Pontifícia Universidade Católica
do Rio de Janeiro

Rômulo César Costa de Sousa

Part-of-Speech Tagging para Português

Dissertação de Mestrado

Dissertação apresentada como requisito parcial para obtenção do grau de Mestre pelo Programa de Pós–graduação em Informática do Departamento de Informática da PUC-Rio.

Orientador: Prof. Hélio Côrtes Vieira Lopes

Rio de Janeiro
Agosto de 2019

Pontifícia Universidade Católica
do Rio de Janeiro

Rômulo César Costa de Sousa

Part-of-Speech Tagging para Português

Dissertação apresentada como requisito parcial para obtenção do grau de Mestre pelo Programa de Pós–graduação em Informática do Departamento de Informática da PUC-Rio. Aprovada pela Comissão Examinadora abaixo assinada.

Prof. Hélio Côrtes Vieira Lopes
Orientador
Departamento de Informática – PUC-Rio

Prof. Eduardo Sany Laber
Departamento de Informatica – PUC-Rio

Prof. Marco Antonio Casanova
Departamento de Informatica – PUC-Rio

Rio de Janeiro, 9 Agosto 2019

Rômulo César Costa de Sousa

Bacharel em Ciência da Computação (2016) na Universidade Federal do Ceará - Campus Quixadá (UFC)

Ficha Catalográfica

Rômulo César Costa de Sousa

Part-of-Speech Tagging para Português / Rômulo César Costa de Sousa; orientador: Hélio Côrtes Vieira Lopes. – Rio de janeiro: PUC-Rio, Departamento de Informática, 2019.

v., 86 f: il. color. ; 30 cm

Dissertação (mestrado) - Pontifícia Universidade Católica do Rio de Janeiro, Departamento de Informática.

Inclui bibliografia

1. Processamento de Linguagem Natural. 2. Anotação Morfossintática. 3. Aprendizado de Máquina. 4. Aprendizagem Profunda. 5. Incorporação de Palavras.
I. Hélio Côrtes Vieira Lopes. II. Pontifícia Universidade Católica do Rio de Janeiro. Departamento de Informática. III. Título.

CDD: 004

Agradecimentos

Primeiramente gostaria de agradecer a Deus, por ter me concedido saúde, força e disposição.

Sou imensamente grato ao Prof. Hélio Lopes grande professor e orientador. Muito obrigado pela paciência, orientação, confiança e dedicação.

Gostaria de agradecer imensamente aos meus pais, Antonio César e Ocilene, as minhas irmãs Jessyca e Isabelly, que nunca negaram apoio durante minha trajetória acadêmica.

A todos os amigos da pós-graduação, especialmente André, Micaele, Dalai, Lauro, Vinicius e Luisa, meu muito obrigado. Vocês foram fundamentais para minha formação, por isso merecem o meu eterno agradecimento.

Agradeço a Welida, sem o seu apoio e companheirismo esse trabalho não seria possível. Obrigado por ser tão atenciosa.

Agradeço a todos os integrantes dos laboratórios DasLab e Ideias, por me proporcionar um ambiente criativo, amigável e colaborativo para os estudos e pesquisas.

O presente trabalho foi realizado com apoio da Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Código de Financiamento 001.

Resumo

Rômulo César Costa de Sousa; Hélio Côrtes Vieira Lopes. **Part-of-Speech Tagging para Português**. Rio de Janeiro, 2019. 86p. Dissertação de Mestrado – Departamento de Informática, Pontifícia Universidade Católica do Rio de Janeiro.

Part-of-speech (POS) tagging é o processo de categorizar cada palavra de uma sentença com sua devida classe morfossintática (verbo, substantivo, adjetivo e etc). *POS tagging* é considerada uma atividade fundamental no processo de construção de aplicações de processamento de linguagem natural (PLN), muitas dessas aplicações, em algum ponto, demandam esse tipo de informação. Nesse trabalho, construímos um *POS tagger* para o Português Contemporâneo e o Português Histórico, baseado em uma arquitetura de rede neural recorrente. Tradicionalmente a construção dessas ferramentas requer muitas *features* específicas do domínio da linguagem e dados externos ao conjunto de treino, mas nosso *POS tagger* não usa esses requisitos. Treinamos uma rede *Bidirectional Long short-term memory* (BLSTM), que se beneficia das representações de *word embeddings* e *character embeddings* das palavras, para atividade de classificação morfossintática. Testamos nosso *POS tagger* em três corpora diferentes: a versão original do corpus Mac-Morpho, a versão revisada do corpus Mac-Morpho e no corpus Tycho Brahe. Nós obtemos um desempenho ligeiramente melhor que os sistemas estado da arte nos três corpora: 97.83% de acurácia para o Mac-Morpho original, 97.65% de acurácia para o Mac-Morpho revisado e 97.35% de acurácia para Tycho Brahe. Conseguimos, também, uma melhora nos três corpora para a medida de acurácia fora do vocabulário, uma acurácia especial calculada somente sobre as palavras desconhecidas do conjunto de treino. Realizamos ainda um estudo comparativo para verificar qual dentre os mais populares algoritmos de criação de *word embedding* (Word2Vec, FastText, Wang2Vec e Glove), é mais adequado para a atividade *POS tagging* em Português. O modelo de *Wang2Vec* mostrou um desempenho superior.

Palavras-chave

Processamento de Linguagem Natural; Anotação Morfossintática; Aprendizado de Máquina; Aprendizagem Profunda; Incorporação de Palavras.

Abstract

Rômulo César Costa de Sousa; Hélio Côrtes Vieira Lopes (Advisor). **Part-of-Speech Tagging for Portuguese**. Rio de Janeiro, 2019. 86p. Dissertação de mestrado – Departamento de Informática, Pontifícia Universidade Católica do Rio de Janeiro.

Part-of-speech (POS) tagging is a process of labeling each word in a sentence with a morphosyntactic class (verb, noun, adjective and etc). POS tagging is a fundamental part of the linguistic pipeline, most natural language processing (NLP) applications demand, at some step, part-of-speech information. In this work, we constructed a POS tagger for Contemporary Portuguese and Historical Portuguese, using a recurrent neural network architecture. Traditionally the development of these tools requires many handcraft features and external data, our POS tagger does not use these elements. We trained a Bidirectional Long short-term memory (BLSTM) network that benefits from the word embeddings and character embeddings representations of the words, for morphosyntactic classification. We tested our POS tagger on three different corpora: the original version of the Mac-Morpho corpus, the revised version of the Mac-Morpho corpus, and the Tycho Brahe corpus. We produce state-of-the-art POS taggers for the three corpora: 97.83% accuracy on the original Mac-Morpho corpus, 97.65% accuracy on the revised Mac-Morpho and 97.35% accuracy on the Tycho Brahe corpus. We also achieved an improvement in the three corpora in out-of-vocabulary accuracy, that is the accuracy on words not seen in training sentences. We also performed a comparative study to test which different types of word embeddings (Word2Vec, FastText, Wang2Vec, and Glove) is more suitable for Portuguese POS tagging. The Wang2Vec model showed higher performance.

Keywords

Natural Language Processing; Part-of-speech Tagging; Machine Learning; Deep Learning; Word Embedding.

Sumário

Lista de figuras

Lista de tabelas

Tudo o que temos de decidir é o que fazer com
o tempo que nos é dado

Gandalf, *Senhor dos Anéis: A Sociedade do Anel*

.

1
Introdução

Entre o nascimento da Internet e 2003, o ano de nascimento das redes sociais (MySpace, LinkedIn e Facebook), foram produzidos apenas algumas dúzias de exabyte de informações em toda a Web. Atualmente, esse mesmo volume de informação é criado semanalmente. O advento das redes sociais ofereceu as pessoas uma nova forma de compartilhamento de conteúdo que permitem a criação e compartilhamento de ideias, opiniões e informações, tudo isso de maneira eficiente em relação a custo e tempo. Essa enorme quantidade de dados, entretanto, é produzida principalmente de forma não estruturada (pois é especificamente feita para o consumo humano) e, portanto, não são facilmente processáveis por máquinas. Processar automaticamente esse tipo de informação envolve uma profunda compreensão da linguagem natural por parte de sistemas de computação (50).

Processamento de linguagem natural (PLN) é a área multidisciplinar que explora como computadores podem usar, entender e manipular dados provenientes da linguagem natural, seja ela escrita ou falada (2). Dentre as principais aplicações de PLN estão incluídas análise de sentimentos, tradução automática, extração e recuperação de informações, sistemas de perguntas e respostas, reconhecimento de fala, verificação ortográfica, entre outras. Na construção de tais aplicações são empregadas ferramentas intermediárias que ajudam na manipulação e estruturação dos dados de linguagem natural, como por exemplo, *Part-of-speech (POS) tagging*, reconhecimento de entidade nomeada, reconhecimento de relações e análise sintática de dependência.

Em especial, *Part-of-speech (POS) tagging* é o processo de categorizar cada palavra de uma sentença com sua devida classe morfossintática. Na Tabela 1.1, temos um exemplo dessa classificação. A palavra “O” é um artigo, “cavalo” é um substantivo, “de” uma adposição, “Napoleão” outro substantivo, “é” verbo e “branco” um adjetivo.

Tabela 1.1: Exemplo de *POS tagging*

Palavras	O	cavalo	de	Napoleão	é	branco
Classes	ART	SUBS	ADP	SUBS	V	ADJ

POS tagging é uma tarefa não trivial devido ao fato de que algumas palavras podem ter mais de uma classe sintática dependendo do contexto em que elas são usadas. *POS tagging* é considerada uma atividade fundamental no processo construção de aplicações de PLN (12). Muitas aplicações, em algum ponto, demandam de informações de *POS tagging* (13). Por exemplo, é possível usar informações de *POS tagging* em sistemas de análise de sentimentos (14), tradução automática (15) e respostas automáticas (16).

Pesquisadores de PLN vêm fazendo um ótimo trabalho e definiram o estado da arte em acurácia de *POS tagging* para mais de 97%. Porém, ainda assim, há espaço para melhorias, já que qualquer pequeno ganho em acurácia resulta em impactos em outras tarefas de PLN. Recentemente, técnicas de redes neurais se tornaram muito usadas para a resolução de problemas de PLN (51), bem como representações de *word embeddings* (25).

Em 2015, Wang et al(52) usaram uma rede neural recorrente (*Bidirectional Long Short-Term Memory*) combinado a *word embeddings* para resolver a tarefa de *POS tagging* para o inglês. A arquitetura foi testada no corpus Penn Treebank WSJ, que atingiu 97.40% de acurácia. Em 2018, Bohnet et al. (1) definiram a pontuação estado da arte para o corpus Penn Treebank WSJ em inglês com 97.96% de acurácia. Dos Santos e Zadrozny (44), construíram um modelo de rede neural para a resolução de *POS tagging* em Português. Eles aplicaram uma camada convolucional para extrair automaticamente *features* das palavras. Eles combinaram essa representação com a representação de *word embeddings*. O modelo foi avaliado em três corpora em Português: Mac-Morpho-v1, Mac-Morpho-v2 no corpus Tycho Brahe.

Este trabalho propõe construir um *POS Tagger* para a Língua Portuguesa usando uma arquitetura de redes neurais recorrentes que já se estabeleceram como estado da arte para *POS Taggers* em outros idiomas. Mais precisamente, a arquitetura proposta tem inspiração nos trabalhos de Ma e Hovy de 2016 (25) e Marulli et al. de 2018 (46).

Treinar *POS taggers* com desempenho estado da arte tradicionalmente requeira muitas *features* construídas de forma manual e dados externos ao corpus de treino. A arquitetura proposta nesse trabalho combina uma rede BLSTM com representações de *word embeddings* pré-treinados e representações de *character embeddings* para realizar a tarefa de *POS tagging*, sem usar quaisquer *features* manuais ou dados externos.

O modelo foi testado no Português Contemporâneo e no Português Histórico, usando três corpora: a versão original do Mac-Morpho corpus (v1) (47), a versão revisada do Mac-Morpho (v2) (48) e o corpus Tycho Brahe (49). Em todas os três corpora apresentamos um desempenho em acurácia e

acurácia FV ligeiramente superior ao estado da arte reportado por Fonseca et al. (45) e Santos e Zadrozny (44). Obtivemos para o corpus Mac-Morpho v1, 97.83% de acurácia em média e 95.48% de acurácia FV em média. Para o corpus Mac-Morpho v2, 97.65% de acurácia em média e 95.23% de acurácia FV em média. E para o corpus Tycho Brahe obtivemos, 97.35% de acurácia em média e 87.40% de acurácia FV em média.

Podemos destacar como as principais contribuições desta dissertação os seguintes pontos:

- Demonstração empírica de que uso de redes BLSTM combinado a representações vetoriais de *word embedding* e *character embedding* são bem adequados e produzem bons resultado para o problema de *POS tagging* em Português.
- A criação de *POS taggers* para a Língua Portuguesa que superam ligeiramente o estado da arte em termos de acurácia e acurácia fora do vocabulário (FV), em três corpora diferentes.
- O estudo comparativo para verificar qual dentre os mais populares algoritmos de criação *word embedding* (Word2Vec, FastText, Wang2Vec e Glove), é mais adequado para a atividade *POS tagging* em Português. O modelo *Wang2Vec* mostrou um desempenho ligeiramente superior.

Este documento está organizado conforme descrito a seguir: nos Capítulos 2 e 3 temos a revisão bibliográfica, descrevendo conceitos sobre redes neurais e processamento de linguagem natural. No Capítulo 4 trazemos os trabalhos relacionados, dividido em sistemas de *POS tagging* para a Língua Portuguesa e sistemas de *POS tagging* para outras línguas. No Capítulo 5 apresentamos a metodologia do trabalho e descrição do modelo de rede neural. No Capítulo 6 apresentamos os corpora usados, as métricas de avaliação, descrevemos os experimentos e comparamos os resultados com o estado da arte. Por fim, no Capítulo 7 apresentamos as conclusões e trabalhos futuros.

2
Redes neurais

Rede neural (RN) artificial tem sua inspiração no processo de aprendizado humano (20). Uma RN é composta de nós (neurônios) interligados, que recebem e fornecem informações. A mais simples versão de uma RN é composta de 2 camadas: camada de entrada e camada de saída, como ilustrado na Figura 2.1. Cada nó na camada de saída executa cálculos de pesos nos valores que recebem dos nós de entrada e então produzem valores de saída usando funções de transformação nos valores obtidos. Os pesos são corrigidos de acordo com erros ou perdas que a rede exibe nos nós de saída.

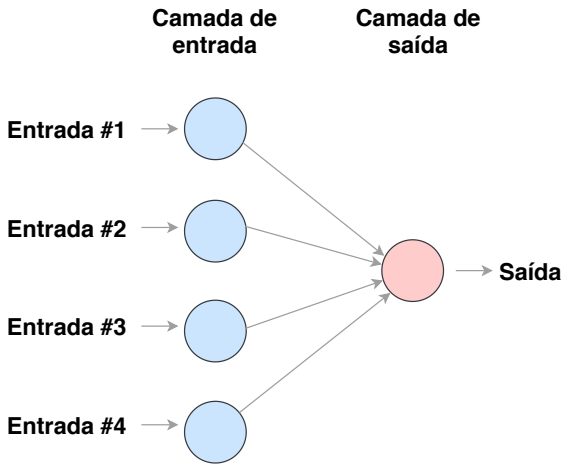

Figura 2.1: Uma rede neural simples

A camada da RN que fica entre a camada de entrada e a camada de saída é conhecida como camada escondida. Com a adição de camadas escondidas é possível separar os dados de uma forma não linear. O número de camadas escondidas e o número de nós em cada camada escondida são especificadas previamente e dependem da arquitetura aplicada. Na Figura 2.2 temos a ilustração de uma RN com uma camada de escondida. Por convenção, *Deep Learning* é a aplicação de redes neurais de múltiplas camadas para realizar tarefas de aprendizado, incluindo regressão, classificação, clusterização e outras (18).

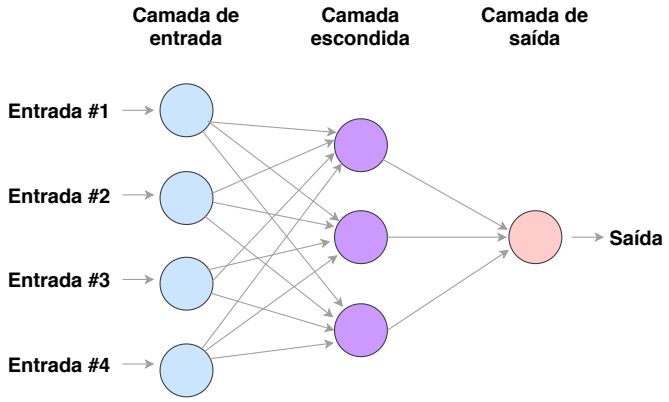

Figura 2.2: Uma rede neural simples com camada escondida

O conceito de RN foi introduzido no século XX (17) e, desde então, avanços na área vêm acontecendo. As técnicas de *deep learning* têm atraído o interesse de pesquisadores por sua capacidade de superar a desvantagens de algoritmos tradicionais que dependem de *features* projetadas manualmente. Abordagens de *deep learning* também se mostraram adequadas para análise em *big data*, aplicações em visão computacional, reconhecimento de padrões, processamento de linguagem natural e sistemas de recomendação (19).

2.1 Recurrent Neural Network (RNN)

Recurrent Neural Network (RNN), é uma família de redes neurais que processam dados sequenciais (23). As RNs tradicionais não consideram a sequência dos dados, porém em muitas aplicações isso se faz necessário. Se, por exemplo, queremos predizer a próxima palavra de uma sentença, é bem conveniente que conheçamos quais palavras ocorreram antes. RNNs comportam uma "memória" para armazenar informação do que já foi computado no passado. Na teoria, RNNs podem fazer o uso de informações de sequências arbitrariamente longas, mas na prática são limitadas apenas a informações de alguns passos anteriores.

RNNs contam com unidades de entrada $\{x_0, x_1, ..., x_t, x_{t+1}, ...\}$, unidades de saída $\{y_0, y_1, ..., y_t, y_{t+1}, ...\}$ e unidades escondidas $\{s_0, s_1, ..., s_t, s_{t+1}, ...\}$. Como ilustrado na Figura 2.3, no passo t, a RNN usa a amostra atual x_t e a unidade escondida passada s_{t-1} como entrada para obter o atual estado de s_t, com $s_t = f(x_t, s_{t-1})$ onde f denota a função de *enconder* (geralmente uma função não-linear, como tanh ou ReLU). Portanto, a RNN captura a dependência entre a amostra atual x_t e a amostra anterior x_{t-1} integrando a

representação da unidade escondida s_{t-1} (55).

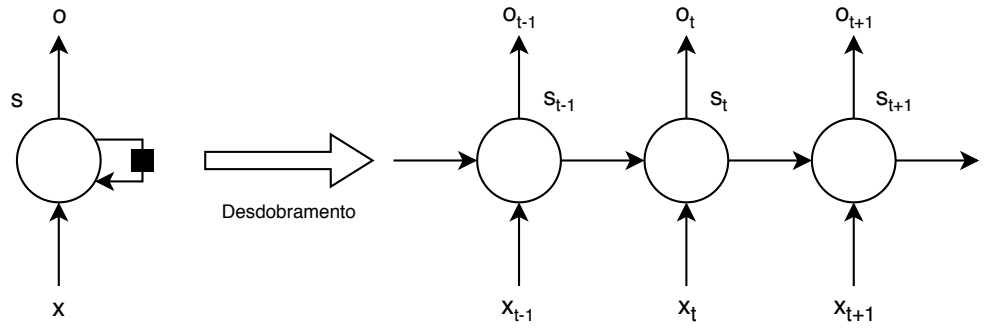

Figura 2.3: Uma rede neural recorrente e o desdobramento no tempo de computação

Durante o processo de treinamento das RNNs, uma função é usada para calcular o erro gerado pela a rede. Então o erro calculado é usado para estimar um gradiente que é usado para atualizar os pesos da rede, para que menos erros sejam cometidos na próxima iteração. Este gradiente é propagado por toda a rede, da camada de saída para a camada de entrada. Porém, na prática, realizar esse processo através de muitas camadas da rede pode fazer com que o valor de gradiente da função de erro se aproxime de zero, tornando toda a rede difícil de treinar. Esse problema é conhecido como *gradient vanishing problem*. RNN são redes poderosas, porém treiná-las tem se provado problemático por conta do problema do gradiente (24).

2.2 Long short-term memory (LSTM)

Long short-term memory (LSTM), proposta por Hochreiter e Schmidhuber em 1997 (22), é uma variante de RNN projetado para lidar com o *gradient vanishing problem*. LSTMs também tornam possível que RNNs lembrem de suas entradas por um maior período de tempo. Isso se deve ao fato que LSTMs contêm uma memória interna especial, que é parecida com a memória de um computador onde é possível ler, gravar e excluir e informações de computações anteriores.

Essa memória também pode ser vista como uma unidade composta por três portas multiplicativas, que controlam o fluxo da informação na rede, em que se pode esquecer informações (excluir) ou passar adiante para a próxima iteração. A Figura 2.4 descreve a estrutura básica de uma unidade de LSTM.

As três portas (*input gates, forget gates e output gates*) são combinadas para calcular o *hidden state* usando as seguintes equações (26):

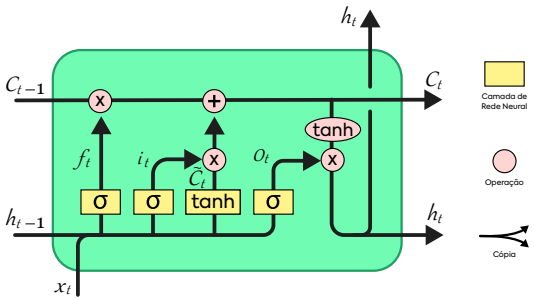

Figura 2.4: Célula LSTM. Adaptada de (56)

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f) \tag{2-1}$$

$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i) \tag{2-2}$$

$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o) \tag{2-3}$$

$$c_t = f_t \odot c_{t-1} + i_t \odot tanh(W_c \cdot [h_{t-1}, x_t] + b_c) \tag{2-4}$$

$$h_t = o_t \odot tanh(c_t) \tag{2-5}$$

O primeiro passo realizado dentro célula de LSTM é decidir quais informações serão descartadas da célula. Esta decisão é feita por uma camada sigmóide chamada de "*forget gate*". O calculo é feito na Equação 2-1, levando em conta o *hidden state* anterior h_{t-1} e valor x_t corrente. Uma saída de valor 1 representa "definitivamente mantenha esse valor" e uma saída de valor 0 representa "definitivamente esqueça esse valor".

O próximo passo é decidir quais novas informações serão armazenadas na célula. Primeiro, uma camada sigmóide chamada "*input gate*" decide quais valores serão atualizados, o calculo do "*input gate*" é feito na Equação 2-2. Depois, o valor c_t corrente é atualizado usando o valor de c_{t-1} anterior e o valor do *forget gate*, descrito na equação 2-4.

Finalmente, é calculado quais valores serão passados adiante na célula. Primeiro, é calculado o valor para "*output gate*" definido na equação 2-3. Após isso, o valor do *hidden state* atual h_t é dado pela aplicação de uma função tanh no valor corrente de c_t multiplicado por o *output gate*, como descrito na Equação 2-5.

Redes LSTM são aplicadas em grande uma variedade de atividades. Por exemplo, para reconhecimento facial em imagem (40), para analisar expressões faciais de indivíduos (53), para transcrever em linguagem natural cenas de vídeos (39), e até mesmo para gerar poemas de forma automática (38).

2.3 Bidirectional Long short-term memory (BLSTM)

Em muitas tarefas com estruturas sequenciais, fazer o uso de informações passadas (à esquerda) e futuras (à direita) da sequência se torna uma vantagem. Porém, a arquitetura LSTM padrão conhece apenas informações de computações prévias (passado), e não conhece nada do futuro da sequência. Então, LSTM bidirecional (BLSTM) se apresenta como uma solução elegante para esse problema. A ideia é computar a sequência de frente para trás e de trás para frente, para capturar informações passadas e futuras (25). Tipicamente, BLSTM são mais poderosas que redes LSTM (41).

Inventada em 1997 por Schuster e Paliwal (54), as BLSTM aumentam a quantidade de informação de entrada disponível para a rede. Os neurônios são divididos em duas direções *forward states* e *backward states*. Na Figura 2.5 temos uma ilustração do processamento de sequência com uma RNN unidirecional e uma RNN bidirecional.

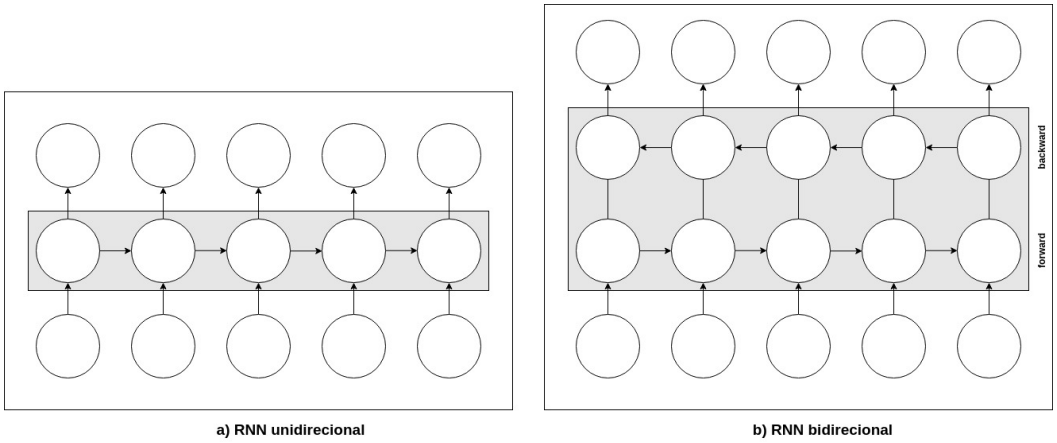

Figura 2.5: Estrutura de RNN unidirecional e bidirecional

2.4 Dropout

Redes neurais com um grande número de parâmetros são técnicas de aprendizado de máquina muito poderosas. No entanto, o *overfitting* é considerado um problema sério em redes neurais. O *dropout* é uma técnica que se propõe mitigar esse problema. A ideia é descartar aleatoriamente nós (juntamente com suas conexões) da rede durante o treinamento. Isso evita que os nós se adaptem demasiadamente aos dados de treino (66).

Na Figura 2.6 temos a ilustração da aplicação de *dropout*. Os nós são retirados de forma aleatória. Nos casos mais simples da aplicação de *dropout*, cada nó é retirado segundo uma probabilidade p independente dos outros

nós. Essa probabilidade p é geralmente determinada usando o conjunto de validação. Segundo (66) 0.5 é um bom palpite para a escolha de p, pois é próximo da escolha ótima de uma grande variedade de redes neurais e tarefas.

O *dropout* é implementado como uma camada de rede neural. Ele pode ser usado combinado a vários tipos de camadas, como camadas densas totalmente conectadas, camadas convolucionais e camadas recorrentes, como LSTM e BLSTM. O *dropout* pode ser implementado em algumas camadas da rede ou até mesmo em todas as camadas escondidas. Geralmente não se usa *dropout* na camada de saída (60).

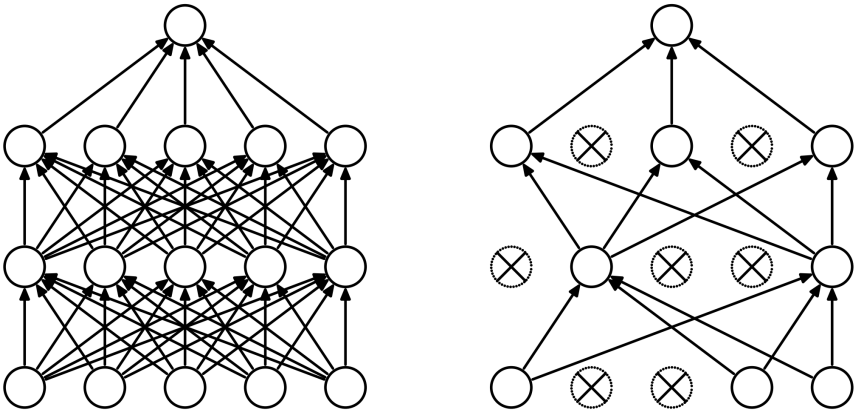

Figura 2.6: **a)** Uma rede neural padrão. **b)** Um exemplo de uma rede produzida pela aplicação de *dropout* na rede à esquerda. (66)

3
Processamento de Linguagem Natural

Processamento de linguagem natural (PLN) é uma área de pesquisa que explora como computadores podem usar, entender e manipular a linguagem natural, seja ela escrita ou falada (2). PLN é uma área multidisciplinar que envolve campos de estudo como linguística, computação e estatística (4). Coppin e Ben (3) afirma que o objetivo final da área de pesquisa em PLN é construir sistemas com conhecimento suficiente para falar com humanos sobre qualquer assunto.

