Departamento de Ciência da Computaçao - UFMG

Trabalho Prático II - Aprendizado Profundo para Processamento de Linguagem Natural

Thiago Malta Coutinho
thiagomaltac@gmail.com

November 26, 2019

1 Introdução

O objetivo desse trabalho prático é estudar a tarefa de POS Tagging, ou classificação gramatical. Nesse estudo uma rede LSTM Bidirecional é implementada para classificar as classes gramaticais do corpus Mac-Morpho, produzido pelo grupo NILC do ICMC-USP. A estrutura do problema e da rede são discutidas assim como a acurácia obtida pelo modelo.

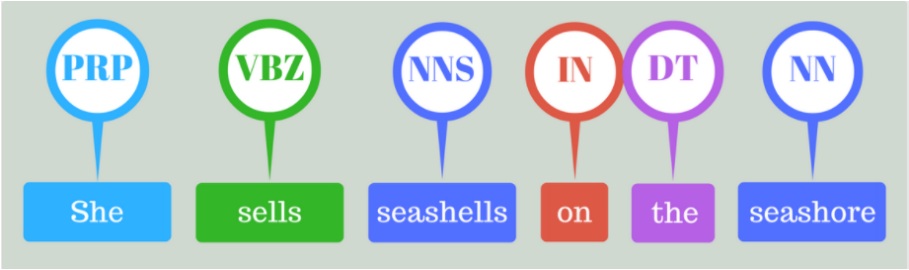

Figure 1.1: Exemplo da tarefa de classificação gramatical, em língua inglesa.

1.1 Corpus Mac-Morpho

A descrição da estrutura e das classes presentes no Corpus é feita no Manual, disponível em http://nilc.icmc.usp.br/macmorpho/macmorpho-manual.pdf. No entanto, as classes utilizadas nesse estudo foram definidas pelo método de pré-processamento de dados descrito na Seção 2.1. Os rótulos e a classe gramatical correspondente são apresentados na Tabela 1.1 abaixo:

RÓTULO	CLASSE GRAMATICAL
N	NOME
PU	PONTUAÇÃO
V	VERBO
NPROP	NOME PRÓPRIO
PREP	PREPOSIÇÃO
ART	ARTIGO
PREP+ART	PREPOSIÇÃO + ARTIGO
ADJ	ADJETIVO
ADV	ADVÉRBIO
KC	CONJUNÇÃO COORDENATIVA
PCP	PARTICÍPIO
NUM	NUMERAL
PROADJ	PRONOME ADJETIVO
KS	CONJUNÇÃO SUBORDINATIVA
PRO-KS	PRONOME CONECTIVO SUBORDINATIVO
PROPESS	PRONOME PESSOAL
PROSUB	PRONOME SUBSTANTIVO
PDEN	PALAVRA DENOTATIVA
CUR	SÍMBOLO DE MOEDA CORRENTE
PREP+PROADJ	PREPOSIÇÃO + PRONOME ADJETIVO
ADV-KS	ADVÉRBIO CONECTIVO SUBORDINATIVO
PREP+PROSUB	PREPOSIÇÃO + PRONOME SUBSTANTIVO
PREP+PROPESS	PREPOSIÇÃO + PRONOME PESSOAL
IN	INTERJEIÇÃO
PREP+PRO-KS	PREPOSIÇÃO + PRONOME CONECTIVO SUBORDINATIVO
PREP+ADV	PREPOSIÇÃO + ADVÉRBIO

Table 1.1: Rótulos e Classes gramaticais analisadas no Corpus Mac-Morpho.

1.2 LSTM

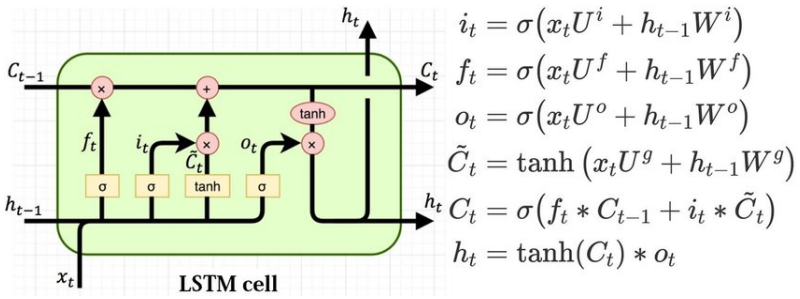

Figure 1.2: Célula LSTM com as operações internas explicitadas.

