Laura Abreu Schulz Vieira - 2017015061

Processamento de Linguagem Natural
TP2 - POS TAGGING

1 Objetivo

O objetivo deste trabalho é a implementação e avaliação de um modelo de **POS Tagging** para a língua portuguesa.

2 Introdução

Uma tarefa de **POS Tagging** (Part-of-speech Tagging) consiste em classificar palavras de uma sentença em sua devida classe gramatical. Em portguês temos, então, classes como *Nome, Nome Próprio, Adjetivo,* etc.

Para implementar um modelo que realize essa tarefa, é imprenscindível que haja contextualização, uma vez que muitas das palavras podem ter diversas classes gramaticais diferentes e a correta só será definidada dado o contexto. Por isso, a arquitetura sugerida para confecção do trabalho é a **LSTM** (Long-Short-Term-Memory).

A LSTM é uma RNN (rede neural recorrente) que, como diz o nome, consegue controlar bem a "quantidade de contexto" dada na realização de uma tarefa. Ela foi, portanto, o modelo selecionado para a realização desse trabalho.

3 Coleta dos dados

Os dados de treino, validação e teste foram todos fornecidos na especificação do trabalho, proveniente de http://nilc.icmc.usp.br/macmorpho/macmorpho-v3.tgz. Correspondem ao corpus Mac-Morpho, produzido pelo grupo NILC em da ICMC USP.

Ambos os três estão organizados de forma que cada sentença está em uma linha e cada palavra da sentença está acoplada à sua respectiva tag por meio de um "_".

Existem 26 possíveis tags, todas devidamente explicadas em

http://nilc.icmc.usp.br/macmorpho/macmorpho-manual.pdf.

4 Metodologia

Para realização do trabalho escolhi a linguagem *Python*, devido à sua grande disponibilidade de bibliotecas que já possuem implementações prévias de modelos de linguagens, além de sua facilidade de entendimento.

A biblioteca que selecionei para utilizar os modelos de linguagem foi a *Keras*, seguindo o seguinte tutorial:

https://nlpforhackers.io/lstm-pos-tagger-keras/

Selecionei essa biblioteca por já possuir uma implementação de *LSTM* e possuir uma boa documentação e muito material sobre ela disponível na internet.

O preprocessamento dos dados foi feito de forma que, para cada corpus (teste, validação e treino) foram criadas duas listas: uma contendo as sentenças e a outra contento as respectivas TAGs. Cada uma dessas listas possuem, dentro delas, outras listas que representam às frases. Após isso, foi feito também um catálogo de palavras e TAGs vistas em tempo de treino.

O *Keras* exige que todas as sentenças tenham o mesmo tamanho (em número de palavras). Por isso, padronizei todas com o tamanho da maior sentença, que é 248 palavras.

Para poder passar pela rede as sentenças foram transformadas em sequências numéricas seguindo o catálogo de palavras feito anteriormente. As tags, por sua vez, passaram pelo processo de *One Hot Encoding*.

Dessa forma, temos a informação da maneira que precisamos para passar pela rede. Portanto, o próximo passo foi configurar a rede para a tarefa em questão. Sua configuração foi:

O modelo em questão possui uma camada de *embedding*, em que computamos nossos vetores de representação das palavras, uma camada sendo uma *LSTM bidirecional*, que analisa o contexto em ambas as direções aumentando sua precisão e uma *Dense Layer*, que faz a predição da TAG. Foi treinado em 10 épocas, utilizando os corpus de teste e validação.

A medida de análise foi a acurácia, que teve que ser feita costumizadamente devido à padronização do tamanho das sentenças; As frases - que, a esta fase, já estão representadas por números - são completas por espaços vazios que sempre pertencem à TAG correspondente ao valor "0". Portanto, se consideradas as frases completas, a acurácia seria falsamente maior do que a real, uma vez que os espaços vazios são muito fáceis de acertar. Portanto, foi feita uma função de parâmetro que ignora a acurácia dos acertos dos espaços que completam o tamanho da frase.

Para análise das TAGs mais recorrentes foi feita uma análise da acurácia para cada TAG,

```
Model: "sequential_2"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding_2 (Embedding)      (None, 248, 128)          6046464
_________________________________________________________________
bidirectional_2 (Bidirection (None, 248, 512)          788480
_________________________________________________________________
time_distributed_2 (TimeDist (None, 248, 27)           13851
_________________________________________________________________
activation_2 (Activation)    (None, 248, 27)           0
=================================================================
Total params: 6,848,795
Trainable params: 6,848,795
Non-trainable params: 0
_________________________________________________________________
```

Figure 1: Sumário do modelo utilizado

ou seja, o percentual de vezes que ela "foi acertada".

5 Análise dos Resultados

5.1 Análise por TAGs

Como dito anteriormente, para cada TAG fiz uma análise das acurácias em tempo de teste, após 6 épocas de treino.