É cada vez mais importante que existam sistemas que possam compreender a linguagem natural de forma automática (3), pois o volume de dados que carregam linguagem natural está alcançando proporções de difícil gerenciamento. Por exemplo, por minuto na internet, mais de 16 milhões de mensagens de texto instantâneas são enviadas, mais de 156 milhões de e-mails são enviados (espera-se que até o fim de 2019 mais 9 bilhões de e-mails já tenham sido enviados) e são realizadas 154.200 chamadas no Skype (7).

A linguagem humana é complexa e diversa, tornando o processamento automático de linguagem uma tarefa não trivial. A ambiguidade da linguagem é um dos principais desafios em PLN.

Atualmente, computadores já são capazes de realizar uma grande variedade de atividades que envolvem linguagem natural, como por exemplo: reconhecimento de linguagem falada, sistemas de conversação automáticos, análise léxica, tradução automática, recuperação de informação, sistemas de perguntas e respostas, analise de sentimentos, geração de linguagem natural e sumarização de linguagem.

Para realizar tarefas tão complexas, as aplicações de PLN geralmente usam no processo de construção outros problemas mais fundamentais de PLN, como por exemplo: classificação gramatical de palavras (*POS tagging*), análise de sentenças e suas estruturas gramaticais, identificação de correlação de menções, reconhecimento de entidades nomeadas, entre outras.

As primeiras técnicas usadas em aplicações de PLN eram baseadas em regras. Cientistas escreviam de forma manual as regras de linguagem. Esse processo se provou uma tarefa difícil devido a ambiguidade presente na linguagem. Com o passar dos anos, PLN foi se transformando, e passaram

a usar com sucesso métodos estatísticos e probabilísticos, juntamente com algoritmos de aprendizado de máquina (6).

Ao longo do tempo, em PLN, várias abordagens de aprendizado de máquina vêm sendo usadas como por exemplo: *naive bayes* (8), *hidden markov models* (9), *support vector machines* (10). Entretanto, durante os últimos anos, vem acontecendo uma nova transformação no campo de PLN, e muitos problemas que antes eram encarados com abordagem de aprendizado de máquina tradicional, agora apresentam bons resultados com aplicação de redes neurais (11).

3.1 Processamento de Linguagem Natural e Redes Neurais

Arquiteturas e algoritmos de redes neurais proporcionaram um grande avanço em áreas do conhecimento, como visão computacional e reconhecimento de padrões. Seguindo essa tendência, recentemente, pesquisadores de PLN estão voltando a atenção para o uso de técnicas de redes neurais. Nos últimos anos, redes neurais combinadas a representações vetoriais de palavras estão produzindo ótimos resultados em muitas atividades de PLN (26).

Em algoritmos de aprendizado de máquina tradicionais, as *features* são projetadas de forma manual, e engenharia de *features*, em muitos casos, é um gargalo pois requer conhecimentos bem específico de domínio. Deng e Liu (57) afirmam que engenharia de *features* era o maior obstáculo em PLN antes do advento de *deep learning*.

As técnicas atuais de *deep learning* contornam o problema de engenharia de *features* explorando poderosas redes neurais de múltiplas camadas que dispensam a necessidade da criação de *features* manuais. Ao contrário de algoritmos de aprendizagem de máquina mais tradicionais, algoritmos de *deep learning* são capazes de aprender representações a partir de dados usando uma hierarquia de múltiplas camadas para a geração automática de *features* (57).

Essa tendência surgiu, em parte, em 2011 com o trabalho pioneiro de Collobert el al. (42). Os autores aplicaram uma rede neural simples em vários problemas de PLN obtendo resultados que superavam o atual estado da arte em *POS tagging*, reconhecimento de entidade nomeada e anotação de papéis semânticos. Desde então redes neurais mais elaboradas vêm sendo empregas em inúmeros problemas de PLN.

3.2 Representação vetorial de palavras

Representação vetorial são basicamente uma forma de representar palavras que liga a compreensão humana da linguagem à de uma máquina. Essas representações são atualmente essenciais para resolver a maioria dos problemas da PLN (58).

3.2.1 *Word Embeddings*

Word Embeddings propõe representar cada palavra como um vetor de números reais em um espaço D-dimensional. Isso permite que palavras com significados similares tenham representações similares. A representação de *Word embeddings* captura informações semânticas e sintáticas das palavras (28).

Essa técnica de representação surge para combater a "maldição da dimensionalidade" (*curse of dimensionality*) (29), já que é possível representar as palavras de um vocabulário de tamanho V com vetores D-dimensionais, em que D pode ser muito menor que V ($D << V$).

É possível medir similaridade entre vetores de palavras usando funções de distância, como distância de cosseno, por exemplo. Também é possível realizar operações de soma e subtração entre pares de vetores e obter resultados com um significado semântico. Podemos, por exemplo subtrair os vetores vec("rei") e vec("homem") e somar com o vetor vec("mulher") e obter um resultado próximo do vetor vec("rainha"). Na Figura 3.1, temos a ilustração da operação de exemplo realizada na Equação 3-1.

$$vec(\text{“}rei\text{”}) - vec(\text{“}homem\text{”}) + vec(\text{“}mulher\text{”}) \approx vec(\text{“}rainha\text{”}). \quad (3\text{-}1)$$

Word embeddings são bastante utilizados como a primeira camada de redes neurais. Normalmente esses vetores são pré-treinados em grande corpus não-supervisionados. (26). Existem uma boa variedade de algoritmos para gerar *word embeddings*: Word2vec (32), FastText (33), Wang2vec (34) e Glove (31) são alguns deles.

As representações de palavras por meio de *word embeddings* são bastante utilizadas na construção de sistemas de PLN (27) essa técnica vem sendo aplicada com bastante sucesso (25) na área.

Por exemplo, Ma e Hovy (25) propuseram uma arquitetura de rede neural que utiliza informações de *word embedding* para resolver de forma eficiente *POS tagging* e reconhecimento de entidades nomeadas. Dickinsonand e Hu (35)

usaram *word embedding* junto com algoritmos de aprendizagem de máquina para identificar sentimentos em tweets.

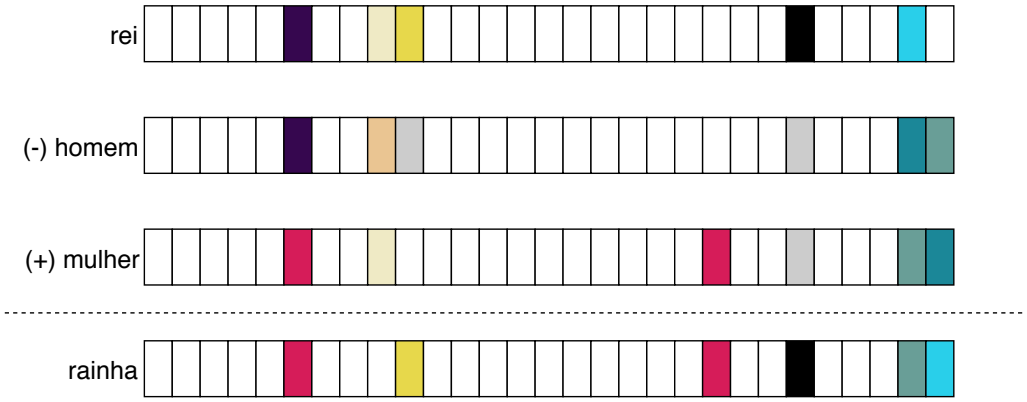

Figura 3.1: Exemplo de operação entre vetores, adaptada de (26)

3.2.1.1 Word2vec

Em 2013, Mikolov et al. (32) propuseram o Word2Vec, um modelo para construção de *word embedding*. Word2vec tem dois modelos de treinamento diferentes: (i) *Continuous Bag-of-Words* (CBOW) e (ii) *skip-gram*.

O modelo CBOW calcula a probabilidade condicional de uma palavra alvo específica, dadas as palavras de contexto que a rodeiam, através de uma janela de contexto de tamanho C. O modelo *skip-gram* faz exatamente o oposto do modelo CBOW, prevendo as palavras do contexto dado à palavra alvo específica.

O modelo CBOW é formado de uma simples rede neural *fully connected* com uma camada escondida, como exemplificado na Figura 3.2. A camada de entrada, que recebe os vetores das palavras do contexto, tem V(tamanho do vocabulário) neurônios enquanto a camada oculta tem N neurônios. A camada de saída é uma função *softmax* de todas as palavras no vocabulário. As camadas são conectas pelas matrizes de pesos W^{VxN} e W'^{NxV} (26) (60).

O modelo *skip-gram* usa a palavra alvo (cujo a representação queremos gerar) para prever as palavras do contexto e no processo, produz a representação vetorial da palavra. Na Figura 3.3 temos a representação do modelo *skip-gram* que também usa uma rede neural *fully connected* com três camadas.

Ambos modelos têm suas vantagens e desvantagens. De acordo com Mikolov (32), o modelo *skip-gram* funciona bem com pequenas quantidades

de dados e representa bem as palavras raras. Entretanto, o modelos CBOW é mais rápido e gera melhores representações para palavras mais frequentes (60).

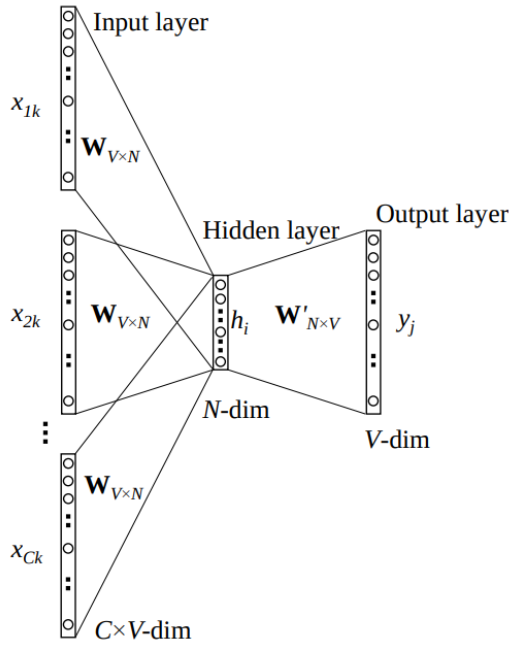

Figura 3.2: Modelo CBOW de (59)

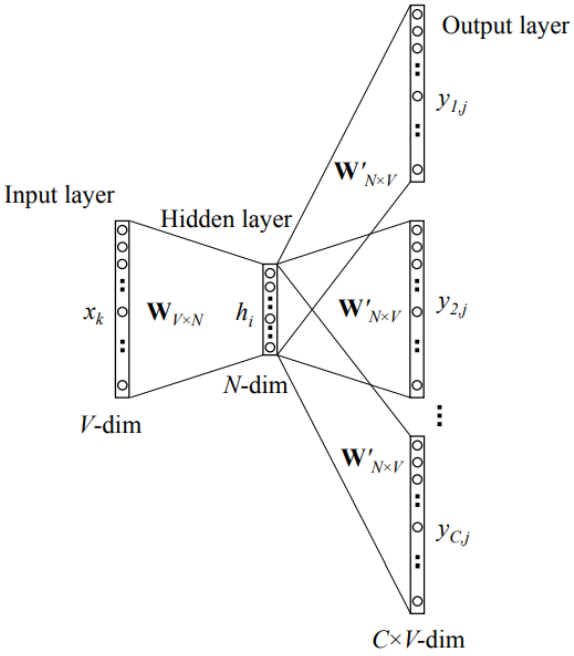

Figura 3.3: Modelo *Skip Gram* de (59)

3.2.1.2 Glove

O método de geração de *word embeddings Global Vectors* (GloVe) foi proposto em 2014 por Pennington et al. em (31). Os autores observam que métodos baseados em janelas de contexto (CBOW e *skip-gram*) sofrem com a desvantagem de não aprender com estatísticas globais do corpus.

Para gera a representação de palavras usando estatísticas globais, o método Glove usa uma matriz de co-ocorrência M que é construída usando as palavras do contexto. Cada elemento M_{ij} na matriz representa a probabilidade da palavra i estar perto da palavra j. Na matriz M, as linhas (ou vetores) são geradas aleatoriamente e treinado por obedecer a equação $P(w_i, w_j) = log(M_ij) = w_i w_j + b_i + b_j$ em que w_i e w_j são vetores e b_i e b_j são os *biases* (27).

3.2.1.3 Wang2Vec

O modelo Wang2Vec foi proposto por Wang Ling et al. (34) em 2015. Wang2Vec é uma modificação do Word2Vec feita a fim de considerar a ordem das palavras, o que não é considerado na arquitetura original. As modificações geram representações vetoriais que carregam um maior valor sintático. Foram feitas duas pequenas modificações. No modelo CBOW, a entrada passa a ser a concatenação de todas as palavras do contexto na exata ordem que elas aparecem. No modelo *skip-gram*, é usado um conjunto diferente de parâmetros para prever cada palavra de contexto, dependendo de sua posição em relação à palavra alvo (27).

3.2.1.4 FastText

O modelo FastText foi proposto em 2017 por Bojanowski et al. em (33). Os autores propuseram uma nova abordagem baseada no modelo *skipgram* de (32), onde cada palavra é representada como um conjunto de *n-grams* de caracteres. A representação vetorial de uma palavra específica é formada pela soma das representações dos *n-grams* que compõem a palavra. Com isso o modelo gera vetores de palavras que tem informações morfológica mais ricas.

3.2.2 Character Embeddings

Character embeddings assim como *word embeddings*, é uma forma de representar palavras em um espaço vetorial de baixa dimensionalidade. Enquanto *word embeddings* são capazes de capturar informações sintáticas e semânticas das palavras, *character embeddings* capturam informações morfológicas e informações estruturais das palavras.

Informações morfológicas de palavras podem ser úteis em tarefas de classificação como *POS tagging*. Prefixos, sufixos, letras maiúsculas, entre outros, podem ajudar a distinguir a classe gramatical de palavras. Por exemplo, o sufixo "mente" da palavra "calmamente" pode indicar a presença de um advérbio. A letra maiúscula em "Maria" indica a presença de um substantivo.

Aplicações que trabalham com idiomas que tem um vocabulário grande, frequentemente lidam com problema de palavras desconhecidas ou palavras fora do vocabulário (*out-of-vocabulary*). Os *character embeddings* tratam bem esse problema, pois uma palavra fora do vocabulário pode ser representada simplesmente como a composição individual de seus caracteres (26).

Character embeddings têm atraído a atenção de pesquisadores, Santos e Gatti (36) usaram a representação de *character embedding* para classificar sentimentos em textos. A representação de *character embedding* foi construída usando uma rede neural convolucional. Kim et el. (37) construíram uma rede neural que usa apenas a representação de *character embedding*. Eles conseguiram resultados competitivos no problema de modelagem de linguagem (prever a probabilidade de uma sequência de *tokens* pertencer a um idioma).

4
Trabalhos Relacionados

Este capítulo apresenta os trabalhos relacionados mais relevantes sobre *POS tagging*, na seção 4.1 são apresentados os principais sistemas publicados destinados a Língua Portuguesa, e na seção 4.2 sistemas destinados a outras línguas.

4.1
POS tagging para Português

No ano 2000 Finger (61) apresentou um *POS tagger* para Português baseado em regras que foi construído usando o Corpus Tycho Brahe de Português Histórico. Finger reportou uma acurácia de 95.43%.

"Basicamente, trata-se de um etiquetador baseado em regras transformacionais, isto é, regras que sucessivamente transformam uma etiquetagem inicial. Tais regras são aprendidas na fase de treinamento a partir de um corpus etiquetado manualmente. Este etiquetador baseia-se em um método proposto por Brill (62) (63). Um exemplo de tal regra seria: Mude a etiqueta da palavra corrente de VB (verbo) para N (nome) se a etiqueta da palavra anterior é D (determinante)." (Finger, 2000) (61)

Em (64) no ano de 2006, os autores propuseram um *POS tagger* para o Português. O etiquetador foi construído usando cadeias de Markov de alcance variável. Eles também usaram o corpus Tycho Brahe de Português Histórico e obtiveram 95.51% de acurácia na avaliação.

Em 2008 Santos, Milidiú e Rentería (65) aplicaram o *Entropy Guided Transformation Learning (ETL)* para resolver o problema de *POS tagging*. O algoritmo ETL consistem em combinar o uso de uma árvore de decisão com o algoritmo de classificação baseado em regras transformacionais de Brill (63). Os autores realizaram os experimentos usando o corpus Mac-Morpho v1 e o corpus Tycho Brahae, reportando uma acurácia de 96.75% e 96.64% para Mac-Morpho v1 e Tycho Brahae, respectivamente.

Fonseca e Rosa em 2013 (48), propuseram uma nova versão do corpus Mac-Morpho. Eles removeram erros do corpus, realizaram limpeza nos dados, juntaram algumas etiquetas. Os autores então treinaram um *POS tagger*

usando uma rede neural perceptron multicamadas. Eles reportaram uma acurácia de 96.48% para o Mac-Morpho revisado (v2).

Em (43) no ano de 2014, os autores usaram um perceptron estruturado de margem máxima para resolver o problema de *POS tagging*. Eles usaram um pequeno conjunto de *features* feitas à mão para melhorar o desempenho da aplicação. O classificador foi testado em duas corpora em Português: a versão original do Mac-Morpho corpus (v1) e na versão revisada do Mac-Morpho corpus (v2). Eles obtiveram o melhor resultado da época em acurácia no Mac-Morpho v1 (97.18%) e resultados bem competitivos com o Mac-Morpho v2 (97.25%).

O modelo proposto por Dos Santos e Zadrozny em 2014 (44) para resolver *POS tagging* foi construído sem nenhum tipo de *features* feitas à mão. Eles usaram uma arquitetura de rede neural que combina vetores de *word embeddings* com vetores de *character embeddings* para representar as palavras. Os vetores de *word embeddings* foram pré-treinados usando Word2Vec, e os *character embeddings* foram gerados a partir de camada de rede neural convolucional. Os autores avaliaram a arquitetura de rede em três corpora em Português: Mac-Morpho v1, Mac-Morpho v2 e Tycho Brahe corpus. E reportaram acurácia de 97.47%, 97.31% e 97.17% para Mac-Morpho v1, Mac-Morpho v2 e Tycho Brahae, respectivamente.

Em 2015, Fonseca et al. (45) usaram uma rede perceptron multicamadas para o *POS tagging*. Essa rede usa vetores de *word embeddings* e algumas *features* manuais. Os autores apresentaram uma nova versão do corpus Mac-Morpho, com a correção de alguns erros e a junção de algumas classes. E ainda, compararam diferentes formas de gerar a representação de *word embeddings*. Fonseca et al (45) reportaram o estado da arte para três corpora em Português. Com 97.57% de acurácia no Mac-Morpho v1 corpus, 97.48% no Mac-Morpho v2, e 97.33% no Mac-Morpho v3.

Na Tabela 4.1 temos um resumo dos trabalhos relacionados apresentados nessa seção. Na tabela marcamos se o trabalho usou *features* manuais, vetores de *word embeddings* ou vetores character embeddings. Ainda apresentamos quais algoritmos foram usados e quais os resultados reportados em acurácia.

Tabela 4.1: Trabalhos relacionados para *POS tagging* em Português

	Features				
	Manual	**Word e.**	**Char. e.**	**Algoritmos**	**Avaliação (Ac. %)**
Finger, 2000	x			Regras Transformacionais	Tycho Brahe: 95.43
Kepler e Finger, 2006	x			Cadeias de Markov	Tycho Brahe: 95.51
Santos, Milidiú e Rentería, 2008	x			Regras Transformacionais/ Árvore de Decisão	Tycho Brahe: 96.64 Mac-Morpho v1: 96.75
Fonseca e Rosa, 2013	x			Perceptron Multicamadas	Mac-Morpho v2: 96.48
Fernandes et al, 2014	x			Perceptron Estruturado de Margem Máxima	Mac-Morpho v1: 97,18 Mac-Morpho v2: 97.25
Santos e Zadrozny, 2014		x	x	CNN/ Algoritmo de Viterbi	Mac-Morpho v1: 97.47 Mac-Morpho v2: 97.3 Tycho Brahe: 97.17
Fonseca et al, 2015	x	x		Perceptron Multicamadas	Mac-Morpho v1: 97.57 Mac-Morpho v2: 97.48 Tycho Brahe: 96.91

4.2 *POS tagging* para outros idiomas

Giménez e Marquez (13) apresentaram em 2004 uma ferramenta para *POS tagging* construída usando o algoritmo de máquina de vetores de suporte. Eles aplicaram um rico conjunto de *features* manuais no processo de aprendizagem e conseguiram resultados competitivos para inglês no corpus Wall Street Journal e para espanhol no corpus LEXESP.

Em (25) os autores propuseram uma arquitetura de rede neural para rotulagem de sequência sem usar *features* manuais. Primeiro, uma camada de rede neural convolucional foi usada para extrair uma representação de *character embeddings*. E então, essa representação foi concatenada com informações de *word embeddings* para ser usada em uma camada de BLSTM. E, em cima da camada de BLSTM, foi usado uma camada de CRF.

A arquitetura de (25) foi testada em duas atividades de rotulagem de sequência: em *POS tagging* com o corpus Wall Street Journal e em reconhecimento de entidade nomeada (REN) com o corpus CoNLL 2003. Eles reportaram estado da arte em acurácia para *POS tagging* e REN.

Em 2018 (46) Marulli et al. usaram uma arquitetura de rede neural para encarar o problema. Eles aplicaram uma camada de BLSTM para extrair as informações de *character embeddings* e então concatenaram com informações de *word embeddings* para então alimentar uma outra camada de BLSTM com essas informações. A arquitetura foi avaliada usando um corpus em Italiano. Os autores reportaram uma acurácia de 98.02%. Eles ainda avaliaram o desempenho do word2vec e FastText, dois algoritmos para calcular *word embeddings*. O modelo que usou o FastText teve um melhor desempenho em acurácia geral e acurácia das palavras fora do vocabulário.

Bohnet et al. (1) reportaram o atual estado da arte para POS Tagging na Língua Inglesa. Eles usam o corpus Penn Treebank WSJ e atingiram 97.96% de

acurácia. Os bons resultados são obtidos com a integração das representações vetoriais de palavra e de caracteres. Um meta-modelo aprende como combinar essas duas representações através de um treinamento sincronizado.

Na Tabela 4.2 temos um resumo dos trabalhos relacionados apresentados nessa seção. Na tabela marcamos se o trabalho usou *features* manuais, vetores de *word embeddings* ou vetores character embeddings. Ainda apresentamos quais algoritmos foram usados e quais os resultados reportados em acurácia.

Tabela 4.2: Trabalhos relacionados para *POS tagging* em outros idiomas

	Features				
	Manual	**Word e.**	**Char. e.**	**Algoritmos**	**Avaliação (Ac. %)**
Giménez e Marquez, 2004	x			SVM	WSJ: 97.16 LEXESP: 96.89
Ma e Hovy, 2016		x	x	CNN / BLSTM / CRF	WSJ: 97.55
Marulli et al., 2018		x	x	BLSTM / BLSTM	UD V2.1 It: 98.00
Bohnet et al., 2018		x	x	Meta BiLSTM	WSJ: 97.96

5
Metodologia

Esse trabalho se propõe construir um *POS tagging* para a Língua Portuguesa usando redes neurais recorrentes que já se estabeleceram como estado da arte para o *POS tagging* em outros idiomas. Nessa abordagem não usamos *features* construídas manualmente ou informações externas ao conjunto de dados. A arquitetura é testada em três *datasets* que abrangem o Português Contemporâneo e o Português Histórico.

5.1
Arquitetura de Rede Neural

A arquitetura usada nesse trabalho é inspirada nos trabalhos de Ma e Hovy de 2016 (25) e Marulli et al. de 2018 (46). Na Figura 6.9, temos a ilustração da arquitetura de rede usada para treinar o modelo de *POS tagging*. Seja a sentença S, decomposta em n *tokens* $t_1, t_2, ..., t_n$ com suas n etiquetas correspondentes $e_1, e_2, ..., e_n$, nós treinamos uma camada de BLSTM para predizer a probabilidade de um *token* pertencer a cada etiqueta. Com isso atribuímos ao *token* t_x a etiqueta e_y de maior probabilidade associada.

Para montar a arquitetura ilustrada na Figura 5.1, é preciso representar cada *token* da sentença como um vetor no espaço R^d, esse vetor de representação é resultado da concatenação da representação de *word embeddings* e da representação de *character embeddings*. Com isso, é possível capturar das palavras informações sintáticas e semânticas (*word embeddings*) e informações morfológicas (*character embeddings*).

A representação de *word embeddings* para um *token* t_x é fornecida por um modelo, já pré-treinado em grandes corpora não supervisionados. A representação de *character embeddings* para um *token* t_x é gerada por uma outra camada de BLSTM. Mais formalmente, como ilustrado na Figura 5.2, a entrada para a camada de BLSTM é um único *token* t_x, e a saída é um vetor v-dimensional de *character embeddings*.

Para gerar a representação de *character embeddings*, é criado um vocabulário V de caracteres que contêm todas a letras maiúsculas, minúsculas e também números e pontuações presentes no corpus de treino. Um *token* t_x de entrada é decomposto em uma sequência de m caracteres $c_1, c_2, ..., c_m$, onde m

representa o tamanho do *token* t_x. Cada caracteres c_i é representado como um vetor de zeros 1_{ci}, com o valor 1 presente no índice que representa o caractere c_i no vocabulário V. Finalmente, a representação do *token* t_x é definida como a concatenação dos estados de *forward* e *backward* provenientes da camada de BLSTM.

Com os *tokens* devidamente representados, as representações de *word embeddings* e *character embeddings* são concatenadas e então passadas para a camada principal de BLSTM. Uma função de ativação *Softmax* é usada no topo da arquitetura, para gerar as probabilidades para as etiquetas.

Durante o treino dessa arquitetura usamos a função de erro *sparse categorical crossentropy*. Essa função compara os valores preditos pela a rede neural, com os valores de classes reais para calcula a perda. A função *sparse categorical crossentropy* é geralmente usada em problemas de classificação em que as classes são mutualmente exclusivas (cada exemplo só pode assumir uma classe por vez). Na Equação 5-1 temos a definição desse função, onde S representa o conjunto de exemplos e C o conjunto de classes possíveis. O $log(s \in c)$ representa a probabilidade predita pelo modelo do s-ésimo exemplo pertencer a c-ésimo classe.

$$-\frac{1}{N}\sum_{s\in S}\sum_{c\in C} 1_{s\in c} log(s \in c) \tag{5-1}$$

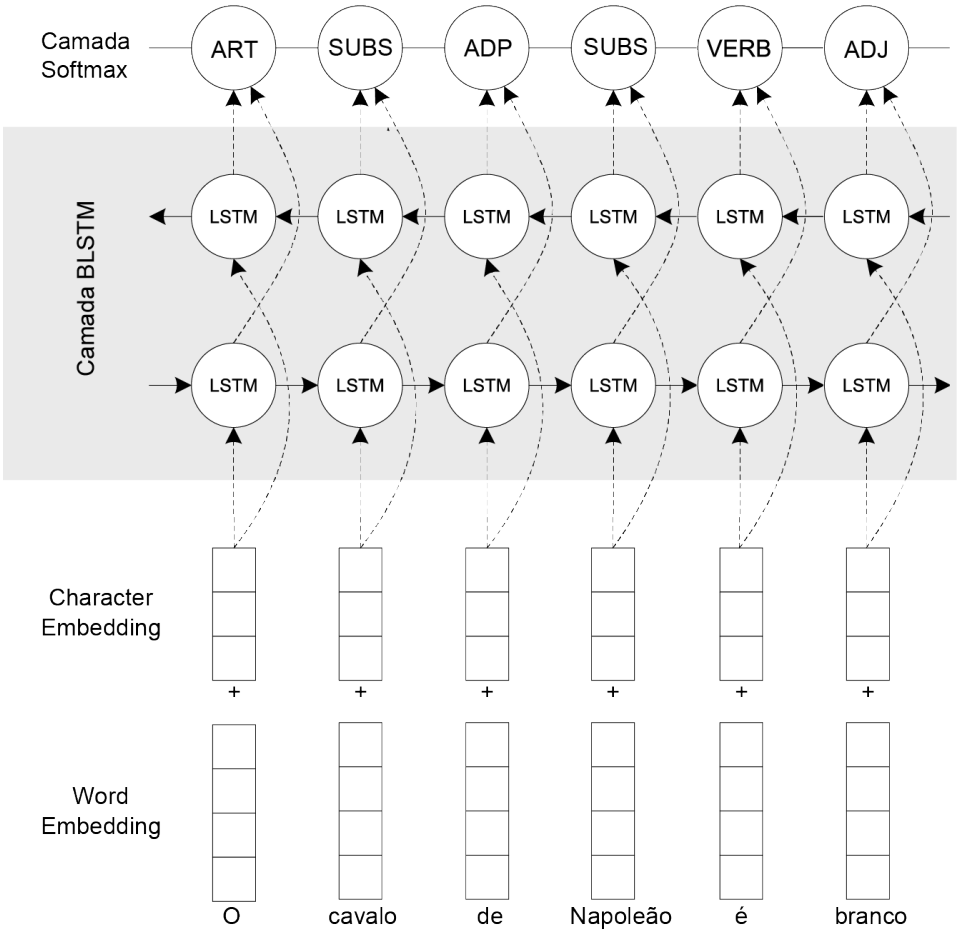

Figura 5.1: Modelo **BLSTM-WE-CE**. Arquitetura da rede neural para *POS tagging*.

