TP1 - Language Modeling

Author: Gabriele LORENZO

Cell C Q1: What is the size of the vocabulary?

The size of the vocabulary is 743.

Cell F Q2: What do you expect the following probabilities to be? Why?

1. I expect P(<eos>|".") to be 1, since a period is always followed by an end of sentence token, (e.g. all sentences end with a period).
2. I expect P(<eos>|<bos>) to be 0, since a beginning of sentence token is never followed by an end of sentence token. This would mean that we have an empty sentence (e.g. <bos><eos>).

Cell H Q3: Run this code to produce a plot. What does this plot show? What is on the x-axis, what is on the y-axis?

This is the plot of the so called Zipf's law. The x-axis shows the rank of the word in the vocabulary, and the y-axis shows the frequency of the word. The plot shows that the frequency of a word is inversely proportional to its rank in the vocabulary.

Cell H Q4: According to this plot, is the ngram assumption justified? If you don't remember what the ngram assumption is, ask Nils.

The ngram assumption is that the probability of a word depends only on the previous n-1 words. In the plot we can see that two distinct trends:

- For small n, the ngram assumption is justified: the n-grams are more frequent and are able to capture more information
- For large n, the ngram assumption is not justified: the frequency of n-grams decreases rapidly, leading to data sparsity. This sparsity makes it difficult to estimate reliable probabilities, as many higher-order n-grams appear rarely or not at all in the dataset

Cell I Q5: Why does this throw an error?

Looking at the assertion message and at the "dev" dataset, we understand that we are trying to compute the following probability: P("was"|"box", "a", "hoped", "liam", "<bos>"). However, the 5-gram "box a hoped liam <bos>" does not appear in the training data. The model has never seen this 5-gram (the assertion checks if the history has been seen at least once in the training data). Therefore, the model cannot compute the requested probability.

To confirm this, we can check the training data to see if the 5-gram "box a hoped liam <bos>" appears in the training data. We can see that the only sentences starting with "<bos> Liam hoped that a" are the following:

- Liam hoped that a beer
- Liam hoped that a judge
- Liam hoped that a girl (appears twice)

Therefore, we are now sure that the 5-gram "box a hoped liam <bos>" does not appear in the training data.

Cell K Q6: Why doesn't the dev perplexity keep decreasing as n increases?

As explained before, as n increases, the n-grams become less frequent and the data sparsity increases. With smoothing we can mitigate this problem, but we have to make trade-offs between the quantity of information we can capture (high n-grams capture more information) and the generalization of the model (low n-grams generalize better).

To summarize, it's not obvious that increasing n will always lead to a lower perplexity.

Cell L Q7: What are the best values for n and smoothing?

The best values for n and smoothing are the ones that minimize the perplexity on the dev set. In this case, the best values are n=2 and smoothing=1.

Cell L Q8: How many parameters does the best ngram model have? Explain how you compute this quantity.

Generally, the number of parameters of an n-gram model is the number of unique n-grams in the training data.

Theoretically, the number of parameters should be given by |V|^n where |V| is the size of the vocabulary and n is the order of the n-gram model. However, in practice, we have to consider that some n-grams may not appear in the training data, so the number of parameters will be less than |V|^n.

In our case, we are working with a bigram model, so the theoretical number of parameters should be given by |V|^2 = 743^2 = 552049. Nevertheless we can see that the real number of parameters is only 11831.

Cell M Q9: What do you notice?

The perplexity of the model is similar for the train, the dev and the test sets. This means that the model generalizes well to unseen data and did not overfit the training data.

However, the perplexity of the model is higher for the gen set, meaning the model doesn't generalize perfectly.

Cell N Q10: How does this RNN LM deal with words outside of its vocabulary?

The model deals with words outside of its vocabulary by replacing them with the <unk> token, as we can see in this snippet of code:

```
if word in self.vocab:
    key = word
else:
    key = self.unk
```

Cell O Q11: How do the number of parameters of this model compare to the number of parameters of the best ngram model?

We have a comparable number of parameters for the RNN model and the best ngram model: the RNN model has 11615 parameters, while the best ngram model has 11831 parameters.

We can also notice that, at least theoretically, the number of parameters of the n-gram model grows exponentially with n, while the number of parameters of the RNN model grows linearly with the size of the vocabulary.

Cell P Q12: How does the following compare to the best ngram model?

All the values of the perplexity are lower for the RNN model than for the best ngram model. This means that the RNN model, with the same number of parameters, is able to give stronger predictions.

Also in this case the perplexity is higher for the gen set, meaning that the model is not able to generalize well to unseen data.

Cell Q Q13: What else could we have done to further push the dev loss down?

Another possible improvement would have been to use more complex embeddings, such as pre-trained embeddings (e.g. GloVe, Word2Vec, etc.).

Cell S Q14: How does the following compare to the other models?

The perplexity is slightly lower. This indicates that increasing the model size can help to improve the performance of the model.

Cell T Q15: What do you notice? Is there a phrase that describes this mismatch between distributions?

In all the cases, the perplexity is higher for the gen set (in the plot the mean of the perplexity distribution is always higher for the gen set than for the test set).

We can see that both of the RNN models are able to reduce the perplexity on the test set, but not on the gen set. This is an example of **distribution shift**, where the test set is drawn from a similar distribution as the training set, while the gen set is drawn from a different distribution.

This shows the limitation of RNNs in generalizing to OOD data.