Aprendizado de Representações para Adaptação de Domínio de Etiquetagem Morfossintática

Irving Muller Rodrigues

SERVIÇO DE PÓS-GRADUAÇÃO DA UFMS

Data de Depósito:

Assinatura:_____

Irving Muller Rodrigues

Orientador: *Prof. Eraldo Luís Rezende Fernandes*

Dissertação de Mestrado apresentada à Faculdade de Computação da Universidade Federal de Mato Grosso do Sul como parte dos requisitos necessários à obtenção do título de Mestre em Ciência da Computação.

UFMS - Campo Grande
Fevereiro/2017

Aos meus pais,
Mario e Gerda,

A minha esposa,
Karoline

Agradecimentos

Agradeço ao meu orientador pela ajuda, paciência, esforço e confiança. Muito grato por tudo. Também agradeço ao professor Ulf Brefeld que me disponibilizou acesso ao cluster localizado na Alemanha. Sem esta ajuda muito do que foi feito no meu mestrado não teria ocorrido.

Resumo

No contexto de aprendizado de máquina, o problema de adaptação de domínio ocorre quando os dados de treinamento advêm de um domínio diferente daquele onde o modelo será aplicado. Neste cenário, a representação dos dados de entrada é um fator crucial para a adaptação do modelo de um domínio para outro. Neste trabalho, três métodos são desenvolvidos para a adaptação de domínio na tarefa de etiquetagem morfossintática. Estes métodos, denominados DLID, DAN e DATT, usam técnicas de aprendizado profundo de representações (*deep learning*) através de uma rede neural denominada CharWNN. O CharWNN detém o estado da arte nos principais corpora da tarefa de etiquetagem morfossintática e sua principal característica é não utilizar atributos manuais. Isto significa que a entrada desta rede é composta exclusivamente pela sequência de palavras de uma frase e a representação desta entrada é aprendida automaticamente. Os métodos desenvolvidos exploram o aprendizado de representações de maneiras distintas, envolvendo treinamento não supervisionado, supervisionado e semissupervisionado. Para avaliar os métodos desenvolvidos, são empregadas duas tarefas de adaptação de domínio – uma em inglês e outra em português – que compreendem diversos pares de domínios origem-alvo. Nos experimentos, os métodos de adaptação de domínio superam substancialmente um baseline que tem acesso exclusivo a dados do domínio origem. Entretanto, estes métodos obtêm desempenho equivalente ao CharWNN quando este usa trivialmente dados externos não anotados. Apesar destes resultados demonstrarem que os métodos desenvolvidos não trazem benefícios, eles também demonstram que a rede CharWNN tem desempenho excelente na adaptação de domínio. Adicionalmente, demonstramos que o problema de adaptação de domínio não supervisionada é difícil e ainda mal resolvido, através de um experimento que fornece, incrementalmente, algumas frases anotadas do domínio alvo para o treinamento da rede CharWNN. Neste experimento, o CharWNN é capaz de superar os melhores sistemas de adaptação do domínio da literatura usando poucas frases anotadas.

Sumário

Lista de Figuras

Lista de Tabelas

CAPÍTULO

1

Introdução

Na internet, é produzido diariamente um exabyte[1] de dados (Lorentz, 2013), grande parte em formato textual. Comentários de redes sociais, notícias, blogs e fóruns são alguns exemplos deste tipo de dados. Estes textos contêm informações relevantes a empresas, pessoas e até indivíduos, que podem ser extraídas com uso de técnicas de *processamento de linguagem natural* (PLN).

O PLN visa desenvolver algoritmos e sistemas computacionais capazes de se comunicar por meio de linguagens naturais (humanas) (Allen, 2013). Isto envolve, por exemplo, capacidades como analisar, compreender e produzir textos. Uma tarefa importante do PLN é a etiquetagem morfossintática (*part-of-speech tagging*). Esta tarefa consiste em etiquetar cada palavra de uma frase de acordo com a classe gramatical exercida pela palavra na frase (Oliveira and de Freitas, 2006). Esta informação é utilizada como entrada na resolução de vários problemas (Fonseca et al., 2015) como análise sintática de dependência, reconhecimento e classificação de entidades nomeadas, análise de sentimentos, resolução de correferência, dentre outros. A etiquetagem morfossintática da frase O gato é bonito é exemplificada na Tabela 1.1.

Palavra	O	gato	é	bonito
Etiqueta	ART	SUBS	VERB	ADJ

Tabela 1.1: Exemplo de etiquetagem morfossintática da frase O gato é bonito. As etiquetas VERB, SUBS, ART e ADJ, representam, respectivamente, verbo, substantivo, artigo e adjetivo.

A maioria dos algoritmos que constituem o estado da arte em PLN são baseados em *aprendizado de máquina* (AM) (Santos and Zadrozny, 2014; Chiu and

[1] Um exabyte é igual a um milhão de terabytes.

Nichols, 2016; Cheng et al., 2016). Boa parte destes algoritmos são treinados de forma supervisionada e, desta forma, pressupõe-se que o modelo aprendido será usado em dados não vistos, mas que seguem a mesma distribuição de probabilidade dos dados de treinamento. Se esta suposição não for observada na prática, então o desempenho alcançado nos dados de treinamento (em um conjunto de validação, por exemplo), geralmente, não é alcançado nos novos dados. Portanto, se um modelo para etiquetagem morfossintática é treinado em textos jornalísticos e aplicado em textos de redes sociais, muito provavelmente, seu desempenho será substancialmente degradado. Este problema ocorre porque os dois domínios seguem distribuições muito distintas. Por exemplo, o uso de gírias (característica comum em redes sociais) é quase inexistente em textos jornalísticos. Textos jornalísticos e textos de redes sociais são considerados dois domínios diferentes, pois o sentido das palavras, o vocabulário e a forma da construção das frases variam muito entre estas fontes de textos.

O problema de *adaptação de domínio* (AD) consiste em utilizar exemplos anotados de um domínio para treinar um modelo que será aplicado em outro domínio (Jiang, 2008). Na adaptação de domínio, considera-se um domínio que possui uma quantidade suficiente de exemplos anotados, denominado *domínio origem*. O domínio no qual o modelo será aplicado, denominado *domínio alvo*, não possui ou possui uma quantidade pequena de exemplos anotados. Existem diferentes paradigmas de adaptação de domínio, dependendo do uso de dados anotados e não anotados do domínio alvo durante o treinamento (Daumé et al., 2010). Na adaptação de domínio *supervisionada*, utiliza-se somente dados anotados do domínio alvo. Já na adaptação de domínio *não supervisionada*, somente dados não anotados são empregados. Um método de adaptação de domínio é denominado *semissupervisionado* quando usa concomitantemente dados anotados e não anotados do domínio alvo. Lembrando que dados anotados do domínio origem estão sempre disponíveis e são utilizados por todos estes paradigmas. O foco deste trabalho é na adaptação de domínio não supervisionada.

Diversos métodos de adaptação de domínio foram propostos na literatura. Alguns destes métodos selecionam um conjunto específico de atributos sobre o qual a distribuição de probabilidade dos domínios origem e alvo se tornem similares. Usando estes atributos *independentes* para representar a entrada, um algoritmo de AM supervisionado se adapta melhor de um domínio para o outro. Por exemplo, na tarefa de etiquetagem morfossintática, um atributo importante é a própria palavra a ser etiquetada. Considerando um problema de adaptação de domínio, onde o domínio origem é composto por textos jornalísticos e o domínio alvo por comentários em redes sociais, um modelo treinado

no domínio origem, provavelmente, atribuiria uma probabilidade alta da palavra curta ser etiquetada como adjetivo (no sentido de pouco comprimento). Entretanto, esta palavra é frequentemente usada como verbo no domínio alvo (imperativo do verbo curtir, como em curta nossa página no Facebook). Outro atributo bastante comum em sistemas de PLN é o conjunto de sufixos da palavra. Um modelo treinado no domínio origem dará uma probabilidade alta para o sufixo mente ser associado à etiqueta de advérbio (cuidadosamente, rapidamente, etc.). Esta relação também é observada no domínio alvo. Portanto, alguns atributos são mais independentes do que outros, quando consideramos a mesma tarefa em domínios diferentes. Isto ilustra a relevância da *representação de dados* na capacidade de um modelo se adaptar a um domínio diferente.

Neste trabalho, técnicas de aprendizado de representações são exploradas para o desenvolvimento de três métodos de adaptação de domínio. Nestes métodos, a representação dos dados não é definida pelo tradicional processo *manual* de engenharia de atributos, que é custoso e lento (Bengio et al., 2013). A representação da entrada é aprendida de forma automática por meio do emprego de métodos de *aprendizado profundo* (*deep learning*). No aprendizado profundo de representações, vários níveis de representação são aprendidos (LeCun et al., 2015). A representação de um nível é aprendida sobre a do nível anterior, o que propicia a geração de representações mais abstratas, que, geralmente, são mais invariantes às mudanças locais dos dados e amplificam aspectos importantes para resolução da tarefa (Bengio et al., 2013).

Algoritmos baseados em aprendizado profundo vêm alcançando resultados surpreendentes em várias tarefas de PLN (Mou et al., 2016). Um destes algoritmos é o CharWNN, que detém o estado da arte nos principais corpora de etiquetagem morfossintática, quando um único domínio é considerado. O CharWNN é uma rede neural proposta por dos Santos e Bianca Zadrozny (2014b) que não utiliza nenhum atributo manual, consistindo em uma das suas vantagens sobre os demais algoritmos. Esta rede consegue aprender automaticamente representações efetivas para a etiquetagem morfossintática utilizando somente os dados de treinamento. Aproveitando-se desta capacidade, desenvolvemos três métodos (DLID, DAN e DATT) a partir da combinação do CharWNN com três métodos de AD propostos na literatura. Estes métodos foram propostos para outras tarefas (classificação de imagens, por exemplo) e/ou usando arquiteturas diferentes (que não envolvem representações distribuídas de palavras ou caracteres, por exemplo). Por motivos de simplicidade, os métodos desenvolvidos neste trabalho possuem as mesmas denominações que os métodos originais.

O método DLID (*Deep Learning for Domain Adaptation by Interpolating be-*

tween Domains) baseia-se em um método proposto por Chopra et al. (2013) para o problema de adaptação de domínio na tarefa de reconhecimento de objetos em imagens. O DLID aprende representações específicas para os domínios origem e alvo, e ainda para um domínio artificial denominado *intermediário*. O domínio intermediário representa uma interpolação entres os domínios origem e alvo. Este domínio artificial é representado por uma mistura de exemplos dos outros dois domínios. No modelo final, a entrada é representada pela concatenação das representações dos três domínios. Espera-se que esta representação capture informações sobre a diferença entre os domínios origem e alvo.

O método DAN (*Domain-Adversarial Training of Neural Networks*) é baseado no método de Ganin et al. (2016), que foi aplicado às tarefas de análise de sentimentos e classificação de imagens. O DAN aprende uma representação que minimiza sua variância entre os domínios origem e alvo. Para isto, cria-se uma rede neural composta por um rotulador e um classificador de domínios que compartilham os primeiros níveis de representação. Toda a rede é treinada em um processo de aprendizado conjunto tal que as representações compartilhadas sejam discriminativas na tarefa em mãos e, ao mesmo tempo, invariantes ao domínio.

O método DATT (*Domain Adaptation With Target and Task Features*) é baseado em Ma et al. (2014) e aborda o problema de adaptação de domínio utilizando representações específicas para o domínio alvo e para a tarefa de etiquetagem morfossintática. Diferentemente do DAN, o aprendizado destas representações é composto por duas etapas. Primeiramente, ocorre o treinamento não supervisionado de um extrator de atributos do domínio alvo utilizando somente os dados não anotados do domínio alvo. Depois disto, acopla-se este extrator ao CharWNN, formando assim uma rede neural, que é treinada usando os dados anotados do domínio origem. Durante este treinamento, aprende-se os atributos da tarefa de etiquetagem morfossintática e os parâmetros do extrator de atributos do domínio alvo não são atualizados, com intuito de evitar a perda do que foi aprendido anteriormente.

Os métodos desenvolvidos foram avaliados em duas tarefas: SANCL e Tycho Brahe. A tarefa SANCL foi proposta na conferência SANCL-2012 (Petrov and McDonald, 2012) e compreende cinco domínios alvo da internet (Answers, Emails, Newsgroups, Reviews e Weblogss) e um domínio origem com textos jornalísticos. Todos os domínios contêm textos em língua inglesa. O objetivo da tarefa compartilhada da SANCL 2012 é treinar um único modelo, usando somente os dados anotados do domínio origem e dados não anotados dos domínios alvo, que tenha um bom desempenho nos domínio Answers, Newsgroups e Reviews.

A tarefa Tycho Brahe utiliza o corpus Histórico do Português Tycho Brahe (Galves and Faria, 2010), que é formado por obras literárias em português de autores nascidos entre 1380 e 1881. Da mesma forma que Yang and Eisenstein (2015), os domínios foram criados pela divisão deste corpus em vários domínios definidos por diferentes períodos. Dois períodos sobrepostos e mais recentes (1800-1849 e 1750-1849) são considerados domínios origem, enquanto os demais períodos são os domínios alvo. Consideramos que o treinamento dos modelos nos períodos 1800-1849 e 1750-1849 formam duas tarefas distintas.

A Tabela 1.2 apresenta um resumo dos resultados obtidos pelos nossos métodos e os sistemas estado da arte nas tarefas de adaptação de domínio utilizadas neste trabalho. O CharWNN/RAND representa um CharWNN que teve exclusivamente acesso aos dados do domínio origem. Este modelo é utilizado como *baseline* do DLID, DAN e DATT na tarefa SANCL e do DAN e DATT nas duas tarefas do Tycho Brahe. Nestas duas tarefas, o DLID tem o seu próprio *baseline*. Por fim, o CharWNN/ALVO representa um CharWNN que usa trivialmente dados não anotados dos domínios alvo. Como podemos observar, DAN, DATT e DLID obtêm desempenhos superiores aos de seus *baselines*. Entretanto, quando o CharWNN utiliza dados não anotados dos domínios alvo (CharWNN/ALVO), o seu desempenho aumenta substancialmente, alcançando acurácias semelhantes, e até superiores, àquelas obtidas pelos métodos de AD. Isto demonstra que os ganhos obtidos pelos métodos de AD são devidos ao uso de representações melhores e não à capacidade destes métodos usarem os dados não anotados. Já o desempenho do CharWNN/ALVO é notável, pois este modelo não utiliza nenhum atributo manual e emprega os dados não anotados de uma forma simples e eficiente.

Método	**SANCL**	**Tycho Brahe (1800-1849)**	**Tycho Brahe (1750-1849)**	**AD supervisionada**
Baseline DLID	-	83,07	-	-
CharWNN/RAND	92,00	86,58	91,65	-
CharWNN/ALVO	92,80	88,84	92,62	93,26
DLID	92,62	83,42	-	-
DAN	92,71	89,03	92,51	-
DATT	92,71	88,86	92,03	-
Yang and Eisenstein (2015)	92,03	89,94	93,56	-
Ma et al. (2014)	93,15	-	-	-

Tabela 1.2: Resumo das acurácias dos métodos e sistemas estado da arte nas tarefas de adaptação de domínio

O sistema de Ma et al. (2014) é o estado da arte de AD não supervisionada na tarefa de etiquetagem morfossintática. Na tarefa SANCL, este sistema obtém um erro médio de 6,85 nos domínios alvo. Este valor é quase três vezes maior que o erro obtido por este sistema no domínio origem (2,56), o que

indica que a adaptação de domínio não supervisionada é um problema difícil. Em vista deste fato, realizamos alguns experimentos considerando um cenário de AD supervisionada (última coluna da Tabela 1.2), no qual o modelo CharWNN/ALVO teve acesso a alguns exemplos anotados do *domínio alvo*. Usando somente 50 frases anotadas de cada domínio alvo do SANCL, o CharWNN/ALVO obteve um desempenho similar ao alcançado por Ma et al. (2014). Isto enfatiza a dificuldade do problema de adaptação de domínio não supervisionada.

Em suma, as principais contribuições deste trabalho foram: (i) desenvolvimento dos métodos DLID, DAN e DATT para AD; (ii) aplicação do CharWNN, uma rede neural profunda que não emprega atributos manuais, no problema de adaptação de domínio; (iii) realização de experimentos que reforçam como o problema de adaptação de domínio não supervisionado é difícil; (iv) um artigo aceito na *2017 International Joint Conference on Neural Networks* (IJCNN 2017) (Rodrigues et al., 2017), no qual a aplicação do CharWNN na AD é descrita.

O restante deste documento será dividido da seguinte maneira. No Capítulo 2, explicamos a rede neural apresentada no trabalho dos Santos e Bianca Zadrozny (2014b) e os conceitos utilizados por ela. No Capítulo 3, descrevemos como é realizado o aprendizado de representações em redes neurais. No Capítulo 4, explicamos o DLID, DAN e DATT, No Capítulo 5, apresentamos os trabalhos relacionados. No Capítulo 6, apresentamos os resultados dos experimentos. No Capítulo 7, apresentamos as conclusões e direções promissoras para trabalhos futuros.

CAPÍTULO

2

Redes Neurais para Etiquetagem de Palavras

Recentemente, as redes neurais têm alcançado resultados impressionantes em diversas tarefas, atingindo o estado da arte em vários problemas de PLN (Goldberg, 2016). A rede neural denominada CharWNN (dos Santos e Bianca Zadrozny, 2014b), por exemplo, detém o estado da arte nos principais corpora de etiquetagem morfossintática para diferentes linguagens. Esta rede, ao contrário de outros métodos propostos para esta tarefa, não utiliza nenhum atributo manual na representação dos dados de entrada. Ou seja, a entrada é composta exclusivamente pelo *texto crú* e a representação é aprendida durante o treinamento. Antes de descrever a própria rede neural CharWNN, na próxima seção, apresentamos os conceitos básicos de uma rede neural. Em seguida, descrevemos a arquitetura do CharWNN.

2.1 Redes Neurais

Ler, interpretar, dirigir e pensar são alguns exemplos de tarefas complexas que o cérebro humano consegue realizar. Os cientistas ainda não entendem completamente o funcionamento deste fascinante orgão (Martin T. Hagan, 2014), porém, baseando-se no conhecimento adquirido até hoje, é possível tentar mimetizar a atividade cerebral. Esta atividade é basicamente realizada por atividades eletroquímicas que acontecem em uma rede complexa de células cerebrais, chamadas de neurônios (Russell and Norvig, 2009).

Os eventos em um neurônio acontecem em espaço de milissegundos, enquanto nas portas lógicas de silício estão na faixa de microssegundos (Haykin,

2009). Porém, o cérebro consegue superar esta desvantagem com uma estrutura altamente paralela, formada por aproximadamente 10 bilhões de neurônios e 60 trilhões de sinapses (Shepherd, 1998), que opera em representações que são distribuídas ao longo de muitos neurônios (Mitchell, 1997).

Com objetivo de tentar mimetizar o funcionamento do cérebro, foi criado um modelo matemático de um neurônio, representado na Figura 2.1, que é dividido basicamente nas três partes listadas abaixo.

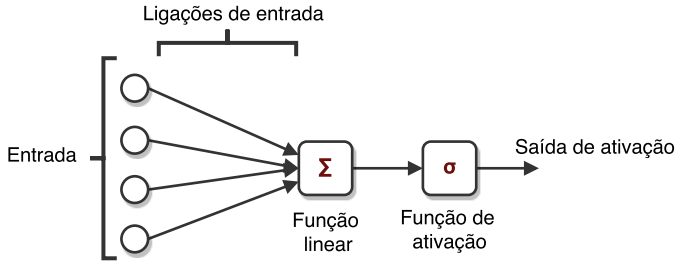


Figura 2.1: Modelo matemático de um neurônio.

- Ligações de entrada: cada ligação recebe um valor de entrada e possui um peso associado (um parâmetro da rede).
- Função linear: realiza o somatório da multiplicação dos pesos pelos valores das ligações de entrada. A saída da função linear é chamada de ativação linear .
- Função de ativação: função *não linear* que recebe como entrada a saída da função linear.
- Saída da ativação: valor de saída da função de ativação.

Estas unidades matemáticas podem ser conectadas umas às outras formando as *redes neurais*. Existem diversos tipos de redes neurais. Nesta seção, trataremos do tipo mais comum que é denominado rede de alimentação direta (Russell, 2009), no qual o fluxo de informação se propaga em um único sentido, ou seja, todos os neurônios recebem como entrada as saídas dos neurônios anteriores. Nestes tipos de redes, não existem ciclos e sua saída depende somente das entradas e dos pesos da rede.

As redes neurais são geralmente divididas em *camadas*, de modo que cada neurônio recebe como entrada as saídas dos neurônios da camada anterior (Russell, 2009). A primeira camada deste tipo de rede é formada por nós fontes, que representam as entradas da rede e recebem o nome de camada de entrada (Haykin, 2009). A última camada da rede é constituída pelos neurônios de saída e chamada de camada de saída. As camadas que ficam entre a camada de entrada e a camada de saída são chamadas de camadas ocultas e são formadas pelos neurônios ocultos. O uso de várias camadas ocultas

possibilita a construção de uma hierarquia complexa de conceitos (Nielsen, 2015). Redes neurais com uma única camada oculta são chamadas de redes rasas, enquanto redes com mais de uma camada oculta são chamadas de redes profundas (Delalleau and Bengio, 2011).

A Figura 2.2 ilustra uma rede com quatro camadas, sendo que os quadrados representam as entradas da rede, os círculos representam os neurônios e as linhas representam as ligações entre os neurônios. Considerando o neurônio em destaque, na camada 3, temos que sua saída de ativação é a_2^3 e o peso da ligação deste neurônio com o primeiro neurônio da camada 2 é o parâmetro W_{21}^3.

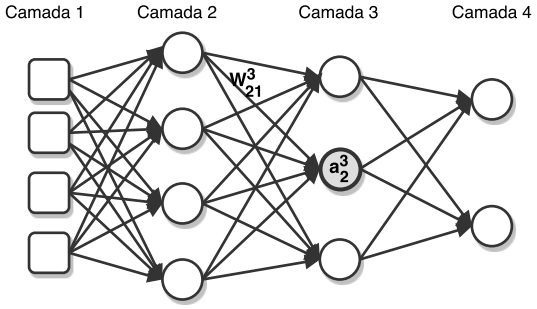


Figura 2.2: Exemplo de uma rede neural com quatro camadas.

A notação utilizada no exemplo acima pode ser generalizada do seguinte modo. As L camadas de uma rede são indexadas pelos inteiros $1, 2, \ldots, L$. Os pesos das ligações entre as camadas $l-1$ e l são representados pela matrix W^l. Desta forma, o peso de uma ligação entre o k-ésimo neurônio da camada $l-1$ e o j-ésimo neurônio da camada l é representado por W_{jk}^l. O j-ésimo neurônio da l-ésima camada tem um valor de *bias* representado por b_j^l e uma *ativação linear* definida como:

$$z_j^l = \sum_k W_{jk}^l a_k^{l-1} + b_j^l,$$

no qual a_k^{l-1} é a *saída de ativação* do k-ésimo neurônio da $(l-1)$-ésima camada. A saída de ativação a_j^l é igual à saída da função de ativação $\sigma^l(\cdot)$ sobre a ativação linear z_j^l, ou seja:

$$a_j^l = \sigma^l(z_j^l).$$

O conjunto de todos os parâmetros da rede é denotado θ e é composto pelos pesos das conexões e os *bias* de todas as camadas da rede. Estes parâmetros são ajustados durante o processo de treinamento da rede.