O modelo LSTM(Long-Short Term Memory) é um modelo que aprende dependências temporais em dados sequenciais. Ele possui representações internas chamadas de estado es-

condido e célula de memória, que são modificadas de acordo com os dados de entrada e as representações anteriores.
Dentro de cada célula LSTM, os dados de entrada e as representações internas anteriores passam por camadas de transformações lineares seguidas de funções não-lineares como a tangente hiperbólica(tanh) e a função sigmoide(σ). Para controlar o fluxo de dados ao longo do tempo, existem três *gates*: i_t, que controla o quanto de informação entra na célula; f_t controla o quanto o estado de memória anterior será suavizado, ou "esquecido"; o_t limita o quanto de informação interna da célula atual será transferida para o estado escondido atual. Além dos *gates* existem duas transformações não-lineares dentro da célula: $\tilde{C}_t$, uma representação intermediária da célula memória atual; e h_t que é o novo estado escondido.

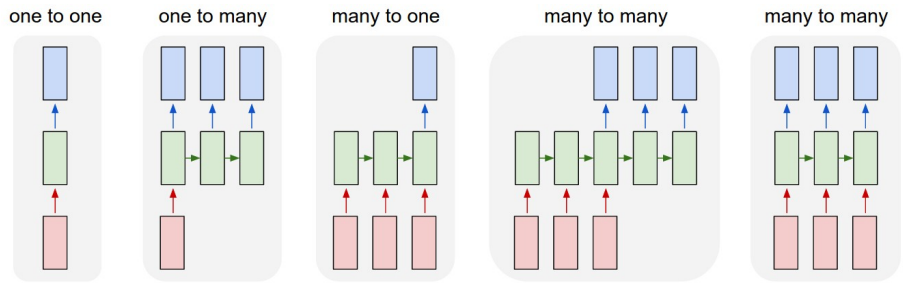

Figure 1.3: Exemplos de diferentes usos da arquitetura LSTM.

A arquitetura da rede neural LSTM pode ser utilizada de diversas maneiras, a Figura 1.3 exemplifica as configurações mais comuns. Para o problema de *POS Tagging* a organização utilizada é a *Many-to-many* com uma saída para cada entrada, pois para cada palavra deverá ser classificada gramaticalmente.
Cada palavra da frase é representada em um vetor, *one-hot* ou *Embedding*, esse vetor passa pelas transformações da célula LSTM gerando um estado escondido para a palavra atual. O estado escondido passa por uma transformação linear gerando um vetor do tamanho do número de classes gramaticais. A função *Softmax* é aplicada sobre o vetor para gerar probabilidades de ocorrêcia de cada classe, a classe com maior probabilidade será atribuída a palavra atual.
É importante ressaltar que cada vetor de probabilidades gerado pela rede LSTM é um vetor de probabilidades condicionais, dado que todas as transformações aplicadas na entrada atual são condicionadas pelo estado escondido anterior e a célula de memória anterior.
Os pesos da rede são atualizados utilizando o BPTT(Back propagation through time) e uma função de custo, como a entropia cruzada, por exemplo.

1.3 LSTM Bidirecional

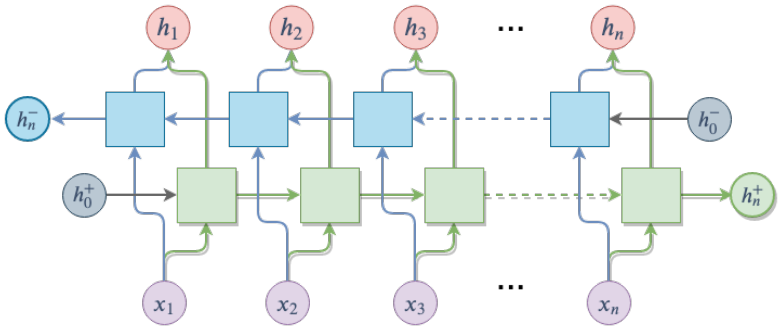

Figure 1.4: Arquitetura LSTM Bidirecional.