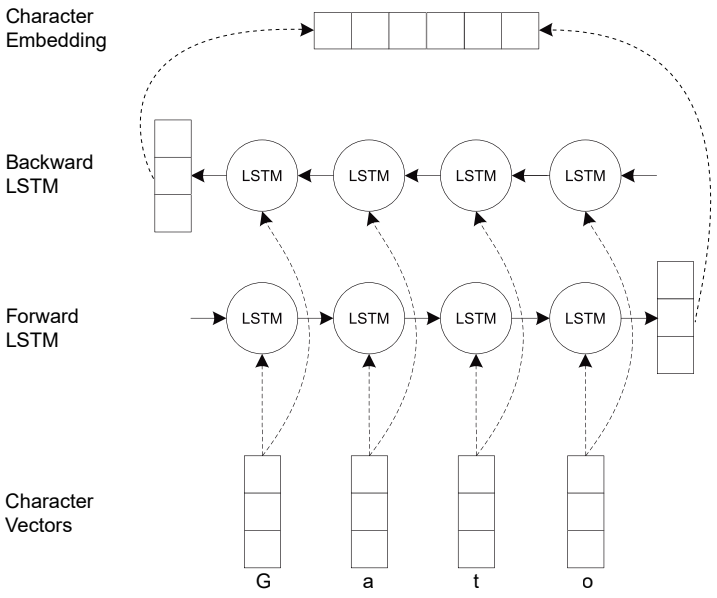

Figura 5.2: Arquitetura do *character embeddings*

6
Experimentos

Neste capítulo descrevemos todos os experimentos realizados, os corpora utilizados e as métricas de avaliação. Usamos os corpora Mac-Morpho v1, Mac-Morpho v2 e Tycho Brahe, todos com as mesmas partições de treino/teste/validação usadas nos trabalhos de Dos Santos e Zadrozny (44) e Fonseca et al. (45). O código dos modelos foram escritos na linguagem Python[1], usando Keras[2], uma biblioteca para rede neural de alto nível. Usamos TensorFlow[3] como *backend* com a biblioteca Keras. Todos os experimentos foram realizados usando um GPU GeForce GTX TITAN X. As arquiteturas de redes neurais descritas nesse Capítulo foram treinadas usando a otimizador "adam" (68) com 5 épocas no treino.

6.1
Corpora

Nós avaliamos o modelo de rede neural em três corpora diferentes: a versão original do Mac-Morpho (47), a versão revisada do Mac-Morpho (48) e o Tycho Brahe (49) corpus. Na Tabela 6.1, mostramos a quantidade de sentenças, *tokens* e classes dos corpora. Usamos as mesmas partições de treino/teste/desenvolvimento que (44) e (45) para comparar os resultados diretamente.

Tabela 6.1: Partições dos Corpora

Corpus		**Treino**	**Validação**	**Teste**	**Total**	**Classes**
Mac-Morpho v1	Sentences	42,022	2,211	9,141	53,374	41
	Tokens	957,439	50,232	213,794	1,221,465	
Mac-Morpho v2	Sentences	42,742	2,249	4,999	49,990	30
	Tokens	807,796	43,141	94,991	945,928	
Tycho Brahe	Sentences	29,162	1,534	10,233	40,929	265
	Tokens	734,889	40,673	259,913	1,035,475	

[1]https://www.python.org/
[2]https://keras.io/
[3]https://www.tensorflow.org/

6.1.1 Mac-Morpho v1

Mac-Morpho é um corpus composto por textos em Português do Brasil com anotação morfossintática dos *tokens*. Os textos foram extraídos de uma coleção textual da Folha de São Paulo[4], que implica num texto contemporâneo de alta qualidade e diferentes autores e domínios (47). O corpus conta com 53.374 sentenças, 1.221.465 *tokens* e 41 classes no total. Na Tabela 6.2 temos a descrição das etiquetas presentes no corpus e alguns exemplos de *tokens* presentes.

Na Tabela 6.3 apresentamos a distribuição dos *tokens* por classe, mostrando a quantidade de exemplos presente nos conjuntos de treino, validação e teste. É possível observar que substantivos (N), preposição (PRED), artigos (ART), substantivos próprio (NPROP) e verbos (V) são as etiquetas mais frequentes. Algumas etiquetas de pontuação não possuem representações no conjunto de validação ou no conjunto de teste, isso torna impossível avaliar o desempenho do modelo para essas classes. Para melhor comparação entre valor, na Figura 6.1, apresentamos a distribuição total das etiquetas mais frequentes no corpus Mac-Morpho v1.

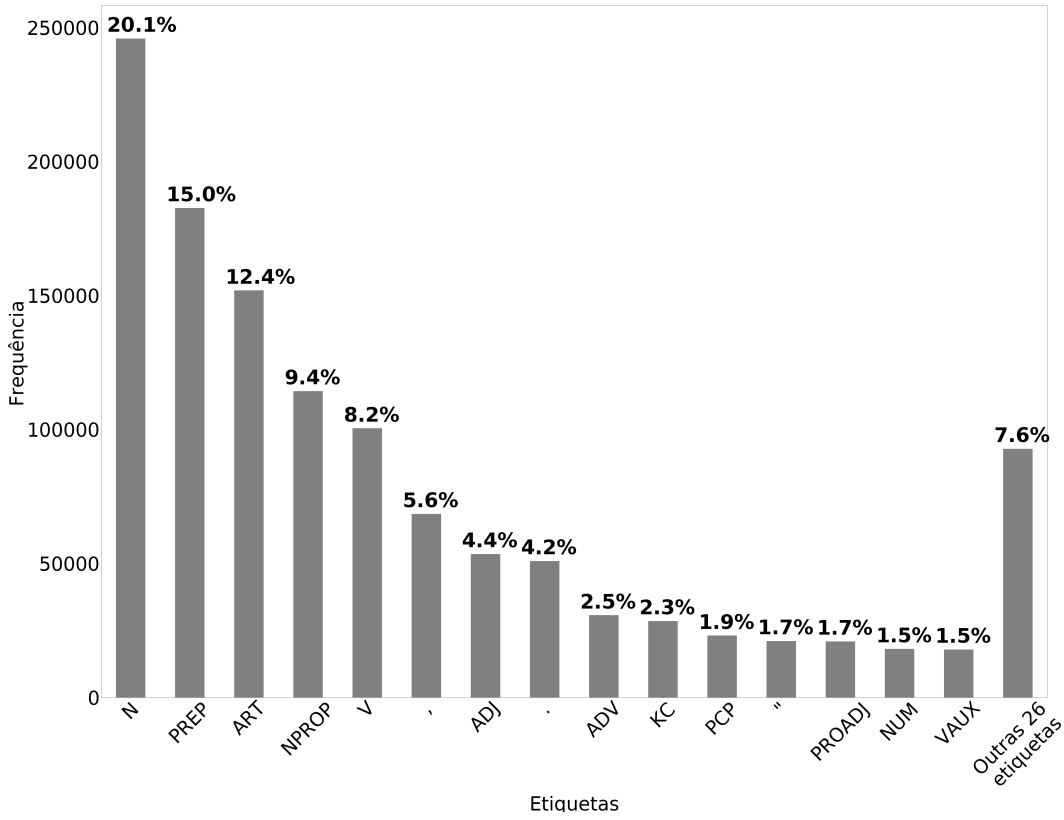

Figura 6.1: Distribuição das etiquetas no Mac-Morpho v1

[4]http://www.folha.uol.com.br/folha

Tabela 6.2: Descrição do corpus Mac-Morpho v1

Etiqueta	**Significado**	**Exemplos**
N	Substantivo	dia, governo , presidente, milhões, país
PREP	Preposição	de, em, a, para, por
ART	Artigo	o, a, as, as, um
NPROP	Substantivos próprio	paulo, são, brasil, eua, rio
V	Verbo	é, foi, tem, disse, são
,	Pontuação	,
ADJ	Adjetivo	maior, grande, primeiro, novo, nova
.	Pontuação	.
ADV	Advérbio	não, mais, amanhã, basicamente, principalmente
KC	Conjunção coordenativa	e, mas, ou, pois, porém
PCP	Particípio	passado, sido, feito, usado, conhecido
"	Pontuação	"
PROADJ	Pronome adjetivo	sua, seu, este, seus, esse
NUM	Numeral	100, 455, dois, mil, um
VAUX	Verbo auxiliar	foi, ser, pode, vai, é
PROPESS	Pronome pessoal	se, ele, eu, eles, ela
KS	Conjunções Subordinativas	que, se, quando, como, porque
PRO-KS-REL	Subordinação do pronome relativo conectivo	que, qual, o, a quais
PROSUB	Pronome nominal	isso, o, um, uma, tudo
)	Pontuação	)
(	Pontuação	(
PDEN	Palavra denotativa	também, só, apenas, mesmo, até
:	Pontuação	:
-	Pontuação	-
CUR	Símbolo de moeda	us, r, cr, rs, u$
PRO-KS	Pronome conectivo de subordinação	que, o, quem, qual, quanto
?	Pontuação	?
;	Pontuação	;
!	Pontuação	!
ADV-KS-REL	Adjetivo relativo de subordinação	onde, quando, que, como, aonde
...	Pontuação	...
	Pontuação	
IN	Interjeição	ora, ah, não, ô, oh
ADV-KS	Advérbio conectivo de subordinação	como, onde, quando, por, que
/	Pontuação	/
$	Pontuação	$
[	Pontuação	[
igual	Pontuação	igual
))	Pontuação	))
((	Pontuação	((
‘	Pontuação	‘

Por conta da ambiguidade presente na linguagem natural, uma única palavra pode assumir diversas classes morfossintáticas dependendo do contexto em que é usada. Por exemplo, na sentença: “Vamos tomar banho de rio?”, a palavra “rio” é usada como substantivo (rio = curso de água fluvial). Na sentença: “Eu rio de suas piadas porque você é hilariante!”, a palavra “rio” é usada como um verbo (rio = verbo rir na 1.ª pessoa do singular).

Na Figura 6.2 temos as 15 palavras do Mac-Morpho v1 com o maior número de classes morfossintáticas atribuídas. A palavra “que” foi rotulada com 17 etiquetas diferentes ao longo do *dataset*, a palavra “a”, “o” e “como” foram associadas com 14, 13 e 12 etiquetas diferentes respectivamente ao longo do *dataset*. Palavras com muitas classes associadas podem ser mais difíceis de classificar.

A Figura 6.3 apresenta uma nuvem de palavras feita com corpus Mac-Morpho v1. Com essa visualização podemos perceber que palavras como “de”,

Tabela 6.3: Distribuição das etiquetas no corpus Mac-Morpho v1

Etiqueta	**Treino**	**Desen.**	**Teste**	**Total**
N	192498	10117	43323	245938 (20.13%)
PREP	142937	7464	32278	182679 (14.96%)
ART	118730	6223	26949	151902 (12.44%)
NPROP	88149	4691	21522	114362 (9.36%)
V	79254	4214	16967	100435 (8.22%)
,	53628	2894	11972	68494 (5.61%)
ADJ	42403	2169	8985	53557 (4.38%)
.	39880	2080	8960	50920 (4.17%)
ADV	24493	1322	4895	30710 (2.51%)
KC	22468	1173	4901	28542 (2.34%)
PCP	18083	861	4148	23092 (1.89%)
"	16305	888	3876	21069 (1.72%)
PROADJ	16621	854	3446	20921 (1.71%)
NUM	13992	749	3371	18112 (1.48%)
VAUX	13908	692	3332	17932 (1.47%)
PROPESS	11889	610	2223	14722 (1.21%)
KS	11380	591	2358	14329 (1.17%)
PRO-KS-REL	8928	436	1983	11347 (0.93%)
PROSUB	6987	368	1243	8598 (0.70%)
)	6063	293	1385	7741 (0.63%)
(	6032	296	1385	7713 (0.63%)
PDEN	5282	297	1209	6788 (0.56%)
:	5511	304	921	6736 (0.55%)
-	2485	120	512	3117 (0.26%)
CUR	2118	103	485	2706 (0.22%)
PRO-KS	1686	80	384	2150 (0.18%)
?	1259	85	166	1510 (0.12%)
;	1120	68	188	1376 (0.11%)
!	860	45	14	919 (0.08%)
ADV-KS-REL	695	42	161	898 (0.07%)
...	538	42	37	617 (0.05%)
’	388	20	102	510 (0.04%)
IN	380	16	20	416 (0.03%)
ADV-KS	295	19	78	392 (0.03%)
/	83	2	10	95 (0.01%)
$	57	3	0	60 (0.0%)
[	18	0	5	23 (0.0%)
=	14	0	0	14 (0.0%)
))	10	0	0	10 (0.0%)
((	10	0	0	10 (0.0%)
‘	2	1	0	3 (0.0%)

“para”, “que”, “em” e “como” são muito frequentes no corpus. Também podemos perceber o tema jornalístico do corpus pela a grande frequência de termos como “disse que”, “políticos”, “brasil”, “milhões de”, “presidente de”,

entre outros.

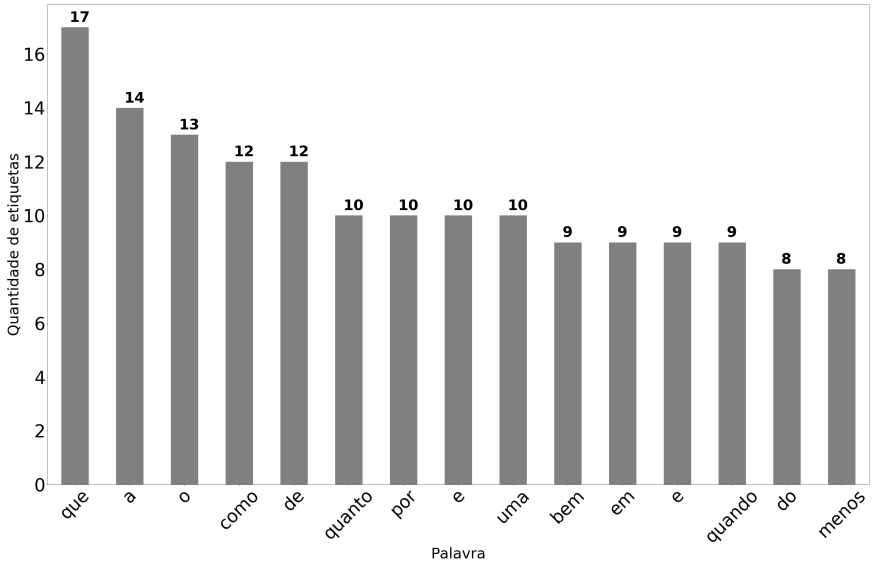

Figura 6.2: Distribuição das palavras com o maior número de diferentes etiquetas associadas. Corpus Mac-Morpho v1.

Figura 6.3: Nuvem de palavras Mac-Morpho v1

6.1.2 Mac-Morpho v2

Mac-Morpho foi revisado em 2013 em (48). A revisão consiste em duas partes principais: limpeza de ruído nos dados e alteração do conjunto de *tags* para incluir contrações de preposições. Foram concatenadas formações como por exemplo: “de + o”, “de + a” que apareciam separadas no corpus, a fim de refletir algo mais parecido com o cenário real. O Mac-Morpho (v2) Revisado

contém 49.990 sentenças, 945,928 *tokens* e 30 classes no total. Na Tabela 6.4 temos as etiquetas presentes no Mac-Morpho v2, seus significados e alguns exemplo de etiquetas. Os *toknes* de pontuação foram reunidos em uma única etiqueta (PU).

Na Tabela 6.5 temos a distribuição das etiquetas nos conjuntos de treino, validação e treino. Substantivos (N), pontuação (PU), substantivos próprio (NPROP), preposição (PRED), verbos (V) e artigos (ART), são as etiquetas mais frequentes no mac-morpho v2. Na Figure 6.4 apresentamos as etiquetas mais frequentes no corpus Mac-Morpho v2.

Também apresentamos a nuvem de palavras para o Mac-Morpho v2 na Figura 6.5. O conteúdo textual dos corpora Mac-Morpho v1 e Mac-Morpho v2 são essencialmente o mesmo, por isso a nuvem de palavras gerada para o Mac-Morpho v2 é muito similar a nuvem do Mac-Morpho v1. Com essa figura podemos ver o carácter jornalístico do corpus Mac-Morpho v2.

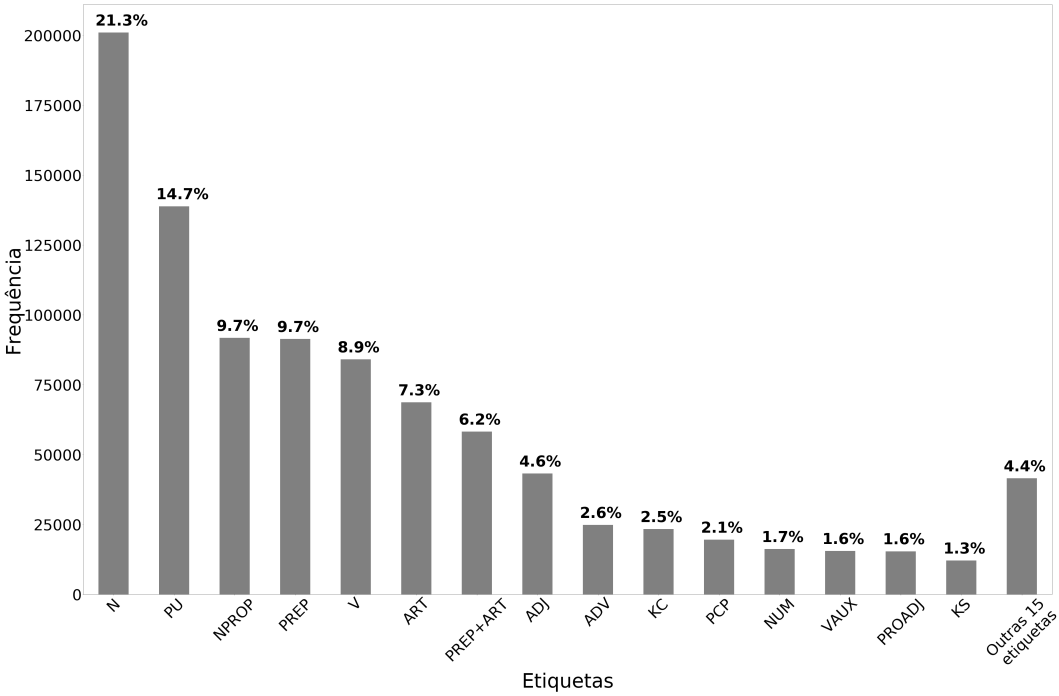

Figura 6.4: Distribuição das etiquetas no Mac-Morpho v2

6.1.3 Corpus Tycho Brahe

O corpus Tycho Brahe foi escrito com um Português Histórico composto por textos em prosa, escritos em Português por falantes nativos, nascidos entre 1550 e 1850. Nesse *dataset* podemos encontrar, por exemplo, sermões do Padre Antônio Vieira, cartas de D. João III e trechos de obras de José de Alencar. Isso implica em um estilo de escrita e vocabulário bem diferente dos que são

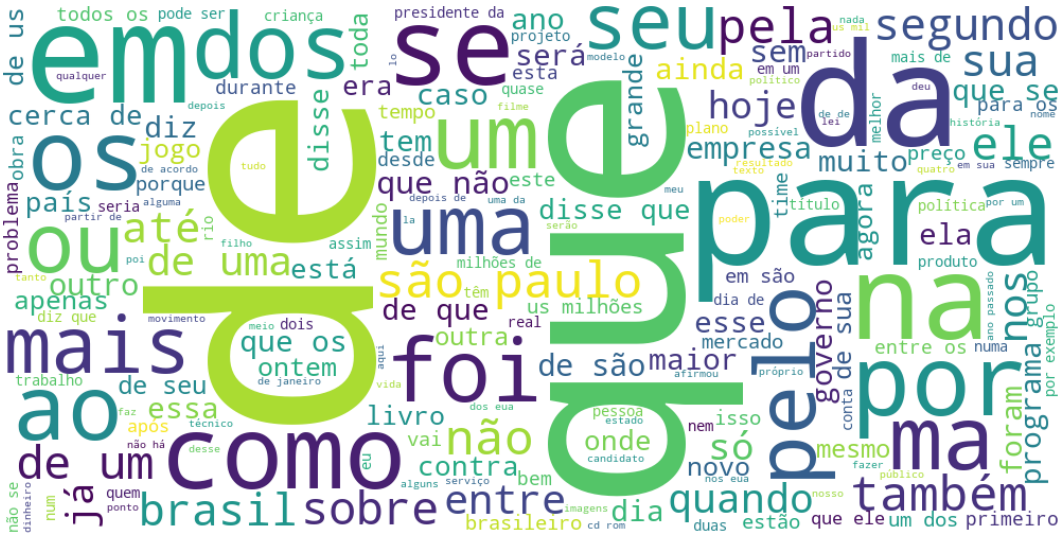

Figura 6.5: Nuvem de palavras Mac-Morpho v2

Tabela 6.4: Descrição do corpus Mac-Morpho v2

Etiqueta	Significado	Exemplos
N	Substantivo	governo, milhões, presidente, país, mercado
PU	Todas as pontuações	. ,"( ) - :
NPROP	Substantivos próprio	silva, santos, sp, pt, fhc
PREP	Preposição	de, em, para, com, a
V	Verbo	estão, ter, afirmou, faz, será
ART	Artigo	o, a, os, um, uma, as, uns, umas
PREP+ART	Preposição + Artigo	do, da, no, na, dos
ADJ	Adjetivo	especial, nacional, segunda, social, anterior, maiores
ADV	Advérbio	arbitrariamente, consideravelmente, predominatemente
KC	Conjunção coordenativa	e, mas, ou, pois, porém, nem
PCP	Particípio	sido, feito, feita, passada, feitas
NUM	Numeral	cinco, 20, 30, uma, 15
VAUX	Verbo auxiliar	foi, ser, pode, vai, é
PROADJ	Pronome adjetivo	sua, seu, seus, mais, suas
KS	Conjunções Subordinativas	que, se, quando, como, porque
PROPESS	Pronome pessoal	me, ele, eu, ela, eles
PRO-KS-REL	Subordinação do pronome relativo conectivo	que, qual, quais, quem, cujo
PROSUB	Pronome nomina	isso, um, o, uma, tudo
PDEN	Palavra denotativa	também, só, apenas, mesmo, até
CUR	Símbolo de moeda	us, r, cr, rs, u$
PRO-KS	Pronome conectivo de subordinação	que, o, quem, qual, quais
PREP+PROADJ	Preposição + pronome adjetivo	deste,neste, nesse, desse, desta
ADV-KS-REL	djetivo relativo de subordinação	onde, quando, que, como, aonde
PREP+PROSUB	Preposição + pronome nomina	disso, do, ao, dos, à
PREP+PROPESS	Preposição + pronome pessoal	delas, dela, comigo, nele, nela
ADV-KS	Advérbio conectivo de subordinação	como, onde, quando, por, que
IN	Interjeição	ah, ora, não, né, ô
PREP+PRO-KS-REL	Preposição + subordinação do pronome relativo conectivo	no, na, dos, pelo, da
PREP+ADV	Preposição + Advérbio	daí, daqui, dali
PREP+PRO-KS	Preposição + pronome conectivo de subordinação	do, no, pelo, ao

usados no Português Contemporâneo apresentado no Mac-Morpho v1 e no Mac-Morpho v2.

O corpus contém 40.929 sentenças, 1.035.475 *tokens* e 265 classes. Esse corpus possui uma quantidade muito maior de etiquetas pois inclui informações morfológicas como gênero, número e tempo verbal (49). Com esse conjunto de etiquetas podemos especificar, por exemplo, o tempo verbal empregado em uma palavra: verbo no infinitivo (VB), verbo no presente (VB-P) ou verbo no passado (VB-D). Também é possível distinguir entre forma e gênero de adjetivos: adjetivo masculino singular (ADJ-S), adjetivo feminino singular (ADJ-S-F), adjetivo masculino plural (ADJ-S-P) e adjetivo feminino plural

Tabela 6.5: Distribuição das etiquetas no corpus Mac-Morpho v2

Etiqueta	**Treino**	**Desen.**	**Teste**	**Total**
N	171690	9145	20181	201016 (21.25%)
PU	118548	6307	14021	138876 (14.68%)
NPROP	78345	4196	9237	91778 (9.70%)
PREP	77771	4332	9296	91399 (9.66%)
V	71874	3812	8415	84101 (8.89%)
ART	58865	3040	6804	68709 (7.26%)
PREP+ART	49927	2652	5680	58259 (6.16%)
ADJ	37055	1954	4264	43273 (4.57%)
ADV	21192	1114	2509	24815 (2.62%)
KC	19967	1067	2333	23367 (2.47%)
PCP	16702	921	1927	19550 (2.07%)
NUM	13708	798	1692	16198 (1.71%)
VAUX	13265	704	1552	15521 (1.64%)
PROADJ	13098	669	1647	15414 (1.63%)
KS	10296	520	1275	12091 (1.28%)
PROPESS	9804	504	1228	11536 (1.22%)
PRO-KS-REL	7891	407	863	9161 (0.97%)
PROSUB	5401	309	672	6382 (0.67%)
PDEN	4858	262	546	5666 (0.60%)
CUR	2106	129	239	2474 (0.26%)
PRO-KS	1512	82	165	1759 (0.19%)
PREP+PROADJ	1474	75	166	1715 (0.18%)
ADV-KS-REL	613	35	69	717 (0.08%)
PREP+PROSUB	608	30	72	710 (0.08%)
PREP+PROPESS	452	27	54	533 (0.06%)
ADV-KS	277	12	31	320 (0.03%)
IN	242	25	17	284 (0.03%)
PREP+PRO-KS-REL	163	5	19	187 (0.02%)
PREP+ADV	66	6	13	85 (0.01%)
PREP+PRO-KS	26	2	4	32 (0.00%)

(ADJ-S-F-P).

Na Figura 6.6, apresentamos o número de exemplos por etiqueta no corpus Tycho Brahe. As etiquetas substantivo singular (N), Virgula (,) e preposição (P) são as etiquetas com mais exemplos em todo o corpus. A descrição e distribuição das 265 etiquetas do corpus Tycho Brahe são detalhadamente apresentadas no Apêndice A.

Na Figura 6.7 apresentamos a nuvem de palavras do Tycho Brahe Corpus, podemos observar a alta frequência de palavras como “que”, “em”, “para” , “se”. Também podemos notar a presença de expressões como “vossa mercê”, “vossa senhoria” e “de vossa” que indica o carácter histórico do português usado no corpus.