Frequentemente, deseja-se que a saída da rede (a ativação da camada de saída) seja uma distribuição de probabilidade (Nielsen, 2015). Por exemplo, em um problema de etiquetagem morfossintática com três etiquetas que representam verbos, substantivos e adjetivos, podemos considerar que a saída de ativação do primeiro neurônio da camada de saída representa a probabili-

dade condicional da etiqueta dos verbos dado a entrada da rede, representado por x. A saída de ativação do segundo neurônio pode ser interpretado como o valor da probabilidade condicional das etiquetas dos substantivos, e assim por diante.

Originalmente, as ativações lineares da camada de saída (z^L) não têm intervalo definido. Para normalizar estas saídas e transformá-las em uma distribuição de probabilidade, utiliza-se a função *softmax* como função de ativação na última camada. Dados uma entrada x e os parâmetros θ, a probabilidade condicional de um rótulo y é denotada $P(y|x;\theta)$. Usando a função *softmax* e a ativação linear do y-ésimo neurônio da camada de saída L, esta probabilidade é calculada como:

$$P(y|x;\theta) = a_y^L = \frac{e^{z_y^L}}{\sum_{y' \in \mathcal{Y}} e^{z_{y'}^L}}, \tag{2.1}$$

no qual $\mathcal{Y}$ é o conjunto de todas as etiquetas possíveis (cada etiqueta está associada a um neurônio da camada de saída). Portanto, a função de ativação *softmax* normaliza as ativações lineares da camada de saída de tal forma que a soma das saídas de ativação é igual a um, isto é, $\sum_{y \in \mathcal{Y}} a_y^L = 1$.

Atualmente, as redes neurais vêm sendo empregadas efetivamente em inúmeras tarefas de PLN (Mou et al., 2016). Nestas tarefas, um dos atributos mais importante são as próprias palavras, que devem ser representadas como valores numéricos para serem repassadas como entrada para redes neurais. Dentre as várias formas de realizar isto, uma das mais promissoras é a representação distribuída de palavras (*word embedding*), que consiste em representar cada palavra como um vetor denso de números reais. Esta representação tem vários benefícios em relação aos outros modos e será melhor explanada na próxima seção.

2.2 Representações Distribuídas

A palavra é a unidade central da linguagem. Com o uso delas, é possível expressar sentimentos e informações factuais. Devido a este papel primordial na linguagem, as palavras são cruciais no processamento de linguagem natural pois, a partir delas e de suas relações, podemos extrair informações importantes. Por isto, as palavras são a base dos modelos usados por algoritmos de PLN. Como o modo que representamos os dados interfere drasticamente no desempenho dos algoritmos de AM (Bengio et al., 2013), é de suma importância representarmos as palavras de uma forma que ajude os algoritmos a realizar as suas tarefas. A maioria dos algoritmos de AM espera que os dados sejam representados como vetores numéricos. No caso do PLN, palavras precisam ser representadas como vetores.

Tradicionalmente, palavras são representadas como vetores binários esparsos $x \in \{0,1\}^N$, através da representação denominada *one-hot* (Qiu et al., 2014), no qual N é o tamanho do dicionário que contém todas as palavras consideradas. Nesta representação, a dimensão dos vetores é igual ao número de palavras no dicionário e cada dimensão corresponde a uma palavra. Assim, o vetor que representa a palavra casa terá apenas uma dimensão com valor igual a 1; todas as outras dimensões terão valor igual a 0. Na Tabela 2.1, é mostrado um exemplo da representação *one-hot*. Neste exemplo, há um dicionário com quatro palavras: casa, neve, sempre e ter. Como só existem quatro palavras no dicionário, o vetor que representa as palavras terá quatro dimensões. A palavra casa será representada pela 1º dimensão do vetor. Com isto, esta palavra será representada por um vetor com zeros em todas dimensões, exceto na primeira que terá o valor igual 1. As representações das outras palavras são mostrada na Tabela 2.1.

Palavra	**Vetor**
casa	$[1,0,0,0]$
neve	$[0,1,0,0]$
sempre	$[0,0,1,0]$
ter	$[0,0,0,1]$

Tabela 2.1: Exemplo da representação *one-hot*.

Devido à grande quantidade de possíveis palavras, os vetores da representação *one-hot* terão centenas de milhares de dimensões. Isto leva esta representação a sofrer com a *maldição da dimensionalidade* (*curse of dimensionality*) (Wang et al., 2014). A maldição da dimensionalidade nasce quando um imenso número de diferentes combinações de valores das variáveis de entrada devem ser discriminadas e o algoritmo de aprendizado precisa de pelo menos um exemplo de cada combinação relevante (Bengio, 2008). Assim, é necessário uma enorme quantidade de dados de treinamento para diminuir os efeitos desta maldição e, com isto, conseguir aprender funções altamente complexas.

Além desta maldição, devido à esparsidade dos dados, a representação *one-hot* não possibilita que os modelos consigam estimar de forma satisfatória os parâmetros ligados às palavras raras no treinamento e não capazes de representar as palavras novas que não estão presentes nos dados de treinamento (Turian et al., 2010). Por último, não é possível, com a representação *one-hot*, comparar as similaridades sintáticas ou semânticas das palavras, pois a distância entre os vetores de duas palavras quaisquer são sempre iguais (Qiu et al., 2014).

Recentemente, um tipo de representação diferente da representação *one-hot*, chamado de representação distribuída de palavras (*word embedding* ou

distributed representation), vem se mostrando um recurso inestimável para resolver diversos problemas de PLN (dos Santos e Bianca Zadrozny, 2014a). Esta técnica representa as palavras como vetores densos de número reais com algumas dezenas de dimensões. Por serem mais compactas do que a representação *one-hot*, as representações distribuídas são menos suscetíveis à maldição da dimensionalidade (Bengio et al., 2003) e à esparsidade de dados (Luong et al., 2013). Nas representações distribuídas de palavras, espera-se que os vetores capturem informações semânticas e sintáticas das palavras (Wang et al., 2014), pois cada dimensão de um vetor representa um atributo latente da palavra. A esperança é que, neste espaço de atributos, as palavras com semelhanças sintáticas e semânticas estarão mais próximas umas das outras (Bengio, 2008).

As palavras não são os únicos símbolos ou objetos que podem ser representados de forma distribuída. Outros objetos frequentemente representados desta maneira são: atributos derivados (Yang and Eisenstein, 2015), documentos (Le and Mikolov, 2014) e caracteres (dos Santos e Bianca Zadrozny, 2014b). Segundo Bengio (2008), a representação distribuída de um objeto é um vetor que descreve o seu significado (Bengio, 2008).

Para utilizar a representação distribuída de um objeto, as redes neurais, geralmente, utilizam uma camada especial chamada *lookup table*, que é exemplificada na Figura 2.3. Na figura, uma camada *lookup table*, denominada LT, contém uma matriz $W^{LT} \in \mathbb{R}^{8\times3}$ na qual cada umas de suas linhas representa uma das oito palavras do vocabulário. A camada LT recebe como entrada uma sequência de palavras (a, origem, do, nome, Maria) e retorna um vetor de número reais contendo a concatenação das representações distribuídas das palavras da entrada.

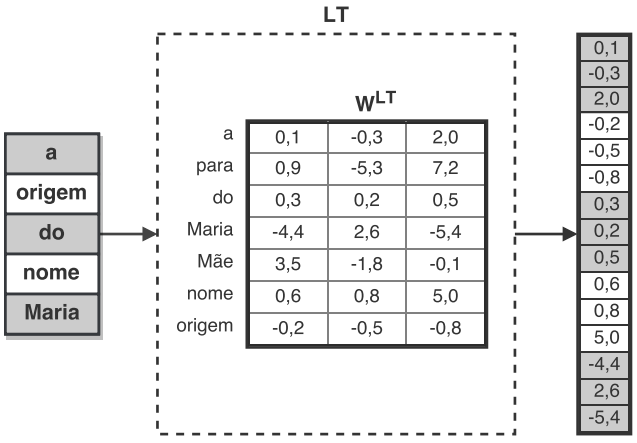


Figura 2.3: Exemplo da camada *lookup table*.

O único parâmetro da camada *lookup table* é uma matriz $W^{LT} \in \mathbb{R}^{N\times d}$, em que cada linha contém a representação distribuída, com d dimensões, de um dos N objetos de um dicionário. O tamanho d das representações e os N objetos

do dicionário devem ser determinados previamente. A entrada da camada *lookup table* consiste em uma lista de objetos e a sua saída é um vetor formado pela concatenação das representações distribuídas de cada um dos objetos da entrada. Esta camada será representada pela função:

$$LT(o) = \left(W_{o_1}^{LT}, W_{o_2}^{LT}, ..., W_{o_l}^{LT}\right), \tag{2.2}$$

em que $o = (o_1, o_2, \ldots, o_l)$ é uma lista de l objetos e $W_{o_i}^{LT}$ é a representação distribuída do objeto o_i.

A maioria das redes empregadas neste trabalho utiliza conjuntamente representações distribuídas de palavras e caracteres. A função $LT^w(x)$ representa a saída de uma camada *lookup table* que corresponde à concatenação das representações distribuídas do conjunto de palavras x. A função $LT^{chr}(z)$ representa a saída de uma camada *lookup table* que corresponde à concatenação das representações distribuídas dos caracteres no conjunto z. As representações distribuídas de uma palavra w e um caractere chr são simbolizados por r^w e r^{chr}.

A maioria das camadas $LT^w(\cdot)$ deste trabalho recebem janelas de palavras como entrada. Na Figura 2.4, são ilustradas duas janelas deste tipo da frase O gato emagreceu demais ano passado. Como se pode observar, estas janelas

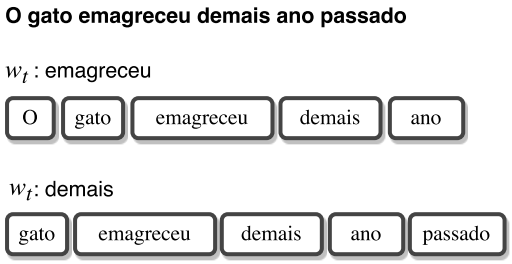


Figura 2.4: Exemplo de janelas de palavras.

são compostas por uma palavra de interesse, que no caso são emagraceu e demais, e pelas duas palavras posteriores e anteriores a elas. É possível criar uma janela de qualquer objeto. Assim, em uma sequência de objetos $o = (o_1, o_2, \ldots, o_n)$, uma janela de objetos é formada pelo objeto de interesse o_t e pelos $k/2$ objetos anteriores e os $k/2$ objetos posteriores a o_t .

Em uma janela, os objetos aos redores do objeto de interesse formam o contexto deste objeto. O *contexto de um objeto* é um conceito que depende do problema em questão, mas, de forma geral, consiste em uma lista de objetos, relacionados a um determinado contexto, que ajudam a classificar corretamente o objeto de interesse. Um outro exemplo de contexto do objeto de interesse o_t em uma sequência de objetos $(o_1, o_2, \ldots, o_n)$ pode ser os k objetos anteriores a o_t, ou seja, $(o_{t-k}, o_{t-k+1}, \ldots, o_{t-1})$. O contexto de um objeto é definido pela função $c(\cdot)$ e a união deste contexto com próprio objeto de interesse

é definido por $c'(\cdot)$.

2.3 WNN

A rede CharWNN utiliza representações distribuídas de palavras e de caracteres. Esta rede é baseada em uma rede proposta por Collobert et al. (2011), denominada aqui WNN, que utiliza somente representações distribuídas de palavras. A arquitetura da rede WNN é ilustrada na Figura 2.5. A entrada desta rede é a palavra de interesse w_t e a sua janela de contexto, ou seja, $c'(w_t)$. Esta entrada é passada para uma camada *lookup table* que é conectada a uma camada oculta tradicional. A saída desta camada oculta serve de entrada para a camada de saída softmax, a qual possui um neurônio representando cada etiqueta do problema. O vetor z^L com as ativações lineares da

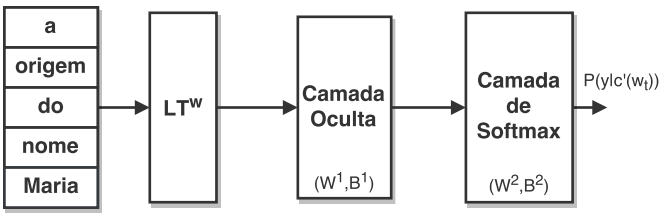


Figura 2.5: Rede proposta por Collobert et al. (2011).

camada de saída é calculado como:

$$z^L = W^2 h(W^1 LT^w(c'(w_t)) + b^1) + b^2, \tag{2.3}$$

sendo W^1 e b^1 os pesos e o bias da camada oculta e W^2 e b^2 os pesos e o bias da camada softmax. A camada de saída softmax calcula, para cada etiqueta $y \in \mathcal{Y}$, a seguinte probabilidade:

$$P(y|c'(w_t);\theta) = \frac{e^{z^L_y}}{\sum_{y' \in \mathcal{Y}} e^{z^L_{y'}}}, \tag{2.4}$$

em que $\mathcal{Y}$ é o conjunto de etiquetas.

Algoritmos baseados no WNN foram aplicados a diferentes problemas de etiquetagem de palavras, obtendo resultados estado da arte (Collobert et al., 2011; Fonseca et al., 2015). Entretanto, para alcançar estes resultados, estas redes necessitam ser aumentadas com atributos manuais que extraem informações morfológicas das palavras como sufixos, capitalização e padrões comuns (datas, valores monetários, etc.). Em problemas como a etiquetagem morfossintática, tanto informações semânticas e sintáticas das palavras, capturadas pelas representações distribuídas de palavras, quanto as informações morfológicas das palavras são importantes.

O modelo CharWNN utiliza a representação distribuída de palavras para capturar as informações sintáticas e semânticas das palavras e introduz uma representação distribuída de *caracteres* que, aliada a uma operação de convolução sobre os caracteres de uma palavra, é capaz de aprender automaticamente os atributos morfológicos das palavras.

2.4 CharWNN

A rede CharWNN, ilustrada na Figura 2.6, estende a WNN, incluindo um módulo de convolução de caracteres. Este módulo recebe a janela de palavras como entrada e extrai atributos morfológicos de cada palavra da janela. A saída do módulo é então concatenada às representações distribuída das palavras (a saída de LT^w) para formar a entrada da camada oculta da rede.

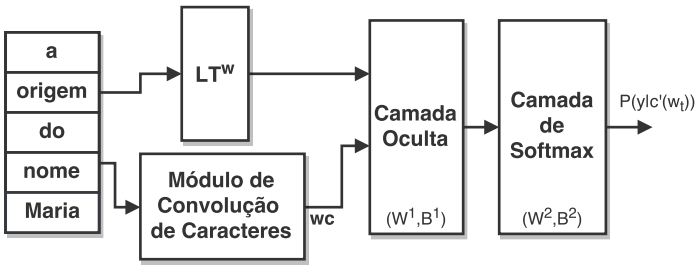


Figura 2.6: Rede proposta por CharWNN.

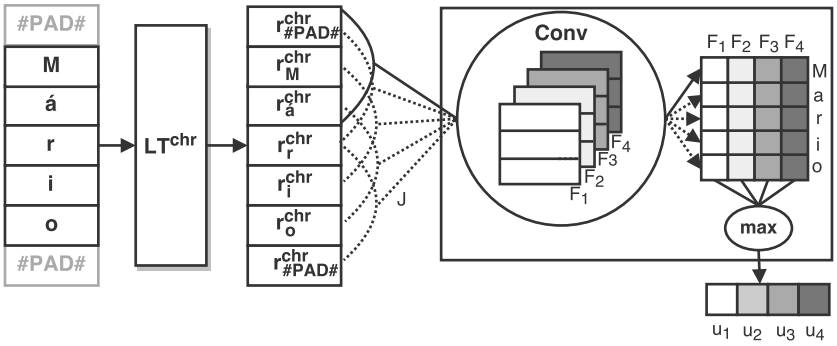


Figura 2.7: Módulo de convolução de caracteres aplicado à palavra Mário.

O módulo convolucional processa cada palavra da entrada de maneira independente. Aqui, descrevemos como uma palavra é processada. O mesmo processo é aplicado a cada palavra da janela de entrada. A Figura 2.7 ilustra a aplicação deste módulo sobre a palavra Mário. O módulo convolucional é composto por uma camada *lookup table* com as representações distribuídas de caracteres, uma operação de convolução sobre janelas de caracteres e uma operação de *max-pooling*. A primeira camada do módulo é uma *lookup table* denominada LT^{chr} que recebe como entrada os caracteres da palavra acrescidos de caracteres de *padding* (#PAD#, M, á, r, i, o, #PAD#) e retorna as

representações distribuídas de cada um destes caracteres, empilhadas uma a uma. Portanto, a saída da LT^{chr} é uma matriz na qual cada linha contém a representação distribuída de um caractere.

A operação de convolução consiste em aplicar um conjunto de filtros ($F1$, $F2$, $F3$ e $F4$) a cada caractere da palavra original (sem *padding*). Na realidade, a entrada para um filtro é uma janela de caracteres ao redor de cada caractere da palavra original. Por isto, usa-se os caracteres de *padding*. No exemplo da figura, estas janelas incluem 3 caracteres. As duas primeiras janelas, portanto, são formadas por $(r^{chr}_{M}, r^{chr}_{á}, r^{chr}_{r})$ e $(r^{chr}_{á}, r^{chr}_{r}, r^{chr}_{i})$. Um filtro é composto por uma matriz de dimensão igual à dimensão da matriz que representa cada uma destas janelas. No exemplo, cada filtro seria uma matriz de dimensão $3 \times d^{chr}$, em que d^{chr} é a dimensão da representação distribuída de caracteres. A aplicação de um filtro em uma janela consiste em multiplicar *elemento-a-elemento*[1] o filtro pela matriz da janela; e, em seguida, somar todos os valores da matriz resultante, obtendo-se um único valor real como resultado. Tem-se então que, para cada filtro, obtém-se um vetor de valores, sendo um valor para cada caractere da palavra de entrada (para cada janela de caracteres). Portanto, a saída da operação de convolução é uma matriz cuja dimensão, no exemplo, é de cinco linhas (pois a palavra de entrada possui cinco caracteres) por quatro colunas (pois o exemplo considera quatro filtros). A matriz resultante da convolução não possui dimensão fixa, pois o comprimento das palavras varia. Por isto, uma operação de *max-pooling* é aplicada à cada coluna desta matriz, produzindo um único valor para cada filtro do módulo convolucional. Esta operação visa selecionar a posição da palavra onde cada filtro foi ativado ao máximo. Desta forma, a camada de convolução introduz uma poderosa invariabilidade de localização.

Intuitivamente, os filtros do módulo convolucional representam detectores de atributos em nível de caractere. Portanto, é natural imaginar que um filtro pode aprender a detectar letras maiúsculas no início das palavras, hífen, e até sufixos que sejam altamente relacionados a etiquetas morfossintáticas específicas. Lembrando que a entrada do módulo convolucional é uma janela de palavras, temos que a saída produzida para cada palavra de entrada é composta por um vetor de valores, sendo um valor para cada filtro. Os vetores produzidos para cada palavra de entrada são finalmente concatenados para produzir um único vetor de saída que, por sua vez, é concatenado à representação distribuída das palavras e o vetor resultante é passado como entrada para a camada oculta da rede CharWNN.

A partir do exemplo acima, generalizamos o módulo de convolução de ca-

[1] A multiplicação elemento-a-elemento entre duas matrizes de mesma dimensão consiste em multiplicar os pares de elementos com mesmo índice um a um, resultando em uma matriz de mesma dimensão dos operandos. Esta operação será representada pelo operador $\odot$.

racteres da seguinte maneira. Uma palavra w da janela de entrada da rede é formada por m caracteres $(chr_1, chr_2, \ldots, chr_m)$ que são repassados à camada LT^{chr}. A camada LT^{chr} fornece como saída uma matriz contendo m vetores empilhados, um para cada caractere da palavra. A partir desta matriz, podemos definir m janelas de caracteres, sendo que cada uma contém k^{chr} linhas da matriz (consideramos $k^{chr}-1$ vetores adicionais de *padding*, metade deles nas primeiras linhas da matriz e a outra metade nas últimas linhas). A l-ésima janela de caracteres forma então uma matriz $J^l \in \mathbb{R}^{k^{chr} \times d^{chr}}$, onde d^{chr} é o número de dimensões das representação distribuída de caracteres e k^{chr} é o comprimento da janela. A janela J^l representa o l-ésimo caractere da palavra de entrada juntamente com $(k^{chr}-1)/2$ caracteres antes e $(k^{chr}-1)/2$ depois dele. A operação de convolução consiste em aplicar cl_u filtros $\{F_1, F_2, \ldots, F_{cl_u}\}$ a cada uma das janelas de caracteres. Cada filtro F_n, para $n = 1, 2, \ldots, cl_u$, consiste em uma matriz $F^n \in \mathbb{R}^{k^{chr} \times d^{chr}}$, ou seja, uma matriz de mesma dimensão de J^l. A aplicação do filtro F_n à janela J_l produz um valor real expressado como:

$$a_n^l = \sum_{i=1,2,\ldots,k^{chr}} \sum_{j=1,2,\ldots,d^{chr}} J_{ij}^l \cdot F_{ij}^n, \tag{2.5}$$

para $l = 1, 2, \ldots, m$ e $n = 1, 2, \ldots, cl_u$; ou seja, multiplicamos as duas matrizes elemento-a-elemento e então somamos todos os valores. Por fim, para cada filtro F_n, aplicamos uma operação de *max-pooling* sobre todas as janelas de caracteres. Esta operação é definida como:

$$u_n = \max\{a_n^1, a_n^2, \ldots, a_n^m\}. \tag{2.6}$$

A concatenação das saídas de todos os filtros irá gerar um vetor $r^{wch} = (u_1, u_2, \ldots, u_{cl_u}) \in \mathbb{R}^{cl_u}$. Para cada palavra de uma janela é gerado um vetor r^{wch}, que são concatenados formando um vetor wc. O vetor wc é utilizado, juntamente com as representações distribuídas das palavras, como entrada da camada oculta da rede CharWNN. Assim, a ativação linear da camada softmax da CharWNN é expressada como:

$$z^L = W^2 h(W^1(LT(c'(w_t)), wc) + b^1) + b^2. \tag{2.7}$$

O CharWNN, sem o uso de atributos manuais, obtém resultados comparáveis, ou até superiores, ao WNN com atributos manuais. Assim, podemos conjecturar que o CharWNN aprende automaticamente a extrair atributos de sufixo, prefixo e capitalização das palavras. Neste trabalho, nós implementamos o CharWNN utilizando a biblioteca Theano (Theano Development Team, 2016).