A rede LSTM processa os dados em apenas um sentido e isso pode gerar limitações nas representações geradas pelo modelo. Diversos problemas dependem de características presentes em dados mais a frente na sequencia, e percorrer os dados em apenas um sentido não captura essa informação. Uma das soluções possíveis para o problema é instanciar outra rede que percorrerá os dados no sentido contrário, possibilitando representações bidirecionais. Os estados escondidos, ou saídas, de cada rede são concatenados antes de entrarem na camada de transformações lineares que antecede a função *Softmax*. Com essa mudança mais informações podem ser aprendidas dos dados.

2 Experimento

O experimento consiste no treinamento de uma rede LSTM Bidirecional no conjuntos de dados de treinamento do Corpus Mac-Morpho e avaliação da acurácia do algoritmo no conjunto de teste do Corpus. Para o processamento e projeto da arquitetura foi utilizado o Framework PyTorch, em linguagem Python.

2.1 Pré-processamento dos dados

O pré-processamento dos dados foi feito no conjunto de treinamento e teste. Os dados foram carregados para a memória e as palavras foram separadas das classes correspondentes. A partir dos vetores de palavras e classes um arquivo de dados Tabulares foi gerado para ser processado pelo módulo `torchtext`, do framework PyTorch. Os dados de treinamento foram divididos em conjunto de treinamento e validação, em uma proporção de 90% para treino e 10% validação.
Utilizando o `torchtext` *batches* de tamanho 64 foram criados ordenados em ordem decrescente de tamanho para otimização do *padding* dentro da rede neural.

2.2 Word Embedding

Semantic Word embeddings são vetores que representam as palavras de forma densa. Eles carregam informações sintáticas e semânticas através do contexto próximo a palavra. Um Word Embedding que se tornou famoso por sua eficácia é o Word2Vec, que existe em duas versões: Skip-gram e CBOW.
Para alimentar a rede um modelo de Word Embedding Word2Vec Skip-gram pré-treinado foi baixado do Repositório de Word Embeddings do NILC. O modelo possui dimensão 100 e produz representação de entrada melhor que o *one-hot encodding*.

2.3 Estrutura da rede e hiperparâmetros

A rede é constituída por uma camada de Embedding, que recebe o Word2Vec 100D pré treinado. Em seguida existem duas camadas de LSTM Bidirecional com uma camada de `Dropout` ao final das camadas da LSTM, afim de regularizar a rede. Após a cada de `Dropout` tem uma camada Linear seguida de uma função *Softmax*. O otimizador utilizado foi o Adam, com regularização L2 na atualização dos pesos. Foi aplicado *Gradient Clipping* de norma 6 para evitar *Exploding Gradient*. A dimensão escolhida para o estado escondido foi 100.
A função de perda utilizada foi a Entropia Cruzada.

2.4 Rotina de treinamento e avaliação

A rotina de treinamento consiste em 40 épocas sem atualização dos pesos do Embedding. Isso porque a rede é inicializada com pesos aleatórios e o gradiente pode atualizar a camada de Embedding de forma a piorar sua representação. Nas primeiras 40 épocas a probabilidade da camada de `Dropout` é igual à 0,25. Após as 40 épocas o algoritmo é treinado por mais 40 épocas mas com atualização dos pesos da camada de Embedding. A probabilidade da camada `Dropout` é atualizada para 0,4. O aumento da probabilidade se deve ao aumento do número de pesos a serem treinados, mais conexões são podadas afim de evitar o *Overfitting* da rede.

3 Resultados

Ao final das 80 épocas de treinamento o modelo foi avaliado no conjunto de teste do Corpus. A acurácia total obtida foi **91,95%**.