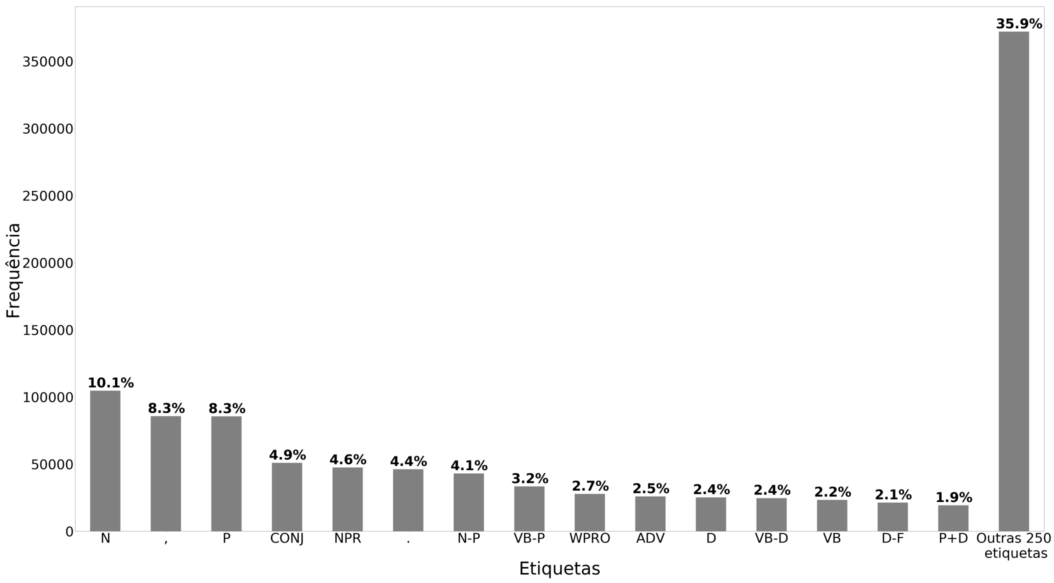

Figura 6.6: Distribuição das etiquetas no Tycho Brahe

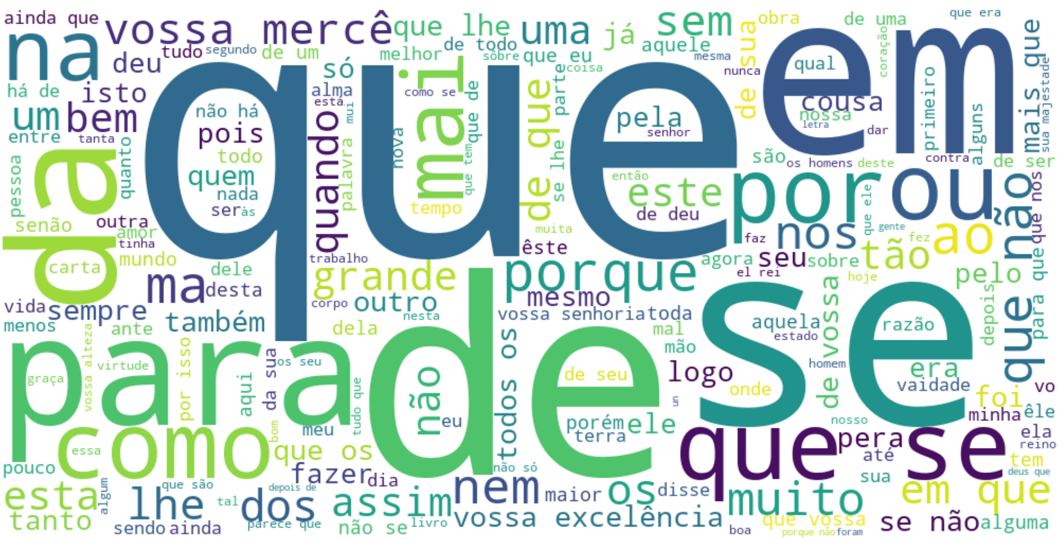

Figura 6.7: Nuvem de palavras Tycho Brahe

6.2 Métricas de avaliação

Para avaliar o desempenho do modelo nós usamos acurácia, formalizada na equação 6-1. Outra métrica de avaliação importante é a acurácia de palavras fora do vocabulário (FV) que calcula uma acurácia local focada apenas em *tokens* não conhecidos pelo conjunto de treino. A acurácia FV captura o desempenho do modelo em palavras desconhecidas, essa métrica é formalizada na equação 6-2. Nós também calculamos *precision*, *recall* e F_1-score (conhecido também como F_1) para cada classe i presente no corpus. As equações 6-3, 6-4 e 6-6 definem *precision*, *recall* e F_1.

$$Acurácia = \frac{\#\ de\ tokens\ corretamente\ classificados}{\#\ total\ de\ tokens} \tag{6-1}$$

$$Acurácia\ FV = \frac{\#\ de\ tokens\ FV\ corretamente\ classificados}{\#\ total\ de\ tokens\ FV} \tag{6-2}$$

$$Precision_i = \frac{TP_i}{TP_i + FP_i} \tag{6-3}$$

$$Recall_i = \frac{TP_i}{TP_i + FN_i} \tag{6-4}$$

$$F_{1i} = \frac{2 * Precision_i * Recall_i}{Precision_i + Recall_i} \tag{6-5}$$

$$Em\ que \begin{cases} TP_i = \#\ de\ anotações\ positivas \\ FP_i = \#\ de\ anotações\ falso-positivas \\ FN_i = \#\ de\ anotações\ falso-negativas \end{cases} \tag{6-6}$$

6.3 Baselines

Para medir o impacto do nosso sistema descrito no Capítulo 5 que usa *word embeddings* e *character embeddings*, nós criamos dois sistemas de *baseline*. Eles são usados como ponto de referência para comparar nossos resultados. Os resultados dos sistemas *baseline* podem informar se o modelo principal realmente agrega valor na classificação morfossintática.

O primeiro sistema *baseline* (**Perceptron-features-manuais**) usa um Perceptron multicamadas com *features* não automáticas.

Na Figura 6.8 temos a estrutura do Perceptron multicamadas utilizado como *baseline*. Usamos uma camada densa totalmente conectada na camada de entrada, camadas com a função de ativação RELU e camadas de *dropout*. Na camada de saída, usamos uma *softmax* como função de ativação. Nós usamos como *features*: a presença de letras maiúsculas, sufixos e prefixo das

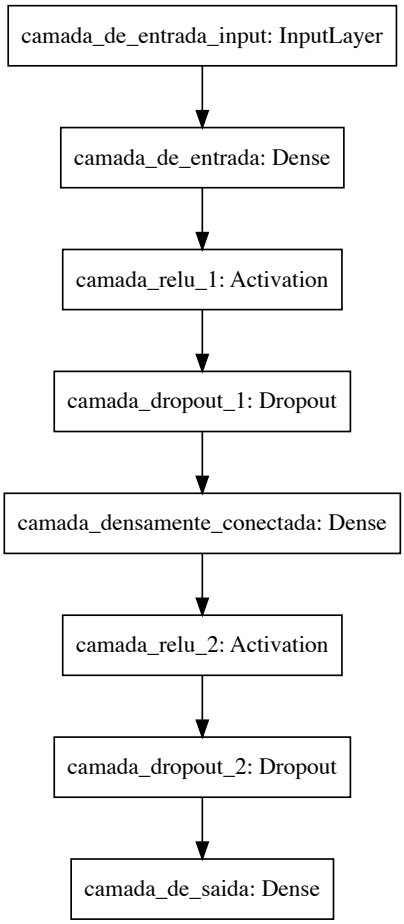

Figura 6.8: Arquitetura do *baseline* **Perceptron-features-manuais**

palavras e ocorrência das palavras (três) anteriores e posteriores. Nesse modelo não usamos qualquer tipo de informação de *word embeddings* ou *character embeddings*.

O segundo sistema *baseline* (**BLSTM-WE**) é semelhante ao nosso modelo principal, porém sem a camada de BLSTM para *character embeddings*, somente vetores de *word embeddings*. Na Figura 6.9 temos a arquitetura do segundo *baseline*. Também usamos uma camada de BLSTM e uma função de ativação *softmax*.

6.4 Ajuste de Hiperparâmetro

Redes Neurais contam com vários hiperparâmetros, esses valores controlam diretamente o processo de treinamento. O tamanho da camada escondida, o valor de *dropout*, e entre outras variáveis das redes neurais impactaram no desempenho dos modelos. Nessa seção procuramos por hiperparâmetros

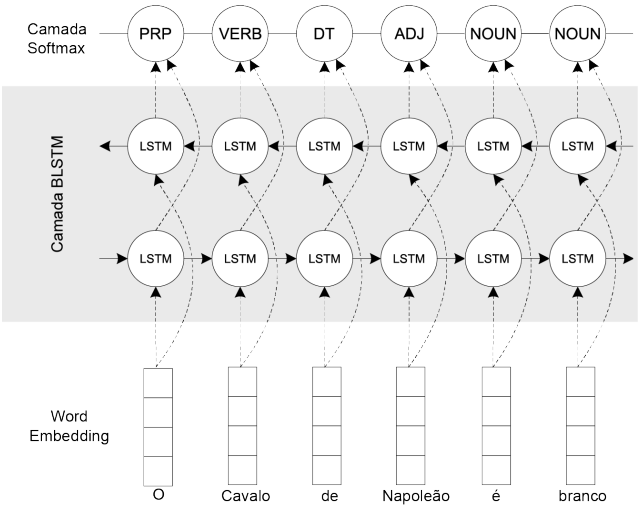

Figura 6.9: Arquitetura de *baseline* **BLSTM-WE**.

que maximizem a acurácia dos dois modelos *baselines* (**Perceptron-features-manuais** e **BLSTM-WE**) e do nosso modelo principal (**BLSTM-WE-CE**), apresentado no Capítulo 5. Usamos o conjunto de validação para procurar por hiperparâmetros e escolhemos o corpus Mac-Morpho v1 para esse processo.

Para o *baseline* **Perceptron-features-manuais**, queremos testar: Qual o tamanho de camada escondida produz melhores resultados? Qual valor de *dropout* produz melhores resultados? Os resultados dos experimentos encontram-se na Tabela 6.6. Primeiro procuramos pelo tamanho da camada escondida, o tamanho 400 obteve o melhor valor em acurácia no conjunto de validação. Depois procuramos pela a probabilidade de *dropout*, 0.4 produziu o melhor resultado. Um alto valor de *dropout* faz com que o modelo "esqueça" muitos neurônios e não aprenda algumas *features* importantes.

Tabela 6.6: Resultados dos experimentos do modelo Perceptron-features-manuais avaliado no conjunto de desenvolvimento do Mac-Morpho v1.

Camada Escondida	**Dropout**	**Acurácia (dev.)**
50	0.1	96.75%
100	0.1	96.77%
200	0.1	96.79%
400	0.1	**96.84**%
600	0.1	96.78%
400	0.2	96.94%
400	0.3	97.00%
400	0.4	**97.02**%
400	0.5	96.96%
400	0.6	96.96%
400	0.7	96.40%
400	0.8	95.79%

Para o *baseline* **BLSTM-WE**, queremos testar: Qual o tamanho de camada escondida produz melhores resultados? Qual valor de *dropout* produz melhores resultados? Qual o melhor tamanho para o vetor de *word embedding*?

Tabela 6.7: Resultados dos experimentos do modelo BLSTM-WE avaliado no conjunto de desenvolvimento do Mac-Morpho v1.

Camada Escondida	**Dropout**	**Dimensão**	**W. embedding**	**Acurácia (dev.)**
10	0.1	50	Word2Vec	95.65%
50	0.1	50	Word2Vec	96.75%
100	0.1	50	Word2Vec	96.92%
200	0.1	50	Word2Vec	96.91%
300	0.1	50	Word2Vec	**97.01**%
400	0.1	50	Word2Vec	96.87%
500	0.1	50	Word2Vec	**97.01**%
1000	0.1	50	Word2Vec	96.91%
1500	0.1	50	Word2Vec	96.85%
2000	0.1	50	Word2Vec	**97.01**%
500	0.2	50	Word2Vec	96.93%
500	0.3	50	Word2Vec	96.87%
500	0.4	50	Word2Vec	97.04%
500	0.5	50	Word2Vec	**97.07**%
500	0.6	50	Word2Vec	97.05%
500	0.7	50	Word2Vec	**97.07**%
500	0.8	50	Word2Vec	97.02%
500	0.5	100	Word2Vec	**97.15**%
500	0.5	300	Word2Vec	**97.15**%

Na Tabela 6.7, temos os resultados dos experimentos realizados usando *baseline* **BLSTM-WE**. Testamos o impacto do tamanho da camada escondida de BLSTM, esse hiperparâmetro tem um impacto limitado dado que a partir do tamanho 100, os resultados em acurácia não variam muito. Escolhemos o valor 500 com hiperparâmetro, esse valor apresenta o melhor resultado em acurácia no conjunto de validação e é um meio termo entre os outros valores que apresentaram também os melhores resultados.

Quando a probabilidade de *dropout*, 0.5 e 0.7 apresentaram os melhores resultados. Escolhemos 0.5 para o nosso sistema. O hiperparâmetro com maior impacto no resultado, foi o tamanho dos vetores de *word embedding*. O tamanho 100 e 300, produziram os mesmos resultados, e melhoram em relação ao tamanho 50 usado nas outras configurações. Escolhemos o tamanho 100 para o nosso sistema, ele produz o mesmo resultado que o tamanho 300 e traz menor complexidade de tamanho para o modelo.

Na Tabela 6.8 temos os resultados dos experimentos realizados para encontrar os hiperparâmetros para nossa arquitetura principal **BLSTM-WE-CE**. Procuramos pelo tamanho da camada escondida do BLSTM principal, o valor de *dropout*, tamanho da camada escondida do BLSTM que gera o character embedding, e o tamanho dos vetores de representação de *word*

Tabela 6.8: Resultados dos experimentos do modelo BLSTM-WE-CE avaliado no conjunto de desenvolvimento do Mac-Morpho v1.

Cam. E.	**Dropout**	**Cam. E. Char.**	**Dimesão**	**W. embedding**	**Acurácia (val.)**
10	0.1	10	50	Word2Vec	95.93%
50	0.1	10	50	Word2Vec	97.38%
100	0.1	10	50	Word2Vec	97.56%
200	0.1	10	50	Word2Vec	**97.73%**
300	0.1	10	50	Word2Vec	97.67%
400	0.1	10	50	Word2Vec	97.67%
500	0.1	10	50	Word2Vec	**97.73%**
1000	0.1	10	50	Word2Vec	97.72%
1500	0.1	10	50	Word2Vec	97.73%
2000	0.1	10	50	Word2Vec	97.66%
500	0.2	10	50	Word2Vec	97.72%
500	0.3	10	50	Word2Vec	97.67%
500	0.4	10	50	Word2Vec	97.61%
500	0.5	10	50	Word2Vec	97.73%
500	0.6	10	50	Word2Vec	**97.79%**
500	0.7	10	50	Word2Vec	97.72%
500	0.8	10	50	Word2Vec	97.54%
500	0.6	20	50	Word2Vec	97.67%
500	0.6	30	50	Word2Vec	97.75%
500	0.6	40	50	Word2Vec	97.69%
500	0.6	50	50	Word2Vec	**97.79%**
500	0.6	100	50	Word2Vec	97.74%
500	0.6	50	100	Word2Vec	**97.80**%
500	0.6	50	300	Word2Vec	97.74%

embedding. Primeiro testamos o tamanho da camada escondida do BLSTM, 200 e 500 apresentaram os melhores resultados em acurácia, e escolhemos o tamanho 500. Depois procuramos pelo o valor de *dropout*, a probabilidade 0.6 gerou o melhor resultado entre as outras probabilidades testadas.

Procuramos também pelo tamanho da camada escondida do BLSTM que gera o character embedding, testamos os tamanhos 10, 20, 30, 40, 50 e 100. O tamanho 50 gerou o melhor resultado em acurácia no conjunto de validação. E por último, analisamos qual o melhor tamanho para o vetor de *word embedding*, testamos os tamanhos 50, 100 e 300, o tamanho 100 produziu o melhor resultado em acurácia.

Na Tabela 6.9, temos o resumo dos hiperparâmetros escolhidos para as arquiteturas dos *baselines* e a arquitetura principal. Todos os experimentos realizados com essas arquiteturas usam essas configurações no processo de treino.

6.5 Análise de word embedding

Nossa arquitetura (**BLSTM-WE-CE**) usa representações de *word embedding*. Esses vetores são pré-treinados em grandes corpora textuais não su-

Tabela 6.9: Hiperparâmetros selecionados para as arquiteturas *baseline* e arquitetura principal.

Perceptron-features-manuais

Hiperparâmetro	**Valor**
Camada Escondida	400
Dropout	0.4

BLSTM-WE

Hiperparâmetro	**Valor**
Camada Escondida (BLSTM)	500
Dropout (BLSTM)	0.5
Dimensão *word embedding*	100

BLSTM-WE-CE

Hiperparâmetro	**Valor**
Camada Escondida (BLSTM)	500
Dropout (BLSTM)	0.6
Camada Escondida (*char. embedding*)	50
Dimensão *word embedding*	100

pervisionadas. Nesse trabalho, analisamos quais algoritmos de treino de *word embedding* geram os melhores resultado em acurácia na classificação morfossintática para a Língua Portuguesa. Analisamos os algoritmos Word2vec, Fasttext, Wang2vec and Glove.

Para nossos experimentos usamos as representações *word embedding* NILC-Embeddings, criadas e disponibilizadas pelo Núcleo Interinstitucional de Linguística Computacional (NILC) no trabalho (28). Esses vetores foram gerados a partir da junção de várias corporas em Português, de diferentes fontes e domínios, a fim de obter representações de gêneros variados. Foram usados 17 corpora diferentes que totalizam 1.395.926.282 *tokens*. O treinamento dos vetores ocorreu com os algoritmos Word2vec, FastText, Wang2vec e Glove e estão disponíveis no repositório online[5].

Nós testamos o desempenho do nosso modelo nos quatro métodos diferentes (Word2vec, FastText, Wang2vec e Glove) e suas versões de CBOW e *Skip-Gram*. Usamos as métricas de acurácia. Na Tabela 6.10 apresentamos os resultados no conjunto de validação para os três corpora.

Todos os experimentos realizados apresentaram alta acurácia (mais que 97%). As representações geradas pelo Wang2Vec obtiveram os melhores resultados em acurácia nos três corpora analisadas. O bom desempenho de Wang2vec pode ser explicado por seu foco na captura de informações sintáticas, que leva em conta a ordem em que as palavras aparecem (34), isso se

[5]http://nilc.icmc.usp.br/embeddings

mostra bastante útil para *POS tagging*.

Para o corpus Mac-Morpho v1, os melhores resultados foram obtidos na configuração que usa o Wang2Vec CBOW, com 97.97% de acurácia no cojunto de validação. Para o corpus Mac-Morpho v2, os melhores resultados em acurácia também foram obtidos com o Wang2Vec CBOW (97.67% de acurácia), com uma pequena margem de 0.02% de diferença para os resultados obtidos com o Wang2Vec *Skip-Gram*. No corpus Tycho Brahe, a configuração que usa o Wang2Vec *Skip-Gram* obteve os melhores resultados em acurácia de 97.30%.

Concluímos que a mudança do tipo algoritmo de *word embedding* resulta em um impacto nos resultados em acurácia para a atividade de *POS tagging*. O modelo Wang2Vec apresentou os melhores resultados em acurácia para os três corpora, porém esses resultados não tiveram um grande distanciamento dos resultados em acurácia obtidos por outros modelos de *word embedding*.

Tabela 6.10: Desempenho do nosso modelo com diferentes *word embeddings* no conjunto validação.

Corpus	**Modelos de Embedding**		**Ac. (%)**
Mac-Morpho v1	Word2Vec	CBOW	97.84
		Skip-Gram	97.72
	FastText	CBOW	97.78
		Skip-Gram	97.80
	Wang2Vec	CBOW	**97.97**
		Skip-Gram	97.84
	Glove		97.70
Mac-Morpho v2	Word2Vec	CBOW	97.56
		Skip-Gram	97.59
	FastText	CBOW	97.44
		Skip-Gram	97.55
	Wang2Vec	CBOW	**97.67**
		Skip-Gram	97.65
	Glove		97.53
Tycho Brahe	Word2Vec	CBOW	96.93
		Skip-Gram	97.01
	FastText	CBOW	96.94
		Skip-Gram	97.07
	Wang2Vec	CBOW	97.17
		Skip-Gram	**97.30**
	Glove		97.11

6.5.1 Comparação com baselines

Para medir o impacto do uso de *word embeddings* e *character embeddings*, nós comparamos a performance do nosso modelo principal com os dois sistemas *baseline* descritos na seção 6.3. Os hiperparâmetros usados nos experimentos são os mesmos apresentados na seção 6.4. O modelo de *word embeddings* usado é o Wang2Vec, tanto para o *baseline* **BLSTM-WE** quanto para o modelo principal **BLSTM-WE-CE**.

Na Tabela 6.11 temos a acurácia e acurácia FV dos três sistemas nos três corpora avaliados. O *baseline* **BLSTM-WE** alcança melhores resultados que **Perceptron-features-manuais** em Mac-Morpho v1 e Mac-Morpho v2. Porém, no corpus Tycho Brahe, o modelo **Perceptron-features-manuais** performa melhor. Isso pode se explicado pelo fato que corpus Tycho Brahe é formado por textos do Português Históricos e algumas palavras que estão presentes no corpus, não estão presentes no modelo pré-treinado de *word embeddings*. Isso faz com que o corpus Tycho Brahe não se beneficie com os outros corpora das informações de *word embeddings*.

Nosso sistema principal **BLSTM-WE-CE** supera os dois sistemas de *baseline* nos três corpora avaliados. Esses resultados suportam os resultados já demonstrados por (44) e (25), de que informações de *character embeddings* são muito importantes para a etiquetagem de sequência em atividades como *POS tagging*.

Comparando nosso sistema principal **BLSTM-WE-CE** e o primeiro sistema *baseline* **Perceptron-features-manuais**, o corpus Mac-Morpho v1 teve um aumento de 1.29 pontos de acurácia e de 7 pontos de acurácia FV. O corpus Mac-Morpho v2 teve um aumento de 1.08 pontos de acurácia e de 4.57 pontos de acurácia FV. Já o corpus Tycho Brahe obteve um aumento de 1.14 em pontos de acurácia e 10.84 em pontos de acurácia FV.

O sistema **BLSTM-WE-CE** tem um maior impacto no corpus Tycho Brahe em acurácia FV. Isso se deve ao fato de que o corpus Tycho Brahe possui muitas classes que descrevem informações morfológicas (gênero, número e tempo verbal) das palavras, justamente o tipo de informação abundante nos de vetores de *character embeddings*.

Os resultados empíricos do nosso sistema proposto (**BLSTM-WE-CE**), mostram que vetores de *character embeddings* gerados por uma camada BLSTM contribuem bastante para a classificação morfossintática da Língua Portuguesa. Observamos um maior ganho em acurácia FV quando usamos *character embeddings* nos três corpora. Esses resultados indicam que o uso de tais representações melhora o desempenho do modelo em palavras que não

fazem parte do vocabulário de treino ou do modelo de *word embeddings*.

Tabela 6.11: Desempenho dos nossos modelos *baseline* no conjunto de teste

Corpus	**Sistema**	**Ac. (%)**	**Ac. FV (%)**
Mac-Morpho v1	Perceptron-features-manuais	96.59	88.59
	BLSTM-WE	96.74	89.76
	BLSTM-WE-CE	**97.88**	**95.59**
Mac-Morpho v2	Perceptron-features-manuais	96.56	90.54
	BLSTM-WE	96.60	90.80
	BLSTM-WE-CE	**97.64**	**95.11**
Tycho Brahe	Perceptron-features-manuais	96.27	76.86
	BLSTM-WE	95.90	75.25
	BLSTM-WE-CE	**97.41**	**87.70**

Na Tabela 6.12 mostrados o tempo computacional gasto para treinar o modelo **BLSTM-WE-CE** nos três diferentes corpora, os experimentos foram realizados usando um GPU GeForce GTX TITAN X. Para treinar os corpora Mac-Morpho v1 e Mac-Morpho v2 gastamos aproximadamente 45 minutos e treinar o corpus Tycho Brahe gastamos aproximadamente 31 minutos.

Tabela 6.12: Tempo de treino aproximado nos três diferentes corpora

Corpus	**Tempo de treino (≈)**
Mac-Morpho v1	45 minutos
Mac-Morpho v2	45 minutos
Tycho Brahe	31 minutos

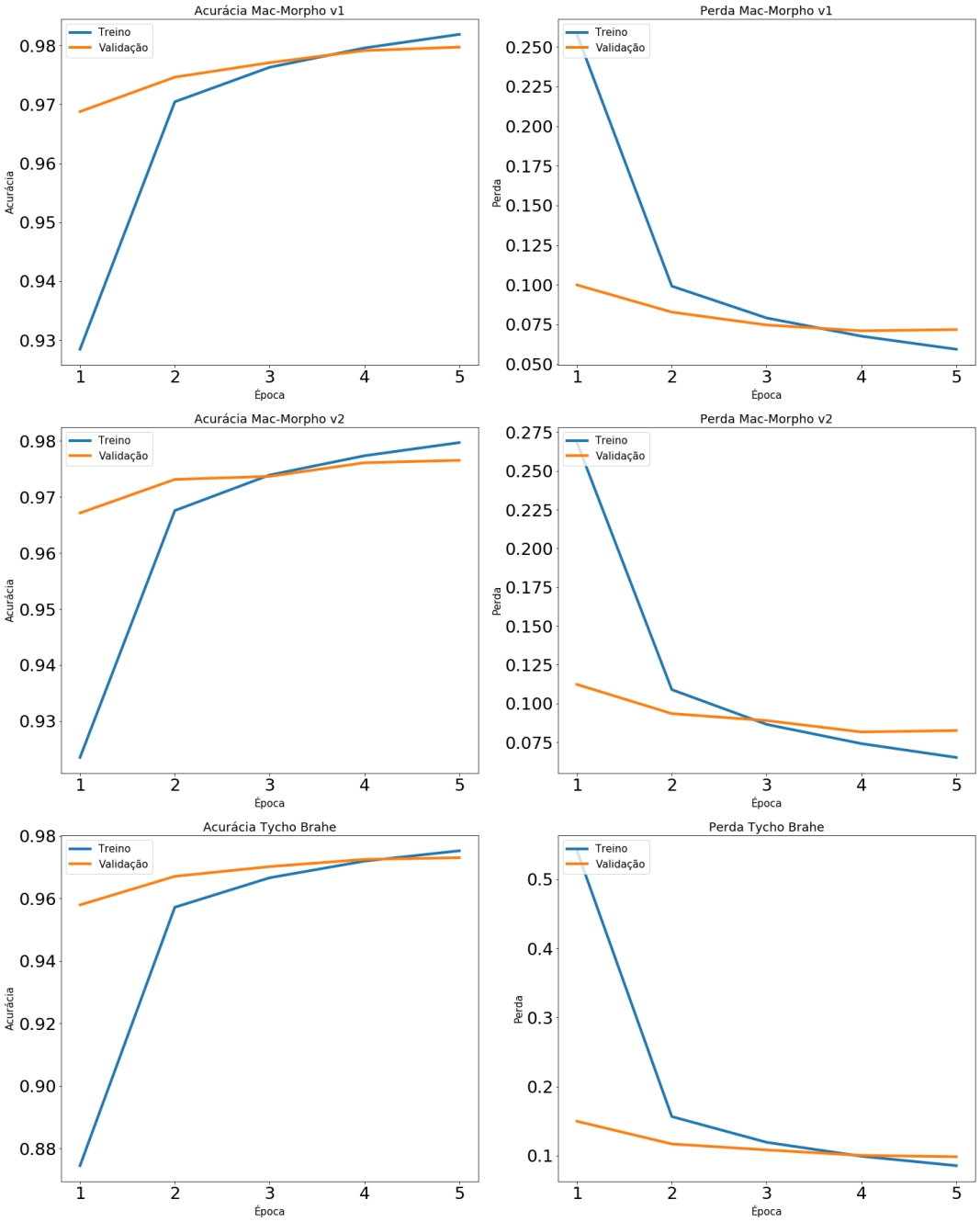

Figura 6.10: Treino do modelo **BLSTM-WE-CE** nas corpora Mac-Morpho v1, Mac-Morpho v2 e Tycho Brahe.

Na Figura 6.10, mostramos acurácia e perda do conjunto de treino e validação durante as épocas do treino do nosso modelo **BLSTM-WE-CE**. A perda foi calcula usando a função *sparse categorical crossentropy*. Para os três corpora, o valor de acurácia do conjunto de treino, ultrapassou o valor de acurácia do conjunto validação depois da terceira época. Isso pode indicar que depois da época três o modelo começa a se especializar no conjunto de treino, podendo perder generalidade.

Podemos observar o mesmo comportamento olhando a perda dos corpora

durante o treino. Por volta da terceira época o valor de perda do conjunto de validação ultrapassou o conjunto de treino. Podendo indicar também o começo de uma especialização no conjunto de treino. Por isso, treinamos com apenas cinco épocas.

6.5.2 Comparação com outros trabalhos

Seguindo a metodologia adotada por Peters et al. (69) e Chiu e Nichols (70), nós treinamos nosso modelo principal **BLSTM-WE-CE** dez vezes com diferentes sementes aleatória e reportamos média e desvio padrão de acurácia e acurácia FV. Isso é importante para estimar a variância dos resultados do nosso modelo, já que durante o treino existe algum fator de aleatoriedade associado, com por exemplo, as camadas de *dropout*.

Tabela 6.13: Resultados no conjunto de teste para dez treinos diferentes com sementes aleatórias.

	Mac-Morpho v1		**Mac-Morpho v2**		**Tycho Brahe**	
	Ac.	**Ac. FV**	**Ac.**	**Ac. FV**	**Ac.**	**Ac. FV**
	97.84	95.35	97.64	95.11	97.37	87.53
	97.86	95.47	97.67	95.28	97.34	88.13
	97.82	95.74	97.63	95.28	97.35	87.28
	97.82	95.54	97.66	94.93	97.41	87.70
	97.85	95.50	97.65	95.51	97.38	87.93
	97.82	95.48	97.67	94.97	97.36	86.95
	97.81	95.51	97.63	95.46	97.30	86.83
	97.83	95.41	97.68	95.11	97.33	87.53
	97.84	95.27	97.68	95.42	97.33	87.14
	97.88	95.59	97.65	95.28	97.37	86.99
$\bar{x}$	97.83	95.48	97.65	95.23	97.35	87.40
σ	0.021	0.129	0.018	0.2	0.030	0.436
min	97.81	95.27	97.63	94.93	97.30	86.83
max	97.88	95.74	97.68	95.51	97.41	88.13

Na Tabela 6.13, reportamos os resultados obtidos no conjunto de teste de dez treinos diferentes como sementes aleatórias. Para as métricas de avaliação acurácia e acurácia FV, calculamos média aritmética ($\bar{x}$), desvio padrão (σ), o valor máximo (max) e o valor mínimo (min).