CAPÍTULO

3

Aprendizado Profundo de Representações

O desempenho das técnicas de aprendizado de máquina é altamente dependente de como os dados de entrada são representados (Bengio et al., 2013). Isto faz com que boa parte dos esforços de desenvolvimento de um algoritmo de aprendizado de máquina sejam despendidos na criação de uma representação adequada para o algoritmo de aprendizado. Na maioria dos casos, a representação é desenvolvida manualmente por especialistas, que é algo caro e lento (Bengio et al., 2013). Devido a isto, diversos algoritmos foram propostos para aprender automaticamente boas representações. Este problema é chamado pela literatura de aprendizado de representações (*representation learning*) ou aprendizado de atributos (*feature learning*).

Uma técnica de aprendizado de representações que tem se destacado recentemente é o *aprendizado profundo* (*deep learning*). Nesta técnica, aprende-se vários níveis de representação sendo que a representação de um nível é aprendida sobre a representação do nível anterior. Isto possibilita o aprendizado de representações mais abstratas, que amplificam os aspectos importantes para resolução da tarefa e são mais invariantes às mudanças locais dos dados (Bengio et al., 2013). Com a utilização dos métodos profundos, muitos algoritmos conseguiram ter desempenho equivalente ou melhor que o estado da arte sem o emprego de atributos manuais, utilizando somente os dados crús. Um exemplo disto é o CharWNN, que utiliza uma convolução sobre as representações distribuídas de caracteres, juntamente com as representações distribuídas de palavras.

O aprendizado das representações nas redes profundas ocorre por meio do

uso do algoritmo de retro-propagação. Este é um algoritmo flexível e genérico, que permite o emprego de inúmeras funções de perda (*loss function*). Explorando esta versatilidade, foram desenvolvidos três métodos de adaptação de domínio neste trabalho, que são descritos no Capítulo 4. Estes métodos utilizam funções de perda baseadas em duas funções, que são descritas neste capítulo e representam as duas formas básicas de aprendizado: supervisionada e não supervisionada.

3.1 Algoritmo de Retro-Propagação

O treinamento de redes neurais profundas é baseado no algoritmo de retro-propagação (*backpropagation*). Este algoritmo é muito versátil, possibilitando o treinamento de redes profundas usando as mais variadas funções de perda. Os métodos desenvolvidos neste trabalho se aproveitam desta versatilidade para treinar diferentes arquiteturas usando dados de diferentes domínios. A versatilidade deste algoritmo permite, por exemplo, o treinamento não supervisionado da rede, ou seja, é possível aprender as representações da entrada a partir de dados não anotados.

O algoritmo de retro-propagação generaliza o algoritmo de descida de gradiente para redes arbitrárias. Considerando um exemplo de entrada que é propagado camada por camada até gerar a saída da rede, uma função de perda (*loss function*) é aplicada à esta saída e fornece um valor de erro. O algoritmo de retro-propagação é responsável por propagar retroativamente este erro desde a camada de saída até a camada de entrada, distribuindo este valor por toda a rede. Os parâmetros de cada camada são atualizados de acordo com o seu impacto (gradiente) no erro retro-propagado.

Segundo Nielsen (2015), alguns pesquisadores, como LeCun et al. (1998), utilizaram a descida de gradiente estocástica e retro-propagação para treinar redes profundas nos anos 80 e 90. Entretanto, estes métodos de treinamento eram lentos e sua calibração era específica para cada arquitetura de rede. Poucos grupos eram capazes de treinar estas redes para problemas arbitrários. Em 2006, Hinton et al. (2006) propuseram uma maneira suficientemente genérica e eficaz para o treinamento de redes profundas. A ideia básica é treinar a rede por camadas e de forma não supervisionada (*greedy layer-wise training*). Ao final deste primeiro passo, a rede completa (todas as camadas) é treinada de maneira supervisionada na tarefa alvo.

Além desta nova técnica de treinamento, outros avanços na área de aprendizado de máquina permitiram o treinamento eficaz de redes neurais profundas. Novas técnicas de regularização, como *dropout*, ajudam a evitar o problema de *overfitting*, algo que redes profundas são suscetíveis devido ao

seu poder de representação. Novos algoritmos de optimização contínua, como Adagrad (Duchi et al., 2011) e Adam (Kingma and Ba, 2014), também foram responsáveis por avanços substanciais nesta área, pois evitam problemas comuns da descida de gradiente em redes profundas. Técnicas de inicialização dos parâmetros da rede, que evitam a saturação dos gradientes e das ativações da rede, compõem outro exemplo de avanço no treinamento de redes profundas. Estes avanços permitem o treinamento de redes profundas de uma maneira eficaz e genérica, viabilizando os resultados surpreendentes alcançados recentemente com redes neurais profundas.

3.2 Treinamento Supervisionado

O treinamento supervisionado é a técnica mais eficaz para se aprender os parâmetros de uma rede profunda. Quando consideramos uma rede cuja camada de saída é do tipo *softmax*, o aprendizado supervisionado consiste em encontrar os parâmetros θ que maximizam a função log-verossimilhança :

$$\theta \mapsto \log P(y|x;\theta), \tag{3.1}$$

no qual y é o rótulo correto associado à entrada x. As redes WNN e CharWNN podem ser treinadas usando-se este critério. Nestes casos, a função de perda pode ser definida como:

$$L_y(y_t, c'(w_t)) = -\log P(y_t|c'(w_t);\theta), \tag{3.2}$$

em que y_t é o rótulo correto da palavra w_t e $c'(w_t)$ é a janela de palavras no contexto de w_t. O procedimento de treinamento consiste então em minimizar esta função de perda para os exemplos de treinamento. Este procedimento é realizado utilizando-se descida de gradiente com o algoritmo de retro-propagação.

Esta função de perda implica que as etiquetas das palavras são independentes, ou seja, a etiqueta dada a uma palavra só depende da própria palavra e das palavras na sua janela de contexto. Além deste modelo, que é denominado de modelo por palavra, (Collobert et al., 2011) propõe outro modelo, chamado de modelo por frase, no qual a etiqueta de uma palavra depende das etiquetas das outras palavras na mesma frase. Considerando que o aprendizado do modelo por frase é computacionalmente mais custoso e os desempenhos do modelo por palavra e por frase são similares (Collobert et al., 2011), somente o modelo por palavra é empregado neste trabalho.

3.3 Treinamento Não Supervisionado

Existem diversas técnicas de aprendizado não supervisionado para redes neurais profundas. Para redes neurais de PLN, as principais técnicas são baseadas em *modelos neurais de linguagem*. Quando um ser humano lê a frase A bola quicou várias, rapidamente, nota-se que algo está faltando no final da frase. Dentre as milhares de palavras do nosso vocabulário, nós intuitivamente constatamos que provavelmente a palavra vezes é a palavra ausente na frase. O cérebro humano consegue fazer isto, pois, de algum modo, consegue modelar a distribuição probabilística das sequências de palavras do Português e, com isto, identifica frases estranhas, com baixa probabilidade de ocorrerem na linguagem, descobrindo possíveis erros. Em outras palavras, o nosso cérebro consegue aprender um modelo, chamado de modelo de linguagem.

Um *modelo de linguagem* determina a probabilidade de expressões, frases ou documentos de uma linguagem (Russell, 2009). Isto permite, por exemplo, predizer a palavra mais provável em um dado contexto. Por exemplo, dado a sequência de palavras A bola quicou várias, um modelo de linguagem pode ser empregado para predizer a próxima palavra mais provável. Existem diversas formas de aprender um modelo de linguagem. Neste capítulo, apresentamos os modelos de linguagem baseados em redes neurais, denominados *modelos neurais de linguagem*. Estes modelos são baseados em redes profundas e, geralmente, utilizam representações distribuídas de palavras.

Mikolov et al. (2013b) propôs duas arquiteturas de modelo neural de linguagem denominados *continuous bag-of-words* (CBOW) e *continuous skip-gram* (skip-gram). A ferramenta *word2vec*[1] (Mikolov, 2013) provê uma implementação eficiente destas arquiteturas para o aprendizado de representações distribuídas de palavras. A arquitetura skip-gram, ilustrada na Figura 3.1, prediz cada uma das palavras da janela em torno de uma palavra de interesse w_t. Por exemplo, dada a janela de palavras: a, origem, do, nome e Maria, a entrada da rede skip-gram seria composta pela palavra é e o objetivo da rede seria predizer as palavras a, origem, nome e Maria.

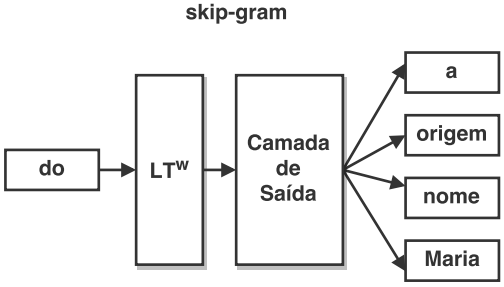


Figura 3.1: Arquitetura do skip-gram.

[1] https://code.google.com/archive/p/word2vec/

Uma forma de treinar este modelo consiste em maximizar a média da log-verossimilhança que é dada por:

$$\frac{1}{T}\sum_{i=1}^{T}\sum_{w\in c(w_i)}\log P(w|w_i,\theta), \tag{3.3}$$

no qual T é o tamanho de uma sequência de palavras do conjunto de treinamento e $c(w_i)$ é a janela de contexto da palavra w_i. Para calcular $P(w|w_i,\theta)$ pode-se utilizar uma camada de softmax. Porém, o grande número de palavras do dicionário torna este cálculo muito custoso. Uma alternativa é a técnica denominada *noise contrastive estimation* (NCE), que foi introduzida por Gutmann and Hyvärinen (2012) e utilizada por Mnih and Teh (2012) em um modelo de linguagem. O NCE também é uma aproximação do softmax. Considerando que o objetivo do *skip-gram* não é aprender um modelo de linguagem propriamente dito, mas sim aprender boas representações distribuídas de palavras, Mikolov et al. (2013a) simplificou o NCE de um modo que mantém a qualidade das representações. Assim, foi proposto um algoritmo baseado em amostragem negativa (*negative sampling*). Na amostragem negativa, o modelo é treinado para discriminar a palavra correta de k palavras ruidosas (*noise words*), considerando um contexto fixo. Assim, o objetivo do modelo skip-gram é maximizar a seguinte função:

$$\log\sigma(v'^{\intercal}_{c(w_t)_j}r^{w_t})+\sum_{i=1}^{k}\mathbb{E}_{w_i\sim P_n(w)}\log\sigma(-v'^{\intercal}_{w_i}r^{w_t}), \tag{3.4}$$

em que w_t é a palavra alvo da janela, $c(w_t)_j$ é palavra a ser predita na janela de palavras $c(w_t)$, v'_{wrd} é a representação da palavra na amostragem negativa, r^w é a representação distribuída da palavra w, $P_n(w)$ é a distribuição de probabilidade de ruído, k é o número de palavras ruidosas amostradas para cada contexto considerado e $\sigma(x)=\frac{1}{1+e^{-x}}$.

As representações distribuídas de palavras foram aprendidas neste trabalho por meio do *word2vec*, utilizando amostragem negativa e o modelo skip-gram. Métodos clássicos de redução de dimensionalidade, como PCA, SVD e LDA, poderiam ser utilizados para aprender tais representações. Entretanto, estes métodos são inviáveis para dados com alta dimensionalidade, como é o caso deste trabalho.

3.4 Treinamento Semissupervisionado

Aliando-se o algoritmo de retro-propagação com técnicas de treinamento não supervisionado, é possível desenvolver uma enorme variedade de métodos de treinamento de redes neurais profundas. Neste trabalho, por exemplo,

as representações distribuídas de palavras são inicialmente aprendidas de forma não supervisionada. Depois disto, geralmente, estas representações são usadas para inicializar os parâmetros de alguma camada *lookup table*, que são posteriormente atualizados pelo treinamento supervisionado, juntamente com os outros parâmetros da rede (camada oculta e camada softmax, por exemplo).

A etapa em que todos ou parte dos parâmetros da rede são aprendidos de forma não supervisionado, como o treinamento inicial das representações distribuídas de palavras do exemplo acima, é chamado de pré-treinamento não supervisionado (Goodfellow et al., 2016) e pode ajudar na otimização e generalização da rede. Depois deste passo, geralmente, é realizado uma outra etapa, chamado de ajuste fino (*fine tuning*), que consiste em atualizar os parâmetros pré-treinados durante o treinamento supervisionado. Normalmente, o ajuste fino é benéfico e traz uma melhora substancial no desempenho do algoritmo de AM.

É possível, durante o processo de aprendizado, realizar o pré-treinamento e não aplicar o ajuste fino. Assim, por exemplo, podemos aprender de forma não supervisionada uma representação distribuída de palavras a partir de um grande conjunto de dados. Posteriormente, durante o aprendizado supervisionado, podemos atualizar apenas a camada oculta e a camada de softmax da rede, deixando fixa a representação distribuída de palavras. Outra possibilidade interessante é combinar mais de uma função de perda concomitantemente. Desta forma, é possível aprender representações considerando dois objetivos distintos (às vezes, até conflitantes).

Estas e outras possibilidades permitem que os diferentes níveis de representação possam ser treinados de vários modos diferentes. Neste trabalho, esta liberdade é explorada na implementação dos métodos de adaptação de domínio.

CAPÍTULO

4

Adaptação de Domínio

Na adaptação de domínio, existe um domínio chamado de origem que possui uma grande quantidade de exemplos anotados e um outro domínio no qual o modelo será aplicado, chamado de domínio alvo, que não possui nenhum ou uma quantidade muito pequena de exemplos anotados. Em ambos domínios, normalmente, os dados não anotados são facilmente adquiridos em grande quantidade. O problema de adaptação de domínio ocorre devido às diferenças das distribuições de probabilidade dos dados de treino e de teste, ou seja, dos dados dos domínios origem e alvo (Li, 2012).

Na literatura, existem três cenários distintos de adaptação de domínio que diferenciam-se pelo uso de dados do domínio alvo (Daumé et al., 2010). Quando utiliza-se somente dados não anotados do domínio alvo, a adaptação de domínio é chamada de não supervisionada. Por outro lado, ela é chamada de supervisionada quando emprega-se exclusivamente poucos dados anotados do domínio alvo. Um método de adaptação de domínio é dito semissupervisionado quando usa concomitantemente dados anotados e não anotados do domínio alvo. A maioria dos métodos da literatura consideram o cenário de adaptação de domínio não supervisionada. Este é o cenário usado neste trabalho.

Neste trabalho definiremos $\mathcal{X}$ como o espaço dos dados de entrada e $\mathcal{Y}$ como o conjunto de rótulos de saída. Portanto, (x, y), sendo $x \in \mathcal{X}$ e $y \in \mathcal{Y}$, constitui um exemplo anotado. Diversos métodos de adaptação de domínio foram propostos na literatura. Para uma melhor compreensão das diferentes abordagens, pode-se dividir estes métodos em três tipos (Li, 2012). Para detalhar estes tipos, definiremos $P_S(x, y; \theta)$ como a distribuição de probabilidade dos dados do domínio origem e $P_T(x, y; \theta)$ a distribuição do domínio alvo, considerando um

modelo θ. Alguns métodos de adaptação de domínio exploram a transformação do espaço de atributos, buscando um conjunto de atributos que torne as probabilidades condicionais $P_S(y|x;\theta)$ e $P_T(y|x;\theta)$ mais similares. Outros métodos desta categoria projetam o espaço de atributos X para um novo espaço X^{Tran}, em que haja alguma correspondência entre $P_S(y|x;\theta)$ e $P_T(y|x;\theta)$. O segundo tipo de abordagem para adaptação de domínio, denominado *prior based adaptation*, explora modelos a priori para diminuir a lacuna da distribuição entre os dois domínios. O terceiro tipo de abordagem, denominado ponderação de instância, utiliza a ponderação dos exemplos de treinamento, com o intuito de dar mais importância para os exemplos do domínio origem que sejam mais informativos em relação ao domínio alvo.

Os métodos estados da arte na adaptação de domínio de etiquetagem morfossintática se encaixam na abordagem que explora a transformação do espaço de atributos. Por isto, os métodos desenvolvidos neste trabalho são baseados nesta abordagem. Desenvolvemos tais métodos combinando-se o CharWNN com três métodos de AD propostos na literatura. Estes métodos foram desenvolvidos para outras tarefas e/ou usando arquiteturas diferentes. Os métodos desenvolvidos aqui aprendem representações usando os dados não anotados do domínio alvo, seguindo o cenário de adaptação de domínio não supervisionada. A expectativa é que estas representações tornem $P(y|x;\theta)$ mais similar a $P_T(y|x;\theta)$ do que as representações originais do CharWNN.

4.1 DLID

O método de interpolação de domínios para etiquetagem de palavras, denominado DLID (*deep learning for domain adaptation by interpolating between domains*), foi proposto por Chopra et al. (2013) para adaptação de domínio no problema de reconhecimento de objetos em imagens. Aqui, desenvolvemos um método inspirado no DLID para a adaptação de domínio no problema de etiquetagem morfossintática. Daqui em diante, por simplicidade, denominaremos o nosso método como DLID.

O DLID possui duas etapas: pré-treinamento não supervisionado e treinamento supervisionado. Na primeira etapa, são considerados três domínios: origem, alvo e intermediário. Para cada domínio, é definido um conjunto de dados não anotados para o pré-treinamento. Os dados dos domínios origem e alvo são compostos por todos os exemplos disponíveis para cada domínio (desconsiderando as etiquetas do domínio origem). O domínio intermediário é um domínio artificial composto por exemplos advindos dos domínios origem e alvo. Uma metade dos dados deste domínio são compostos por textos do domínio origem, e a outra metade por textos do domínio alvo. Pode-se inter-

pretar o domínio intermediário como uma interpolação dos domínios origem e alvo. Cada um destes conjuntos de dados são utilizados pela ferramenta *word2vec* para aprender uma representação distribuída de palavras para um determinado domínio. Assim, cada palavra w possui três representações, uma para cada domínio. Caso a palavra w não exista no vocabulário de um domínio, então esta palavra é substituída pela palavra artificial UUUNKKK. A palavra UUUNKKK representa palavras desconhecidas (ou muito raras) em um vocabulário e cada um dos domínios possui uma representação diferente para esta palavra.

Depois do pré-treinamento, as representações aprendidas nos três domínios são usadas como entrada para a rede CharWNN, juntamente com a convolução de caracteres, como ilustrado na Figura 4.1. Este CharWNN mo-

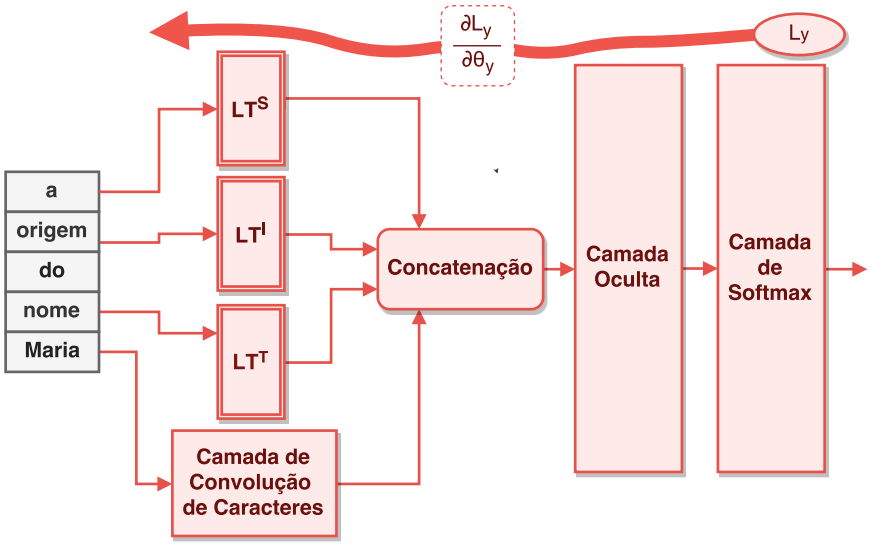


Figura 4.1: Arquitetura da rede neural do DLID.

dificado é treinado de forma supervisionada nos dados do domínio origem, sendo composto por três camadas *lookup table* em paralelo que recebem a mesma janela de palavras. Cada uma destas camadas está associada a uma representação distribuída de palavras de um domínio, sendo que as camadas relacionadas com o domínio origem, alvo e intermediário são denominadas, respectivamente, $LT^S(\cdot)$, $LT^T(\cdot)$ e $LT^I(\cdot)$. Na figura, estas camadas são ilustradas como retângulos de borda dupla, indicando que seus parâmetros foram pré-treinados usando um treinamento não supervisionado. Esta notação de retângulos com borda dupla é usada em outras figuras deste capítulo, para indicar quais parâmetros são pré-treinados nos métodos de AD. A saída de cada uma das camadas LT são concatenadas, juntamente com a convolução de caracteres, em um único vetor que compõe a entrada para a camada oculta da CharWNN. O restante da rede segue a arquitetura CharWNN original.

No treinamento supervisionado, é utilizado a função de perda $L_y(\cdot)$, definida

como:

$$L_y(y, c'(w_t)) = -\log P(y|c'(w_t); \theta). \tag{4.1}$$

Esta é a função padrão de log-verossimilhança quando se usa uma camada softmax e será utilizada por outros métodos deste capítulo. Observe que na Figura 4.1 possui uma seta ligada associada à L_y que passa sobre todas as camadas da rede. Esta notação indica que os parâmetros destas camadas serão atualizadas de acordo com o gradiente $\frac{\partial L_y}{\partial \theta_y}$ de $L_y(\cdot)$ através do algoritmo de retro-propagação. Deste modo, será realizado o ajuste fino das representações de palavras, que foram pré-treinadas anteriormente.

É importante ressaltar que, tanto no DLID quanto nos outros dois métodos de AD desenvolvidos neste trabalho, as representações de caracteres são aprendidas exclusivamente durante o treinamento supervisionado, ou seja, não há pré-treinamento para estas representações. Isto se deve a dois fatores. Primeiro, a ferramenta *word2vec* não provê treinamento de representação de caracteres, apenas de palavras. Segundo, um método de treinamento não supervisionado foi implementado neste trabalho e poderia ter sido usado para pré-treinar representações de caracteres. Entretanto, esta implementação foi realizada no final do trabalho e, por isto, não houve tempo hábil para realizar tal pré-treinamento. De qualquer maneira, acreditamos que esta omissão não ocasiona perda de desempenho pois, geralmente, atributos morfológicos das palavras são mais invariantes entre domínios.