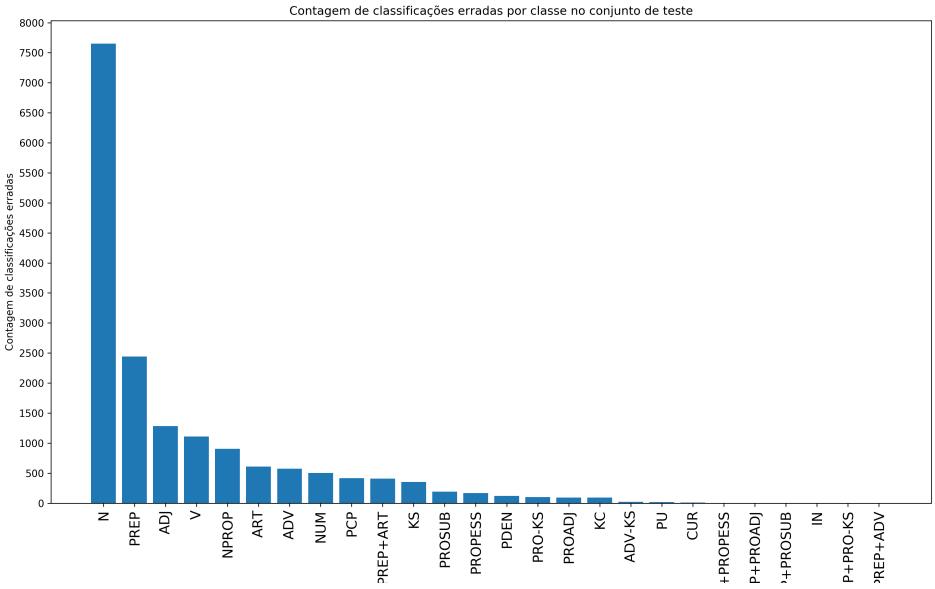

Figure 3.1: Contagem de classificações erradas por classe no conjunto de teste

Para avaliar a acurácia do modelo, a contagem de erros por classe foi feita para visualização da quantidade de erros em cada classe.

Como cada classe tem número de ocorrências distinto, a contagem em si não fornece resultados conclusivos. A acurácia por classe foi avaliada para levar em consideração o número de amostras de cada classe.

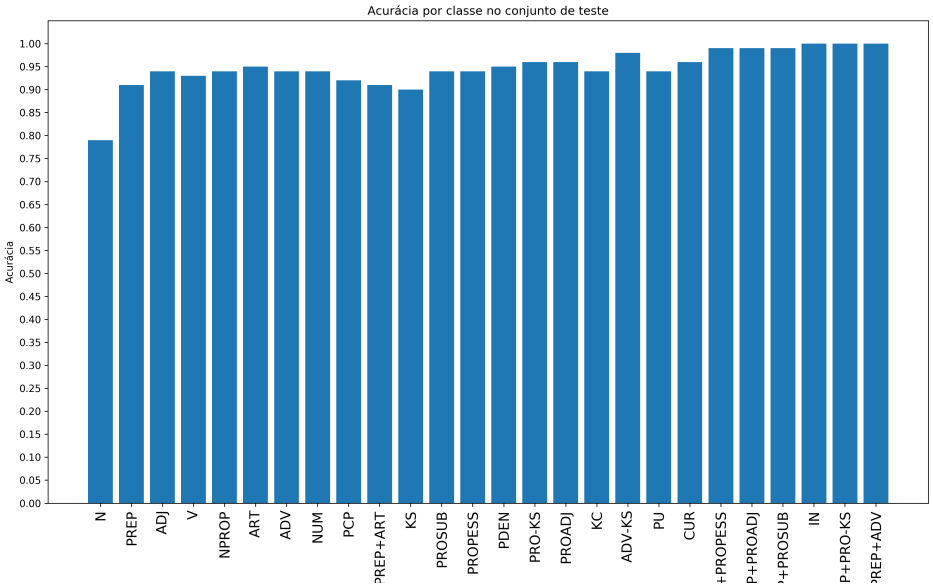

Figure 3.2: Acurácia por classe no conjunto de teste

A partir da Figura 3.2, é possível verificar o comportamento observado na Figura 3.1 tem correlação com a acurácia por classe. A discrepância entre as classes diminuiu ao avaliar a acurácia mas mesmo assim as classes que erraram mais também possuem acurácia menor, em média.