Na Figura 6.11, apresentamos a visualização das distribuições descritas na Tabela 6.13 com o gráfico *boxplot*. Podemos observar no *boxplot* do corpus Mac-Morpho v1, que os valores de acurácia reportados nos experimentos apresentam menor variabilidade que os valor de acurácia FV. Para o corpus Mac-Morpho v1, temos um valor médio de acurácia de 97.83% com desvio

padrão de apenas 0.021% e um valor médio de acurácia FV de 95.48% com um desvio padrão de 0.129%.

O corpus Mac-Morpho v2 apresenta uma avaliação com características semelhantes ao Mac-Morpho v1, como é possível notar na Figura 6.11. Mac-Morpho v2 também apresenta um espalhamento menor dos resultados de acurácia se comparados aos resultados de acurácia FV. Essa diferença talvez possa ser explicada pelo fato do conjunto de palavras fora do vocabulário ser bem menor que o conjunto todas as palavras. E conjuntos pequenos podem ser mais sensíveis a variações randômicas que possam acontecer no modelo no momento do treino. O Mac-Morpho v2 obteve na avaliação, acurácia média de 97.65% com um valor de desvio padrão de 0.018% e acurácia FV média de 95.23% com um desvio padrão de 0.2%.

Para o corpus Tycho Brahe reportamos 97.35% de acurácia média com um desvio padrão de 0.030% e 87.40% de acurácia FV com desvio padrão de 0.436%. Nesse corpus também observamos um menor espalhamento do valor de acurácia se comparado ao espalhamento do valor de acurácia FV.

Nós comparamos os resultados do modelo **BLSTM-WE-CE** com dois sistemas de melhores performance reportadas para a Língua Portuguesa. O primeiro trabalho de Dos Santos e Zadrozny (44) que usaram, representações *word embeddings* e *character embeddings*, combinados ao algoritmo de Viterbi para fazer a predição das etiquetas. E o trabalho de Fonseca et al. (45), que usa uma rede perceptron multicamadas combinada a *word embeddings* e algumas *features* manuais. Usamos a mesma partição treino/teste/validação que foram usadas nesses trabalhos.

Tabela 6.14: Comparação com outros *POS taggers* em Português

Corpus	**Sistema**	**Ac. (%)±σ**	**Ac. FV (%)**
Mac-Morpho v1	**Este Trabalho**	**97.83±0.021**	**95.48±0.129**
	Fonseca et al. 2015 (45)	97.57	93.38
	Santos e Zadrozny 2014 (44)	97.47	92.49
Mac-Morpho v2	**Este Trabalho**	**97.65±0.018**	**95.23±0.2**
	Fonseca et al. 2015 (45)	97.48	94.34
	Santos e Zadrozny 2014 (44)	97.31	93.43
Tycho Brahe	**Este Trabalho**	**97.35±0.030**	**87.40± 0.436**
	Fonseca et al. 2015 (45)	96.91	84.14
	Santos e Zadrozny 2014 (44)	97.17	86.58

Na Tabela 6.14, nós comparamos acurácia média e acurácia FV média dos resultados obtidos com os valores reportados nos trabalhos anteriores para a Língua Portuguesa. No corpus Mac-Morpho v1 aumentamos 0.26 pontos de acurácia em média, e 2.02 de acurácia FV em média, se compararmos com o resultado reportado por Fonseca et al. (45).

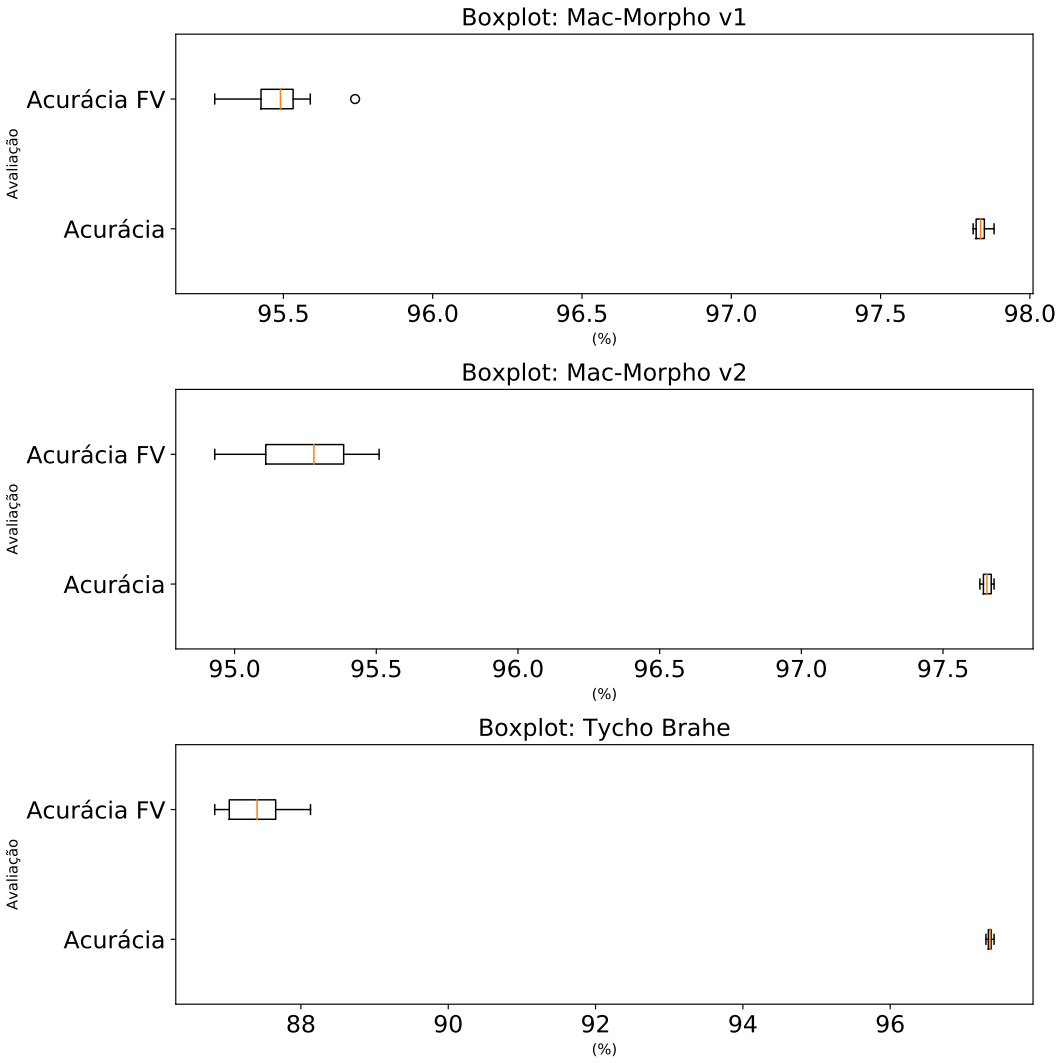

Figura 6.11: *Boxplot* dos resultados de 10 diferentes treinos com o modelo BLSTM-WE-CE nos corpora Mac-Morpho v1, Mac-Morpho v2 e Tycho Brahe.

No Mac-Morpho v2 aumentamos 0.17 pontos de acurácia em média, e 1.8 pontos de acurácia FV em média, se compararmos com o resultado reportado por Fonseca et al (45). Já no corpus Tycho Brahe aumentamos 0.18 pontos de acurácia em média, e 0.82 pontos de acurácia FV em média, se compararmos com o resultado reportado por Santos e Zadrozny (44). Os resultados definem um novo estado da arte nos três corpora usados, demonstrando assim a efetividade do uso de BLSTM combinado a *word embeddings* e *character embeddings* para *POS tagging* em Português.

6.6 Análise por etiquetas

Nós usamos F_1-score, *precision* e *recall* para avaliar os 10 experimentos realizados com o modelo *BLSTM-WE-CE*. Também realizamos uma análise do desempenho médio do modelo por etiquetas. Na Tabela 6.15 mostramos a

média de avaliação das etiquetas para os três corpora.

Tabela 6.15: Valor médio de F_1, *Precision* e *Recall* dos experimentos realizados com o BLSTM-WE-CE no conjunto de teste

Corpus	$\boldsymbol{F_1} \pm \sigma$	**Precision**$\pm\sigma$	**Recall**$\pm\sigma$
Mac-Morpho v1	0.948±0.094	0.953±0.075	0.945±0.108
Mac-Morpho v2	0.902±0.011	0.908±0.022	0.899±0.014
Tycho Brahe	0.740±0.045	0.753±0.054	0.743±0.049

O corpus Mac-Morpho v1 obteve a melhor avaliação média (F_1=0.948, *precision*=0.953, *recall*=0.945). Esse corpus possui muitas etiquetas de pontuação, tais etiquetas são fáceis de classificar e quase sempre o modelo acerta 100% dos exemplos dessas classes, isso faz com que a média final do corpus seja influenciada por esses valores. Na Tabela 6.16 apresentamos os valores de F_1, *precision* e *recall* médio para cada etiqueta do corpus Mac-Morpho v1. Nessa tabela podemos ver que as etiquetas de pontuação quase sempre obtêm 100% de F_1.

Na Figura 6.12, para melhor comparação entre valores, mostramos o F_1 médio das classes presentes no conjunto de treino, e que não são classes de pontuação do corpus Mac-Morpho v1. A maioria das etiquetas obtiveram um F_1 de mais de 95%. As piores avaliações foram das etiquetas ADV-KS (Advérbio conectivo de subordinação) com 52% de F_1 e IN (Interjeição) com 66.1% de F_1. Essas duas etiquetas, estão entre as classes de mais baixa representação no corpus Mac-Morpho v1. A etiqueta ADV-KS representa apenas 0.034% do total de *tokens*, já a etiqueta IN representa apenas 0.032% do total. Devido às baixas distribuições, o modelo não conseguiu aprender a classificar essas etiquetas tão bem quanto as outras.

O corpus Mac-Morpho v2 obteve a segunda melhor avaliação média (F_1=0.902, *precision*=0.908 e *recall*=0.899) no conjunto de treino. Na Tabela 6.17 apresentamos os valores de F_1, *precision* e *recall* médio para cada uma das 30 etiquetas do corpus Mac-Morpho v2. E na Figura 6.13, para melhor comparação entre os resultados, apresentamos o F_1 médio das classes presentes no Mac-Morpho v2.

Podemos ver na Figura 6.13 que a etiqueta com a pior avaliação no Mac-Morpho v2 é PREP+PRO-KS (Preposição + subordinação do pronome relativo conectivo). Essa etiqueta obteve apenas 9.9% de F_1. O modelo não aprendeu a reconhecer bem a classe PREP+PRO-KS, provavelmente devido a baixa representação no corpus. Essa etiqueta tem a pior representatividade, com apenas 32 exemplos em todo o corpus.

Ainda na Figura 6.13 vemos que a etiqueta ADV-KS obteve 64.4% de F_1, essa classe representa apenas 0.03% dos *tokens* presentes no Mac-

Tabela 6.16: Valor médio de F_1, *Precision* e *Recall* por etiqueta no conjunto de treino do corpus Mac-Morpho v1

Classe	$\boldsymbol{F_1} \pm \sigma$	$\boldsymbol{Precision} \pm \sigma$	$\boldsymbol{Recall} \pm \sigma$
!	1.000±0.000	1.000±0.000	1.000±0.000
"	1.000±0.000	1.000±0.000	1.000±0.000
’	1.000±0.000	1.000±0.000	1.000±0.000
(	1.000±0.000	1.000±0.000	1.000±0.000
)	1.000±0.000	1.000±0.000	1.000±0.000
,	1.000±0.000	1.000±0.000	1.000±0.000
-	0.981±0.003	0.971±0.006	0.990±0.000
.	1.000±0.000	1.000±0.000	1.000±0.000
...	1.000±0.000	1.000±0.000	1.000±0.000
/	1.000±0.000	1.000±0.000	1.000±0.000
:	1.000±0.000	1.000±0.000	1.000±0.000
;	1.000±0.000	1.000±0.000	1.000±0.000
?	1.000±0.000	1.000±0.000	1.000±0.000
[	1.000±0.000	1.000±0.000	1.000±0.000
ADJ	0.950±0.000	0.949±0.006	0.957±0.005
ADV	0.940±0.000	0.941±0.006	0.934±0.005
ADV-KS	0.585±0.024	0.764±0.063	0.481±0.048
ADV-KS-REL	0.877±0.016	0.882±0.042	0.877±0.048
ART	0.990±0.000	0.990±0.000	0.993±0.005
CUR	1.000±0.000	0.991±0.003	1.000±0.000
IN	0.661±0.039	0.666±0.071	0.665±0.053
KC	0.980±0.000	0.987±0.005	0.978±0.004
KS	0.915±0.005	0.916±0.013	0.911±0.012
N	0.978±0.004	0.978±0.004	0.974±0.005
NPROP	0.968±0.004	0.970±0.000	0.960±0.000
NUM	0.970±0.000	0.967±0.007	0.967±0.007
PCP	0.970±0.000	0.959±0.006	0.982±0.006
PDEN	0.910±0.005	0.917±0.016	0.901±0.017
PREP	0.990±0.000	0.990±0.000	0.990±0.000
PRO-KS	0.749±0.014	0.787±0.034	0.718±0.049
PRO-KS-REL	0.940±0.000	0.927±0.013	0.953±0.012
PROADJ	0.980±0.000	0.977±0.005	0.980±0.000
PROPESS	0.980±0.000	0.980±0.000	0.985±0.005
PROSUB	0.884±0.005	0.869±0.019	0.901±0.014
V	0.990±0.000	0.986±0.005	0.990±0.000
VAUX	0.960±0.000	0.967±0.007	0.955±0.005
Média±	**0.948±0.094**	**0.953±0.075**	**0.945±0.108**

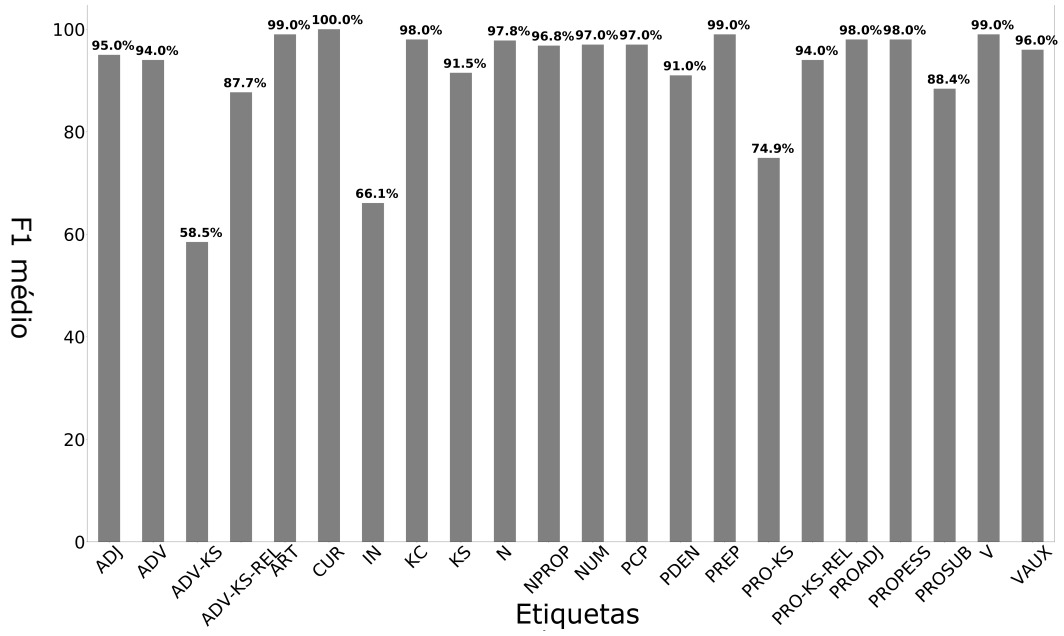

Figura 6.12: Valor médio de F_1 dos 10 experimentos realizados com o modelo BLSTM-WE-CE no corpus Mac-Morpho v1.

Morpho v2. A etiqueta IN obteve 70.0% de F_1, essa etiqueta também tem uma baixa representação no corpus de apenas 0.03%. Essas classes obtiveram as piores avaliações, o modelo não aprendeu a reconhecer bem essas classes provavelmente devido a baixa representação. Para as outras classes do Mac-Morpho v2 temos um valor médio de F_1 entorno de 90%.

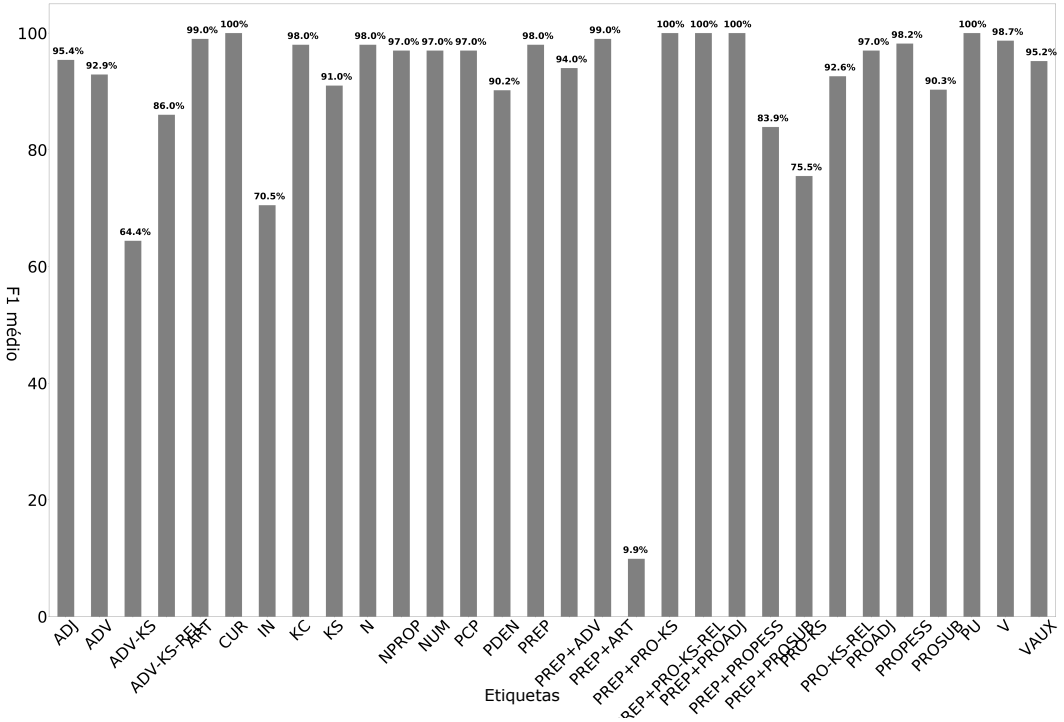

Figura 6.13: Valor médio de F_1 dos 10 experimentos realizados com o modelo BLSTM-WE-CE no corpus Mac-Morpho v2.

Tabela 6.17: Valor médio de F_1, *Precision* e *Recall* por etiqueta no conjunto de treino do corpus Mac-Morpho v2

Classe	$\boldsymbol{F_1} \pm \sigma$	***Precision*** $\pm\sigma$	***Recall*** $\pm\sigma$
ADJ	0.954±0.005	0.952±0.004	0.957±0.005
ADV	0.929±0.003	0.936±0.008	0.919±0.009
ADV-KS	0.644±0.030	0.706±0.056	0.600±0.062
ADV-KS-REL	0.860±0.013	0.853±0.020	0.868±0.013
ART	0.990±0.000	0.986±0.005	0.990±0.000
CUR	1.000±0.000	1.000±0.000	1.000±0.000
IN	0.705±0.046	0.780±0.119	0.650±0.000
KC	0.980±0.000	0.991±0.003	0.972±0.004
KS	0.910±0.007	0.917±0.011	0.906±0.013
N	0.980±0.000	0.980±0.000	0.975±0.005
NPROP	0.970±0.000	0.969±0.006	0.972±0.006
NUM	0.970±0.000	0.970±0.009	0.968±0.008
PCP	0.970±0.000	0.963±0.005	0.976±0.005
PDEN	0.902±0.006	0.907±0.016	0.892±0.010
PREP	0.980±0.000	0.980±0.000	0.982±0.004
PREP+ADV	0.940±0.021	0.936±0.034	0.944±0.039
PREP+ART	0.990±0.000	0.990±0.000	0.992±0.004
PREP+PRO-KS	0.099±0.159	0.150±0.242	0.075±0.121
PREP+PRO-KS-REL	1.000±0.000	1.000±0.000	1.000±0.000
PREP+PROADJ	1.000±0.000	0.990±0.000	1.000±0.000
PREP+PROPESS	1.000±0.000	1.000±0.000	1.000±0.000
PREP+PROSUB	0.839±0.016	0.867±0.022	0.814±0.035
PRO-KS	0.755±0.015	0.726±0.055	0.795±0.066
PRO-KS-REL	0.926±0.005	0.903±0.014	0.952±0.006
PROADJ	0.970±0.000	0.968±0.004	0.973±0.005
PROPESS	0.982±0.004	0.989±0.003	0.980±0.000
PROSUB	0.903±0.005	0.906±0.021	0.900±0.016
PU	1.000±0.000	1.000±0.000	1.000±0.000
V	0.987±0.005	0.989±0.003	0.982±0.004
VAUX	0.952±0.004	0.949±0.006	0.959±0.003
Média±	**0.902±0.011**	**0.908±0.022**	**0.899±0.014**

O corpus Tycho Brahe obteve a pior avaliação média (F_1=0.740, *precision*=0.753 e *recall*=0.743) se comparado aos outros corpora. Esse corpus tem um sistema de etiquetas muito mais granular, são 265 classes diferente. E como podemos ver no Apêndice A, muitas etiquetas apresentam uma baixa representação, algumas delas contam com menos de uma dezena de exemplos em todo o corpus. E as etiquetas com baixa representação tendem a obter resultados piores, fazendo com que a média de F_1 seja possivelmente "puxada"' para baixo por essas classes.

Os valores de F_1, *precision* e *recall* médio de todas as classes presentes no conjunto de teste do corpus Tycho Brahe, estão detalhados no Apêndice B. Em

geral, as classes com uma boa representação tiveram um bom desempenho. Na Figura 6.14 mostramos os valores de F_1 para as 15 etiquetas com as maiores representações no corpus Tycho Brahe, a maioria das classes alcançaram valor de F_1 entorno de 98%. Para melhor entender o desempenho de todas as 233 etiquetas presentes no conjunto de teste, montamos um histograma de frequência (Figura 6.15) das etiquetas em 5 grupos avaliação. A maioria das etiquetas (158) obtiveram um valor de F_1 entre 80% e 100%. 38 etiquetas foram avaliadas com um F_1 entre 0% e 20%. Embora tenhamos algumas dezenas de etiquetas com um pobre desempenho ainda temos a maioria delas com uma boa avaliação.

No corpus Mac-Morpho v1 e no corpus Mac-Morpho v2 notamos que as classes com os piores desempenhos apresentavam uma baixa representatividade de exemplo. Para o corpus Tycho Brahe, que têm 265 classes diferentes, montamos um gráfico para investigar se existe algum tipo de correlação entre a grandeza do número de exemplos de uma classe e seu desempenho em F_1. Na Figura 6.16 apresentamos um gráfico de correlação, no eixo y temos o logaritmo do número de exemplo da uma classe, e no eixo x temos o desempenho da classe em F_1. Calculamos o coeficiente de correlação de Pearson das duas variáveis e descobrimos uma correlação positiva de $p = 0.701$.

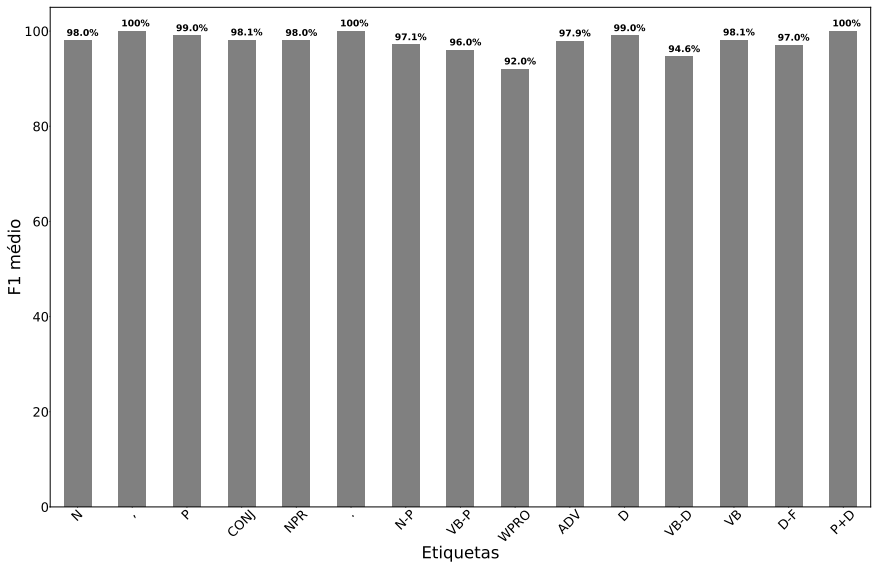

Figura 6.14: Valor médio de F_1 dos 10 experimentos realizados com o modelo BLSTM-WE-CE no corpus TychoBrahe para as 15 etiquetas mais frequentes.

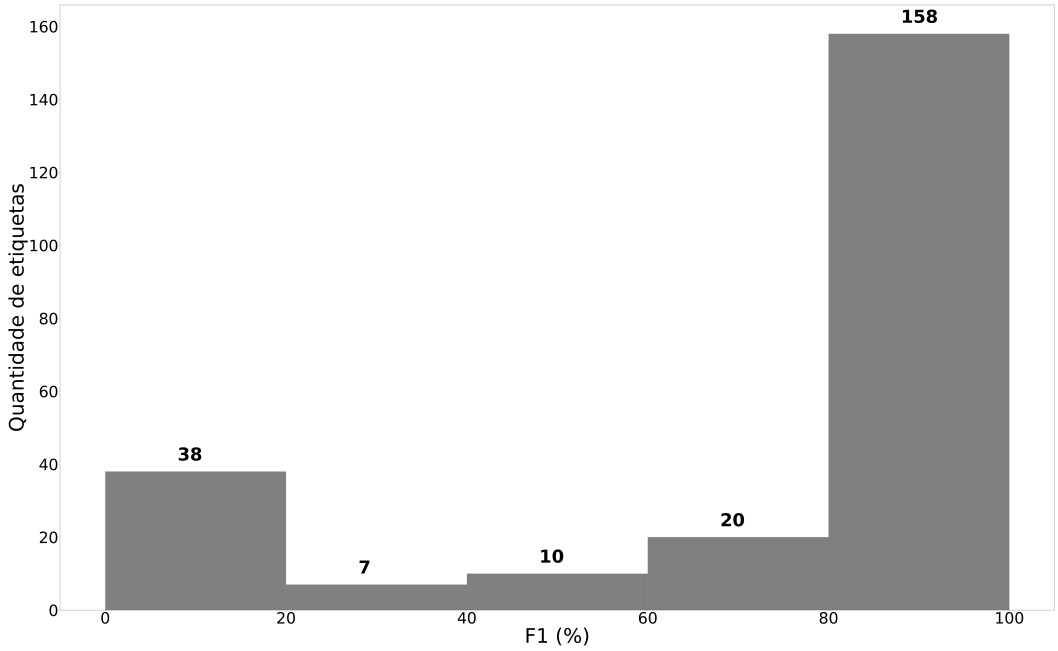

Figura 6.15: Histograma: quantidade de etiquetas em 5 grupo de avaliação de F_1

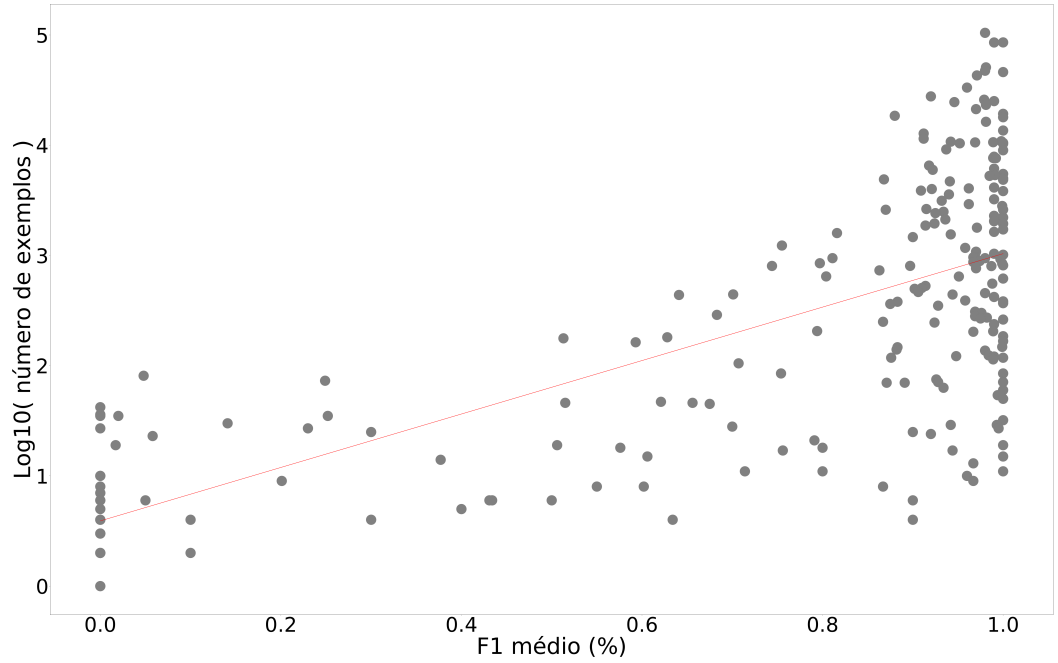

Figura 6.16: Gráfico de correlação do valor F_1 da etiqueta pelo logaritmo do número de exemplos.