4.1.1 DLID-M

Considerando dois domínios distintos, é natural pensar que uma representação distribuída de palavras treinada na união dos dois domínios é equivalente à média entre duas representações distribuídas de palavras treinadas em cada domínio individualmente. Esta ideia está relacionada a modelos de mistura (Bickel and Scheffer, 2005) e comitês de modelo por média (Haykin, 1998). Baseando-se em tal ideia e considerando que o domínio intermediário no DLID representa a união entre os domínios origem e alvo, podemos gerar a representação do domínio intermediário através da média das representações dos domínios origem e alvo. Assim, uma palavra w é representada no domínio intermediário por um vetor cuja dimensão j é igual à média dos valores encontrados na dimensão j de cada uma das representações de w nos domínios origem e alvo. Por exemplo, suponha que a representação distribuída da palavra casa é igual a $(10,4,2,3)$ no domínio origem e $(12,22,6,11)$ no domínio alvo. Então, esta palavra será representada pelo vetor $(11,13,8,7)$ no domínio intermediário. Denominamos este método *DLID-M*, para diferenciá-lo do método anterior, o DLID. O método DLID-M tem a vantagem de evitar o pré-

treinamento no domínio intermediário, pois a representação específica deste domínio é calculada por uma simples média entre duas outras representações. O cálculo desta média é, claramente, computacionalmente muito mais barato do que o treinamento não supervisionado no domínio intermediário.

4.2 DAN

Na literatura de adaptação de domínio, alguns métodos buscam descobrir atributos invariantes à mudança de domínio. A maioria destes métodos identificam os atributos invariantes em uma etapa de pré-processamento, ou seja, antes do treinamento supervisionado. Nestes métodos, os atributos gerados no pré-treinamento não são atualizados no treinamento supervisionado. O método DAN, proposto em Ganin et al. (2016), aprende representações invariantes entre dois domínios ao mesmo tempo que treina supervisionadamente um classificador para a tarefa alvo. Aqui, propomos um método (denominaremos este método simplesmente DAN) que combina a rede CharWNN com esta ideia.

Na Figura 4.2, apresentamos uma ilustração da arquitetura do método proposto. Neste método existem três módulos que são retratados com cores

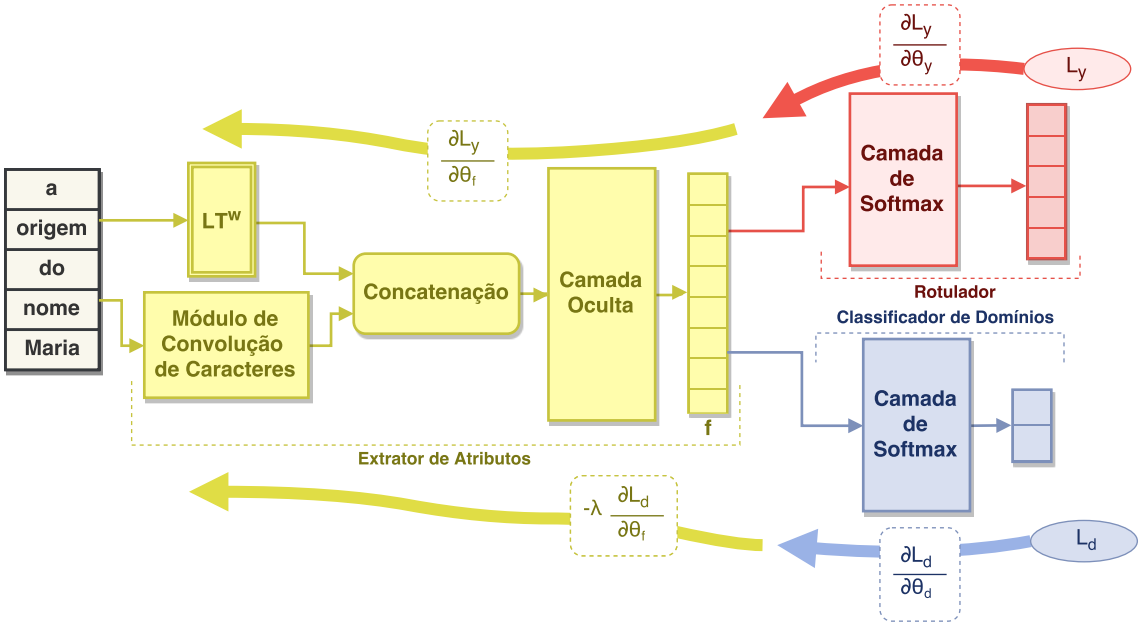


Figura 4.2: Arquitetura da rede neural do DAN.

diferentes, sendo que as camadas em amarelo e vermelho correspondem ao CharWNN original. As camadas em vermelho representam o rotulador de etiquetas morfossintática, que é treinado usando os dados anotados do domínio origem e tem como objetivo etiquetar a entrada. O classificador de dominios (azul), treinado usando os dados domínio origem e alvo, prediz de qual domínio a entrada pertence. O rotulador e o classificador de domínios são conectados

a um extrator de atributos (amarelo), que utiliza para o seu treino os dados do domínio origem e alvo.

De forma geral, O DAN tem como objetivo treinar um rotulador e classificador de domínios e aprender um extrator de atributos capaz de gerar representações dos dados que sejam boas para a etiquetagem morfossintática e, ao mesmo tempo, variem pouco de um domínio para outro. Como representações invariantes ao domínio podem ser aprendidas tornando as indistinguível em relação a sua origem, o erro do classificador de domínios será utilizado para mensurar o quão fácil é descobrir a origem do dado a partir de sua representação. Além disto, usaremos o erro do rotulador para medir a qualidade das representações para a tarefa final. Deste modo, para atingir os seus objetivos, o DAN irá, concomitantemente, ajustar os parâmetros do extrator de atributos para diminuir o erro do rotulador e aumentar o erro do classificador de domínios. Ao mesmo tempo, também serão ajustados os parâmetros do rotulador e classificador de domínios para diminuir os seus respectivos erros.

Pode-se considerar o extrator de atributos como uma função, denominada de $G_f(\cdot)$, que transforma uma janela de palavras em um vetor de número reais, denominado de f. Esta função é definida matematicamente da seguinte forma:

$$G_f(c'(w_t)) = h(W^1(LT^w(c'(w_t)), wc) + b^1), \tag{4.2}$$

em que $c'(w_t)$ é janela de palavras, W^1 e b^1 são os pesos e o bias da camada oculta, $h(\cdot)$ é uma função tangente hiperbólica, wc é a saída da convolução realizada nos caracteres das palavras e $LT^w(\cdot)$ é a função que representa uma camada *lookup table* contendo as representações distribuídas de palavras.

A saída deste extrator, que consiste no vetor f, serve como entrada para o rotulador de etiquetas morfossintáticas e também para o classificador de domínios. O rotulador de etiquetas morfossintática rotula com uma etiqueta y a palavra que se encontra no meio da janela de palavras e tem $L_y(\cdot)$ (apresentada no Capítulo 2) como função de perda. O classificador de domínio, que também recebe o vetor f como entrada, prediz se uma janela de palavras pertence ao domínio origem ou alvo. A sua função de perda deste classificador é definida como a log-verossimilhança negativa abaixo:

$$L_d(y_d, c'(w_t)) = -\log P(y_d|c'(w_t); \theta), \tag{4.3}$$

em que $y_d \in \mathcal{Y}_d$, $\mathcal{Y}_d$ é o conjunto contendo os dois possíveis rótulos para o classificador de domínios e θ é o conjunto de parâmetros do extrator de atributos, do rotulador de etiquetas morfossintática e do classificador de domínios que são, respectivamente, simbolizados por θ_f, θ_y e θ_d. Ambos os módulos de saída são compostos por apenas uma camada softmax, que são utilizadas para calcular

$P(y|c'(w_t);\theta)$ e $P(y_d|c'(w_t);\theta)$.

No DAN, busca-se aprender: (i) um rotulador de etiquetas morfossintática; (ii) um classificador de domínios; e (iii) um extrator capaz de extrair atributos discriminativos e invariantes. O dois primeiros objetivos podem ser alcançados pela busca de parâmetros θ_y e θ_d que, respectivamente, minimizem $L_y(\cdot)$ e $L_d(\cdot)$. Enquanto isto, visando aprender atributos discriminativos e invariantes, os parâmetros θ_f são otimizados para minimizar $L_y(\cdot)$ e maximar $L_d(\cdot)$. A primeira otimização visa encontrar parâmetros que sejam bons para a tarefa de etiquetagem morfossintática. Enquanto, a segunda direcionada o aprendizado dos parâmetros para tornar a representação f indistinguível em relação à sua origem, o que segundo Ganin et al. (2016) induz ao aprendizado de atributos invariantes.

Matematicamente a otimização dos parâmetros de cada módulo pode ser definido da seguinte maneira:

$$\theta_y \mapsto L_y(y, c'(w_t)), \tag{4.4}$$

$$\theta_d \mapsto L_d(y_d, c'(w_t)), \tag{4.5}$$

$$\theta_d \mapsto L_y(y, c'(w_t)) - \lambda_d L_d(y_d, c'(w_t)), \tag{4.6}$$

no qual $\lambda_d \in \mathbb{R}$ é um hiper-parâmetro que pondera a influência de $L_d(\cdot)$ na otimização de θ_d. Estas otimizações são realizadas utilizando-se o algoritmo de descida de gradiente estocástico, no qual as atualizações dos parâmetros ocorrem da seguinte forma:

$$\theta_y \leftarrow \theta_y - \mu \frac{\partial L_y(y, c'(w_t))}{\partial \theta_y}, \tag{4.7}$$

$$\theta_d \leftarrow \theta_d - \mu \frac{\partial L_d(y_d, c'(w_t))}{\partial \theta_d}, \tag{4.8}$$

$$\theta_f \leftarrow \theta_f - \mu \left(\frac{\partial L_y(y, c'(w_t))}{\partial \theta_y} - \lambda_d \frac{\partial L_d(y_d, c'(w_t))}{\partial \theta_d}\right), \tag{4.9}$$

em que μ é a taxa de aprendizado.

As otimizações e atualizações descritas acima podem ser implementadas de diversas formas. Considerando que um dos termos da função de perda do extrator de atributos ($-\lambda_d L_d(\cdot,\cdot)$) é igual à função de perda do classificador de domínios ($L_d(\cdot,\cdot)$) multiplicada por $-\lambda_d$, um modo eficiente de implementar isto é por meio do uso de uma camada especial, chamada de *camada de inversão de gradiente*. Esta camada é inserida entre o classificador de domínio e o ex-

trator de atributos e aplica uma transformação de identidade em uma entrada x na fase de propagação. Assim, o valor da entrada é diretamente repassado para a próxima camada. Já durante a fase de retro-propagação, a camada de inversão de gradiente multiplica por $-\lambda_d$ o gradiente da camada de softmax.

A rede neural do DAN é treinada usando um mini-lote (*mini-batch*) contendo um exemplo do conjunto de dados anotados do domínio origem, simbolizado por $c'(w_S)$, e outro exemplo do conjunto de dados não anotados do domínio alvo, denominado de $c'(w_T)$. Assim, a função de perda final a ser otimizada é:

$$L_y(y, c'(w_S)) + L_d(y_d = 1, c'(w_S)) + L_d(y_d = 0, c'(w_T)), \tag{4.10}$$

no qual $y_d = 1$ quando o exemplo origina-se do domínio origem e $y_d = 0$ quando o exemplo advém do domínio alvo.

4.3 DATT

Vários trabalhos sobre a adaptação de domínio não supervisionada, como Glorot et al. (2011), Chen et al. (2012) e Ma et al. (2014), aprendem extratores de representação que possuem múltiplas camadas a partir dos dados não anotados dos domínios. A utilização de várias camadas gera, frequentemente, representações mais abstratas. Tais representações contêm informações mais genéricas, o que implica em uma melhor adaptação do modelo. Com esta finalidade, Ma et al. (2014) propuseram uma rede neural composta por dois módulos, no qual um deles é uma rede com várias camadas que é treinada de forma não supervisionada.

Baseando-se neste trabalho, desenvolvemos um método, chamado de DATT, que conecta um extrator de atributos do domíno alvo, denominado de EXT-WEB, a camada de saída softmax do CharWNN, resultando em uma rede neural ilustrada na Figura 4.3. Nesta rede neural, podemos considerar a camada de saída softmax como um classificador que irá etiquetar os dados utilizando as representações geradas na camadas anteriores. Uma desta representações será criada por EXT-WEB, que ao ser treinado de forma não supervisionada, irá aprender atributos que podem ser específicos para somente o domínio alvo ou genéricos a mais de dois domínios. Esperamos que EXT-WEB aprenda mais o segundo tipo de atributos com uso de duas camadas e que, assim, consiga melhor representar os dados para adaptação de domínio.

Porém, uso de somemte o EXT-WEB não é o bastante para produzir um bom classificador. Precisamos de atributos mais particulares a tarefa, como sufixo e capitalização. Isto é aprendido pelas camadas em amarelo do CharWNN. Assim, estas camadas são conectada a camada de saída de softmax e formam um extrator, denominado de EXT-POS, que extrai os atributos impor-

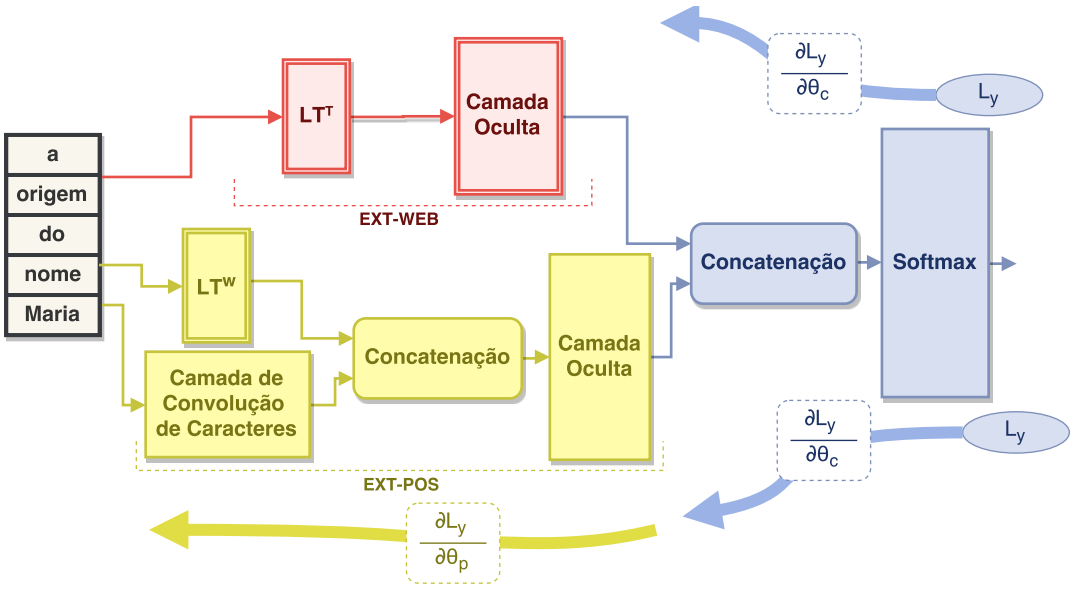


Figura 4.3: Arquitetura da rede neural do DATT.

tantes para a tarefa de etiquetagem morfossintática.

O EXT-POS pode ser definido matematicamente como sendo a seguinte função:

$$G_{pos}(c'(w_t)) = h(W^3(LT^w(c'(w_t)), wc) + b^3), \tag{4.11}$$

sendo $c'(w_t)$ a janela de palavras, LT^w a camada *lookup table*, wc a saída da camada de convolução de caracteres, W^3 e b^3 os pesos e o bias da camada oculta e $h(\cdot)$ é uma função tangente hiperbólica. Sendo $G_T(\cdot)$ a função que gera a saída de EXT-WEB, então a camada de saída de softmax irá calcular $P(y|c'(w_t);\theta)$ utilizando um vetor que concatena $G_T(\cdot)$ e $G_{pos}(\cdot)$. É importante salientar que na Figura 4.2 os parâmetros da camada de saída de softmax e do EXT-POS são representados, respectivamente, por θ_c e θ_p.

A rede neural do DATT é treinada usando os dados anotados do domínio alvo. Durante o aprendizado, tenta-se encontrar parâmetros da camada de saída e do EXT-POS que minimizem $L_y(\cdot)$. De forma semelhante a Ma et al. (2014), os parâmetros do EXT-WEB não são atualizados durante este treinamento. Inspirado nas técnicas empregadas no *word2vec*, estes parâmetros são treinados de forma não supervisionada utilizando dados não anotados do domínio alvo. EXT-WEB é formado por uma camada *lookup table*, denominada de LT^T, e uma camada oculta. Este extrator representa uma janela de palavras como um vetor de números reais no qual esta transformação pode ser definida pela seguinte função:

$$G_T(c'(w_t)) = h(W^1 LT^T(c'(w_t) + b^1), \tag{4.12}$$

sendo $h(\cdot)$ uma função tangente hiperbólica e W^1 e b^1 os pesos e bias da camada oculta.

Para o treinamento não supervisionado do EXT-WEB, empilhamos uma camada de saída, que possui um neurônio e uma função sigmoide como função de ativação, sobre este extrator. Isto resulta em uma rede neural, ilustrada na Figura 4.4, que tem como tarefa predizer se uma janela de palavras é

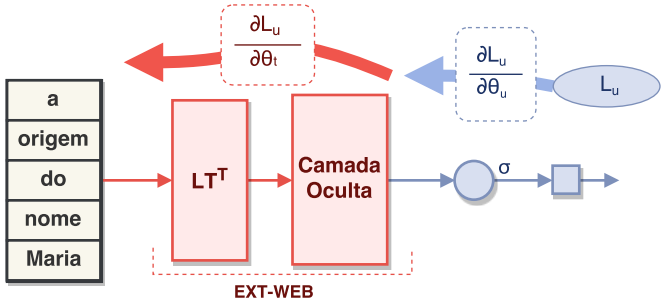


Figura 4.4: Arquitetura da rede neural usada para o treinamento do extrator dos atributos do domínio alvo.

plausível ou não de ocorrer no domínio de alvo. A saída desta rede pode ser interpretada como sendo a probabilidade de uma janela de palavras $c'(w_t)$ ser plausível. Sendo y o rótulo de uma janela de palavras e $y = 1$ quando uma janela é plausível, então $P(y = 1|c'(w_t);\theta)$ é definido da seguinte maneira:

$$P(y = 1|c'(w_t);\theta_u) = \sigma(W^2 G_T(c'(w_t)) + b^2), \tag{4.13}$$

no qual W^2 e b^2 são os pesos e o bias da camada de saída, θ_u são os parâmetros da rede e σ é uma função sigmoide.

Durante o processo de aprendizado, a rede neural da Figura 4.4 recebe como entrada n janelas de palavras, no qual somente uma destas janelas ocorre no conjunto de dados não anotados do domínio alvo, ou seja, é plausível. As demais $n-1$ janelas são construídas pela substituição da palavra no centro da janela plausível por outras palavras, chamadas de palavras de ruído. Assim, baseando-se no método da amostragem negativa (Mikolov et al., 2013a), a rede neural é treinada para minimizar a seguinte função de perda:

$$L_u(c'(w_t)) = \log P(y = 1|c'(w_t);\theta_u) + \sum_{i=1}^{k} \mathbb{E}_{w_i \sim P_n(w)} \log(1 - P(y = 1|c'(w_i);\theta_u)), \tag{4.14}$$

sendo $c'(w_t)$ a janela de palavras plausível, $P_n(w)$ a distribuição de probabilidade das palavras de ruído, k o número de palavras de ruído selecionadas de acordo com $P_n(w)$ e $c'(w_i)$ a janela de palavras não plausível. Do mesmo que em Mikolov et al. (2013a), $P_n(w)$ é igual a distribuição dos unigramas $U(w)$ elevada por $\frac{3}{4}$, sendo que $U(w)$ é calculada usando o conjunto de dados de treinamento.

Como no *word2vec*, empregamos no treinamento não supervisionado duas técnicas: poda de palavras raras (*rare-word pruning*) e sub-amostragem (*sub-sampling*). A primeira técnica retira do vocabulário as palavras que aparecem

menos que n vezes no conjunto de treinamento. Estas palavras são substituídas pelo termo UUUNKKK, que representa as palavras que não estão no vocabulário. A segunda técnica descarta uma janela de palavras $c'(w_t)$ de acordo com a seguinte probabilidade:

$$P(c'(w_t)) = 1 - \sqrt{\frac{t}{f(w_t)}}, \tag{4.15}$$

sendo $f(w_t)$ o valor da frequência da palavra w_t dividido pelo tamanho do conjunto de treinamento e t um limite (*threshold*) a ser escolhido, que possui valores entre 0 e 1. O emprego da Equação 4.15 ocasiona um maior descarte das palavras mais frequentes que possuem $f(w_t)$ maior que t. Segundo Mikolov et al. (2013a), isto torna o treinamento mais rápido e, em alguns casos, melhora qualidade das representações.

CAPÍTULO

5

Trabalhos Relacionados

Adaptação de domínio é um importante problema na área de PLN (Blitzer et al., 2006). Segundo Glorot et al. (2011), o problema de adaptação de domínio foi formalizado primeiramente por Daume III and Marcu (2006), porém vários trabalhos anteriores estudaram este problema utilizando diferentes nomes. A adaptação de domínio é chamada de supervisionada e não supervisionada quando utiliza-se, respectivamente, somente dados anotados e não-anotados do domínio alvo. Quando este dois tipos de dados são usados, então adaptação de domínio é dita semissupervisionado.

Os métodos estado da arte na adaptação de domínio de etiquetagem morfossintática e os métodos estudados e desenvolvidos neste trabalho se enquadram em uma abordagem que busca representações dos dados mais adequadas para adaptação de domínio. Além desta abordagem, que existem outras duas bastante exploradas na literatura, que são adaptação a priori e ponderação de instâncias. Os trabalhos Finkel and Manning (2009), Chelba and Acero (2004) e Chan and Ng (2006) utilizam a primeira abordagem, enquanto os trabalhos Axelrod et al. (2011), Xu et al. (2011) e Liu et al. (2012) empregam a segunda abordagem. Uma discussão mais detalhada dos trabalhos de adaptação de domínio em PLN é encontrada em Li (2012)

Todos os métodos desenvolvidos neste trabalho aprendem as representações a partir de métodos de aprendizado profundo. Assim, esta revisão bibliográfica será focado nos trabalhos que utilizam os métodos de aprendizado profundo ou técnicas de aprendizado de representação para adaptação de domínio para PLN, com exceção de um método de visão computacional no qual DLID foi inspirado. Boa parte destes trabalhos na literatura são sobre adaptação de domínio não supervisionada, sendo que um dos mais citados é Blitzer

et al. (2006). Este trabalho desenvolveu um método, denominado de *Structural Correspondence Learning* (SCL), que consegue aprender uma representação de atributos através da utilização dos dados não anotados. Isto é feito pelo uso dos atributos pivôs, que possuem o mesmo comportamento durante aprendizagem discriminativa nos dois domínios. Com estes atributos, consegue-se aprender um mapeamento do espaço original dos atributos dos domínios para um espaço de atributos com valores reais, compartilhado e com poucas dimensões. Os atributos desses dois espaços são usados durante o treinamento supervisionado. Segundo Li (2012), as principais vantagens do SCL são a capacidade correlacionar atributos em diferentes domínios e a não necessidade de qualquer dado rotulado no domínio alvo. Porém, as principais desvantagens deste método são a necessidade da escolha dos atributos pivôs e a falta de noção de quanto a qualidade destes atributos interfere no desempenho.