O gráfico a seguir ilustra as respectivas acurácias para cada TAG:

A partir dessa figura, podemos observar que a classe com um pior desempenho considerável em relação as outras foi a TAG *NPROP*, que representa os nomes próprios. Isso pode ser facilmente explicados pela menor ocorrência dos nomes próprios em relação as demais palavras, uma vez que muitos deles ocorrem raras ou apenas uma vez ao longo dos corpus. Ademais, podem haver algumas ambiguidades, já que podemos transformar quase qualquer palavra em nome próprio, o que pode causar confusões.

Pode-se observar, também, que a maior acurácia foi de *ART*, que representam os artigos.

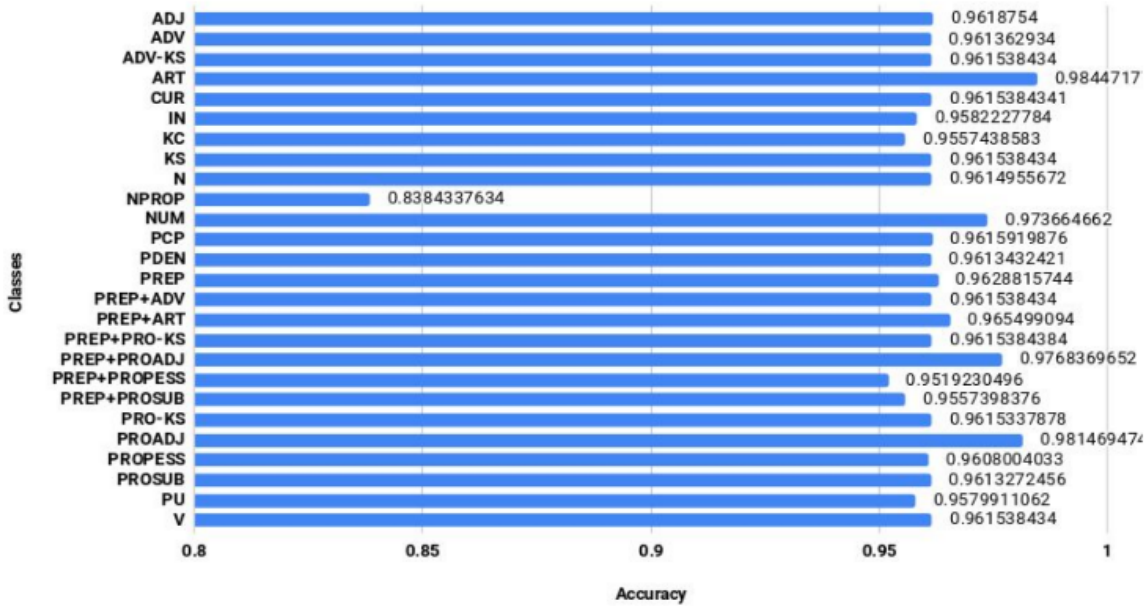

Figure 2: Gráfico de acurácias de cada TAG

Isso pode ser explicado por ser o caso inverso do observado acima: os artigos são a classe gramatical mais recorrente em um texto e existem pouca diversidade entre eles. Portanto, se torna muito mais simples o aprendizado eficiente dessa classe.

5.2 Análise geral

Ao longo das 6 épocas que foi rodado, pudemos observar o crescimento de ambas as acurácias: em tempo de treino e em tempo de validação. A visualização desses dados está em ambos os gráficos abaixo:

Nas primeiras iterações foi quando houve o maior crescimento de acurácias, em ambos os gráficos. A partir da quarta iteração ainda se observa melhora nos valores, ainda que pequenos. Entretando, vale ressaltar que, a partir da quarta iteração, a acurácia em tempo de testes se tornou superior (significativamente) que em tempo de validação. Isso pode evidenciar um início de processo de overfitting, em que a máquina "decora" os dados. Entretanto, o overfitting em si ainda não ocorre, já que a acurácia dos dados de validação ainda sobem e o resultado da acurácia em tempo de testes segue a mesma lógica, sendo muito semelhante à final da validação: 0.9431, enquanto o de validação é 0.9457. Possivelmente, com mais algumas poucas iterações ainda haveria melhora no resultado, ainda que pouca.