6.7 Análise de erro

Para a nossa arquitetura **BLSTM-WE-CE**, analisamos nos três corpora os erros mais frequentes. Nas Tabelas 6.18, 6.19 e 6.20 temos as palavras que mais obtiveram erros de marcação e as etiquetas que elas possuem no conjunto de teste, respectivamente para os corpora Mac-Morpho v1, Mac-Morpho v2 e Tycho Brahe.

As palavras mais difíceis de se atribuir etiquetas morfossintáticas são "que" e "a", tanto para o Português Contemporâneo, quanto para o Português Histórico. Essas duas palavras apareceram no topo das tabelas de erros dos três corpora. A complexidade de marcação, e consequentemente a maior quantidade de erros nessas palavras pode ser atribuída, provavelmente, ao grande número de classes que as palavras "que" e "a" podem ser relacionadas no conjunto de dados.

No corpus Mac-Morpho v1, a palavra com o maior número de diferentes classes associadas, é justamente a palavra "que", com 17 classes. A segunda é a palavra "a", com 14 classes (em Figura 6.2). O mesmo acontece no corpus Mac-Morpho v2, a palavra "que", tem 16 diferentes classes associadas, o maior número de todo corpus, e a palavra "a" tem 13, o segundo maior número de classes associadas. O mesmo vale para o corpus de Português Histórico (Tycho Brahe), a palavra "que", com 16 classes. A segunda é a palavra "a", com 13 classes, sendo essas a duas palavras com mais classes nesse corpus.

Também encontramos no topo das tabelas de erros, palavras como "de", "o", "se", "um", "nem", "como" e "até". Por conta da ambiguidade presente na linguagem humana, essas palavras também assumem diferentes classes morfossintática de acordo com o contexto em que são usadas.

6.8 Análise de character embedding

Nessa subseção realizamos uma análise nos vetores de *character embedding* gerados durante o treino do modelo BLSTM-WE-CE com o corpus Mac-Morpho v1. Tais vetores, geram representações para as palavras do conjunto de treino por meio de uma camada BLSTM. Essas representações são capazes de capturar informações morfológicas de, por exemplo, tamanho, prefixo, sufixo e vocabulário de caracteres.

Nós computamos a similaridade de cosseno de algumas palavras w_{oov}, que estão fora do vocabulário de treino, para as palavras w_v, que fizeram parte do treino do modelo. Na Tabela 6.21, apresentamos as palavras w_{oov} e as 5 palavras w_v mais próximas, segundo suas respectivas similaridades de cosseno.

Tabela 6.18: Os *tokens* com mais erros e as etiquetas que eles possuem no conjunto de teste do Mac-Morpho v1.

Palavra	**Etiquetas**	**# de erros**
que	NPROP, ADV-KS, PRO-KS-REL, ADV, PROADJ, ADV-KS-REL, PROSUB, PREP, PRO-KS, KS, PDEN	281
a	N, KC, PRO-KS-REL, PROADJ, ADV, ART, PROSUB, PREP, NPROP, KS, PROPESS	234
de	N, KC, ADV, PREP, NPROP, KS, PDEN	162
o	NPROP, ADV-KS, PRO-KS-REL, ADV, PROADJ, ART, PROSUB, PRO-KS, KS, PROPESS	123
como	KC, NPROP, ADV-KS, ADV-KS-REL, PREP, ADV, KS	92
um	N, NPROP, NUM, PROADJ, ART, PROSUB, ADV	70
até	NPROP, PREP, ADV, KS, PDEN	64
se	NPROP, KS, PROPESS	41
é	V, VAUX, PROSUB, NPROP, PDEN	40
uma	N, NUM, ADV, ART, PROSUB, PREP, NPROP, KS	39

Tabela 6.19: Os *tokens* com mais erros e as etiquetas que eles possuem no conjunto de teste do Mac-Morpho v2.

Palavra	**Etiquetas**	**# de erros**
que	NPROP, ADV-KS, PRO-KS-REL, PROADJ, ADV-KS-REL, ADV, PROSUB, PRO-KS, KS, PDEN	154
a	KC, PRO-KS-REL, PROADJ, ADV, ART, PROSUB, PREP, NPROP, PROPESS	95
de	N, ADV, PREP, NPROP, KS, PDEN	67
o	PRO-KS-REL, ADV, ART, PRO-KS, PROSUB, NPROP, PROPESS	53
mais	N, KC, NPROP, PROADJ, PROSUB, PREP, ADV	34
como	KC, ADV-KS, ADV, ADV-KS-REL, PREP, NPROP, KS	33
até	ADV, KS, PDEN, PREP	28
um	N, NUM, ADV, PROADJ, ART, PROSUB, NPROP	28
do	N, PREP+ART, NPROP, KS, PREP+PROSUB	21
da	NPROP, PREP+ART, PREP+PROSUB, N	20

Com a Tabela 6.21 podemos notar a capacidade dos vetores de *character embedding* gerado com BLSTM de capturar prefixos e sufixos. Para a palavra fora do vocabulário "imprecisão", por exemplo, temos palavras próximas com prefixos parecidos ("im", "i") e também sufixos parecidos ("são", "ão"). Para a palavras "recrutada", vemos também semelhanças de prefixos e sufixos com as palavras próximas, e também uma semelhança de tamanho, todas a palavras têm 9 caracteres. Podemos notar essas semelhanças para a outras palavras w_{oov} da tabela.

Tabela 6.20: Os *tokens* com mais erros e as etiquetas que eles possuem no conjunto de teste do Tycho Brahe.

Palavra	**Etiquetas**	**# de erros**
que	CONJ, WPRO, FW, WD, C, D, CONJS, WD-P	1208
a	N, CL, FW, D, NPR, P, D-F	230
se	WQ, CONJ, FW, WD, CONJS, SE	189
porque	WADV, CONJ, P+WPRO, CONJS	127
como	WADV, CONJ, WPRO, WD, CONJS, VB-P	90
nem	WADV, CONJ, NEG, P, CONJ-NEG	73
o	N, CL, FW, D, NPR, P	63
as	D-P, CL, D-F-P, D-F, P	55
foi	VB-D, SR-D	42
até	P, FP	31

Tabela 6.21: Cinco palavras fora do vocabulário e suas palavras mais similares de acordo com a representação de *character embedding*

imprecisão	**recrutada**	**pré-requisito**	**seccional**
iluminação	rejeitada	pré-campanhas	seleciona
imaginação	resgatado	pré-suposições	semifinal
insinuação	recontada	pernambucano	sertaneja
tranmissão	resultado	curta-metragem	sensorial
implicação	recortada	pressupostos	soluciona

7
Conclusões

7.1
Resumo do trabalho

Nessa dissertação, desenvolvemos um etiquetador morfossintático (*POS Tagger*) para a Língua Portuguesa, usando redes neurais recorrentes. Nossa arquitetura se baseia no trabalho de Ma e Hovy (25), que apresentou em 2016 o *POS Tagger* estado da arte para a língua Inglesa, e no trabalho de Marulli et al. (46) que apresentou em 2018 o *POS Tagger* estado da arte para Língua Italiana.

A arquitetura do nosso *POS Tagger* de Língua Portuguesa, não depende de *features* manuais, ou *features* específicas do domínio ou dados externos ao conjunto de treino. Usamos uma rede neural recorrente BLSTM para resolver o problema de *POS Tagging* para o Português. E que tem como entrada representações de *word embedding* carregados de modelos pré-treinados, combinados a representações de *character embedding* gerados por outra camada de BLSTM.

Aplicamos a arquitetura **BLSTM-WE-CE** para *POS Tagging* em três corpora de Língua Portuguesa. Em todas os três corpora apresentamos um desempenho em acurácia e acurácia FV ligeiramente superior ao estado da arte reportado por Fonseca et al. (45) e Santos e Zadrozny (44). Obtivemos para o corpus Mac-Morpho v1, 97.83% de acurácia em média e 95.48% de acurácia FV em média. Para o corpus Mac-Morpho v2, 97.65% de acurácia em média e 95.23% de acurácia FV em média. E para o corpus Tycho Brahe obtivemos, 97.35% de acurácia em média e 87.40% de acurácia FV em média.

Avaliamos ainda separadamente o desempenho de cada etiqueta nos três corpora com as medidas de F_1-score, *precision* e *recall*. Notamos que as etiquetas com um baixo número de exemplos no conjunto de treino (baixa representatividade), apresentam em muitos casos, um baixo desempenho em F_1-score. Porém, a maioria das etiquetas nos três corpora apresentam uma boa avaliação, em torno de 90% de F_1-score. Tiramos a média das etiquetas por corpus e vimos que o corpus Mac-Morpho v1 obteve a melhor avaliação média (F_1=0.948, *precision*=0.953, *recall*=0.945), seguido do corpus Mac-Morpho v2

(F_1=0.902, *precision*=0.908 e *recall*=0.899) e corpus Tycho Brahe (F_1=0.740, *precision*=0.753 e *recall*=0.743).

7.2 Contribuições

Nesse trabalho demonstramos empiricamente que a arquitetura proposta (BLSTM-WE-CE), que usa uma rede BLSTM para classificar as etiquetas, e uma outra rede BLSTM para gerar representações de *character embedding*, é bem adequada para treinar *POS Taggers* para a Língua Portuguesa, tanto para o Português Contemporâneo, quando para o Português Histórico.

Criamos *POS Taggers* para a Língua Portuguesa que superam ligeiramente o estado da arte em termos de acurácia e acurácia FV. Avaliamos três corpora diferentes, o Mac-Morpho v1 (Ac.=97.83%, Ac. FV= 95.48) o Mac-Morpho v2 (Ac.=97.65%, Ac. FV=95.23), que são compostos de sentenças do Português Contemporâneo. E ainda o corpus Tycho Brahe (Ac.=97.35, Ac. FV= 87.40), composto de sentenças do Português Histórico.

Realizamos ainda um estudo empírico para descobrir qual dentre os mais populares algoritmos de criação *word embedding*, é mais adequado para a atividade *POS Tagging* em Português. Analisamos os modelos Glove e Word2Vec, FastText e Wang2Vec em suas duas versões (CBOW e Skip-Gram). O modelo Wang2Vec apresentou o melhor desempenho em acurácia para os três corpora analisadas. O bom desempenho de modelo Wang2vec, pode vir provavelmente, de seu foco na captura de informações sintáticas, que leva em conta a ordem em que as palavras aparecem, o que não é feito pelo os outros modelos. E esse tipo de informação se mostrou bastante útil para a classificação morfossintática em Português com nossa arquitetura de rede neural.

7.3 Trabalhos futuros

Existem algumas possibilidades de trabalhos futuros. Poderíamos aplicar a arquitetura apresentada nesse trabalho em outras atividades de PLN em Português, como exemplo reconhecimento de entidades nomeadas em diversos domínios (em textos de redes sociais, no meio jurídico e em aplicações da indústria de óleo de gás).

Nessa dissertação investigamos o uso de diferentes modelos de *word embeddings* (Word2Vec, FastText, Wang2Vec e Glove) no contexto da atividade de *POS Tagging*. Recentemente, em 2018 Peters et al. (71) propuseram uma nova técnica para gerar representações de *word embeddings*, o chamado ELMo (*Embeddings from Language Models*). Diferentes dos modelos investigados nesse

trabalho, o modelo ELMo gerar representações vetoriais contextualizadas para as palavras, portanto a mesma palavra pode ter diferentes vetores dependendo do contexto que é usada. Essa técnica melhorou significativamente o estado da arte em uma ampla variedade de problemas de PLN. Uma possibilidade de trabalho futuro seria aplicar os vetores de *embeddings* contextualizados ELMo na arquitetura de *POS tagging* em Português.

No campo da visão computacional, os pesquisadores mostraram repetidamente o valor do chamado *transfer learning*. Em que redes neurais são treinas para uma certa atividade usando grande bancos de imagens. Então os pesos gerados por essas redes são usados como um *basis* inicial para treinar novas redes em outras atividades especificas.

Também no ano de 2018, pesquisados do *Google AI Language* propuseram o BERT (72) (*Bidirectional Encoder Representations from Transformers*), um modelo que pode ser usado como pré-treino para redes neurais em atividades de PLN, realizando o *transfer learning*. Com o BERT os pesquisados alcançaram o estado da arte em 11 atividade de PLN. Uma possibilidade de trabalho futuro seria estudar o impacto do uso do modelo BERT em atividades de PLN em Português.

Referências bibliográficas

[1] BOHNET, B.; MCDONALD, R.; SIMÕES, G.; ANDOR, D.; PITLER, E. ; MAYNEZ, J.. **Morphosyntactic tagging with a meta-bilstm model over context sensitive token encodings**. In: PROCEEDINGS OF THE 56TH ANNUAL MEETING OF THE ASSOCIATION FOR COMPUTATIONAL LINGUISTICS (VOLUME 1: LONG PAPERS), volumen 1, p. 2642–2652, 2018.

[2] CHOWDHURY, G. G.. **Natural language processing**. Annual review of information science and technology, 37(1):51–89, 2003.

[3] COPPIN, B.. **Artificial intelligence illuminated**. Jones and Bartlett Publishers, 2004.

[4] RANJAN, N.; MUNDADA, K.; PHALTANE, K. ; AHMAD, S.. **A survey on techniques in nlp**. International Journal of Computer Applications, 134(8):6–9, 2016.

[6] HIRSCHBERG, J.; MANNING, C. D.. **Advances in natural language processing**. Science, 349(6245):261–266, 2015.

[7] MARR, B.. **How much data do we create every day? the mind-blowing stats everyone should read**, 2018.

[8] GO, A.; BHAYANI, R. ; HUANG, L.. **Twitter sentiment classification using distant supervision**. CS224N Project Report, Stanford, 1(12), 2009.

[9] DE HOLANDA MAIA, M. R.; XEXÉO, G. B.. **Part-of-speech tagging of portuguese using hidden markov models with character language model emissions**. In: PROCEEDINGS OF THE 8TH BRAZILIAN SYMPOSIUM IN INFORMATION AND HUMAN LANGUAGE TECHNOLOGY, 2011.

[10] ZHANG, D.; LEE, W. S.. **Question classification using support vector machines**. In: PROCEEDINGS OF THE 26TH ANNUAL INTERNATIONAL ACM SIGIR CONFERENCE ON RESEARCH AND DEVELOPMENT IN INFORMAION RETRIEVAL, p. 26–32. ACM, 2003.

[11] GOLDBERG, Y.. **A primer on neural network models for natural language processing**. Journal of Artificial Intelligence Research, 57:345–420, 2016.

[12] GIMPEL, K.; SCHNEIDER, N.; O'CONNOR, B.; DAS, D.; MILLS, D.; EISENSTEIN, J.; HEILMAN, M.; YOGATAMA, D.; FLANIGAN, J. ; SMITH, N. A.. **Part-of-speech tagging for twitter: Annotation, features, and experiments**. In: PROCEEDINGS OF THE 49TH ANNUAL MEETING OF THE ASSOCIATION FOR COMPUTATIONAL LINGUISTICS: HUMAN LANGUAGE TECHNOLOGIES: SHORT PAPERS-VOLUME 2, p. 42–47. Association for Computational Linguistics, 2011.

[13] GIMÉNEZ, J.; MARQUEZ, L.. **Svmtool: A general pos tagger generator based on support vector machines**. In: IN PROCEEDINGS OF THE 4TH INTERNATIONAL CONFERENCE ON LANGUAGE RESOURCES AND EVALUATION. Citeseer, 2004.

[14] DAS, O.; BALABANTARAY, R. C.. **Sentiment analysis of movie reviews using pos tags and term frequencies**. International Journal of Computer Applications, 96(25), 2014.

[15] MA, J.; LIU, H.; HUANG, D. ; SHENG, W.. **An english part-of-speech tagger for machine translation in business domain**. In: NATURAL LANGUAGE PROCESSING ANDKNOWLEDGE ENGINEERING (NLP-KE), 2011 7TH INTERNATIONAL CONFERENCE ON, p. 183–189. IEEE, 2011.

[16] WANG, W.; AUER, J.; PARASURAMAN, R.; ZUBAREV, I.; BRANDYBERRY, D. ; HARPER, M.. **A question answering system developed as a project in a natural language processing course**. In: PROCEEDINGS OF THE 2000 ANLP/NAACL WORKSHOP ON READING COMPREHENSION TESTS AS EVALUATION FOR COMPUTER-BASED LANGUAGE UNDERSTANDING SYTEMS-VOLUME 6, p. 28–35. Association for Computational Linguistics, 2000.

[17] SCHMIDHUBER, J.. **Deep learning in neural networks: An overview**. Neural networks, 61:85–117, 2015.

[18] HATCHER, W. G.; YU, W.. **A survey of deep learning: Platforms, applications and emerging research trends**. IEEE Access, 6:24411–24432, 2018.

[19] LIU, W.; WANG, Z.; LIU, X.; ZENG, N.; LIU, Y. ; ALSAADI, F. E.. **A survey of deep neural network architectures and their applications**. Neurocomputing, 234:11–26, 2017.

[20] ABRAHAM, A.. **Artificial neural networks**. handbook of measuring system design, 2005.

[22] HOCHREITER, S.; SCHMIDHUBER, J.. **Long short-term memory**. Neural computation, 9(8):1735–1780, 1997.

[23] GOODFELLOW, I.; BENGIO, Y. ; COURVILLE, A.. **Deep Learning**. MIT Press, 2016. `http://www.deeplearningbook.org`.

[24] LECUN, Y.; BENGIO, Y. ; HINTON, G.. **Deep learning**. nature, 521(7553):436, 2015.

[25] MA, X.; HOVY, E.. **End-to-end sequence labeling via bi-directional lstm-cnns-crf**. In: PROCEEDINGS OF THE 54TH ANNUAL MEETING OF THE ASSOCIATION FOR COMPUTATIONAL LINGUISTICS (VOLUME 1: LONG PAPERS), volumen 1, p. 1064–1074, 2016.

[26] YOUNG, T.; HAZARIKA, D.; PORIA, S. ; CAMBRIA, E.. **Recent trends in deep learning based natural language processing**. ieee Computational intelligenCe magazine, 13(3):55–75, 2018.

[27] HARTMANN, N.; FONSECA, E.; SHULBY, C.; TREVISO, M.; SILVA, J. ; ALUÍSIO, S.. **Portuguese word embeddings: Evaluating on word analogies and natural language tasks**. In: PROCEEDINGS OF THE 11TH BRAZILIAN SYMPOSIUM IN INFORMATION AND HUMAN LANGUAGE TECHNOLOGY, p. 122–131, 2017.

[28] LAI, S.; LIU, K.; HE, S. ; ZHAO, J.. **How to generate a good word embedding**. IEEE Intelligent Systems, 31(6):5–14, 2016.

[29] BENGIO, Y.; DUCHARME, R.; VINCENT, P. ; JAUVIN, C.. **A neural probabilistic language model**. Journal of machine learning research, 3(Feb):1137–1155, 2003.

[31] PENNINGTON, J.; SOCHER, R. ; MANNING, C.. **Glove: Global vectors for word representation**. In: PROCEEDINGS OF THE 2014 CONFERENCE ON EMPIRICAL METHODS IN NATURAL LANGUAGE PROCESSING (EMNLP), p. 1532–1543, 2014.

[32] MIKOLOV, T.; CHEN, K.; CORRADO, G. S. ; DEAN, J.. **Efficient estimation of word representations in vector space**. CoRR, abs/1301.3781, 2013.

[33] BOJANOWSKI, P.; GRAVE, E.; JOULIN, A. ; MIKOLOV, T.. **Enriching word vectors with subword information**. Transactions of the Association for Computational Linguistics, 5:135–146, 2017.

[34] LING, W.; DYER, C.; BLACK, A. W. ; TRANCOSO, I.. **Two/too simple adaptations of word2vec for syntax problems**. In: PROCEEDINGS OF THE 2015 CONFERENCE OF THE NORTH AMERICAN CHAPTER OF THE ASSOCIATION FOR COMPUTATIONAL LINGUISTICS: HUMAN LANGUAGE TECHNOLOGIES, p. 1299–1304. Association for Computational Linguistics, 2015.

[35] DICKINSON, B.; HU, W.. **Sentiment analysis of investor opinions on twitter**. Social Networking, 4(03):62, 2015.

[36] DOS SANTOS, C.; GATTI, M.. **Deep convolutional neural networks for sentiment analysis of short texts**. In: PROCEEDINGS OF COLING 2014, THE 25TH INTERNATIONAL CONFERENCE ON COMPUTATIONAL LINGUISTICS: TECHNICAL PAPERS, p. 69–78, 2014.

[37] KIM, Y.; JERNITE, Y.; SONTAG, D. ; RUSH, A. M.. **Character-aware neural language models.** In: AAAI, p. 2741–2749, 2016.

[38] GHAZVININEJAD, M.; SHI, X.; CHOI, Y. ; KNIGHT, K.. **Generating topical poetry**. In: PROCEEDINGS OF THE 2016 CONFERENCE ON EMPIRICAL METHODS IN NATURAL LANGUAGE PROCESSING, p. 1183–1191, 2016.

[39] VENUGOPALAN, S.; XU, H.; DONAHUE, J.; ROHRBACH, M.; MOONEY, R. ; SAENKO, K.. **Translating videos to natural language using deep recurrent neural networks**. In: PROCEEDINGS OF THE 2015 CONFERENCE OF THE NORTH AMERICAN CHAPTER OF THE ASSOCIATION FOR COMPUTATIONAL LINGUISTICS: HUMAN LANGUAGE TECHNOLOGIES, p. 1494–1504, 2015.

[40] CORRÊA, D. C.; SALVADEO, D. H.; LEVADA, A. L.; SAITO, J. H. ; DA, N.. **Using lstm network in face classification problems**, 2008.

[41] GRAVES, A.; SCHMIDHUBER, J.. **Framewise phoneme classification with bidirectional lstm and other neural network architectures**. Neural Networks, 18(5-6):602–610, 2005.

[42] COLLOBERT, R.; WESTON, J.; BOTTOU, L.; KARLEN, M.; KAVUKCUOGLU, K. ; KUKSA, P.. **Natural language processing (almost) from scratch**. Journal of Machine Learning Research, 12(Aug):2493–2537, 2011.

[43] FERNANDES, E. R.; RODRIGUES, I. M. ; MILIDIÚ, R. L.. **Portuguese part-of-speech tagging with large margin structure learning**. In: INTELLIGENT SYSTEMS (BRACIS), 2014 BRAZILIAN CONFERENCE ON, p. 25–30. IEEE, 2014.

[44] DOS SANTOS, C. N.; ZADROZNY, B.. **Training state-of-the-art portuguese pos taggers without handcrafted features**. In: INTERNATIONAL CONFERENCE ON COMPUTATIONAL PROCESSING OF THE PORTUGUESE LANGUAGE, p. 82–93. Springer, 2014.

[45] FONSECA, E. R.; ROSA, J. L. G. ; ALUÍSIO, S. M.. **Evaluating word embeddings and a revised corpus for part-of-speech tagging in portuguese**. Journal of the Brazilian Computer Society, 21(1):2, 2015.

[46] MARULLI, F.; POTA, M. ; ESPOSITO, M.. **A comparison of character and word embeddings in bidirectional lstms for pos tagging in italian**. In: INTERNATIONAL CONFERENCE ON INTELLIGENT INTERACTIVE MULTIMEDIA SYSTEMS AND SERVICES, p. 14–23. Springer, 2018.

[47] ALUÍSIO, S.; PELIZZONI, J.; MARCHI, A. R.; DE OLIVEIRA, L.; MANENTI, R. ; MARQUIAFÁVEL, V.. **An account of the challenge of tagging a reference corpus for brazilian portuguese**. In: INTERNATIONAL WORKSHOP ON COMPUTATIONAL PROCESSING OF THE PORTUGUESE LANGUAGE, p. 110–117. Springer, 2003.

[48] FONSECA, E. R.; ROSA, J. L. G.. **Mac-morpho revisited: Towards robust part-of-speech tagging**. In: PROCEEDINGS OF THE 9TH BRAZILIAN SYMPOSIUM IN INFORMATION AND HUMAN LANGUAGE TECHNOLOGY, 2013.

[49] TEMPONI, C. N.; OTHERS. **O corpus anotado do português histórico: um avanço para as pesquisas em lingüística histórica do português**. Revista Virtual de Estudos da Linguagem: ReVEL, 2(3):1, 2004.

[50] CAMBRIA, E.; WHITE, B.. **Jumping nlp curves: A review of natural language processing research**. IEEE Computational intelligence magazine, 9(2):48–57, 2014.

[51] JUNG, S.; LEE, C. ; HWANG, H.. **End-to-end korean part-of-speech tagging using copying mechanism**. ACM Transactions on Asian and Low-Resource Language Information Processing (TALLIP), 17(3):19, 2018.

[52] WANG, P.; QIAN, Y.; SOONG, F. K.; HE, L. ; ZHAO, H.. **Part-of-speech tagging with bidirectional long short-term memory recurrent neural network**. arXiv preprint arXiv:1510.06168, 2015.

[53] SRINIVASA, K.; ANUPINDI, S.; SHARATH, R. ; CHAITANYA, S. K.. **Analysis of facial expressiveness captured in reaction to videos**. In: 2017 IEEE 7TH INTERNATIONAL ADVANCE COMPUTING CONFERENCE (IACC), p. 664–670. IEEE, 2017.

[54] SCHUSTER, M.; PALIWAL, K. K.. **Bidirectional recurrent neural networks**. IEEE Transactions on Signal Processing, 45(11):2673–2681, 1997.

[55] ZHANG, Q.; YANG, L. T.; CHEN, Z. ; LI, P.. **A survey on deep learning for big data**. Information Fusion, 42:146–157, 2018.

[56] OLAH, C.. **Understanding lstm networks**, 2015.

[57] DENG, L.; LIU, Y.. **Deep Learning in Natural Language Processing**. Springer, 2018.

[58] GUPTA, S.. **Word embeddings in nlp and its applications**, 2019.

[59] RONG, X.. **word2vec parameter learning explained**. arXiv preprint arXiv:1411.2738, 2014.

[60] KARANI, D.. **Introduction to word embedding and word2vec**, 2018.

[61] FINGER, M.. **Técnicas de otimizaçao da precisao empregadas no etiquetador tycho brahe**. Proceedings of PROPOR, Sao Paulo, p. 141–154, 2000.

[62] BRILL, E. D.. **A corpus-based approach to language learning**. IRCS Technical Reports Series, p. 191, 1993.

[63] BRILL, E.. **Transformation-based error-driven learning and natural language processing: A case study in part-of-speech tagging**. Computational linguistics, 21(4):543–565, 1995.

[64] KEPLER, F. N.; FINGER, M.. **Part-of-speech tagging of portuguese based on variable length markov chains**. In: INTERNATIONAL

WORKSHOP ON COMPUTATIONAL PROCESSING OF THE PORTUGUESE LANGUAGE, p. 248–251. Springer, 2006.

[65] DOS SANTOS, C. N.; MILIDIÚ, R. L. ; RENTERÍA, R. P.. **Portuguese part-of-speech tagging using entropy guided transformation learning**. In: INTERNATIONAL CONFERENCE ON COMPUTATIONAL PROCESSING OF THE PORTUGUESE LANGUAGE, p. 143–152. Springer, 2008.

[66] SRIVASTAVA, N.; HINTON, G.; KRIZHEVSKY, A.; SUTSKEVER, I. ; SALAKHUTDINOV, R.. **Dropout: a simple way to prevent neural networks from overfitting**. The Journal of Machine Learning Research, 15(1):1929–1958, 2014.

[68] KINGMA, D. P.; BA, J.. **Adam: a method for stochastic optimization. iclr (2015)**, 2015.

[69] PETERS, M.; AMMAR, W.; BHAGAVATULA, C. ; POWER, R.. **Semi-supervised sequence tagging with bidirectional language models**. In: PROCEEDINGS OF THE 55TH ANNUAL MEETING OF THE ASSOCIATION FOR COMPUTATIONAL LINGUISTICS (VOLUME 1: LONG PAPERS), p. 1756–1765, 2017.