Glorot et al. (2011) tenta resolver o problema de adaptação de domínio com uma abordagem que utiliza técnicas de aprendizado profundo (*deep learning*). Tais técnicas são capazes de aprender atributos mais genéricos que, provavelmente, são mais representativos para os domínios origem e alvo. Para extração destes atributos de forma não supervisionada, é empregado o *Stacked Denoising Autoencoder* (SDA), que é treinado usando o SGD. Logo depois do treino do SDA, é realizado o treino supervisionado de um classificador linear, um SVM no caso do artigo, usando os dados do domínio alvo modificados pelo extrator de atributos. Baseando-se neste trabalho, Chen et al. (2012) utilizou para o treinamento não supervisionado o *Marginalized Stacked Denoising Autoencoder* (*m*SDA). Diferente do SDA, *m*SDA não corrompe de forma aleatória os atributos. Isto o torna mais rápido de ser treinado que o primeiro. Tanto Glorot et al. (2011) quanto Chen et al. (2012), experimentaram seus métodos de adaptação de domínio para a tarefa de análise de sentimentos.

Para adaptação de domínio na tarefa de etiquetagem morfossintática, Schnabel and Schütze (2014) utilizaram uma representação do contexto local de um palavra para realizar a tarefa de etiquetagem morfossintática. Esta representação é feita pela concatenação da representação da palavra a ser rotulada com as representações das palavras vizinhas. Cada palavra é representada por um vetor formado pelos atributos binários de sufixo, pelos atributos binários de *shape* (atributos relacionados a ortografia e morfologia), e pelos atributos de distribuição (*distributional features*), que são atributos relacionados a frequência do bigrama formado pela palavra atual e a palavra vizinha. As representações dos atributos de distribuição são aprendidas de forma não supervisionada, que são utilizadas junto com os outros atributos no treinamento supervisionado.

Na área de PLN, é comum organizar os atributos em gabaritos (*templates*),

sendo que, para um dado exemplo, existe exatamente um atributo ativo por gabarito. Yang and Eisenstein (2015) propõem, para adaptação de domínio, utilizar as estruturas dos gabaritos para aprender de forma não supervisionada as representações distribuídas dos atributos. Na aprendizagem não supervisionada, eles utilizam o modelo *skip-gram*, proposto por Mikolov et al. (2013b). Porém, diferentemente de Mikolov et al. (2013b), o modelo de linguagem não tenta predizer as palavras adjacentes a uma palavra, mas os atributos ativos dos outros gabaritos dado um determinado atributo de um gabarito. Yang and Eisenstein (2015) utilizam os dados não anotados dos domínios alvo e origem no treinamento não supervisionado. Depois deste treinamento, é realizado o treinamento supervisionado de um *SVM*, que usa como entrada as representações dos atributos ativos para cada gabarito e um conjunto de atributos básicos. Yang and Eisenstein (2015) é o estado da arte na adaptação de domínio de etiquetagem morfossintática de uma das tarefas utilizadas neste trabalho.

O DLID, DAN e DATT foram baseados, respectivamente, em três trabalhos: Chopra et al. (2013), Ma et al. (2014), Ganin et al. (2016). Chopra et al. (2013) propuseram um método para adaptação de domínio de um problema de reconhecimento de objetos em imagens. O método de Chopra et al. (2013) aprende de forma não supervisionado as representações para o domínio origem, alvo e intermediário, utilizando um algoritmo chamado *Predictive Sparse Decomposition* (Jarrett et al., 2009). Para o treinamento supervisionado, eles criam uma única rede no qual os módulos, que geram as representações, são conectados a um classificador de regressão logística multinomial. Os parâmetros destes módulos são atualizados neste treinamento. Diferente de Chopra et al. (2013), o DLID utiliza uma rede para etiquetagem morfossintática. Além disso, utilizamos as representações distribuídas de palavras que são repassadas a uma camada oculta e, não, diretamente a camada responsável por classificar.

O método de Ma et al. (2014) utiliza uma rede neural, composta por dois módulos unidos por uma camada de saída. Um dos módulos, chamado de módulo dos atributos da *web*, é especializado em extrair os atributos relacionados ao domínio da internet. Este módulo é treinado de forma não supervisionada usando uma técnica de factorização do *Word Representation Restricted Boltzmann machine* (WRRBM). O outro módulo, nomeado de módulo dos atributos esparsos, recebe como entrada os atributos manuais e possui somente uma camada oculta. Este módulo juntamente com a camada de saída da rede neural são treinadas usando os dados anotados do domínio origem. Nesta fase, os parâmetros do modulo dos atributos da *web* não são atualizados. Existem duas diferenças principais entre Ma et al. (2014) e o DATT. A primeira ocorre no treinamento não supervisionado, no qual o DATT utiliza um método ba-

seado no word2vec, enquanto Ma et al. (2014) emprega um WRRBM. A outra diferença é que o DATT não utiliza nenhum atributo manual.

O método de Ganin et al. (2016), que baseamos para cria o DAN, tenta aprender atributos invariantes e discriminativos e um rotulador utilizando um único processo de treinamento. O modo como realiza isto é semelhante ao que descrito na Seção 4.2, no qual descremos sobre o DAN. Para validar o seu método, Ganin et al. (2016) o aplica nas tarefas de análise de sentimento e classificação de imagens. Na análise de sentimentos, ele utiliza uma rede simples, com somente uma camada oculta, Isto consiste na principal diferença ente os dois métodos. O DAN emprega uma rede profunda, que tem uma unidade de convolução e emprega dois tipos de representações distribuídas, no qual uma delas, normalmente, é pré-treinada. Em outras palavras, possui um treinamento mais complexo que o Ganin et al. (2016).

Como se pode observar, os estudos apresentados até este momento não utilizam dados anotados dos domínios alvo. Durante a pesquisa bibliográfica, foram encontrados dois trabalhos relevantes de adaptação de domínio supervisionada e semissupervisionada em PLN, que utilizam a abordagem focada na transformação do espaço de atributos. Daumé III (2007) considera que a distribuição de probabilidade de um domínio é formada pela mistura de duas distribuições: uma específica ao domínio e a outra refere-se ao domínio geral. Esta distribuição do domínio geral pode ser aplicada tanto no domínio origem quanto no domínio alvo. Dado esta suposição, este método tenta descobrir um componente comum aos dois domínios e dois componentes específicos a cada domínio (Jiang, 2008). Estes componentes são utilizados no treinamento de um modelo de entropia máxima e criados a partir dos dados anotados dos dois domínios. Daumé et al. (2010) estende este método permitindo também o uso de dados não anotados do domínio alvo.

Chen et al. (2011) utiliza *co-learning* para adaptação de domínio. O *co-learning* é uma técnica que usa um ou mais classificadores, no caso deste estudo foram empregados dois classificadores, que basicamente ensinam uns a outros a partir de seus erros. Na técnica original, cada classificador utiliza a uma visão. Estas visões devem ser condicionalmente independentes entre si e suficientes. Diferente do *co-learning* original, Chen et al. (2011) cria as visões a partir da separação do espaço de atributos em duas partes exclusivas. Os autores consideram que a necessidade das visões serem condicionalmente independentes é muito rígida e aliviam isto usando ε-*expandability*. O aprendizado da melhor forma de realizar a divisão dos espaço de atributos, respeitando as restrições da exclusividade e do ε-*expandability*, e o treinamento dos classificadores são realizados ao mesmo tempo utilizando dados anotados dos domínios fonte e alvo e dados não anotados dos domínios alvo.

Na Tabela 5.1, realizamos uma breve comparação entre os principais trabalhos descritos neste capítulo considerando o cenário de adaptação de domínio em que o método foi desenvolvido (supervisionado, semissupervisionado e não supervisionado); a técnica de aprendizado de representação empregada; e as tarefas em que o método foi aplicado. Além disto, para cada método, indicamos a ideia principal utilizada.

Método	**AD**	**AR não supervisionado**	**Ideia Geral**	**Tarefa(s)**
Blitzer et al. (2006)	Não supervisionada	*Structural Correspondence Learning* (SCL)	Aprendizado de representação com atributos pivôs	Etiquetagem morfossintática
Glorot et al. (2011)	Não supervisionada	*Stacked Denoising Autoencoder*	Aprendizado de representação utilizando *Stacked Denoising Autoencoder*	Análise de sentimentos
Chen et al. (2012)	Não supervisionada	*Marginalized Stacked Denoising Autoencoder*	Aprendizado representação utilizando *Marginalized Stacked Denoising Autoencoder*	Análise de sentimentos
Schnabel and Schütze (2014)	Não supervisionada	*TF-weighting* (Huang and Yates, 2009)	Atributos de distribuição para AD	Etiquetagem morfossintática
Yang and Eisenstein (2015)	Não supervisionada	Skip-gram	Representação distribuídas de atributos	Etiquetagem morfossintática
Chopra et al. (2013)	Não supervisonada e Semissupervisionada	*Predictive Sparse Decomposition* (Jarrett et al., 2009)	Aprendizado de representação para o domínios origem, alvo e intermediário	Reconhecimento de objetos em imagens
Ma et al. (2014)	Não supervisionada	Word Representation Restricted Boltzmann machine (WRRBM)	Módulo dos atributos esparsos e módulo dos atributos da web para AD	Etiquetagem morfossintática
Ganin et al. (2016)	Não supervisionada	-	Aprendizado de atributos invariantes e discriminativos e de um rotulador em um único processo de aprendizado.	Análise de sentimentos e classificação de imagens
Daumé et al. (2010)	Supervisionado e Semissupervisionada	-	Descobrir um componente comum aos dois domínios e dois componentes específicos a cada domínio	Etiquetagem morfossintática e reconhecimento de entidades nomeadas
Chen et al. (2011)	Semissupervisionada	-	Co-learning para AD	Análise de sentimentos

Tabela 5.1: Tabela comparativa dos trabalhos descritos neste capítulo.

CAPÍTULO

6

Avaliação Experimental

Neste capítulo, são apresentados e discutidos os experimentos realizados com o intuito de avaliar os três métodos de adaptação de domínio desenvolvidos neste trabalho. Comparamos os desempenhos destes métodos com algumas versões do CharWNN original e também com sistemas estado da arte no problema de adaptação de domínio. Adicionalmente, realizamos um estudo que compara as representações de caracteres aprendidas pelo CharWNN e os atributos manuais clássicos (sufixo e capitalização) na adaptação de domínio. Ainda apresentamos uma análise dos erros do modelo CharWNN em comparação com os atributos manuais clássicos. Por fim, apresentamos um estudo onde o CharWNN é treinado com uma pequena quantidade de frases anotadas do domínio alvo e seu desempenho supera os melhores sistemas de adaptação do domínio da literatura. Isto demonstra que o problema de adaptação de domínio não supervisionada é difícil e ainda mal resolvido.

Nestes experimentos, foram empregadas duas tarefas de adaptação de domínio do problema de etiquetagem morfossintática. As características destas tarefas e os dados usados são descritos a seguir.

6.1 Tarefas e Corpora

O corpus Histórico do Português Tycho Brahe (Galves and Faria, 2010) é um corpus formado por obras literárias de autores nascidos de 1380 a 1881. Este corpus contém quase 1,5 milhão de palavras anotadas manualmente com 341 etiquetas morfossintáticas diferentes. A distribuição das 50 etiquetas mais frequentes neste corpus é apresentada na Tabela 6.1. Seguindo a metodologia apresentada em Yang and Eisenstein (2015), dividimos este corpus em

períodos de 50 anos, resultando em sete domínios no final, e definimos duas tarefas de adaptação de domínio. Na primeira tarefa, o período 1800-1849 é o domínio origem e outros seis períodos são os domínios alvo. Na segunda tarefa, o domínio origem consiste no período 1750-1849, que é formado pela união dos períodos 1750-1799 e 1800-1849, e outros cinco períodos são os domínios alvo. As Estatísticas básicas de cada um dos sete períodos são apresentada na Tabela 6.2.

Etiqueta	Porcentagem
N	10,20
P	8,95
,	8,05
NPR	5,42
.	4,95
CONJ	4,55
N-P	4,04
VB-P	3,00
D	2,69
ADV	2,61
WPRO	2,56
VB-D	2,50
D-F	2,29

Etiqueta	Porcentagem
VB	2,15
C	1,69
P+D	1,65
CL	1,64
P+D-F	1,54
NEG	1,20
ADJ	1,18
D-P	1,10
SR-P	1,02
CONJS	0,99
ADJ-G	0,98
PRO	0,94
ADV-R	0,89

Etiqueta	Porcentagem
SE	0,89
PRO$-F	0,89
ADJ-F	0,89
D-F-P	0,77
FW	0,75
Q	0,71
NUM	0,65
P+D-P	0,58
VB-AN	0,55
PRO$	0,55
D-UM	0,52
VB-G	0,51
VB-SP	0,51

Etiqueta	Porcentagem
(	0,48
NPR-P	0,45
ADJ-P	0,43
P+D-F-P	0,42
SR-D	0,41
D-UM-F	0,40
Q-P	0,39
ADJ-G-P	0,35
VB-R	0,34
TR-P	0,34
WADV	0,32

Tabela 6.1: Distribuição das 50 etiquetas que mais aparecem no corpus Tycho Brahe.

	Tipo de Texto				
Período	**Narrativo**	**Carta**	**Dissertação**	**Teatro**	**Total**
1800-1849	91.582	34.137	0	0	125.719
1750-1799	57.477	84.465	0	60.404	202.346
1700-1749	0	130.327	148.519	0	278.846
1650-1699	83.938	115.062	49.194	0	248.194
1600-1649	117.515	115.252	62.387	0	295.154
1550-1599	148.061	0	0	0	148.061
1500-1549	126.516	0	55.692	0	182.208
Total	625.089	479.243	315.792	60.404	1.480.528

Tabela 6.2: Estatísticas do corpus Tycho Brahe.

A tarefa conjunta proposta na conferência SANCL-2012 (Petrov and McDonald, 2012) consiste em resolver a análise sintática de constituintes e de dependência em domínios da internet, usando dados anotados provenientes de textos jornalísticos. Além das anotações de árvores sintáticas, estes corpora também incluem etiquetas morfossintáticas; informação que é utilizada nos experimentos deste trabalho. A tarefa SANCL, como será denominada aqui, engloba seis domínios. Nesta tarefa, o domínio origem é composto por documentos jornalísticos advindos das seções 2 a 21 do *Wall Street Journal* (WSJ) do Ontonotes 4.0. Os dados deste domínio são subdivididos em três partes: treinamento, desenvolvimento e teste. Os outros cinco domínios são os domínios alvo e compreendem textos advindos de diferentes fontes da internet, a

saber: Newsgroups, Weblogs, Reviews, Answers, Emails. Cada domínio é subdividido em três partes: um conjunto grande de textos não anotados e dois pequenos conjuntos de desenvolvimento e teste contendo dados anotados.

O objetivo da tarefa SANCL é treinar um único modelo a ser aplicado nos conjuntos de teste dos domínio Answers, Newsgroups e Reviews. O treinamento deste modelo pode usar somente os dados anotados do WSJ e os dados não anotados dos domínios alvo. Os conjuntos de desenvolvimento dos domínios Email e Weblogs são usados para calibração de hiper-parâmetros. Na Tabela 6.3, são apresentados o número de *tokens* e de frases não anotados antes e depois do pré-processamento dos dados de cada um dos 6 domínios. Na Tabela 6.4, são mostrados as estatísticas dos dados anotados dos corpora SANCL. Estes corpora compreendem 50 etiquetas morfossintáticas. As distribuições destas etiquetas na união de todos os domínios alvo (WEB) e no domínio origem (WSJ) são apresentadas na Tabela 6.5.

	Antes do pré-processamento		**Depois do pré-processamento**	
	Nº de frases	**Nº de tokens**	**Nº de frases**	**Nº de frases**
WSJ	100.000	2.453.182	99.123	2.449.911
Answers	27.274	424.188	23.104	409.711
Emails	1.194.173	17.046.119	853.972	16.248.131
NewsGroups	1.000.000	18.423.581	992.723	18.376.299
Reviews	1.965.350	29.288.941	1.725.423	28.474.640
Weblogs	524.834	10.261.131	452.936	9.862.459

Tabela 6.3: Estatísticas dos dados não anotados dos corpora SANCL.

	Treinamento		**Desenvolvimento**		**Teste**	
	Nº frases	**Nº tokens**	**Nº frases**	**Nº tokens**	**Nº frases**	**Nº frases**
WSJ	30.060	731.678	1.336	32.092	1.640	39.590
Answers	-	-	1.745	25.180	1.744	28.823
Emails	-	-	2.450	29.131	2.450	28.676
NewsGroups	-	-	1.196	22.398	1.195	20.651
Reviews	-	-	1.907	27.504	1.906	28.086
Weblogs	-	-	1.016	24.025	1.015	20.356

Tabela 6.4: Estatísticas dos dados anotados dos corpora SANCL.

6.2 Pré-Processamento de Dados

Todos os dados não anotados da Tycho Brahe passaram por um pré-processamento similar ao realizado por Santos and Zadrozny (2014) que é composto por três passos:

1. frases compostas por menos que 20 caracteres (incluindo espaços em branco) ou menos que cinco *tokens* são removidas;

Etiqueta	WEB (%)	WSJ (%)
NN	13,0106	14,4045
IN	9,9927	10,9743
DT	8,1199	8,5317
NNP	6,1148	9,5565
PRP	5,9916	1,9076
JJ	5,8868	6,1356
RB	5,3291	3,3961
.	5,1570	4,0729
VB	4,7638	2,8280
NNS	4,1472	6,2804
,	4,0033	4,9273
CC	3,3561	2,4838
VBP	2,6799	1,3936
VBD	2,4488	2,8872
VBZ	2,3157	2,3554
VBN	1,9646	2,2468
CD	1,8261	3,1803

Etiqueta	WEB (%)	WSJ (%)
MD	1,6915	1,0441
TO	1,6425	1,4488
VBG	1,6379	1,6081
PRP$	1,5362	0,8608
-RRB-	0,4720	0,1515
-LRB-	0,4598	0,1500
WDT	0,4330	0,4769
WRB	0,4055	0,2406
RP	0,3672	0,3038
“	0,3611	0,7796
WP	0,3573	0,2699
’	0,3550	0,7573
UH	0,3374	0,0131
POS	0,3114	0,8720
:	0,2999	0,5376
HYPH	0,2961	1,1655
JJR	0,2257	0,3484

Etiqueta (%)	WEB (%)	WSJ (%)
NNPS	0,2196	0,2737
NFP	0,2111	0,0017
JJS	0,2027	0,1924
EX	0,1721	0,1012
ADD	0,1438	0,0000
RBR	0,1400	0,1815
GW	0,1362	0,0000
$	0,0949	0,4578
PDT	0,0910	0,0476
SYM	0,0819	0,0309
RBS	0,0788	0,0690
FW	0,0581	0,0268
LS	0,0451	0,0088
AFX	0,0199	0,0004
WP$	0,0054	0,0182
XX	0,0023	0,0001

Tabela 6.5: Distribuição dos rótulos no conjunto de dados da Tarefa SANCL

2. todas as letras são convertidas para minúsculas[1];

3. e todos os dígitos são substituídos por zero.

Além da execução dos passos listados na tarefa Tycho Brahe, nós pré-processamos os dados não anotados da tarefa SANCL usando os seguintes passos:

1. aspas neutras são transformadas em aspas abertas ou fechadas;

2. *tokens* começando com www e http ou terminando com .org e .com são substituídos pelo símbolo #URL;

3. pontuações repetidas (!!! ou ???) são removidas;

4. colchetes esquerdos (<, { e () são convertidos para -LRB-. Analogamente, colchetes direitos são convertidos para -RRB-;

5. palavras contendo mais de quatro letras maiúsculas tem todas as suas letras convertidas para minúsculas;

6. dígitos consecutivos dentro de uma palavra são substituídos por #DIG.

Em razão dos textos advindos da internet possuírem muitos ruídos como erros de escrita, *emoticons* e inconsistência na capitalização, também usamos os seis passos acima para pré-processar os dados anotados. Segundo (Ma et al., 2014), este pré-processamento melhora significativamente os resultados dos modelos. Os dois conjuntos de filtros descritos acima são aplicados nos dados

[1]Este filtro é aplicado somente à entrada da camada de representação de palavras. Ele não é aplicado à entrada do módulo de convolução de caracteres, pois este módulo se preocupa com a morfologia das palavras.

não anotados que serão usados pelo *word2vec* e pelo extrator de atributos do domínio alvo no DATT. Nos dados não anotados usados pela rede neural DAN serão aplicados somente os seis passos definidos na SANCL, não utilizando assim os definidos em dos Santos e Bianca Zadrozny (2014b).

6.3 Calibração de Hiper-Parâmetros

Seguindo Yang and Eisenstein (2015), na tarefa Tycho Brahe, o ajuste dos hiper-parâmetros é realizado com 5% dos dados do domínio origem. Já na tarefa SANCL, seguimos a tarefa original (Petrov and McDonald, 2012) e usamos dois domínios alvo (Emails e Weblogs) para ajuste de hiper-parâmetros e seleção de modelos. Nesta última tarefa, usamos a média das acurácias nestes dois domínios como critério de seleção dos melhores modelos.

Para a calibração dos hiper-parâmetros dos algoritmos e métodos, usamos a busca aleatória (*random search*) (Bergstra and Bengio, 2012). Este método consiste em executar N testes, escolhendo aleatoriamente os valores dos hiper-parâmetros em cada teste. O valor de N pode ser regulado de acordo com a quantidade de hiper-parâmetros, o tamanho do domínio dos hiper-parâmetros e o tempo e o poder computacional disponível. Segundo Bergstra and Bengio (2012), a busca aleatória é mais eficiente do que a tradicional busca em grade (*grid search*). Os valores dos hiper-parâmetros foram gerados de acordo com sua natureza, como descrito abaixo.

- Valores inteiros foram gerados por uma distribuição uniforme em um intervalo determinado.
- Valores reais foram gerados por uma distribuição exponencial, sendo que o expoente foi limitado a um intervalo determinado.

Na Tabela 6.6, são mostradas as distribuições e os intervalos usadas para gerar os valores dos hiper-parâmetros.