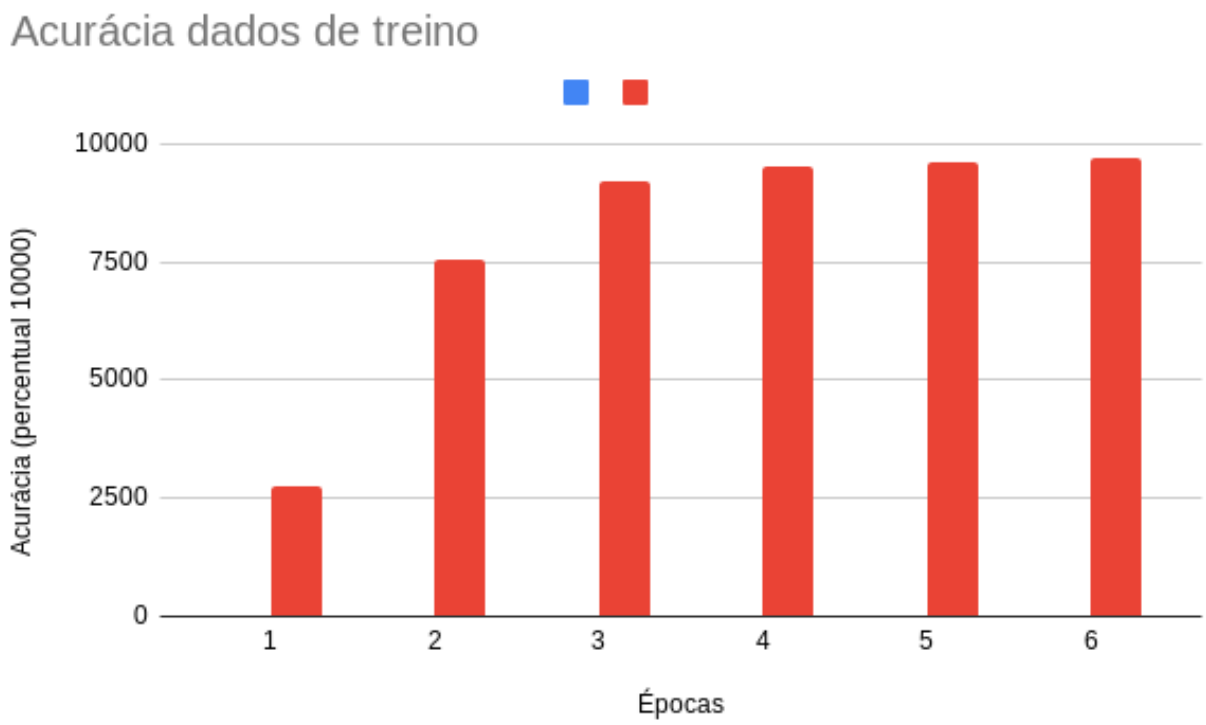

Figure 3: Gráfico de acurácias por época em tempo de teste

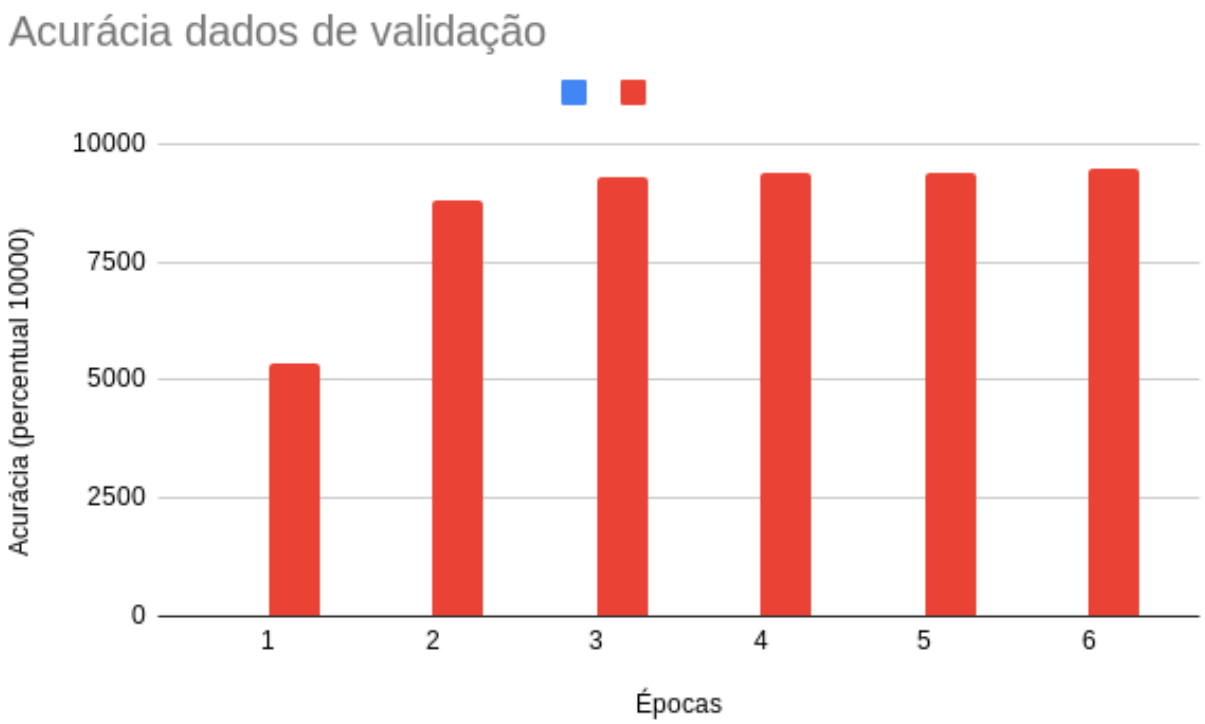

Figure 4: Gráfico de acurácias por época em tempo de validação

6 Conclusão

A LSTM possui um desempenho muito bom para a realização da tarefa de *POS Tagging*. Para apenas 6 épocas, o modelo atingiu uma acurácia de aproximadamente 94%, o que é um valor alto considerando a complexidade da tarefa.