using Averaged SGE.

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

The Transformer
The Transformer is a state-of-the-art 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).

Then, a dot product between each query Q and every other key K is performed to compute a score per Q * K. Finally, that score is used to weight the values V.

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

Results

<table>
<thead>
<tr>
<th>Model</th>
<th>Macro F-Score</th>
<th>Micro F-Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>Transformer</td>
<td>78.3</td>
<td>84.6</td>
</tr>
<tr>
<td>Baseline</td>
<td>77.3</td>
<td>82.6</td>
</tr>
</tbody>
</table>

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 parameters space might yield even 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 augment the original text) would help.

References

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