Tanto na tarefa SANCL quanto na Tycho Brahe, utilizamos os mesmos valores para os hiper-parâmetros no *word2vec* no CharWNN e no treinamento não supervisionado do DATT. Tais hiper-parâmetros são apresentados, respectivamente, nas Tabelas 6.7, 6.8 e 6.9. Vale ressaltar que os valores dos hiper-parâmetros apresentados nas Tabelas 6.7 e 6.8 são idênticos ao utilizado por dos Santos e Bianca Zadrozny (2014b). A única exceção é o tamanho da camada oculta do CharWNN, no qual utilizamos 100 em vez de 300 como valor para este hiper-parâmetro. Tal redução do número de neurônios na camada oculta não afetou o desempenho do CharWNN e tornou mais rápido o seu treinamento. Outro ponto importante a ser mencionado é que devido à quantidade de dados não anotados na Tycho Brahe, houve a necessidade de

Hiper-parâmetros	**Algoritmos**	**Distribuição**	**Intervalo**
Tamanho da janela	DATT	Uniforme	$[3;7]$
Número de épocas	DATT	Uniforme	$[3;10]$
Mínimo número de vezes que a palavra deve aparecer para ser colocada no dicionário	DATT	Uniforme	$[2;10]$
Tamanho da representação distribuída de palavras	DATT	Uniforme	$[75;300]$
Número de palavras ruídosas usados na amostragem negativa	DATT	Uniforme	$[2;8]$
Taxa de aprendizado	DATT	Exponencial	$[1\times10^{-4};1]$
Tamanho da camada oculta	DATT	Uniforme	$[100;300]$
Subamostragem (t)	DATT	Exponencial	$[1\times10^{-6};0,1]$
λ_{DAN}	DAN	Exponencial	$[1\times10^{-6};1]$
Taxa de aprendizado	Todos	Exponencial	$[1\times10^{-4},1]$

Tabela 6.6: Distribuição e intervalos usados nos hiper-parâmetros.

reduzir de 10 para 5 o número de vezes que uma palavra deve aparecer para ser incluída no dicionário do *word2vec*.

Hiper-parâmetro	**Valor**
Tamanho da janela	5
Número de iterações	5
Mínimo número de vezes que a palavra deve aparecer para ser colocada no dicionário	10
Tamanho da representação distribuída de palavras	100
Modelo	Skip-Gram
Número de palavras ruídosas usados na amostragem negativa	5

Tabela 6.7: Hiper-parâmetros do *word2vec*.

Hiper-parâmetro	**Valor**
Tamanho do mini-batch	1
Tamanho da janela de caracteres	5
Número de filtros na convolução	50
Tamanho da camada oculta	100
Tamanho da janela de palavras	5
Tamanho da representação distribuída de caracteres	10

Tabela 6.8: Hiper-parâmetros do CharWNN.

Os valores das taxas de aprendizado usados em todos os métodos são apresentados na Tabela 6.10. No método DAN, o hiper-parâmetro λ_d também foi ajustado, sendo $1,888\times10^{-3}$, 5×10^{-5} e $2,8\times10^{-4}$ os melhores valores encontrados para este parâmetro na tarefa SANCL e nas duas tarefas Tycho Brahe, respectivamente.

Hiper-parâmetro	**Valor**
Tamanho da janela	5
Número de épocas	5
Mínimo número de vezes que a palavra deve aparecer para ser colocada no dicionário	4
Tamanho da representação distribuída de palavras	100
Número de palavras ruídosas usados na amostragem negativa	5
Taxa de aprendizado	0,01
Tamanho da camada oculta	300
Subamostragem (t)	1×10^{-5}

Tabela 6.9: Hiper-parâmetros do treinamento não supervisionado do DATT.

Método	**SANCL**	**TB (1800-1849)**	**TB (1750-1849)**
CharWNN	0,0075	0,0100	0,0100
WNN+Ftrs	0,0075	0,0250	0,0100
WNN	0,0050	0,0250	0,0250
DLID/PLN	0,0075	0,0250	-
DLID/PLN-M	0,0075	0,0250	-
DATT	0,0075	0,0250	0,0100
DAN	0,0072	0,0182	0,0088

Tabela 6.10: Taxa de aprendizado de todos os métodos nas três tarefas.

Os experimentos foram realizados em um cluster com 108 processadores do modelo Intel(R) Xeon(R) E5-2620 e 544 GB (gigabytes) de memória RAM.

6.4 Baselines

Visto que os métodos DLID, DAN e DATT utilizam redes neurais baseadas no CharWNN, então o *baseline* mais adequado para estes métodos é o CharWNN original. Uma característica do CharWNN é que as representações distribuídas de palavras podem ser pré-treinadas em qualquer conjunto de dados. Neste trabalho, utilizamos o *word2vec* para realizar este pré-treinamento em diversos conjuntos de dados. Uma opção comum é utilizar os artigos da Wikipedia para este fim, pois esta é uma fonte multilíngue com uma grande quantidade de documentos. Nós usaremos diferentes inicializações para as representações de palavras do CharWNN. Para mostrar como os parâmetros de uma camada LT^w foram inicializados, utilizamos os seguintes valores: RAND, ALVO e WIKI.

A opção RAND indica que as representações foram geradas de forma aleatória. Denominaremos um modelo CharWNN treinado com esta inicialização como CharWNN/RAND e este é o baseline mais justo pois não utiliza nenhum dado do domínio alvo. A opção ALVO assinala que as representações foram

pré-treinadas pelo *word2vec* utilizando os dados dos domínios alvo. No caso da SANCL, são usados os dados não anotados de todos os domínios alvo. No caso da Tycho Brahe são utilizados todos os dados do corpus, pois os domínios deste corpus são pequenos. A opção WIKI significa que foram empregados dados da Wikipedia e de outros corpora. No caso da SANCL, usamos as mesmas representações empregadas em Pennington et al. (2014)[2], que foram pré-treinadas por uma aplicação chamada GLoVe usando dados da Wikipedia[3] e do Gigaword[4]. Na Tycho Brahe, utilizamos as representações de dos Santos e Bianca Zadrozny (2014b)[5], que foram pré-treinadas pelo *word2vec* usando dados da Wikipedia em português e dos corpora CETENFolha[6] e CETEMPúblico[7]. Para indicar no texto a forma como foram inicializadas as representações distribuídas de palavras em um método, o nome do método será concatenado ao símbolo / e o nome da opção. Por exemplo, WNN/RAND indica que os parâmetros da camada LT^w do WNN foram gerados aleatoriamente.

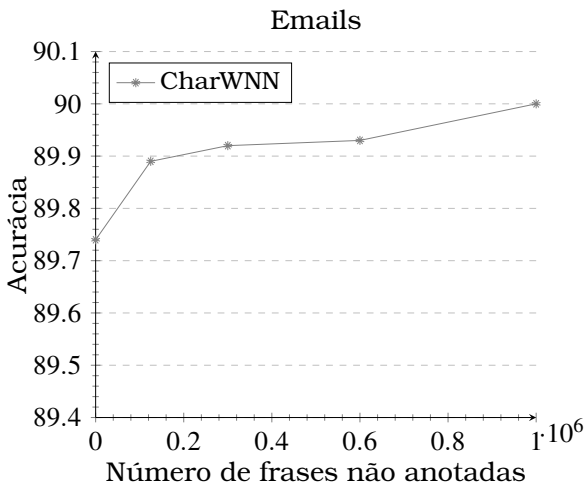
6.5 Treinamento Não Supervisionado

Os três métodos de AD desenvolvidos neste trabalho utilizam dados não anotados do domínio alvo. Entretanto, por questões práticas e por características das tarefas de AD, existem diferenças na maneira que cada método utiliza os dados não anotados. Na tarefa Tycho Brahe, os dados não anotados são gerados a partir da retirada dos rótulos dos dados anotados dos domínios origem e alvo. Este não é o cenário ideal para adaptação de domínio não supervisionada, pois estes conjuntos são pequenos para aprendizado não supervisionado. Na maioria dos domínios, textos não anotados estão disponíveis em abundância. Entretanto, neste caso, que compreende obras literárias antigas, é difícil obter muitos textos. Por outro lado, na tarefa SANCL, há uma grande quantidade de dados não anotados para cada domínio alvo, o que constitui um cenário mais adequado e aderente para a AD não supervisionada.

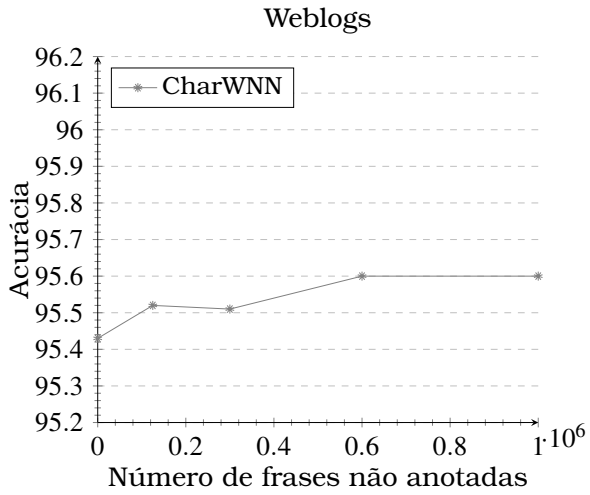
O DAN seleciona aleatoriamente 40 mil frases não anotadas dos domínios alvo da tarefa SANCL, devido a limitações computacionais. O DATT também não utiliza todos os dados não anotados na SANCL, devido ao tempo de processamento necessário para isto. Para determinar uma quantidade adequada de frases não anotadas para estes métodos, foram realizados diversos experimentos nos domínios Emails e Weblogs usando diferentes quantidades de frases. Como se pode observar nas Figuras 6.1a e 6.1b, para o domínio We-

[2]http://nlp.stanford.edu/projects/glove/
[3]https://dumps.wikimedia.org/enwiki/
[4]https://catalog.ldc.upenn.edu/LDC2011T07
[5]As representações distribuídas foram cedidas pelo autores.
[6]http://www.linguateca.pt/cetenfolha/
[7]http://www.linguateca.pt/CETEMPublico/

blogs, a utilização de 600 mil frases se mostra suficiente. Entretanto, no domínio Emails, houve um pequeno aumento no desempenho quando se utilizou um milhão de frases. Desta forma, decidiu-se utilizar um milhão de frases no DATT para a tarefa SANCL. Já na tarefa Tycho Brahe, tanto DATT quanto DAN utilizam todos os dados não anotados.



(a) Acurácia do DATT no Emails.



(b) Acurácia do DATT no Weblogs.

O DLID é empregado de forma diferente nas duas tarefas. Na Tycho Brahe, treinamos as representações distribuída de palavras para cada domínio alvo e intermediário. Assim, por exemplo, para testar este método no período 1700-1749 usando como treinamento o período 1800-1849, são empregadas as representação distribuídas de palavras de 1700-1750, de 1800-1849 e do domínio intermediário entre 1700-1750 e 1800-1849. Porém, isto não é possível de ser realizado na SANCL, em razão da exigência de que somente um modelo deve ser testado nos domínios. Portanto, neste caso, treinamos somente uma representação distribuídas de palavras para o domínio alvo usando todos os dados não anotado dos domínios da internet. Em ambas tarefas, os conjunto de dados não anotados dos domínios intermediário são criados a partir da seleção de um mesmo número de frases no domínio alvo e origem.

6.6 Avaliação dos Métodos de AD

Os métodos avaliados neste capítulo possuem componentes aleatórios como: a inicialização dos pesos das redes neurais, a seleção de um conjunto de frases dos dados não anotados, o treinamento das representações distribuídas de palavras, e o próprio SGD. Para verificar a estabilidade dos métodos diante destes componentes aleatórios, são realizadas três execuções de cada experimento nos conjuntos de teste. Assim, nas tabelas de resultados deste capítulo, apresentamos a acurácia média das três execuções e o erro padrão

da média (entre colchetes).

Na Tabela 6.11, apresentamos os desempenhos dos três métodos de adaptação de domínio na tarefa SANCL. Esta tabela também inclui os desempenhos de dois baselines: CharWNN/RAND e CharWNN/ALVO. Estes dois baselines usam a rede CharWNN mas, como apresentado na seção 6.4, a inicialização das representações distribuídas de palavras varia de um para outro. Os três métodos de AD apresentam ganhos significativos em relação ao *baseline* CharWNN/RAND. Entretanto, o baseline CharWNN/ALVO, que utiliza os dados dos domínios alvo, obtém acurácias semelhantes (até um pouco superiores) às obtidas pelos métodos de adaptação de domínio. Portanto, podemos concluir que os ganhos dos métodos DLID, DAN e DATT sobre o *baseline* são devidos ao uso de representações distribuídas de palavras melhores e não a fatores inerentes aos métodos.

	Answer	**Newsgroup**	**Reviews**	**WSJ-test**	**Média**
CharWNN/RAND	91,21 [0,0341]	92,75 [0,0791]	92,04 [0,0750]	97,20 [0,0316]	92,00
DLID	91,56 [0,0744]	93,34 [0,0689]	92,97 [0,0402]	97,33 [0,0329]	92,62
DAN	91,79 [0,0579]	93,37 [0,0628]	92,98 [0,0191]	97,28 [0,0232]	92,71
DATT	91,79 [0,0535]	93,39 [0,0602]	92,97 [0,1106]	97,25 [0,0482]	92,71
CharWNN/ALVO	91,80 [0,0212]	93,47 [0,0354]	93,14 [0,0338]	97,39 [0,0325]	92,80

Tabela 6.11: Comparação das acurácias do CharWNN, DLID, DAN e DATT na tarefa SANCL.

Comparando os métodos DLID e DLID-M, não é possível concluir muito. Mesmo que as acurácias destes métodos sejam bem próximas, não é possível saber se isto ocorre devido à semelhança entre as representações do domínio intermediário. Este resultado pode ser simplesmente porque o CharWNN funciona bem nos dois casos, pois os desempenhos do DLID e do DLID-M são semelhantes ao do CharWNN/ALVO.

Com relação às duas tarefas Tycho Brahe, as acurácias obtidas pelo DAN e DATT são exibidas nas Tabelas 6.12 e 6.13. Do mesmo modo que na SANCL, o CharWNN teve um melhor desempenho quando utilizou-se uma representação distribuída de palavras não iniciada aleatoriamente. O CharWNN/RAND alcançou uma acurácia média 2,26 e 0,97 inferior que o CharWNN/ALVO treinado, respectivamente, no período 1800-1849 e 1750-1849. Assim, para realmente descobrir se os métodos de adaptação de domínio realmente funcionaram, eles serão comparados com o CharWNN/ALVO.

Na primeira tabela, o DAN possui uma acurácia média 0,19 maior que a do CharWNN/ALVO, porém, devido aos valores relativamente altos dos erros padrão do DAN, não temos indícios se o uso deste método trouxe ou não um ganho significativo. Por outro lado, podemos considerar que o desempenho do DATT e CharWNN/ALVO são similares. A segunda tabela mostra um mesmo cenário encontrado na SANCL, no qual o uso dos métodos parece não ocasi-

	1500-1549	1550-1599	1600-1649	1650-1699	1700-1749	1750-1799	Média
CharWNN/RAND	84,30 [0,1626]	88,20 [0,1403]	85,02 [0,2965]	85,77 [0,1120]	88,87 [0,1143]	87,32 [0,1372]	86,58
CharWNN/ALVO	86,87 [0,1055]	90,08 [0,0318]	88,47 [0,0749]	87,54 [0,0521]	90,81 [0,1321]	89,28 [0,0980]	88,84
Baseline DLID	73,98 [0,1031]	86,03 [0,1400]	82,67 [0,1365]	83,89 [0,1119]	87,16 [0,0999]	84,73 [0,0669]	83,07
DAN	86,88 [0,2122]	90,40 [0,1452]	88,83 [0,2105]	87,42 [0,5729]	91,23 [0,2097]	89,42 [0,0915]	89,03
DATT	86,70 [0,0779]	90,16 [0,0087]	88,42 [0,0601]	87,36 [0,8116]	91,10 [0,0415]	89,43 [0,0560]	88,86
DLID	75,13 [0,1441]	86,31 [0,2022]	83,81 [0,3645]	83,51 [0,1263]	86,90 [0,2410]	84,87 [0,0756]	83,42

Tabela 6.12: Comparação das acurácias do CharWNN, DAN e DATT na tarefa Tycho Brahe quando treinado no período 1800-1849.

	1500-1549	1550-1599	1600-1649	1650-1699	1700-1749	Média
CharWNN/RAND	89,48 [0,0084]	92,77 [0,1036]	91,69 [0,1618]	89,98 [0,1898]	94,31 [0,1236]	91,65
CharWNN/ALVO	90,51 [0,0678]	93,49 [0,0887]	93,34 [0,0382]	90,80 [0,0087]	94,94 [0,0514]	92,62
DAN	90,47 [0,0676]	93,45 [0,0822]	93,19 [0,0289]	90,53 [0,0653]	94,90 [0,0544]	92,51
DATT	90,57 [0,0156]	93,52 [0,0098]	93,19 [0,0494]	90,82 [0,0128]	94,98 [0,0399]	92,62

Tabela 6.13: Comparação das acurácias do CharWNN, DAN e DATT na tarefa Tycho Brahe quando treinado no período 1750-1849.

onar nenhum aumento no desempenho, podendo até ser pernicioso algumas vezes. Pode-se notar na Tabela 6.12 que os valores da acurácia do *baseline* do DLID estão bem abaixo das acurácias conseguidas pelo *baseline* de Yang and Eisenstein (2015). O que é estranho, pois o CharWNN/ALVO consegue acurácias próximas do obtido pelo Yang and Eisenstein (2015).

A única diferença entre CharWNN/ALVO e o *baseline* do DLID é a quantidade de exemplos não anotados usados no Word2Vec, no qual o primeiro utiliza todos os dados não anotados disponíveis na Tycho Brahe, aproximadamente 1.432.000 *tokens*, enquanto o segundo emprega somente os dados não anotados do período 1800-1849, que são em torno de 128.000 *tokens*. Assim, podemos concluir que a quantidade de dados em cada um dos domínios Tycho Brahe não é o bastante para o *word2vec* aprender boas representações. Esta necessidade é um fator conhecido dos métodos de aprendizado não supervisionado de representações. Deste modo, o DLID só faz sentido em ambientes que tenham uma quantidade razoável de dados não anotados nos domínios origem e alvo. Tornando assim inviável a comparação do DLID com Yang and Eisenstein (2015) neste cenário. Pelas razões citadas acimas, também não foi possível testar o DLID-M no Tycho Brahe.

Tanto na tarefa SANCL quanto na Tycho Brahe, o DLID, DAN e DATT não aumentaram o desempenho em relação ao CharWNN, e até alguns casos foram prejudiciais. Acreditamos que os métodos não funcionaram devido a dificuldade da tarefa. Provavelmente, se dedicássemos mais tempo para investigar, poderíamos fazê-los funcionar, mesmo que marginalmente. Mas, como demonstrado adiante, mesmo o estado da arte não apresenta uma melhora expressiva.

6.7 CharWNN na Adaptação de Domínio

O CharWNN é uma rede profunda e, por isto, é capaz de aprender automaticamente representações dos dados. Devido a esta capacidade, gostaríamos de compreender se os atributos aprendidos conseguem ser eficazes em um cenário de adaptação de domínio. Assim, para isto, iremos comparar os resultados obtidos pelo CharWNN com os sistemas estado da arte, o WNN e o WNN com sufixo e capitalização, denominado de WNN+Ftrs.

Da mesma forma que Santos and Zadrozny (2014), os atributos de capitalização no WNN+Ftrs têm 5 possíveis valores: (i) todas letras estão em minúsculo, (ii) somente a primeira letra é maiúscula, (iii) todas as letras são maiúsculas, (iv) contém uma letra maiúscula, e (v) outros casos. Nós usamos o tamanho do sufixo igual a dois para as tarefas em inglês e três para as tarefas em português. Para ambos atributos, as representações distribuídas tem 5 dimensões.

As acurácias do CharWNN na tarefa SANCL e de vários outros sistemas estado da arte neste ambiente são mostradas na Tabela 6.14, que tem na sua última coluna a média da acurácia somente dos domínios alvo. Como pode-se observar, o CharWNN/ALVO possui um desempenho superior aos sistemas de Yang and Eisenstein (2015) e Tang et al. (2012). Entretanto, o CharWNN/ALVO tem um desempenho ligeiramente abaixo do atingido por Le Roux et al. (2012) - as acurácias destes dois métodos são similares nos domínios Answers e Reviews - e inferior ao obtido por Ma et al. (2014). Vale dizer que, apesar do CharWNN/ALVO ser superado por Le Roux et al. (2012) e Ma et al. (2014), seus resultados são expressivos (somente 0,35 abaixo da média do melhor sistema), já que não utiliza atributos manuais e emprega os dados não anotados de modo simples e eficaz.

	Answer	**Newsgroup**	**Reviews**	**WSJ-test**	**Média**
WNN/RAND	89,04 [0,1795]	90,07 [0,0271]	90,78 [0,0865]	95,58 [0,0185]	89,96
WNN/WIKI	90,15 [0,0769]	91,51 [0,0331]	91,82 [0,0208]	96,28 [0,0204]	91,16
WNN/ALVO	90,81 [0,0416]	91,99 [0,0547]	92,85 [0,0184]	96,12 [0,0565]	91,88
WNN+Ftrs/RAND	91,03 [0,1389]	92,76 [0,0650]	92,05 [0,0524]	97,10 [0,0295]	91,95
WNN+Ftrs/WIKI	91,20 [0,0807]	93,08 [0,0507]	92,41 [0,1189]	97,35 [0,0489]	92,23
WNN+Ftrs/ALVO	91,74 [0,0699]	93,44 [0,0339]	93,06 [0,0504]	97,27 [0,0168]	92,75
CharWNN/RAND	91,21 [0,0341]	92,75 [0,0791]	92,04 [0,0750]	97,20 [0,0316]	92,00
CharWNN/WIKI	91,39 [0,0446]	93,25 [0,0434]	92,49 [0,0832]	97,38 [0,0419]	92,38
CharWNN/ALVO	91,80 [0,0212]	93,47 [0,0354]	93,14 [0,0338]	97,39 [0,0325]	92,80
Yang and Eisenstein (2015)	91,35	92,60	92,15	–	92,03
Tang et al. (2012)	91,76	92,91	91,94	97,49	92,20
Le Roux et al. (2012)	91,79	93,81	93,11	97,29	92,90
Ma et al. (2014)	92,37	93,59	93,62	97,44	93,15

Tabela 6.14: Comparação do WNN, WNN+Ftrs e CharWNN com sistemas estado na arte na tarefa SANCL.

Nas Tabelas 6.15 e 6.16, nós comparamos o CharWNN com o melhor sistema conhecido na Tycho Brahe. Pode-se notar que o CharWNN/ALVO não

possui acurácias tão abaixo dos valores obtidos por Yang and Eisenstein (2015) nas duas tabelas. A diferença na acurácia média entre os dois é, respectivamente, de 1,00 e 0,94 quando é empregado, como domínio fonte, o período 1800-1849 e 1750-1849. Diferentemente dos sistemas estados da arte na SANCL e na Tycho Brahe, o CharWNN emprega os dados não anotados de modo simples e eficaz e não utiliza nenhum atributo manual. Mesmo assim, consegue atingir acurácias próximas, principalmente na tarefa SANCL, aos obtidos por estes sistemas.