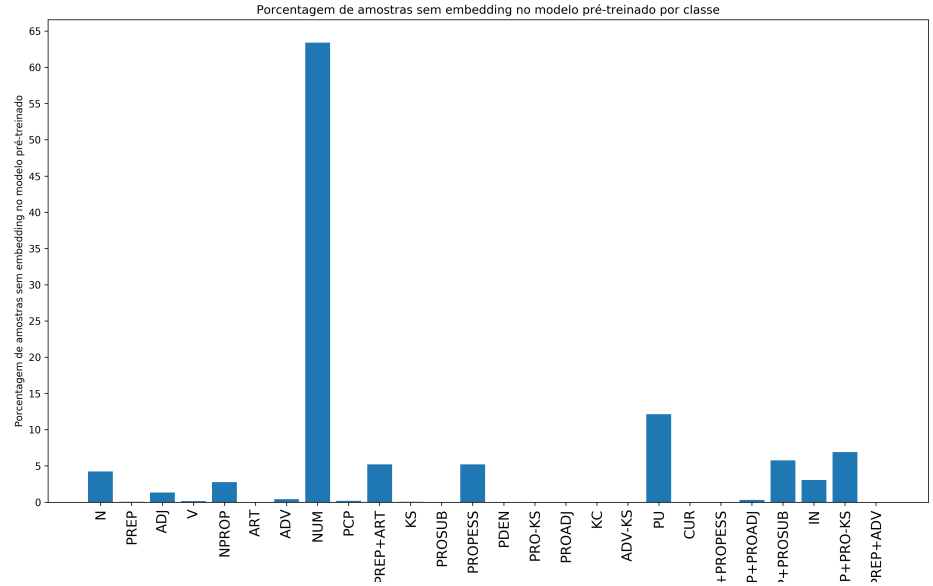

Figure 3.3: Porcentagem de amostras sem embedding no modelo pré-treinado, por classe

Partindo dessa observação, uma hipótese levantada foi a de que a classe de **Nomes(N)**

possui muitos elementos e pode não estar sendo representada corretamente pelo Word2Vec pré-treinado. Foi verificado a porcentagem de palavras que existiam vetores pré-treinados no Word Embedding, os resultados separados por classe estão na Figura 3.3. Os resultados apontam que boa parte das palavras estavam representadas no Embedding. É possível que se todas as palavras do vocabulário de teste estivessem representadas no Embedding os resultados fossem ligeiramente melhores.

4 Conclusões

Esse trabalho estudou o problema de POS Tagging. Nele foi implementado um modelo de aprendizado de máquina baseado na LSTM Bidirecional para classificação gramatical no Corpus Mac-Morpho.
O modelo foi treinado e obteve resultados satisfatórios, alcançando acurácia geral de **91,95%**. Ao observar as acurácias por classe, o modelo deixa a desejar na classe **Nome(N)**. Um estudo para verificar se existiam representações de Embedding para essa classe e não foi encontrado falta de representação significativa.
O resultado final pode ser devido à própria arquitetura do modelo e não os dados, mas um estudo mais aprofundado deve ser feito para formar conclusões. Uma melhoria a ser feita no modelo é adicionar uma representação dos caracteres, trazendo mais informação para a classificação.
Nesse trabalho foi possível estudar o modelo de linguagem *Word2Vec* e entender melhor seu funcionamento. O aluno se familiarizou com a implementação fornecida pelo professor e apresentou uma rotina de avaliação do algoritmo ao variar os parâmetros especificados.

5 Referências

1 - Notas de Aula do Professor Adriano Veloso, disciplina de Aprendizado Profundo para Processamento de Linguagem Natural.
2 - https://medium.com/@raghavaggarwal0089/bi-lstm-bc3d68da8bd0
3 - https://pytorch.org/
4 - https://torchtext.readthedocs.io/en/latest/
5- https://www.researchgate.net/publication/306093394_Multilingual_Part-of-Speech_Tagging_with_Bid
Term_Memory_Models_and_Auxiliary_Loss