[70] CHIU, J. P.; NICHOLS, E.. **Named entity recognition with bidirectional lstm-cnns**. Transactions of the Association for Computational Linguistics, 4:357–370, 2016.

[71] PETERS, M. E.; NEUMANN, M.; IYYER, M.; GARDNER, M.; CLARK, C.; LEE, K. ; ZETTLEMOYER, L.. **Deep contextualized word representations**. In: PROCEEDINGS OF NAACL-HLT, p. 2227–2237, 2018.

[72] DEVLIN, J.; CHANG, M.-W.; LEE, K. ; TOUTANOVA, K.. **Bert: Pre-training of deep bidirectional transformers for language understanding**. In: PROCEEDINGS OF THE 2019 CONFERENCE OF THE NORTH AMERICAN CHAPTER OF THE ASSOCIATION FOR COMPUTATIONAL LINGUISTICS: HUMAN LANGUAGE TECHNOLOGIES, VOLUME 1 (LONG AND SHORT PAPERS), p. 4171–4186, 2019.

A
Descrição das etiquetas do Tycho Brahe corpus

Este apêndice contém tabelas com informações de descrição e distribuição das 265 etiquetas do Tycho Brahe corpus descrito no Capítulo 6 desta dissertação. A Tabela A.1 mostra a distribuição das etiquetas nos conjuntos de treino, desenvolvimento e teste.

Tabela A.1: Distribuição das etiquetas no corpus Tycho Brahe

Etiquetas	Treino	Desen.	Teste	Total
N	74197	4102	26300	104599 (10.1%)
,	60818	3370	21447	85635 (8.27%)
P	60610	3430	21465	85505 (8.26%)
CONJ	35923	2048	12879	50850 (4.91%)
NPR	33743	1763	11886	47392 (4.58%)
.	32809	1746	11520	46075 (4.45%)
N-P	30342	1662	10963	42967 (4.15%)
VB-P	23569	1308	8445	33322 (3.22%)
WPRO	19677	1127	6926	27730 (2.68%)
ADV	18360	1023	6490	25873 (2.5%)
D	17841	953	6331	25125 (2.43%)
VB-D	17515	937	6161	24613 (2.38%)
VB	16426	932	5875	23233 (2.24%)
D-F	15162	817	5272	21251 (2.05%)
P+D	13608	689	4964	19261 (1.86%)
C	13214	737	4540	18491 (1.79%)
P+D-F	12739	710	4492	17941 (1.73%)
CL	11620	630	4063	16313 (1.58%)
NEG	9656	563	3373	13592 (1.31%)
ADJ	9035	494	3232	12761 (1.23%)
CONJS	8112	464	2886	11462 (1.11%)
D-P	7787	415	2672	10874 (1.05%)
ADJ-G	7695	425	2649	10769 (1.04%)
ADV-R	7492	464	2690	10646 (1.03%)
SE	7577	445	2580	10602 (1.02%)
SR-P	7487	407	2614	10508 (1.01%)
FW	7679	390	2332	10401 (1.0%)
PRO$-F	7349	458	2530	10337 (1.0%)
ADJ-F	6572	360	2228	9160 (0.88%)
PRO	6354	360	2282	8996 (0.87%)
Q	5608	325	1934	7867 (0.76%)
D-F-P	5352	271	2043	7666 (0.74%)
P+D-P	5325	313	1996	7634 (0.74%)
VB-SP	4696	297	1543	6536 (0.63%)
NUM	4487	212	1425	6124 (0.59%)
VB-AN	4236	260	1486	5982 (0.58%)
D-UM	3887	208	1395	5490 (0.53%)
P+D-F-P	3803	205	1348	5356 (0.52%)
VB-G	3770	203	1295	5268 (0.51%)
PRO$	3530	185	1292	5007 (0.48%)
ADJ-P	3523	191	1183	4897 (0.47%)
(	3405	199	1257	4861 (0.47%)
NPR-P	3341	152	1204	4697 (0.45%)
Q-P	2932	182	1017	4131 (0.4%)
SR-D	2882	137	1039	4058 (0.39%)
ADJ-G-P	2815	159	1030	4004 (0.39%)
VB-R	2661	175	1038	3874 (0.37%)
D-UM-F	2709	156	971	3836 (0.37%)
ADJ-F-P	2510	133	935	3578 (0.35%)
TR-P	2287	146	799	3232 (0.31%)
VB-AN-F	2216	121	798	3135 (0.3%)
VB-SD	2020	132	768	2920 (0.28%)
P+PRO	1968	110	732	2810 (0.27%)
VB-AN-P	1836	107	694	2637 (0.25%)
Q-F-P	1816	82	705	2603 (0.25%)
WADV	1808	112	677	2597 (0.25%)
ADJ-R-G	1788	85	618	2491 (0.24%)
FP	1659	101	661	2421 (0.23%)
Q-F	1597	125	566	2288 (0.22%)
PRO$-P	1553	108	548	2209 (0.21%)
DEM	1524	79	572	2175 (0.21%)

Continuação da Tabela A.1

Etiquetas	Treino	Desen.	Teste	Total
CONJ-NEG	1482	75	564	2121 (0.2%)
SR	1471	82	501	2054 (0.2%)
HV-P	1450	85	508	2043 (0.2%)
VB-PP	1363	86	500	1949 (0.19%)
QT	1401	58	480	1939 (0.19%)
VB+CL	1317	82	468	1867 (0.18%)
TR-D	1285	65	436	1786 (0.17%)
PRO$-F-P	1252	55	414	1721 (0.17%)
ET-P	1158	81	397	1636 (0.16%)
VB-RA	1147	59	390	1596 (0.15%)
VB-AN-F-P	1090	65	397	1552 (0.15%)
VB-F	1051	49	367	1467 (0.14%)
VB-SR	869	56	305	1230 (0.12%)
Q-NEG	804	42	325	1171 (0.11%)
OUTRO-P	770	54	257	1081 (0.1%)
HV-D	772	38	226	1036 (0.1%)
Q-G	725	38	252	1015 (0.1%)
OUTRO	680	36	250	966 (0.09%)
SR-R	704	32	226	962 (0.09%)
VB-G+CL	681	31	233	945 (0.09%)
OUTRO-F	636	35	274	945 (0.09%)
SR-SP	671	32	225	928 (0.09%)
ET-D	635	31	223	889 (0.09%)
VB+SE	628	33	210	871 (0.08%)
VB-D+CL	594	39	217	850 (0.08%)
HV-P+P	587	33	225	845 (0.08%)
SENAO	577	27	209	813 (0.08%)
VB-P+SE	572	35	197	804 (0.08%)
VB-P+CL	582	23	198	803 (0.08%)
TR	562	40	199	801 (0.08%)
OUTRO-F-P	554	28	183	765 (0.07%)
VB-D+SE	537	20	175	732 (0.07%)
WD	452	29	164	645 (0.06%)
ADJ-R-G-P	455	17	172	644 (0.06%)
SR-G	456	19	142	617 (0.06%)
P+DEM	391	16	148	555 (0.05%)
VB-G+SE	354	27	148	529 (0.05%)
ADJ-S	369	23	115	507 (0.05%)
ADV-NEG	344	19	135	498 (0.05%)
SR-SD	319	19	128	466 (0.05%)
NUM-F	306	22	127	455 (0.04%)
VB-I	330	13	100	443 (0.04%)
P+ADV	303	19	120	442 (0.04%)
WQ	330	13	95	438 (0.04%)
P+WADV	306	11	103	420 (0.04%)
CL+CL	279	11	100	390 (0.04%)
HV	265	16	102	383 (0.04%)
WPRO-P	259	19	102	380 (0.04%)
D-UM-P	266	12	91	369 (0.04%)
ADJ-S-F	246	31	86	363 (0.04%)
INTJ	248	14	88	350 (0.03%)
Q-NEG-F	222	12	75	309 (0.03%)
TR-SP	208	7	86	301 (0.03%)
VB-SP+CL	211	12	66	289 (0.03%)
ADJ-R-F-P	209	9	63	281 (0.03%)
ADJ-R-F	196	9	68	273 (0.03%)
ET	183	10	75	268 (0.03%)
ADJ-R-P	184	13	65	262 (0.03%)
SR-RA	191	3	56	250 (0.02%)
SR-SR	183	8	55	246 (0.02%)
TR-R	174	7	57	238 (0.02%)
VB-SP+SE	132	10	64	206 (0.02%)
D-UM-F-P	134	11	60	205 (0.02%)
SR-F	141	9	53	203 (0.02%)
TR-G	137	4	44	185 (0.02%)
ADJ-R	123	4	54	181 (0.02%)
VB-I+CL	127	6	44	177 (0.02%)
WPRO$-F	112	8	47	167 (0.02%)
WD-P	119	8	36	163 (0.02%)
HV-SP	107	7	34	148 (0.01%)
P+D-UM	117	3	27	147 (0.01%)
ADJ-S-P	99	3	38	140 (0.01%)
TR-SD	94	8	35	137 (0.01%)
HV-G	93	1	30	124 (0.01%)
P+D-UM-F	78	8	36	122 (0.01%)
ET-G	87	5	29	121 (0.01%)
WPRO$	87	3	28	118 (0.01%)
P+NPR	85	5	28	118 (0.01%)
HV-SD	77	4	33	114 (0.01%)
HV-R	83	2	20	105 (0.01%)
ADJ-S-F-P	57	4	24	85 (0.01%)
SR-PP	59	2	24	85 (0.01%)
D-G	61	3	17	81 (0.01%)
TR-RA	52	3	20	75 (0.01%)
VB-R!CL	54	1	18	73 (0.01%)
ET-R	54	1	16	71 (0.01%)
ET-SP	50	3	18	71 (0.01%)
TR-F	52	0	18	70 (0.01%)
TR-SR	47	3	20	70 (0.01%)
HV-RA	40	4	19	63 (0.01%)
WPRO$-P	39	1	20	60 (0.01%)
HV-SR	42	2	10	54 (0.01%)
WPRO$-F-P	31	3	16	50 (0.0%)

Continuação da Tabela A.1

Etiquetas	Treino	Desen.	Teste	Total
WD-F-P	34	2	11	47 (0.0%)
VB-F+SE	33	1	12	46 (0.0%)
P+WPRO	31	3	12	46 (0.0%)
P+CL	31	0	14	45 (0.0%)
VB-F+CL	30	1	11	42 (0.0%)
D-G-P	28	2	6	36 (0.0%)
XX	28	2	5	35 (0.0%)
VB-P+SE+CL	27	0	8	35 (0.0%)
VB+CL+CL	20	1	14	35 (0.0%)
ET-SR	25	1	6	32 (0.0%)
VB-R!SE	18	2	10	30 (0.0%)
ET-RA	24	0	5	29 (0.0%)
P+OUTRO-F	20	2	7	29 (0.0%)
TR-PP	20	1	7	28 (0.0%)
ADV-S	20	2	5	27 (0.0%)
ET-SD	21	0	6	27 (0.0%)
HV-F	17	0	10	27 (0.0%)
ET-F	14	0	11	25 (0.0%)
Q-NEG-P	11	1	13	25 (0.0%)
P+OUTRO-F-P	16	2	6	24 (0.0%)
VB-G+SE+CL	18	0	5	23 (0.0%)
WD-F	16	0	5	21 (0.0%)
VB-D+SE+CL	14	0	5	19 (0.0%)
TR-P+CL	13	0	6	19 (0.0%)
HV-G+SE	13	1	5	19 (0.0%)
WPRO-F-P	12	2	4	18 (0.0%)
HV-PP	11	2	5	18 (0.0%)
ET-PP	10	1	6	17 (0.0%)
P+OUTRO	12	1	4	17 (0.0%)
P+OUTRO-P	7	3	5	15 (0.0%)
Q-G-P	11	0	4	15 (0.0%)
TR+CL	10	1	3	14 (0.0%)
TR-G+CL	12	0	1	13 (0.0%)
HV+CL	9	0	2	11 (0.0%)
Q-NEG-F-P	8	2	1	11 (0.0%)
TR-D+CL	8	2	1	11 (0.0%)
TR-I	6	1	4	11 (0.0%)
SR+CL	7	1	2	10 (0.0%)
VB-RA+CL	7	0	3	10 (0.0%)
VB+SE+CL	7	1	2	10 (0.0%)
VB-RA+SE	5	0	5	10 (0.0%)
HV-G+CL	8	0	1	9 (0.0%)
ET-I	7	0	2	9 (0.0%)
TR-SP+CL	6	0	2	8 (0.0%)
HV-P+CL	6	0	2	8 (0.0%)
SR-P+CL	6	1	1	8 (0.0%)
SR-G+CL	6	1	1	8 (0.0%)
P+Q-F	3	2	2	7 (0.0%)
TR-AN	1	0	6	7 (0.0%)
P+D-UM-P	3	1	2	6 (0.0%)
P+Q	5	0	1	6 (0.0%)
P+Q-P	6	0	0	6 (0.0%)
WPRO-F	5	0	1	6 (0.0%)
SR-D+CL	5	0	1	6 (0.0%)
P+Q-F-P	5	0	1	6 (0.0%)
ET-P+CL	5	0	1	6 (0.0%)
VB-P+P	5	0	1	6 (0.0%)
ADV+CL	4	0	1	5 (0.0%)
HV-AN-P	3	0	2	5 (0.0%)
VB-SP+SE+CL	3	0	1	4 (0.0%)
HV+SE	2	0	2	4 (0.0%)
TR-G+SE	2	1	1	4 (0.0%)
TR+SE	2	0	2	4 (0.0%)
TR-P+SE	2	1	1	4 (0.0%)
SR-I	0	0	4	4 (0.0%)
VB-SP+CL+CL	4	0	0	4 (0.0%)
HV-I	3	0	1	4 (0.0%)
P+WD	4	0	0	4 (0.0%)
VB-G+CL+CL	4	0	0	4 (0.0%)
VB-SR+CL	4	0	0	4 (0.0%)
P+D-UM-F-P	4	0	0	4 (0.0%)
TR-I+CL	1	1	1	3 (0.0%)
VB-D+CL+CL	3	0	0	3 (0.0%)
HV-AN-F	1	0	2	3 (0.0%)
ET-P+SE	3	0	0	3 (0.0%)
VB-SR+SE	3	0	0	3 (0.0%)
D+OUTRO	1	0	2	3 (0.0%)
TR-F+CL	2	0	1	3 (0.0%)
VB-P+CL+CL	1	0	1	2 (0.0%)
VB-F+CL+CL	1	0	1	2 (0.0%)
HV-F+SE	1	0	1	2 (0.0%)
VB-SD+CL	1	0	1	2 (0.0%)
SR-I+CL	1	0	1	2 (0.0%)
TR-D+SE	1	0	1	2 (0.0%)
SR-R!CL	1	0	1	2 (0.0%)
HV-P+P+SE	2	0	0	2 (0.0%)
VB+P	2	0	0	2 (0.0%)

Continuação da Tabela A.1				
Etiquetas	**Treino**	**Desen.**	**Teste**	**Total**
HV-P+P+CL	2	0	0	2 (0.0%)
HV-P+SE	2	0	0	2 (0.0%)
VB-R+CL	2	0	0	2 (0.0%)
SR-SP+CL	1	0	1	2 (0.0%)
D+OUTRO-F	2	0	0	2 (0.0%)
HV-F+CL	2	0	0	2 (0.0%)
D+OUTRO-P	2	0	0	2 (0.0%)
TR-R!CL	1	0	1	2 (0.0%)
VB-I+CL+CL	2	0	0	2 (0.0%)
HV-G+CL+CL	0	0	1	1 (0.0%)
HV-SP+SE	1	0	0	1 (0.0%)
W	1	0	0	1 (0.0%)
TR+SE+CL	1	0	0	1 (0.0%)
HV-P+SE+CL	1	0	0	1 (0.0%)
D+OUTRO-F-P	1	0	0	1 (0.0%)
HV-AN-F-P	1	0	0	1 (0.0%)
TR-AN-P	1	0	0	1 (0.0%)
ET+CL	1	0	0	1 (0.0%)
SR-RA+CL	1	0	0	1 (0.0%)
ET-D+CL	1	0	0	1 (0.0%)
ET-D+SE	1	0	0	1 (0.0%)
HV-RA+SE+CL	1	0	0	1 (0.0%)
SR-F+CL	1	0	0	1 (0.0%)
HV-AN	0	0	1	1 (0.0%)

Na Tabela A.2 temos a descrição das 265 etiquetas presentes no corpus, é possível encontra uma descrição completa em: `http://www.tycho.iel.unicamp.br/corpus/manual/pos2016.html`.

Tabela A.2: Descrição e exemplos das etiquetas do corpus Tycho Brahe

Etiqueta	**Significado**	**Exemplos**
(	Pontuação	- ( )
,	Pontuação	, ¿
.	Pontuação	. ; : ? !
ADJ	Adjetivo masculino singular	mesmo, bom, certo, primeiro, necessário
ADJ-F	Adjetivo feminino singular	mesma, boa, primeira, divina, segunda
ADJ-F-P	Adjetivo feminino plural	boas, mesmas, novas, primeiras, necessárias
ADJ-G	Adjetivo duplo gênero singular	grande, real, natural, presente, só
ADJ-G-P	Adjetivo duplo gênero plural	grandes, semelhantes, diferentes, particulares, ilustres
ADJ-P	Adjetivo masculino plural	mesmos, antigos, bons, primeiros, próprios
ADJ-R	Adjetivo exclamativo/comparativo masculino singular	tanto, mesma, tamanho, mínimo, primeiro
ADJ-R-F	Adjetivo exclamativo/comparativo feminino singular	tanta, tamanha, mínima, sêca, tão-pouca
ADJ-R-F-P	Adjetivo exclamativo/comparativo feminino plural	tantas, mesmas
ADJ-R-G	Adjetivo exclamativo/comparativo duplo gênero singular	maior, melhor, tal, menor, bastante
ADJ-R-G-P	Adjetivo exclamativo/comparativo duplo gênero plural	maiores, tais, melhores, menores, bastantes
ADJ-R-P	Adjetivo exclamativo/comparativo masculino plural	tantos, superiores, tantas, maiores
ADJ-S	Adjetivo superlativo masculino singular	excelentíssimo, cristianíssimo, augustíssimo, reverendíssimo, eminentíssimo
ADJ-S-F	Adjetivo superlativo feminino singular	reverendíssima, ilustríssima, grandíssima, santíssima, cristianíssima
ADJ-S-F-P	Adjetivo superlativo feminino plural	grandíssimas, justíssimas, altíssimas, grossíssimas, lindíssimas
ADJ-S-P	Adjetivo superlativo masculuno plural	doutíssimos, sapientíssimos, gravíssimos, nobilíssimos, altíssimos
ADV	Adverbio	ainda, assim, bem, já, sempre
ADV+CL	Advérbio Clítico	eis-me, ei-lo, ei-los
ADV-NEG	Advérbio de negação	nunca, jamais, jámais, jàmais
ADV-R	Advérbio exclamativo/comparativo	mais, tão, menos, tanto, melhor
ADV-S	Advérbio superlativo	gravissimamente, grandissimamente, perfeitissimamente, mais, ardentìssimamente
C	Complementizer	que, quais, ache, uma, deve
CL	Clítico	me, lhe, o, nos, a
CL+CL	Contração Clítico	lho, lha, mo, lhas, lhos
CONJ	Conjunção	e, mas, ou, porque, pois
CONJ-NEG	Conjunção de negação	nem
CONJS	Conjunção	como, se, quando, porque, que
D	Determinante masculino singular	o, este, êste, aquele, esse
D+OUTRO	Determinante contraído masculino singular	estoutro, destoutro
D+OUTRO-F	Determinante contraído feminino singular	aqueloutra, estoutra
D+OUTRO-F-P	Determinante contraído feminino plural	destoutras
D+OUTRO-P	Determinante contraído masculino plural	êstoutros, estoutros
D-F	Determinante femino singular	a, esta, aquela, essa, as
D-F-P	Determinante masculino plural	as, estas, aquelas, essas, ás
D-G	Determinante duplo gênero singular	tal
D-G-P	Determinante duplo gênero plural	tais
D-P	Determinante masculino plural	os, estes, aqueles, êstes, esses
D-UM	Determinante indefinido masculino singular	um, hum, uma, un
D-UM-F	Determinante indefinido feminino singular	uma, huma
D-UM-F-P	Determinante indefinido feminino plural	umas, humas
D-UM-P	Determinante indefinido masculino plural	uns, huns
DEM	Determinante demonstrativo invariável	isto, isso, aquilo, tal, iso
ET	Verbo ESTAR infinitivo (contextos verbais e nominais)	estar, star, esteve, stá
ET+CL	Verbo ESTAR clítico	está-lhe
ET-D	Verbo ESTAR passado	estava, estavam, esteve, estive, stava
ET-D+CL	Verbo ESTAR passado clítico	stava-me
ET-D+SE	-	estava-se
ET-F	Verbo ESTAR infinitivo flexionado	estarem, estares, steverem, starem
ET-G	Verbo ESTAR gerúndio	estando, stando
ET-I	Verbo ESTAR Imperativo	está, estai, esteja, estae
ET-P	Verbo ESTAR presente	está, estão, estou, estamos, stá
ET-P+CL	Verbo ESTAR presente clítico	estão-vos, estou-me, está-lhe, estais-vos
ET-P+SE	Verbo ESTAR	está-se, estão-se
ET-PP	Verbo ESTAR particípio perfeito	estado
ET-R	Verbo ESTAR futuro e condicional	estará, estarão, estaria, estarei, estareis
ET-RA	Verbo ESTAR morfema inflexional -ra	estivera, estiveram
ET-SD	Verbo ESTAR subjuntivo do passado	estivesse, estivessem, stevessem, tomasse, steve
ET-SP	Verbo ESTAR	esteja, estejam, estejão, estejamos, steia
ET-SR	Verbo ESTAR	estiver, estiverem, stever, steverem, estivermos
FP	Partícula de foco	só, mesmo, até, sòmente, mesma
FW	Palavra Estrangeira	et, etc, in, non, est
HV	Verbo HAVER infinitivo (contextos verbais e nominais)	haver, aver, á, he
HV+CL	Verbo HAVER clítico	havê-lo, haver-me, havê-la, havia-o, haver-lhe
HV+SE	Verbo HAVER + SE	haver-se
HV-AN	Verbo HAVER particípio passivo	havido
HV-AN-F	Verbo HAVER particípio passivo feminino	havida
HV-AN-F-P	Verbo HAVER particípio passivo feminino plural	havidas
HV-AN-P	Verbo HAVER particípio passivo masculino plural	havidos
HV-D	Verbo HAVER passado	havia, houve, haviam, houveram, havíamos
HV-F	Verbo HAVER infinitivo flexionado	haverem, havermos, haver
HV-F+CL	Verbo HAVER infinitivo flexionado clítico	haverem-nos
HV-F+SE	Verbo HAVER infinitivo flexionado	haverem-se
HV-G	Verbo HAVER gerúndio	havendo
HV-G+CL	Verbo HAVER gerúndio clítico	havendo-a, havendo-lhe, havendo-nos, havendo-me
HV-G+CL+CL	Verbo HAVER gerúndio contração clítico	havendo-mo
HV-G+SE	Verbo HAVER gerúndio	havendo-se
HV-I	Verbo HAVER imperativo	havei, haja
HV-P	Verbo HAVER presente	há, ha, havemos, hão, hei
HV-P+CL	Verbo HAVER presente clítico	ha-de-nos, haveis-mo, há-as, hei-me, há-me
HV-P+P	Verbo HAVER presente P	há-de, ha-de, hão-de, hei-de, hás-de
HV-P+P+CL	Verbo HAVER presente + P clítico	hei-de-lhe
HV-P+P+SE	Verbo HAVER presente + P + SE	hão-de-se
HV-P+SE	Verbo HAVER presente + SE	hamse, hão-se
HV-P+SE+CL	Verbo HAVER presente + SE clítico	ha-se-lhe
HV-PP	Verbo HAVER particípio perfeito	havido, há, servido
HV-R	Verbo HAVER futuro e condicional	haverá, haveria, haverei, haverão, heveria
HV-RA	Verbo HAVER morfema inflexional -ra	houvera, houvéramos, houveram, houveramos
HV-RA+SE+CL	Verbo HAVER morfema inflexional -ra + SE + clítico	negarselhe-há
HV-SD	Verbo HAVER subjuntivo do passado	houvesse, houvessem, houvéssemos, houvessemos
HV-SP	Verbo HAVER presente do subjuntivo	haja, hajam, há, hajamos, haveis

Continuação da Tabela A.2

Etiqueta	Significado	Exemplos
HV-SP+SE	Verbo HAVER presente do subjuntivo + SE	hajam-se
HV-SR	Verbo HAVER subjuntivo Futuro	houver, houverem
INTJ	Interjeção	oh, ó, adeus, ah, ai
N	Substantivo singular	tempo, parte, mundo, vida, dia
N-P	Substantivo plural	homens, vezes, anos, cousas, dias
NEG	Negação	não, nem, nam, pois, nao
NPR	Substantivo proprio singular	deus, mercê, senhor, excelência, dom
NPR-P	Substantivo proprio plural	reis, portugueses, padres, índios, príncipes
NUM	Número cardeal geral	mil, dois, três, quatro, dous
NUM-F	Número	duas, quinhentas, duzentas, quatrocentas, oitocentas
OUTRO	Tag OUTRO masculino	outro, reverendíssima, ilustríssima, outros, liso
OUTRO-F	Tag OUTRO feminino	outra, outras
OUTRO-F-P	Tag OUTRO feminino plural	outras, outra, essoutras
OUTRO-P	Tag OUTRO masculino plural	outros, *outros, outro, estoutros
P	Preposição	de, a, em, com, por
P+ADV	Preposição contração + ADV	daqui, dali, daí, d'aqui, d'ali
P+CL	Preposição contração + CL	pelo, pelos, polo, pelas, pela
P+D	Preposição contração + D	do, no, ao, pelo, neste
P+D-F	Preposição contração + D-F	da, na, à, pela, desta
P+D-F-P	Preposição contração + D-F	das, nas, às, pelas, destas
P+D-P	Preposição contração + D-P	dos, aos, nos, pelos, destes
P+D-UM	Preposição contração + D-UM	dum, num, d'um, n'um, numa
P+D-UM-F	Preposição contração + D-UM-F	duma, numa, d'uma, n'uma
P+D-UM-F-P	Preposição contração + D-UM-F-P	dumas
P+D-UM-P	Preposição contração + D-UM-P	duns, n'uns, nuns
P+DEM	Preposição contração + DEM	nisto, disto, disso, nisso, porisso
P+NPR	Preposição contração + NPR	d'elrei, delrei, d'el-rei, del-rei, delrey
P+OUTRO	Preposição contração + OUTRO	noutro, doutro, d'outro, n'outro
P+OUTRO-F	Preposição contração + OUTRO-F	noutra, d'outra, doutra, n'outra
P+OUTRO-F-P	Preposição contração + OUTRO-F-P	d'outras, noutras, n'outras, doutras, noutra
P+OUTRO-P	Preposição contração + OUTRO-P	noutros, d'outros, doutros, doutos, n'outros
P+PRO	Preposição contração + PRO	dela, dele, nela, delas, deles
P+Q	Preposição contração + Q	d'algum, d'outrem, dalgum, doutrém, nalgum
P+Q-F	Preposição contração + Q-F	d'alguma, n'alguma, dalguma
P+Q-F-P	Preposição contração + Q-F-P	d'algumas, entr'ambas, dalgumas
P+Q-P	Preposição contração + Q-P	d'alguns, d'ambos
P+WADV	Preposição contração + WADV	donde, aonde, d'onde, adonde, daqui
P+WD	Preposição contração + WD	porque
P+WPRO	Preposição contração + WPRO	porque, porquê
PRO	Pronome oblíquo/pessoal	eu, ele, êle, si, ela
PRO$	Pronome masculino singular	seu, meu, nosso, vosso, teu
PRO$-F	Pronome singular feminino	vossa, sua, minha, nossa, tua
PRO$-F-P	Pronome plural feminino	suas, minhas, nossas, vossas, tuas
PRO$-P	Pronome masculino plural	seus, nossos, meus, vossos, teus
Q	Quantificador	muito, tudo, todo, pouco, algum
Q-F	Quantificador feminino	alguma, toda, muita, tôda, pouca
Q-F-P	Quantificador feminino plural	todas, muitas, algumas, tôdas, poucas
Q-G	Quantificador duplo gênero singular	cada, qualquer, algum
Q-G-P	Quantificador duplo gênero plural	quaisquer, quaesquer
Q-NEG	Quantificador de negação	nada, nenhum, ninguém, ninguem, alguém
Q-NEG-F	Quantificador de negação feminino	nenhuma
Q-NEG-F-P	Quantificador de negação feminino plural	nenhumas
Q-NEG-P	Quantificador de negação plural	alguns, nenhuns
Q-P	Quantificador Plural	todos, muitos, alguns, poucos, ambos
QT	Aspa	"
SE	Se	se
SENAO	Senão	senão, senã
SR	Verbo SER infinitivo (contextos verbais e nominais)	ser, é, ri, haverá, desejo
SR+CL	Verbo SER clítico	sê-lo, ser-lhes, ser-lhe, ser-me, ser-vos
SR-D	Verbo SER passado	era, foi, eram, foram, fui
SR-D+CL	Verbo SER passado clítico	foi-lhe, era-me, foi-me, foi-lhes
SR-F	Verbo SER futuro	serem, serdes, sermos, seres, ser
SR-F+CL	Verbo SER futuro clítico	serem-lhe
SR-G	Verbo SER gerúndio	sendo
SR-G+CL	Verbo SER gerúndio clítico	sendo-lhe, sendo-me
SR-I	Verbo SER imperativo	sede, sê
SR-I+CL	Verbo SER imperativo clítico	achai-vos, sêde-lhes
SR-P	Verbo SER presente	é, são, sou, sois, somos
SR-P+CL	Verbo SER presente clítico	é-nos, são-me, é-me, é-lhes, são-lhe
SR-PP	Verbo SER particípio perfeito	sido
SR-R	Verbo SER futuro e condicional	será, seria, serão, haverá, seriam
SR-R!CL	Verbo SER mesóclise	ser-me-ia, serlhes-ha
SR-RA	Verbo SER morfema inflexional -ra	fôra, fora, foram, fôras, fôreis
SR-RA+CL	Verbo SER morfema inflexional -ra clítico	fôra-o
SR-SD	Verbo SER subjuntivo do passado	fosse, fôsse, fossem, fôssem, fôsseis
SR-SP	Verbo SER presente do subjuntivo	seja, sejam, sejais, sejão, sejas
SR-SP+CL	Verbo SER presente do subjuntivo clítico	seja-me
SR-SR	Verbo SER subjuntivo futuro	for, fôr, forem, fôrem, formos