	1500-1549	**1550-1599**	**1600-1649**	**1650-1699**	**1700-1749**	**1750-1799**	**Média**
WNN/RAND	77,17 [0,1808]	81,81 [0,1061]	78,71 [0,2260]	79,66 [0,5329]	83,76 [0,2480]	82,29 [0,1461]	80,57
WNN/WIKI	81,12 [0,0675]	86,70 [0,0309]	84,37 [0,0399]	83,42 [0,2531]	89,21 [0,0310]	86,48 [0,0438]	85,21
WNN/ALVO	80,55 [0,0852]	84,95 [0,0441]	83,35 [0,0527]	82,78 [0,2955]	86,14 [0,0299]	84,28 [0,0757]	83,68
WNN+Ftrs/RANDOM	84,29 [0,1230]	88,75 [0,0459]	85,83 [0,1959]	86,25 [0,0524]	89,99 [0,0425]	87,98 [0,0128]	87,18
WNN+Ftrs/WIKI	84,88 [0,0380]	89,95 [0,0110]	87,29 [0,0090]	86,66 [0,0934]	91,74 [0,0312]	89,15 [0,0334]	88,28
WNN+Ftrs/ALVO	86,22 [0,1051]	89,97 [0,0639]	88,23 [0,1547]	87,90 [0,4093]	91,00 [0,1331]	89,09 [0,0600]	88,74
CharWNN/RAND	84,30 [0,1626]	88,20 [0,1403]	85,02 [0,2965]	85,77 [0,1120]	88,87 [0,1143]	87,32 [0,1372]	86,58
CharWNN/WIKI	85,59 [0,0285]	90,14 [0,0122]	88,05 [0,0509]	86,73 [0,0375]	91,74 [0,0839]	89,39 [0,0336]	88,61
CharWNN/ALVO	86,87 [0,1055]	90,08 [0,0318]	88,47 [0,0749]	87,54 [0,0521]	90,81 [0,1321]	89,28 [0,0980]	88,84
Yang and Eisenstein (2015)	89,29	91,47	89,39	87,64	91,61	90,25	89,94

Tabela 6.15: Comparação do WNN, CharWNN e WNN+Ftrs com o sistema estado da arte na tarefa Tycho Brahe quando Treinado no Período 1800-1849.

	1500-1549	**1550-1599**	**1600-1649**	**1650-1699**	**1700-1749**	**Média**
WNN/RAND	83,88 [0,2098]	87,90 [0,1160]	87,10 [0,2213]	87,10 [0,2213]	90,05 [0,0836]	87,20
WNN/WIKI	85,32 [0,0162]	89,90 [0,0383]	89,39 [0,0543]	87,74 [0,0511]	92,57 [0,0159]	88,99
WNN/ALVO	85,24 [0,0289]	88,90 [0,0465]	88,80 [0,0254]	86,93 [0,0579]	90,41 [0,0293]	88,05
WNN+Ftrs/RAND	89,02 [0,0757]	92,59 [0,0467]	91,47 [0,1416]	89,61 [0,0710]	94,02 [0,0137]	91,34
WNN+Ftrs/WIKI	88,98 [0,0487]	92,96 [0,0161]	92,19 [0,0025]	91,73 [0,0393]	95,04 [0,0148]	92,18
WNN+Ftrs/ALVO	89,91 [0,0833]	93,17 [0,0987]	92,96 [0,0250]	90,84 [0,3240]	94,63 [0,0373]	92,30
CharWNN/RAND	89,48 [0,0084]	92,77 [0,1036]	91,69 [0,1618]	89,98 [0,1898]	94,31 [0,1236]	91,65
CharWNN/WIKI	89,90 [0,0424]	93,53 [0,0275]	93,03 [0,0228]	91,15 [0,0095]	95,40 [0,0353]	92,60
CharWNN/ALVO	90,51 [0,0678]	93,49 [0,0887]	93,34 [0,0382]	90,80 [0,0087]	94,94 [0,0514]	92,62
Yang and Eisenstein (2015)	92,05	94,23	93,80	92,56	95,14	93,56

Tabela 6.16: Comparação do WNN, CharWNN e WNN+Ftrs com o sistema estado da arte na tarefa Tycho Brahe quando Treinado no Período 1750-1849.

Como podemos observar nas Tabelas 6.14, 6.15 e 6.16, o pré-treinamento trouxe ganhos significativos nos desempenhos de todos os algoritmos. Na SANCL, as representações pré-treinadas nos domínios alvo se mostraram mais eficazes que em domínios mais gerais. Na Tycho Brahe, o mesmo comportamento ocorreu no CharWNN e WNN+Ftrs. Porém, no WNN, as representações pré-treinadas em domínios mais gerais foram que obtiveram os melhores resultados. Além disto, se comparamos a diferença do desempenho de cada modelo sem pré-treinamento com os melhores valores obtidos com pré-treinamento, podemos notar que esta diferença no WNN é consideravelmente maior que no CharWNN e WNN+Ftrs, o que indica que o WNN é mais dependente da qualidade das representações distribuídas de palavras. Esta constatação é algo esperado, pois o WNN não possui atributos relacionados à morfologia como os demais modelos, baseando suas decisões somente nas representações distribuídas de palavras.

De uma forma geral, podemos verificar que o emprego dos atributos manuais (sufixo e capitalização), invariavelmente, implica em melhora substancial de desempenho. O modelo CharWNN, apesar de não utilizar nenhum atributo manual, consegue ter desempenho similar ou superior ao WNN/Ftrs tanto na tarefa SANCL quanto na Tycho Brahe. Isto demonstra que o CharWNN aprende representações tão boas quanto os atributos manuais clássicos. O fato do CharWNN não utilizar atributos manuais é uma característica importante e desejável, principalmente para domínios ou linguagens com pouco recursos.

6.8 Análise de Erros: CharWNN e WNN+Ftrs

Com intuito de melhor compreender a similaridade dos atributos aprendidos pelo CharWNN e os atributos manuais do WNN+Ftrs, nós realizamos uma análise dos erros destes dois modelos. Esta análise consiste em quantificar, por etiqueta, o número erros comuns e exclusivos destas duas redes. Tal análise foi realizada nos conjuntos de devolvimento dos domínios Emails, Weblogs e WSJ da tarefa SANCL. A soma do número de erros exclusivos ao CharWNN, exclusivo ao WNN+Ftrs e que os dois modelos erraram em conjunto foi, respectivamente, 3358, 1364 e 1316 no Emails, Weblogs e WSJ. As 20 etiquetas com mais erros em cada domínio são apresentadas na Figura 6.1. Nesta figura, po-

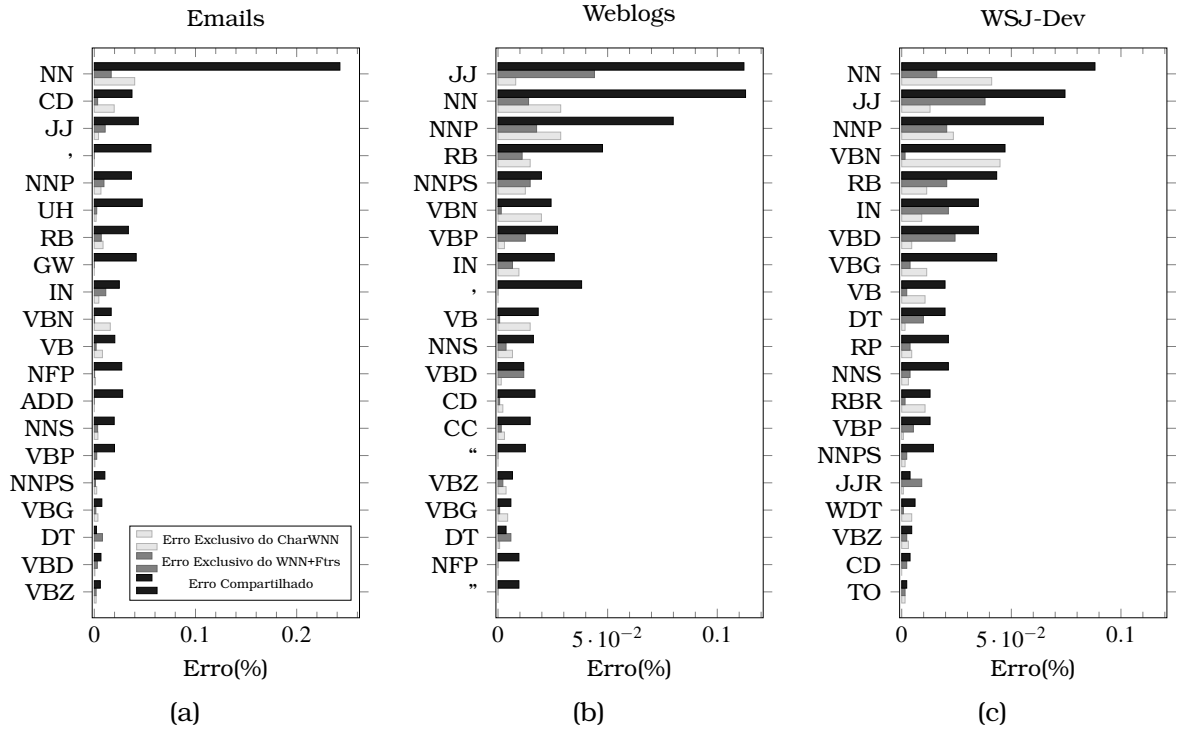


Figura 6.1: Análise de erro das 20 etiquetas com mais erros do WNN+Ftrs e o CharWNN.

demos notar que os atributos aprendidos pelo CharWNN parecem ajudar mais o modelo em algumas etiquetas como: os adjetivos (ADJ); verbos no passado simples (VBD); verbos que não estão na terceira pessoa do singular (VBP); e artigos (DT). Entretanto, os substantivos simples (NN); verbos no passado particípio (VBN); e verbos no infinitivo (VB) parecem ser etiquetados de forma mais correta quando utilizamos os atributos manuais. Apesar da similaridade das duas representações, os resultados apresentados acima indicam que cada uma tem características específicas. Esta características ajudam os modelos em pontos diferentes da adaptação de domínio.

Por fim, nós realizamos um estudo que tenta compreender quanto interferem na adaptação de domínio o tamanho da janela de caracteres do CharWNN e o tamanho do sufixo. Este último é um importante hiper-parâmetro no WNN+Ftrs que tem os valores ótimos conhecidos no cenário intra-domínio de cada língua. A Figura 6.2 apresenta as acurácias obtidas pelo WNN+Ftrs e CharWNN, quando variamos o tamanho do sufixo e da janelas de caracteres, nos conjuntos de teste do Answers, Newsgroups e Reviews e nos conjuntos de desenvolvimento do Emails e Weblogs. Podemos observar que o desempe-

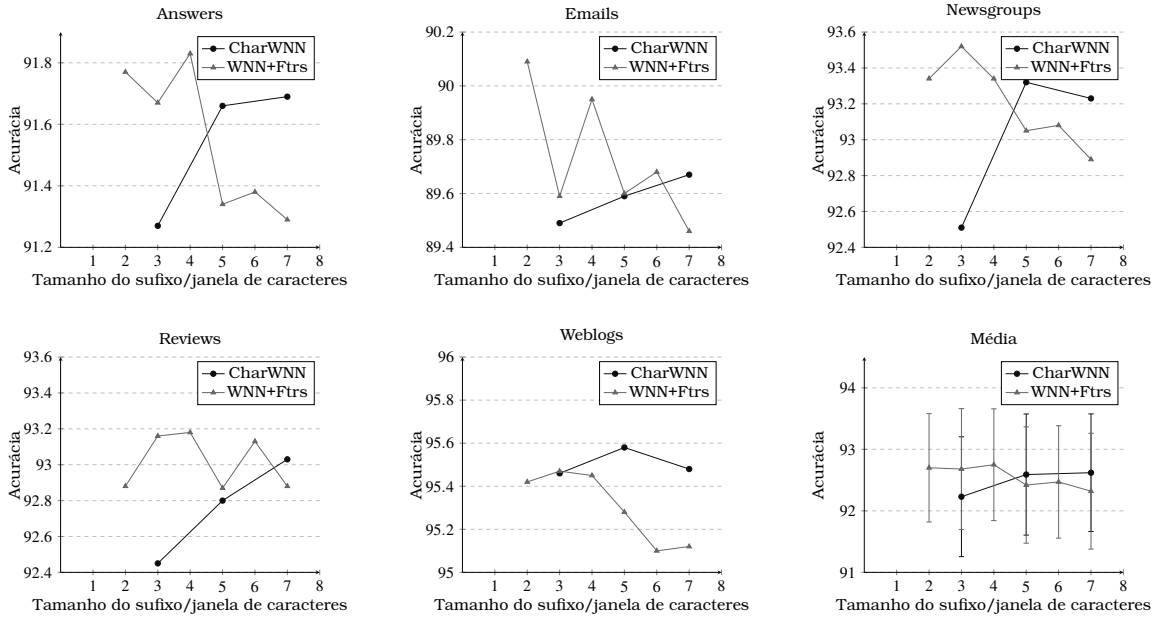


Figura 6.2: Acurácias do CharWNN e WNN na tarefa SANCL com diferentes sufixos e tamanhos da janela de caracteres.

nho do WNN+Ftrs é altamente dependente do tamanho do sufixo. Ainda mais importante, temos que os melhores desempenhos são alcançados com diferentes valores para alguns domínios (3 para Newsgroups e 4 para Weblogs). Enquanto que para os demais domínios o comportamento é bastante caótico. Por outro lado, o CharWNN apresenta um comportamento mais consistente, no qual os melhores desempenhos geralmente ocorrem nas janelas de caracteres maiores.

Nós realizamos os mesmos experimentos na tarefa Tycho Brahe utilizando os cenários cujos domínios origem são 1800-1849 e 1750-1849. Nas Figuras 6.3 e 6.4, são apresentados, respectivamente, os resultados de tais cenários. Podemos notar, novamente, que o comportamento CharWNN é mais estavél do que o do WNN+Ftrs.

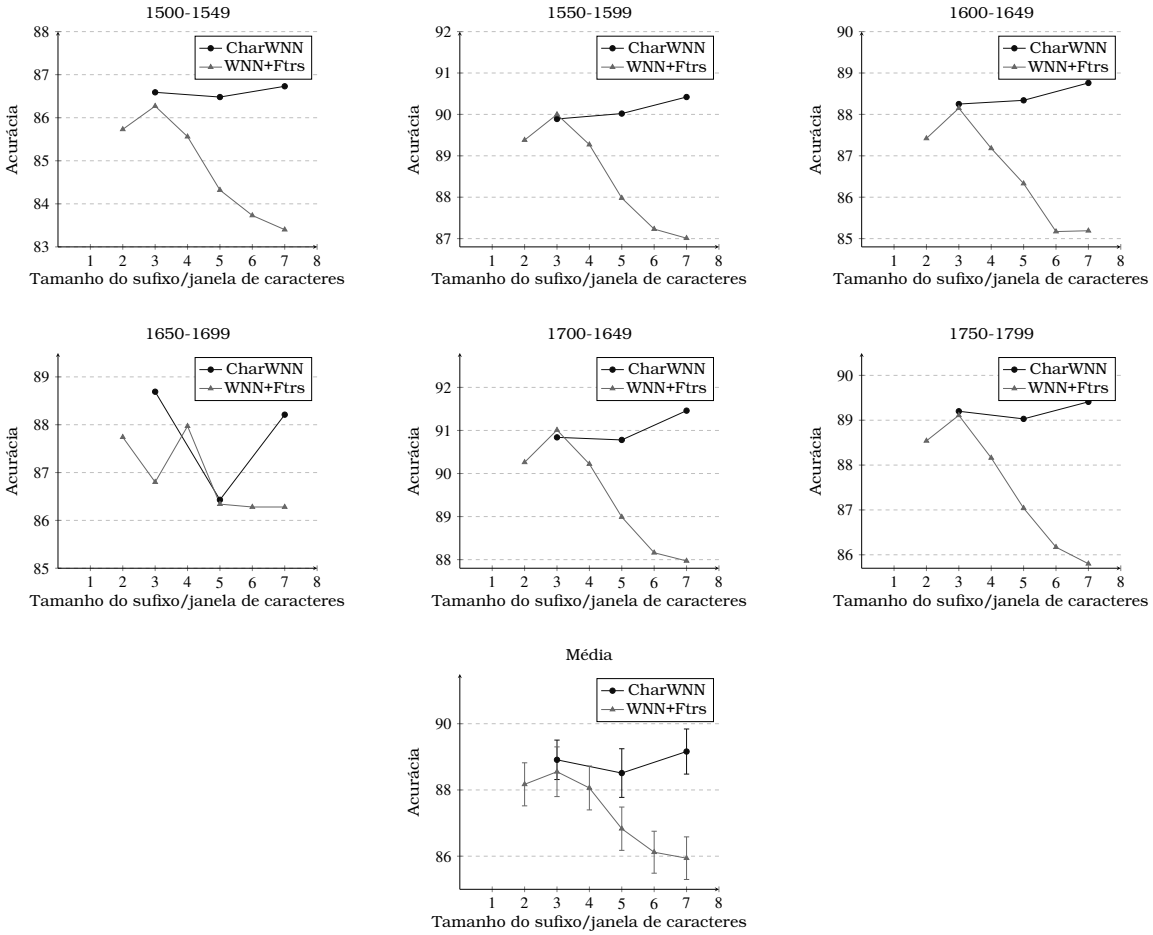


Figura 6.3: Acurácias do CharWNN e WNN na tarefa Tycho Brahe quando treinado no período 1800-1849 com diferentes sufixos e tamanhos da janela de caracteres.

6.9 Adaptação de Domínio Supervisionada

O sistema estado da arte de AD na tarefa de etiquetagem morfossintática (Ma et al., 2014) possui um erro médio de 6,85 nos domínios alvo da tarefa SANCL (veja Tabela 6.14). Este valor é quase três vezes maior do que o erro obtido pelo mesmo sistema no domínio origem (WSJ), que é de 2,56. Esta diferença indica que o problema de adaptação de domínio não supervisionada é difícil. Inspirados por esta observação, realizamos um estudo para identificar a quantidade de *dados anotados dos domínios alvo* que são necessários para que o CharWNN alcance um desempenho similar ao sistema de Ma et al.

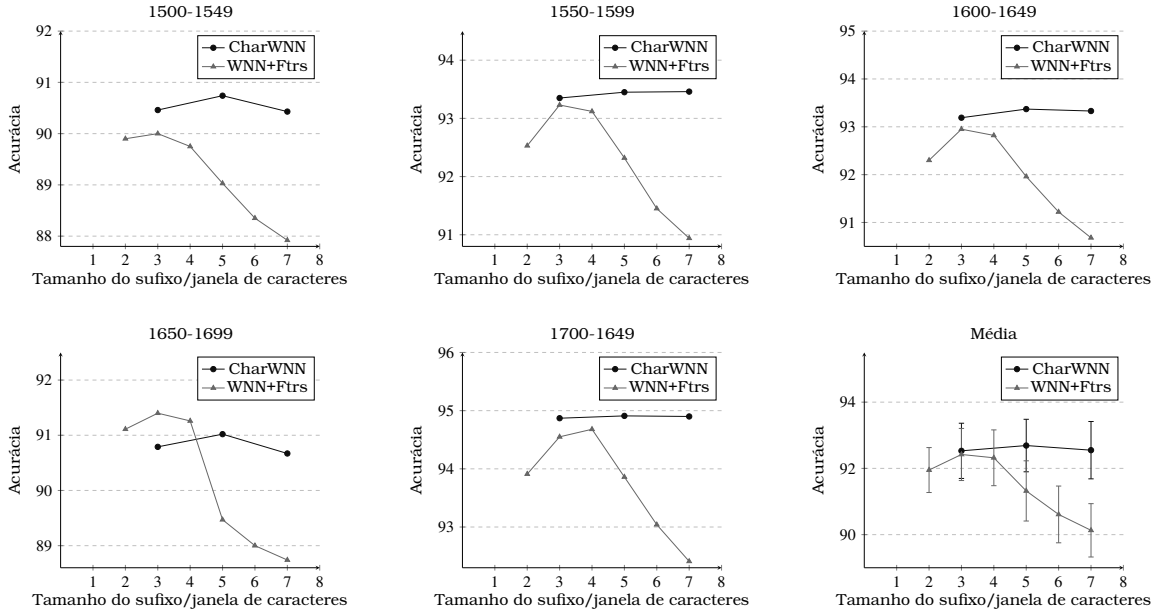


Figura 6.4: Acurácias do CharWNN e WNN na tarefa Tycho Brahe quando treinado no período 1750-1849 com diferentes sufixos e tamanhos da janela de caracteres.

(2014). Nestes experimentos, as representações distribuídas de palavras do CharWNN foram pré-treinadas nos dados não anotados dos domínios alvo. Para isto, treinamos o CharWNN na tarefa SANCL empregando dados anotados do domínio origem e dos domínios alvo. Considerando o objetivo da tarefa original, onde um único modelo deve ser aplicado em todos os domínios alvo, optamos por utilizar dados anotados dos cinco domínios alvo. Realizamos experimentos com 1, 10, 50, 100 e 250 frases anotadas de cada domínio alvo. Denominamos o número de frases anotadas de cada domínio como N_t. Estas frases foram obtidas a partir dos conjuntos de desenvolvimento dos domínios Answers, Newsgroups e Reviews e conjuntos de teste dos domínios Emails e Weblogs.

Em razão do desbalanceamento do número de exemplos no domínio origem e no domínio alvo, realizamos uma amostragem não uniforme entre estes dois conjuntos de dados durante o treinamento. Ao selecionar um exemplo de treinamento, primeiramente, um dos domínios é probabilisticamente selecionado, sendo que λ_S é a probabilidade de selecionar o domínio origem. Consequentemente, a probabilidade de selecionar o domínio alvo é igual a $1 - \lambda_S$. Em seguida, um exemplo é sorteado uniformemente dentre os exemplos do domínio selecionado. Devido à existência de aleatoriedade no treinamento, em especial na seleção de exemplos, realizamos dez execuções de cada experimento. Nas tabelas e figuras desta seção, apresentamos a acurácia média e o erro padrão da média destas dez execuções.

O hiper-parâmetro λ_S e a taxa de aprendizado foram calibrados no con-

junto de desenvolvimento dos domínios Emails e Weblogs para cada um dos valores de N_t. Os melhores valores encontrados para este hiper-parâmetro são apresentados na Tabela 6.17. Na Figura 6.5, são mostrados os resultados do CharWNN nos domínios de validação. Como podemos notar, o uso de poucos dados anotados trouxe uma melhora considerável no desempenho, especialmente no Emails, chegando a ultrapassar o método estado da arte ao utilizar 50 frases anotados de cada um dos 5 domínio alvo. Os resultados do

N_t	**Taxa de aprendizado**	λ_S
1	0,0041	0,9804
10	0,0089	0,4132
50	0,0068	0,5285
100	0,0005	0,8000
250	0,0078	0,6452

Tabela 6.17: Hiper-parâmetros da adaptação de domínio supervisionada.