Continuação da Tabela A.2

Etiqueta	Significado	Exemplos
TR	Verbo TER infinitivo (contextos verbais e nominais)	ter, têem, ternas, desejo, posso
TR+CL	Verbo TER clítico	tê-la, ter-nos, tê-lo, ter-lhe, ter-la
TR+SE	Verbo TER + SE	ter-se, terminar-se
TR+SE+CL	Verbo TER + SE clítico	ter-se-lhe
TR-AN	Verbo TER particípio passivo	tido
TR-AN-P	Verbo TER particípio passivo presente	tidos
TR-D	Verbo TER passado	tinha, teve, tinham, tive, tiveram
TR-D+CL	Verbo TER passado clítico	tinha-me, tinha-o, tinha-lhe, tinhalhe, tinhão-o
TR-D+SE	Verbo TER passado + SE	tinha-se
TR-F	Verbo TER futuro	terem, termos, terdes, téremos
TR-F+CL	Verbo TER futuro clítico	terem-no, terem-nos, terem-na
TR-G	Verbo TER gerúndio	tendo
TR-G+CL	Verbo TER gerúndio clítico	tendo-o, tendo-me, tendo-vos, tergiversandolhe, tendo-os
TR-G+SE	Verbo TER gerúndio + SE	tendo-se
TR-I	Verbo TER imperativo	tende, tem
TR-I+CL	Verbo TER imperativo clítico	tende-o
TR-P	Verbo TER presente	tem, tenho, têm, temos, tendes
TR-P+CL	Verbo TER presente clítico	tenho-o, tem-me, tem-na, tenho-me, tem-no
TR-P+SE	Verbo TER presente + SE	tem-se, têm-se, têem-se
TR-PP	Verbo TER particípio perfeito	tido
TR-R	Verbo TER futuro	terá, teria, terei, terão, teriam
TR-R!CL	Verbo TER futuro mesóclise	telo-hão, tel-o-hão
TR-RA	Verbo TER morfema inflexional -ra	tivera, tevera, tiveram, tevéramos, tiveras
TR-SD	Verbo TER subjuntivo do passado	tivesse, tivessem, tivéssemos, tevessemos, teverem
TR-SP	Verbo TER presente do subjuntivo	tenha, tenham, tenhamos, tenhais, tenhão
TR-SP+CL	Verbo TER presente do subjuntivo	tenha-me, tenha-o, tenha-lhe
TR-SR	Verbo TER subjuntivo futuro	tiver, tiverem, tever, tivermos, tiverdes
VB	Verbo	fazer, dizer, dar, ver, ir
VB+CL	Verbo clítico	dar-lhe, fazê-lo, dizer-lhe, servi-lo, dar-me
VB+CL+CL	Verbo contração clítico	dar-lhas, dizer-lho, querer-lh'a, insinuar-lho, fazer-no-lo
VB+P	Verbo + P	hás-de
VB+SE	Verbo + SE	fazer-se, servir-se, ver-se, lembrar-se, conservar-se
VB+SE+CL	Verbo + SE clítico	cortar-se-lhe, conferir-se-lhe, oferecem-se-lhe, dar-se-lhe, comunicar-se-nos
VB-AN	Verbo particípio passivo masculino singular	posto, obrigado, passado, visto, dito
VB-AN-F	Verbo particípio passivo feminino singular	feita, dita, querida, passada, conhecida
VB-AN-F-P	Verbo particípio passivo feminino plural	feitas, escritas, ditas, passadas, pintadas
VB-AN-P	Verbo particípio passivo masculino plural	passados, mortos, obrigados, ditos, feitos
VB-D	Verbo passado	disse, fez, podia, mandou, deu
VB-D+CL	Verbo passado clítico	disse-lhe, disse-me, deu-lhe, pareceu-me, mandou-lhe
VB-D+CL+CL	Verbo passado contração clítico	acendeu-lho, tornou-mo, quebrou-lhos
VB-D+SE	Verbo passado + SE	foi-se, seguiu-se, viu-se, fez-se, pôs-se
VB-D+SE+CL	Verbo passado + SE clítico	ofereceu-se-lhe, foi-se-lhe, pegou-se-me, quebrou-se-lhe, meteu-se-lhe
VB-F	Verbo infinitivo Infletido	fazerem, darem, poderem, irem, virem
VB-F+CL	Verbo infinitivo Infletido clítico	darem-lhe, darem-me, estimarem-nos, preguntarem-me, recusarem-lhe
VB-F+CL+CL	Verbo infinitivo Infletido clítico contração clítico	dardes-mas, darmos-lhas
VB-F+SE	Verbo infinitivo Infletido + SE	irem-se, darem-se, acharem-se, deixarem-se, verem-se
VB-G	Verbo gerúndio	fazendo, vendo, dizendo, dando, deixando
VB-G+CL	Verbo gerúndio clítico	dando-lhe, dizendo-lhe, pedindo-lhe, fazendo-lhe, mandando-lhe
VB-G+CL+CL	Verbo gerúndio contração clítico	permittindo-lho, dando-lha, quadruplicandolho, mandando-ma
VB-G+SE	Verbo gerúndio + SE	vendo-se, fazendo-se, achando-se, metendo-se, dando-se
VB-G+SE+CL	Verbo gerúndio + SE clítico	dando-se-lhe, oferecendo-se-lhe, pagando-se-lhe, oferecendose-lhe, preguntando-se-lhe
VB-I	Verbo Imperativo	vêde, fazei, sabei, dá, vinde
VB-I+CL	Verbo Imperativo clítico	perdoai-me, dai-me, dizei-me, lembrai-vos, dizei-lhe
VB-I+CL+CL	Verbo Imperativo contração clítico	guardai-ma, mostrai-lho
VB-P	Verbo presente	pode, faz, parece, deve, diz
VB-P+CL	Verbo presente clítico	parece-me, parece-nos, julgam-vos, lembra-me, digo-vos
VB-P+CL+CL	Verbo presente contração clítico	pedem-no-la, digo-vo-lo
VB-P+P	Verbo presente + P	hás-de, dá-de
VB-P+SE	Verbo presente + SE	deve-se, pode-se, segue-se, acha-se, devem-se
VB-P+SE+CL	Verbo presente + SE clítico	dá-se-lhe, deita-se-lhe, suspendeu-se-lhe, segue-se-lhe, regula-se-lhe
VB-PP	Verbo particípio perfeito	feito, dito, dado, visto, passado
VB-R	Verbo futuro	poderá, fará, poderia, direi, verá
VB-R!CL	Verbo futuro mesóclise	chamar-me-eis, obrigá-los-ia, dar-lhe-ia, dir-me-ão, dir-me-á
VB-R!SE	Verbo futuro + SE	salvar-se-ha, farse-ha, continuar-se-ia, enfadar-se-ía, livrarse-ha
VB-R+CL	Verbo futuro clítico	mandá-la, passá-los
VB-RA	Verbo morfema inflexional -ra	pudera, quisera, fizera, tomara, dera
VB-RA+CL	Verbo morfema inflexional -ra clítico	apresentara-me, quisera-me, atrevera-me, tomarame, estimara-o
VB-RA+SE	Verbo morfema inflexional -ra + SE	pudera-se, implicara-se, familiarizara-se, podera-se, perdera-se
VB-SD	Verbo subjuntivo do passado	fizesse, pudesse, desse, quisesse, fossem
VB-SD+CL	Verbo subjuntivo do passado clítico	esqueceu-me, condemnassem-no
VB-SP	Verbo presente do subjuntivo	guarde, faça, possa, queira, dê
VB-SP+CL	Verbo presente do subjuntivo clítico	dê-me, mande-me, encomende-me, diga-me, diga-lhe
VB-SP+CL+CL	Verbo presente do subjuntivo contração clítico	lance-ma, mande-mas, dê-vo-la, perdoe-mo
VB-SP+SE	Verbo presente do subjuntivo + SE	lembre-se, sirva-se, faça-se, contente-se, veja-se
VB-SP+SE+CL	Verbo presente do subjuntivo + SE clítico	satisfaçam-se-me, deva-se-lhes, contem-selhe, ponha-se-me
VB-SR	Verbo subjuntivo futuro	quiser, puder, der, fizer, disser
VB-SR+CL	Verbo subjuntivo futuro clítico	fazerem-me, serem-lhe, saberdes-vos
VB-SR+SE	Verbo subjuntivo futuro + SE	pintarem-se, passarem-se, verem-se
W	nido	nido
WADV	Elemento interrogativo	onde, quanto, como, porque, quão
WD	Determinante interrogativo masculino	que, qual, quanto, se, quê
WD-F	Determinante interrogativo feminino	quanta, quais, cuja
WD-F-P	Determinante interrogativo feminino plural	quantas
WD-P	Determinante interrogativo masculino plural	quais, quantos, quantas, que, quaes
WPRO	Elemento relativo	que, quem, qual, quanto, quê
WPRO$	Elemento relativo CUJO masculino	cujo
WPRO$-F	Elemento relativo CUJO feminino	cuja
WPRO$-F-P	Elemento relativo CUJO feminino plural	cujas, cuja
WPRO$-P	Elemento relativo CUJO masculino plural	cujos, quais
WPRO-F	Elemento relativo eminino	quanta
WPRO-F-P	Elemento relativo feminino plural	quantas, quais
WPRO-P	Elemento relativo plural	quais, quantos, quaes, quantas
WQ	Elemento exclamativo/interrogativo	se, como
XX	Palavras desconhecidas	x, tamalavez, a, al, dist

B
Avaliação modelo BLSTM-WE-CE no corpus Tycho Brahe.

Na Tabela B.1 apresentamos os valores de F_1, *precision* e *recall* médio para cada etiqueta do corpus Tycho Brahe. Os valores médio foram calculados a partir de 10 experimentos realizados com inicialização de sementes aleatórias.

Tabela B.1: Valor médio de F_1, *Precision* e *Recall* por etiqueta no conjunto de treino do corpus Tycho Brahe

Classe	$F_1 \pm \sigma$	Precision$\pm\sigma$	Recall$\pm\sigma$
(	1.000±0.000	1.000±0.000	1.000±0.000
,	1.000±0.000	1.000±0.000	1.000±0.000
.	1.000±0.000	1.000±0.000	1.000±0.000
ADJ	0.912±0.004	0.922±0.010	0.905±0.016
ADJ-F	0.937±0.007	0.939±0.015	0.937±0.008
ADJ-F-P	0.940±0.008	0.935±0.017	0.944±0.012
ADJ-G	0.942±0.004	0.938±0.008	0.945±0.008
ADJ-G-P	0.921±0.006	0.924±0.012	0.921±0.017
ADJ-P	0.868±0.004	0.891±0.021	0.843±0.020
ADJ-R	0.628±0.148	0.856±0.043	0.520±0.170
ADJ-R-F	0.982±0.010	0.982±0.024	0.982±0.006
ADJ-R-F-P	0.969±0.003	0.998±0.006	0.940±0.000
ADJ-R-G	0.934±0.016	0.913±0.013	0.959±0.039
ADJ-R-G-P	0.951±0.003	0.947±0.008	0.961±0.014
ADJ-R-P	1.000±0.000	1.000±0.000	1.000±0.000
ADJ-S	0.910±0.038	0.902±0.074	0.925±0.014
ADJ-S-F	0.875±0.007	0.876±0.031	0.873±0.032
ADJ-S-F-P	0.754±0.061	0.665±0.109	0.886±0.044
ADJ-S-P	0.882±0.018	0.878±0.025	0.885±0.041
ADV	0.979±0.003	0.980±0.005	0.978±0.004
ADV+CL	0.000±0.000	0.000±0.000	0.000±0.000
ADV-NEG	0.902±0.053	0.877±0.015	0.935±0.111
ADV-R	0.989±0.003	0.988±0.004	0.989±0.003
ADV-S	0.000±0.000	0.000±0.000	0.000±0.000
C	0.880±0.000	0.868±0.015	0.890±0.016
CL	0.981±0.003	0.983±0.005	0.986±0.005
CL+CL	0.958±0.004	0.967±0.007	0.958±0.004
CONJ	0.981±0.003	0.990±0.000	0.980±0.000
CONJ-NEG	0.936±0.005	0.885±0.005	0.992±0.010
CONJS	0.912±0.006	0.942±0.012	0.885±0.017
D	0.990±0.000	0.994±0.005	0.990±0.000
D+OUTRO	0.000±0.000	0.000±0.000	0.000±0.000
D-F	0.970±0.000	0.969±0.006	0.978±0.004
D-F-P	0.989±0.003	0.977±0.005	0.995±0.005
D-G	0.048±0.081	0.084±0.164	0.087±0.216
D-G-P	0.000±0.000	0.000±0.000	0.000±0.000
D-P	0.998±0.004	0.990±0.000	0.999±0.003
D-UM	1.000±0.000	0.990±0.000	1.000±0.000
D-UM-F	1.000±0.000	1.000±0.000	1.000±0.000
D-UM-F-P	0.989±0.003	0.980±0.000	0.998±0.006
D-UM-P	1.000±0.000	1.000±0.000	1.000±0.000
DEM	0.998±0.004	0.998±0.004	0.991±0.003
ET	0.975±0.005	0.986±0.014	0.968±0.012
ET-D	0.974±0.005	0.961±0.006	0.989±0.011
ET-F	0.900±0.000	1.000±0.000	0.820±0.000
ET-G	0.990±0.011	0.985±0.016	1.000±0.000
ET-I	0.201±0.324	0.300±0.483	0.150±0.242
ET-P	0.990±0.000	0.989±0.003	0.990±0.000
ET-P+CL	0.050±0.158	0.033±0.104	0.100±0.316
ET-PP	0.944±0.068	1.000±0.000	0.899±0.117
ET-R	1.000±0.000	1.000±0.000	1.000±0.000
ET-RA	0.942±0.085	0.899±0.143	1.000±0.000
ET-SD	0.230±0.221	0.199±0.180	0.284±0.296
ET-SP	0.928±0.019	0.955±0.051	0.903±0.045
ET-SR	1.000±0.000	1.000±0.000	1.000±0.000
FP	0.925±0.005	0.935±0.008	0.912±0.015
FW	0.952±0.006	0.963±0.014	0.943±0.009
HV	1.000±0.000	1.000±0.000	1.000±0.000
HV+CL	0.800±0.270	0.736±0.348	1.000±0.000
HV+SE	0.634±0.457	0.650±0.474	0.650±0.474
HV-AN	0.000±0.000	0.000±0.000	0.000±0.000
HV-AN-F	0.000±0.000	0.000±0.000	0.000±0.000

Continuação da Tabela B.1

Classe	$F_1 \pm \sigma$	Precision$\pm\sigma$	Recall$\pm\sigma$
HV-AN-P	0.400±0.516	0.400±0.516	0.400±0.516
HV-D	0.990±0.000	0.989±0.003	1.000±0.000
HV-F	0.995±0.016	1.000±0.000	0.990±0.032
HV-F+SE	0.100±0.316	0.100±0.316	0.100±0.316
HV-G	0.984±0.008	0.976±0.013	1.000±0.000
HV-G+CL	0.967±0.104	0.950±0.158	1.000±0.000
HV-G+CL+CL	0.000±0.000	0.000±0.000	0.000±0.000
HV-G+SE	1.000±0.000	1.000±0.000	1.000±0.000
HV-I	0.100±0.316	0.100±0.316	0.100±0.316
HV-P	0.990±0.000	0.980±0.000	0.990±0.000
HV-P+CL	0.000±0.000	0.000±0.000	0.000±0.000
HV-P+P	0.999±0.003	0.999±0.003	1.000±0.000
HV-PP	0.800±0.000	0.800±0.000	0.800±0.000
HV-R	0.707±0.130	0.653±0.051	0.815±0.252
HV-RA	0.934±0.024	1.000±0.000	0.882±0.043
HV-SD	0.989±0.014	0.991±0.020	0.987±0.024
HV-SP	0.999±0.003	0.997±0.009	1.000±0.000
HV-SR	0.994±0.019	1.000±0.000	0.989±0.035
INTJ	0.928±0.004	0.918±0.004	0.933±0.009
N	0.980±0.000	0.979±0.003	0.980±0.000
N-P	0.971±0.003	0.971±0.006	0.978±0.004
NEG	1.000±0.000	1.000±0.000	1.000±0.000
NPR	0.980±0.000	0.979±0.003	0.980±0.000
NPR-P	0.941±0.006	0.933±0.017	0.952±0.008
NUM	0.990±0.000	0.989±0.003	0.990±0.000
NUM-F	0.980±0.007	0.990±0.005	0.971±0.014
OUTRO	0.969±0.003	0.949±0.003	0.980±0.000
OUTRO-F	0.980±0.000	0.950±0.000	1.000±0.000
OUTRO-F-P	0.970±0.000	0.950±0.000	0.990±0.000
OUTRO-P	0.970±0.000	0.950±0.000	1.000±0.000
P	0.990±0.000	0.993±0.005	0.990±0.000
P+ADV	0.944±0.007	0.942±0.017	0.951±0.006
P+CL	0.675±0.109	0.858±0.081	0.579±0.156
P+D	1.000±0.000	1.000±0.000	1.000±0.000
P+D-F	1.000±0.000	0.991±0.003	1.000±0.000
P+D-F-P	0.991±0.003	0.991±0.003	0.999±0.003
P+D-P	0.992±0.004	0.998±0.004	0.992±0.004
P+D-UM	0.883±0.013	0.809±0.047	0.974±0.039
P+D-UM-F	0.948±0.006	0.899±0.003	0.997±0.009
P+D-UM-P	0.434±0.473	0.500±0.527	0.400±0.459
P+DEM	0.988±0.004	0.990±0.000	0.988±0.004
P+NPR	0.876±0.025	0.925±0.061	0.839±0.041
P+OUTRO	0.756±0.117	0.749±0.047	0.775±0.184
P+OUTRO-F	0.993±0.022	0.988±0.038	1.000±0.000
P+OUTRO-F-P	0.920±0.139	1.000±0.000	0.875±0.212
P+OUTRO-P	0.606±0.076	1.000±0.000	0.440±0.084
P+PRO	0.999±0.003	0.999±0.003	0.998±0.004
P+Q	0.000±0.000	0.000±0.000	0.000±0.000
P+Q-F	0.000±0.000	0.000±0.000	0.000±0.000
P+Q-F-P	0.000±0.000	0.000±0.000	0.000±0.000
P+WADV	0.990±0.000	0.980±0.000	1.000±0.000
P+WPRO	0.515±0.117	0.785±0.178	0.416±0.157
PRO	1.000±0.000	1.000±0.000	1.000±0.000
PRO$	1.000±0.000	1.000±0.000	1.000±0.000
PRO$-F	1.000±0.000	1.000±0.000	1.000±0.000
PRO$-F-P	1.000±0.000	1.000±0.000	1.000±0.000
PRO$-P	1.000±0.000	1.000±0.000	1.000±0.000
Q	0.990±0.000	0.992±0.004	0.988±0.004
Q-F	0.990±0.000	0.990±0.000	0.990±0.000
Q-F-P	1.000±0.000	1.000±0.000	1.000±0.000
Q-G	1.000±0.000	1.000±0.000	1.000±0.000
Q-G-P	1.000±0.000	1.000±0.000	1.000±0.000
Q-NEG	0.958±0.004	0.950±0.016	0.971±0.011
Q-NEG-F	0.969±0.003	0.951±0.003	0.996±0.013
Q-NEG-P	0.300±0.107	0.852±0.132	0.192±0.083
Q-P	0.990±0.000	0.989±0.003	1.000±0.000
QT	1.000±0.000	1.000±0.000	1.000±0.000
SE	0.969±0.003	0.950±0.011	0.986±0.005
SENAO	1.000±0.000	1.000±0.000	0.990±0.000
SR	0.990±0.000	0.980±0.000	0.990±0.000
SR+CL	0.960±0.084	0.934±0.139	1.000±0.000
SR-D	0.962±0.004	0.948±0.012	0.978±0.012
SR-D+CL	0.500±0.527	0.500±0.527	0.500±0.527
SR-F	0.967±0.012	0.972±0.025	0.962±0.006
SR-G	1.000±0.000	1.000±0.000	1.000±0.000
SR-G+CL	0.550±0.497	0.533±0.502	0.600±0.516
SR-I	0.000±0.000	0.000±0.000	0.000±0.000
SR-I+CL	0.000±0.000	0.000±0.000	0.000±0.000
SR-P	1.000±0.000	0.991±0.003	1.000±0.000
SR-P+CL	0.867±0.322	0.850±0.337	0.900±0.316
SR-PP	1.000±0.000	1.000±0.000	1.000±0.000
SR-R	0.967±0.005	0.959±0.016	0.975±0.013
SR-R!CL	0.000±0.000	0.000±0.000	0.000±0.000
SR-RA	0.867±0.011	0.856±0.021	0.879±0.021
SR-SD	0.906±0.012	0.888±0.029	0.925±0.042
SR-SP	0.997±0.005	0.993±0.007	1.000±0.000
SR-SP+CL	0.000±0.000	0.000±0.000	0.000±0.000
SR-SR	0.924±0.007	0.923±0.016	0.928±0.017
TR	0.987±0.005	0.984±0.010	0.990±0.000
TR+CL	0.377±0.416	0.467±0.502	0.334±0.386
TR+SE	0.000±0.000	0.000±0.000	0.000±0.000
TR-AN	0.000±0.000	0.000±0.000	0.000±0.000
TR-D	0.971±0.003	0.964±0.005	0.983±0.009
TR-D+CL	0.714±0.274	0.620±0.345	1.000±0.000
TR-D+SE	0.000±0.000	0.000±0.000	0.000±0.000
TR-F	0.891±0.019	0.963±0.051	0.832±0.025
TR-F+CL	0.000±0.000	0.000±0.000	0.000±0.000
TR-G	1.000±0.000	1.000±0.000	1.000±0.000
TR-G+CL	0.967±0.104	0.950±0.158	1.000±0.000

Continuação da Tabela B.1

Classe	$F_1 \pm \sigma$	Precision$\pm\sigma$	Recall$\pm\sigma$
TR-G+SE	0.900±0.316	0.900±0.316	0.900±0.316
TR-I	1.000±0.000	1.000±0.000	1.000±0.000
TR-I+CL	0.000±0.000	0.000±0.000	0.000±0.000
TR-P	0.990±0.000	0.980±0.000	0.997±0.005
TR-P+CL	0.506±0.160	0.547±0.081	0.515±0.253
TR-P+SE	0.300±0.483	0.300±0.483	0.300±0.483
TR-PP	0.700±0.000	0.540±0.000	1.000±0.000
TR-R	0.990±0.000	0.980±0.000	1.000±0.000
TR-R!CL	0.000±0.000	0.000±0.000	0.000±0.000
TR-RA	0.926±0.013	0.942±0.004	0.902±0.025
TR-SD	0.980±0.000	1.000±0.000	0.970±0.000
TR-SP	0.976±0.005	0.970±0.011	0.981±0.003
TR-SP+CL	0.602±0.088	0.800±0.258	0.500±0.000
TR-SR	0.871±0.023	0.894±0.052	0.850±0.046
VB	0.981±0.003	0.982±0.004	0.987±0.005
VB+CL	0.914±0.022	0.873±0.048	0.958±0.010
VB+CL+CL	0.020±0.063	0.033±0.104	0.014±0.044
VB+SE	0.967±0.012	0.961±0.020	0.974±0.013
VB+SE+CL	0.000±0.000	0.000±0.000	0.000±0.000
VB-AN	0.922±0.004	0.917±0.013	0.932±0.011
VB-AN-F	0.932±0.006	0.934±0.010	0.931±0.007
VB-AN-F-P	0.942±0.010	0.937±0.022	0.946±0.008
VB-AN-P	0.915±0.005	0.928±0.011	0.904±0.016
VB-D	0.946±0.005	0.938±0.014	0.958±0.006
VB-D+CL	0.797±0.016	0.848±0.070	0.759±0.066
VB-D+SE	0.863±0.016	0.861±0.069	0.875±0.054
VB-D+SE+CL	0.017±0.054	0.014±0.044	0.020±0.063
VB-F	0.900±0.014	0.890±0.037	0.917±0.035
VB-F+CL	0.000±0.000	0.000±0.000	0.000±0.000
VB-F+CL+CL	0.000±0.000	0.000±0.000	0.000±0.000
VB-F+SE	0.656±0.101	0.905±0.113	0.530±0.125
VB-G	0.985±0.005	0.985±0.005	0.984±0.005
VB-G+CL	0.811±0.052	0.780±0.075	0.855±0.075
VB-G+SE	0.914±0.029	0.921±0.050	0.909±0.052
VB-G+SE+CL	0.058±0.078	0.048±0.070	0.080±0.103
VB-I	0.701±0.012	0.821±0.069	0.617±0.043
VB-I+CL	0.513±0.040	0.700±0.151	0.428±0.098
VB-P	0.960±0.000	0.964±0.007	0.958±0.004
VB-P+CL	0.744±0.037	0.740±0.114	0.763±0.052
VB-P+CL+CL	0.000±0.000	0.000±0.000	0.000±0.000
VB-P+P	0.900±0.316	0.900±0.316	0.900±0.316
VB-P+SE	0.897±0.014	0.895±0.035	0.900±0.016
VB-P+SE+CL	0.252±0.077	0.579±0.310	0.185±0.092
VB-PP	0.924±0.005	0.936±0.017	0.913±0.016
VB-R	0.909±0.011	0.910±0.036	0.913±0.026
VB-R!CL	0.249±0.110	0.375±0.207	0.225±0.115
VB-R!SE	0.141±0.177	0.304±0.375	0.100±0.133
VB-RA	0.816±0.017	0.897±0.047	0.751±0.022
VB-RA+CL	0.000±0.000	0.000±0.000	0.000±0.000
VB-RA+SE	0.000±0.000	0.000±0.000	0.000±0.000
VB-SD	0.962±0.006	0.959±0.017	0.962±0.010
VB-SD+CL	0.000±0.000	0.000±0.000	0.000±0.000
VB-SP	0.918±0.010	0.938±0.033	0.901±0.017
VB-SP+CL	0.683±0.048	0.684±0.131	0.705±0.064
VB-SP+SE	0.794±0.069	0.819±0.165	0.800±0.077
VB-SP+SE+CL	0.000±0.000	0.000±0.000	0.000±0.000
VB-SR	0.755±0.021	0.832±0.049	0.695±0.058
WADV	0.870±0.007	0.916±0.020	0.830±0.023
WD	0.804±0.013	0.819±0.037	0.793±0.034
WD-F	0.791±0.083	0.854±0.175	0.760±0.084
WD-F-P	0.621±0.248	0.672±0.094	0.636±0.334
WD-P	0.593±0.033	0.666±0.151	0.557±0.049
WPRO	0.920±0.000	0.917±0.007	0.929±0.012
WPRO$	1.000±0.000	1.000±0.000	1.000±0.000
WPRO$-F	1.000±0.000	1.000±0.000	1.000±0.000
WPRO$-F-P	1.000±0.000	1.000±0.000	1.000±0.000
WPRO$-P	1.000±0.000	1.000±0.000	1.000±0.000
WPRO-F	0.431±0.249	0.299±0.185	0.800±0.422
WPRO-F-P	0.576±0.153	0.496±0.099	0.701±0.247
WPRO-P	0.883±0.018	0.851±0.011	0.920±0.042
WQ	0.641±0.030	0.748±0.073	0.566±0.049
XX	0.000±0.000	0.000±0.000	0.000±0.000