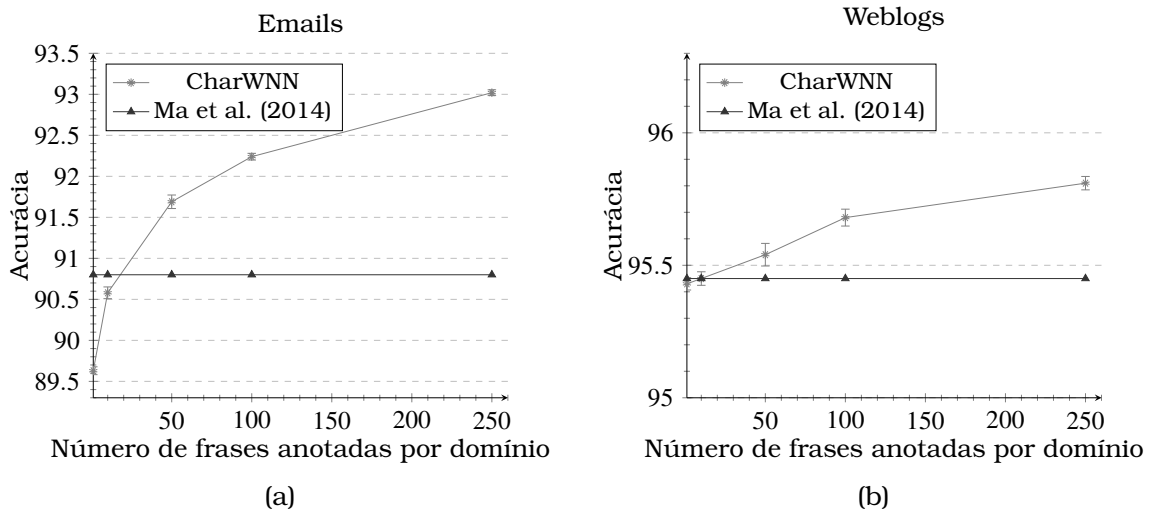


Figura 6.5: Resultados do CharWNN na adaptação de domínio supervisionada no Emails e Weblogs.

CharWNN nos conjuntos de teste (Answers, Newsgrouops e Reviews) são exibidos na Figura 6.6. Além disto, a Tabela 6.18 apresenta as diferenças entre as acurácias do CharWNN e de Ma et al. (2014) (valores positivos significam que o CharWNN foi melhor do que seu concorrente). Como podemos observar, do mesmo modo que aconteceu nos conjuntos de validação, o desempenho do CharWNN melhora a medida que mais dados anotados do domínio alvo são considerados. O uso de somente 50 frases anotadas de cada domínio (250 no total) é suficiente para que o CharWNN obtenha um desempenho equivalente ao estado da arte. Com o aumento da quantidade de frases de cada domínio para 250 (1250 no total), o CharWNN consegue superar significativamente o estado da arte. Estes resultados demostram como é difícil aprender informa-

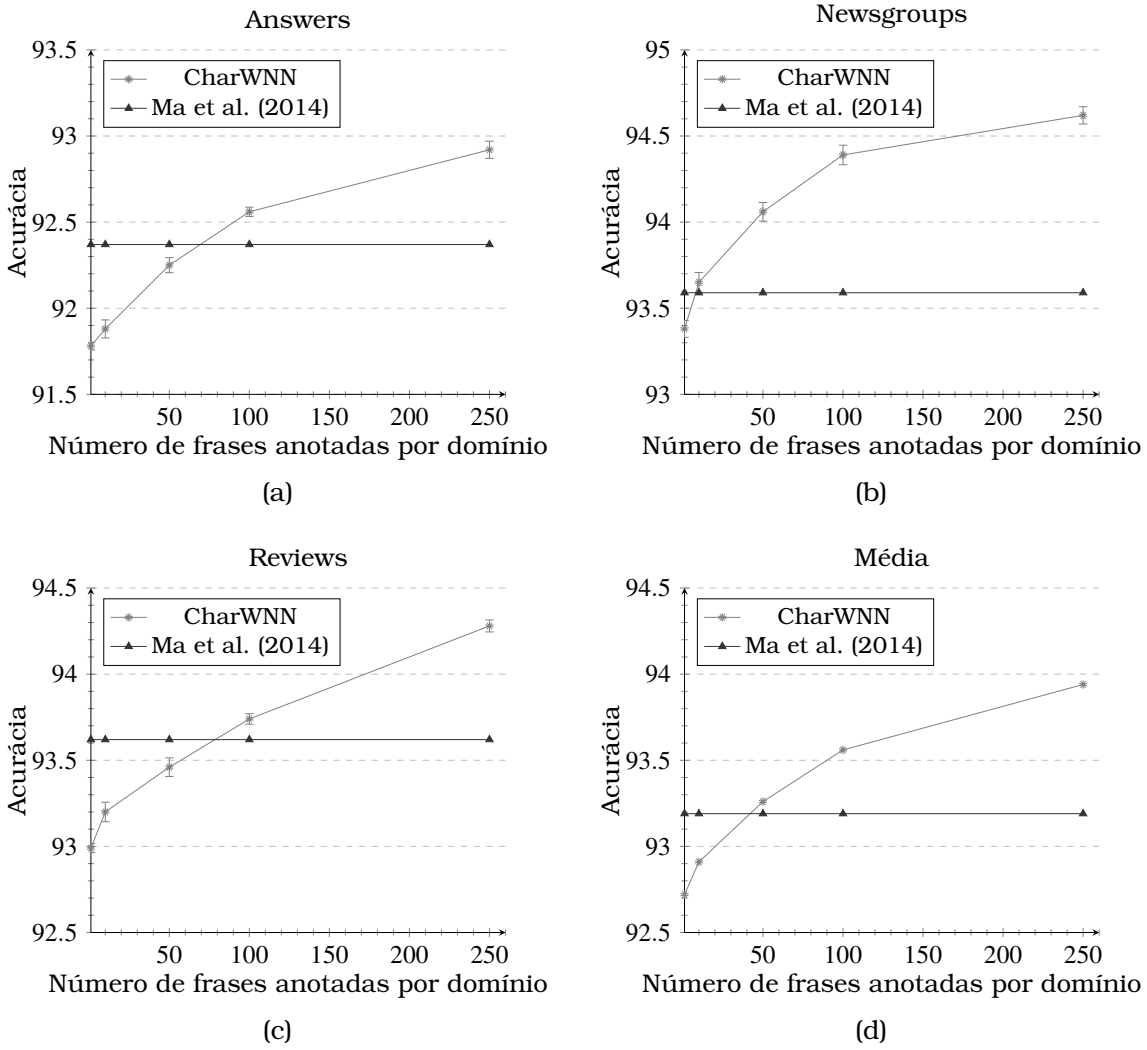


Figura 6.6: Resultados do CharWNN na adaptação de domínio supervisionada.

N_t	**Answers**	**Newsgroups**	**Reviews**	**Média**
1	-0,59	-0,21	-0,63	-0,47
10	-0,49	0,06	-0,42	-0,28
50	-0,12	0,47	-0,16	0,06
100	0,19	0,80	0,12	0,37
250	0,55	1,03	0,66	0,74

Tabela 6.18: Diferença das acurácias do CharWNN e de Ma et al. (2014), quando o primeiro usa N_t frases anotadas dos domínios alvo.

ções relevantes de uma tarefa nos domínios origem e alvo utilizando somente dados não anotados. Alcançamos o mesmo desempenho do estado da arte em AD não supervisionada usando poucos dados anotados. Isto demonstra a dificuldade do problema de adaptação de domínio não supervisionada. Além disto, dado o custo em se desenvolver e aplicar um método forte de AD, podemos concluir que, em muitos casos, pode ser mais eficiente e eficaz anotar dados dos domínios alvo do que usar um método de AD.

CAPÍTULO

7

Conclusão

Os métodos desenvolvidos neste trabalho abordam o problema de adaptação de domínio não supervisionada utilizando o aprendizado profundo de representações. Estes métodos foram baseados em trabalhos anteriores para AD. Entretanto, estes métodos prévios foram propostos para outras tarefas e/ou utilizavam arquiteturas de rede neural distintas. O desenvolvimento de novas arquiteturas e a criação de métodos de AD que não utilizam atributos manuais na etiquetagem morfossintática são contribuições importantes deste trabalho. Porém, os métodos de AD desenvolvidos não trouxeram ganhos significativos. Acreditamos que os métodos de AD desenvolvidos neste trabalho não funcionaram devido à dificuldade da tarefa. Provavelmente, se dedicássemos mais tempo para investigar, poderíamos fazê-los funcionar, mesmo que marginalmente. Por este motivo, gostaríamos de investigar, principalmente através de análises de erro, a razão do mau funcionamento destes métodos.

Todos os métodos de AD usaram o modelo CharWNN como base. Este modelo não utiliza atributos manuais, tomando como entrada apenas o texto cru, o que é uma importante vantagem deste modelo. Além disto, quando este modelo é inicializado com representações distribuídas de palavras pré-treinadas em dados não anotados dos domínios alvo, ele obtém um desempenho próximo dos melhores sistemas disponíveis na literatura. Este é um feito relevante, dado que este modelo não necessita de atributos manuais e usa os dados não anotados dos domínios alvo de uma maneira trivial. O fato do CharWNN em não usar atributos manuais é algo importante e desejável, principalmente para domínios ou linguagens com pouco recursos. Este resultado consiste em uma das contribuições deste trabalho. Outra contribuição foi a confecção de um artigo descrevendo os resultados do CharWNN deste trabalho. Este artigo

foi aceito para publicação na *2017 International Joint Conference on Neural Networks* (IJCNN 2017) (Rodrigues et al., 2017).

Futuramente, pretendemos realizar estudos para compreender melhor o que realmente são os atributos aprendidos pelo CharWNN e como eles se adaptam a um domínio diferente. Além disto, gostaríamos de estudar mais profundamente como as representações distribuídas de palavras e o modo como a treinamos impactam na adaptação de domínio.

Por último, realizamos alguns experimentos que evidenciam, de uma maneira simples e clara, a dificuldade do problema de adaptação de domínio não supervisionada; algo que não está explícito na literatura. Pretendemos no futuro ampliar a nossa pesquisa no campo de AD e desenvolver também métodos para o cenário semissupervisionado e supervisionado. Uma ideia simples para estes cenários é realizar o ajuste fino do CharWNN usando somente os exemplos anotados do domínio alvo, aliado a taxas de aprendizado baixas para evitar a total sobrescrita das representações aprendidas no treinamento com dados do domínio origem. Outra ideia é aplicar métodos com ideias similares ao DLID, DAN e DATT no cenário de adaptação de domínio semissupervisionado.

Referências Bibliográficas

Allen, J. F. (2013). Natural language processing. In *Encyclopedia of Computer Science*, páginas 1218–1222. John Wiley and Sons Ltd., Chichester, UK. Citado na página 1.

Axelrod, A., He, X., e Gao, J. (2011). Domain adaptation via pseudo in-domain data selection. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, EMNLP '11, páginas 355–362, Stroudsburg, PA, USA. Association for Computational Linguistics. Citado na página 37.

Bengio, Y. (2008). Neural net language models. *Scholarpedia*, 3(1):3881. Citado nas páginas 11 e 12.

Bengio, Y., Courville, A., e Vincent, P. (2013). Representation learning: A review and new perspectives. *IEEE Trans. Pattern Anal. Mach. Intell.*, 35(8):1798–1828. Citado nas páginas 3, 10, e 19.

Bengio, Y., Ducharme, R., Vincent, P., e Janvin, C. (2003). A Neural Probabilistic Language Model. *The Journal of Machine Learning Research*, 3:1137–1155. Citado na página 12.

Bergstra, J. e Bengio, Y. (2012). Random search for hyper-parameter optimization. *The Journal of Machine Learning Research (JMLR)*, 13:281–305. Citado na página 47.

Bickel, S. e Scheffer, T. (2005). Estimation of mixture models using co-em. In *Machine Learning: ECML 2005, 16th European Conference on Machine Learning, Porto, Portugal, October 3-7, 2005, Proceedings*, páginas 35–46. Springer Berlin Heidelberg. Citado na página 28.

Blitzer, J., McDonald, R., e Pereira, F. (2006). Domain adaptation with structural correspondence learning. In *Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing*, EMNLP '06, páginas 120–128,

Stroudsburg, PA, USA. Association for Computational Linguistics. Citado nas páginas 37 e 41.

Chan, Y. S. e Ng, H. T. (2006). Estimating class priors in domain adaptation for word sense disambiguation. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*, páginas 89–96, Sydney, Australia. Association for Computational Linguistics. Citado na página 37.

Chelba, C. e Acero, A. (2004). Adaptation of maximum entropy capitalizer: Little data can help a lot. In Lin, D. e Wu, D., editors, *Proceedings of EMNLP 2004*, páginas 285–292, Barcelona, Spain. Association for Computational Linguistics. Citado na página 37.

Chen, M., Weinberger, K. Q., e Blitzer, J. (2011). Co-training for domain adaptation. In Shawe-Taylor, J., Zemel, R. S., Bartlett, P. L., Pereira, F., e Weinberger, K. Q., editors, *Advances in Neural Information Processing Systems 24*, páginas 2456–2464. Curran Associates, Inc. Citado nas páginas 40 e 41.

Chen, M., Xu, Z., Weinberger, K., e Sha, F. (2012). Marginalized denoising autoencoders for domain adaptation. In Langford, J. e Pineau, J., editors, *Proceedings of the 29th International Conference on Machine Learning (ICML-12)*, ICML '12, páginas 767–774, New York, NY, USA. ACM. Citado nas páginas 32, 38, e 41.

Cheng, H., Fang, H., He, X., Gao, J., e Deng, L. (2016). Bi-directional attention with agreement for dependency parsing. In *Proceedings of EMNLP-2016*, pagina 2204–2214. Citado na página 2.

Chiu, J. P. C. e Nichols, E. (2016). Named entity recognition with bidirectional lstm-cnns. *TACL*, 4:357–370. Citado na página 1.

Chopra, S., Balakrishnan, S., e Gopalan, R. (2013). Dlid: Deep learning for domain adaptation by interpolating between domains. In *ICML workshop on challenges in representation learning*. Citado nas páginas 4, 26, 39, e 41.

Collobert, R., Weston, J., Bottou, L., Karlen, M., Kavukcuoglu, K., e Kuksa, P. (2011). Natural language processing (almost) from scratch. *The Journal of Machine Learning Research*, 12:2493–2537. Citado nas páginas xiii, 14, e 21.

Daumé, III, H., Kumar, A., e Saha, A. (2010). Frustratingly easy semi-supervised domain adaptation. In *Proceedings of the 2010 Workshop on Domain Adaptation for Natural Language Processing*, DANLP 2010, páginas

53–59, Stroudsburg, PA, USA. Association for Computational Linguistics. Citado nas páginas 2, 25, 40, e 41.

Daumé III, H. (2007). Frustratingly easy domain adaptation. In *Conference of the Association for Computational Linguistics (ACL)*, Prague, Czech Republic. Citado na página 40.

Daume III, H. e Marcu, D. (2006). Domain adaptation for statistical classifiers. *Journal of Artificial Intelligence Research*, 26:101–126. Citado na página 37.

Delalleau, O. e Bengio, Y. (2011). Shallow vs. deep sum-product networks. In *Advances in Neural Information Processing Systems*, páginas 666–674. Citado na página 9.

dos Santos e Bianca Zadrozny, C. N. (2014a). Learning character-level representations for part-of-speech tagging. In *Proceedings of the 31th International Conference on Machine Learning, ICML 2014, Beijing, China, 21-26 June 2014*, páginas 1818–1826. Citado na página 12.

dos Santos e Bianca Zadrozny, C. N. (2014b). Training state-of-the-art portuguese POS taggers without handcrafted features. In *Computational Processing of the Portuguese Language - 11th International Conference, PROPOR 2014, São Carlos/SP, Brazil, October 6-8, 2014. Proceedings*, páginas 82–93. Citado nas páginas 3, 6, 7, 12, 47, e 50.

Duchi, J., Hazan, E., e Singer, Y. (2011). Adaptive subgradient methods for online learning and stochastic optimization. *The Journal of Machine Learning Research (JMLR)*, 12:2121–2159. Citado na página 21.

Finkel, J. R. e Manning, C. D. (2009). Hierarchical bayesian domain adaptation. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, NAACL '09, páginas 602–610, Stroudsburg, PA, USA. Association for Computational Linguistics. Citado na página 37.

Fonseca, E. R., G Rosa, J. L., e Aluísio, S. M. (2015). Evaluating word embeddings and a revised corpus for part-of-speech tagging in portuguese. *Journal of the Brazilian Computer Society*, 21(1):1–14. Citado nas páginas 1 e 14.

Galves, C. e Faria, P. (2010). Tycho brahe parsed corpus of historical portuguese. Citado nas páginas 5 e 43.

Ganin, Y., Ustinova, E., Ajakan, H., Germain, P., Larochelle, H., Laviolette, F., Marchand, M., e Lempitsky, V. (2016). Domain-adversarial training of neural networks. *The Journal of Machine Learning Research (JMLR)*, 17(1):2096–2030. Citado nas páginas 4, 29, 31, 39, 40, e 41.

Glorot, X., Bordes, A., e Bengio, Y. (2011). Domain adaptation for large-scale sentiment classification: A deep learning approach. In *In Proceedings of the Twenty-eight International Conference on Machine Learning, ICML*. Citado nas páginas 32, 37, 38, e 41.

Goldberg, Y. (2016). A primer on neural network models for natural language processing. *Journal of Artificial Intelligence Research (JAIR)*, 57:345–420. Citado na página 7.

Goodfellow, I., Bengio, Y., e Courville, A. (2016). *Deep Learning*. MIT Press. http://www.deeplearningbook.org. Citado na página 24.

Gutmann, M. U. e Hyvärinen, A. (2012). Noise-contrastive estimation of unnormalized statistical models, with applications to natural image statistics. *The Journal of Machine Learning Research*, 13(1):307–361. Citado na página 23.

Haykin, S. (1998). *Neural Networks: A Comprehensive Foundation*. Prentice Hall PTR, Upper Saddle River, NJ, USA, 2nd edition. Citado na página 28.

Haykin, S. (2009). *Neural Networks and Learning Machines*. Number v. 10 in Neural networks and learning machines. Prentice Hall. Citado nas páginas 7 e 8.

Hinton, G. E., Osindero, S., e Teh, Y.-W. (2006). A fast learning algorithm for deep belief nets. *Neural computation*, 18(7):1527–1554. Citado na página 20.

Huang, F. e Yates, A. (2009). Distributional representations for handling sparsity in supervised sequence-labeling. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 1 - Volume 1*, ACL '09, páginas 495–503, Stroudsburg, PA, USA. Association for Computational Linguistics. Citado na página 41.

Jarrett, K., Kavukcuoglu, K., Ranzato, M., e LeCun, Y. (2009). What is the best multi-stage architecture for object recognition? In *Computer Vision, 2009 IEEE 12th International Conference on*, páginas 2146–2153. Citado nas páginas 39 e 41.

Jiang, J. (2008). *Domain Adaptation in Natural Language Processing*. PhD thesis, University of Illinois at Urbana-Champaign, Champaign, IL, USA. AAI3337811. Citado nas páginas 2 e 40.

Kingma, D. e Ba, J. (2014). Adam: A Method for Stochastic Optimization. *International Conference on Learning Representations*, páginas 1–13. Citado na página 21.

Le, Q. V. e Mikolov, T. (2014). Distributed representations of sentences and documents. In *Proceedings of the 31th International Conference on Machine Learning, ICML 2014, Beijing, China, 21-26 June 2014*, páginas 1188–1196. Citado na página 12.

Le Roux, J., Foster, J., Wagner, J., Kaljahi, R. S. Z., e Bryl, A. (2012). DCU-Paris13 Systems for the SANCL 2012 Shared Task. In *NAACL 2012*, páginas 1–4, Montréal, Canada. Citado na página 54.

LeCun, Y., Bengio, Y., e Hinton, G. (2015). Deep learning. *Nature*, 521(7553):436–444. Citado na página 3.

LeCun, Y., Bottou, L., Bengio, Y., e Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324. Citado na página 20.

Li, Q. (2012). Literature Survey : Domain Adaptation Algorithms for Natural Language Processing Qi. Citado nas páginas 25, 37, e 38.

Liu, C., Wu, S., Jiang, S., e Tung, A. K. H. (2012). Cross domain search by exploiting wikipedia. In *Proceedings of the 2012 IEEE 28th International Conference on Data Engineering*, ICDE '12, páginas 546–557, Washington, DC, USA. IEEE Computer Society. Citado na página 37.

Lorentz, A. (2013). With big data, context is a big issue. Citado na página 1.

Luong, T., Socher, R., e Manning, C. D. (2013). Better word representations with recursive neural networks for morphology. In *Proceedings of the Seventeenth Conference on Computational Natural Language Learning, CoNLL 2013, Sofia, Bulgaria, August 8-9, 2013*, páginas 104–113. Citado na página 12.

Ma, J., Zhang, Y., e Zhu, J. (2014). Tagging the web: Building a robust web tagger with neural network. In *ACL (1)*, páginas 144–154. Citado nas páginas xvi, 4, 5, 6, 32, 33, 39, 40, 41, 46, 54, 58, 60, e 61.

Martin T. Hagan, Howard B. Demuth, M. H. B. (2014). *Neural Network Design*. Martin Hagan, 2rd edition. Citado na página 7.

Mikolov, T. (2013). word2vec. Citado na página 22.

Mikolov, T., Chen, K., Corrado, G., e Dean, J. (2013a). Distributed Representations of Words and Phrases and their Compositionality. *Nips*, páginas 1–9. Citado nas páginas 23, 34, e 35.

Mikolov, T., Corrado, G., Chen, K., e Dean, J. (2013b). Efficient Estimation of Word Representations in Vector Space. *Proceedings of the International Conference on Learning Representations (ICLR 2013)*, páginas 1–12. Citado nas páginas 22 e 39.

Mitchell, T. M. (1997). *Machine Learning*. McGraw-Hill, Inc., New York, NY, USA, 1 edition. Citado na página 8.

Mnih, A. e Teh, Y. W. (2012). A fast and simple algorithm for training neural probabilistic language models. In *In Proceedings of the International Conference on Machine Learning*. Citado na página 23.

Mou, L., Meng, Z., Yan, R., Li, G., Xu, Y., Zhang, L., e Jin, Z. (2016). How transferable are neural networks in NLP applications? In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, páginas 478–489. Empirical Methods in Natural Language Processing (EMNLP). Citado nas páginas 3 e 10.

Nielsen, M. A. (2015). *Neural Networks and Deep Learning*. Determination Press. Citado nas páginas 9 e 20.

Oliveira, C. e de Freitas, M. C. (2006). Classes de palavras e etiquetagem na lingüística computacional. *Calidoscópio*, 4:79–188. Citado na página 1.

Pennington, J., Socher, R., e Manning, C. D. (2014). Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, páginas 1532–1543. Citado na página 50.

Petrov, S. e McDonald, R. (2012). Overview of the 2012 shared task on parsing the web. In *Notes of the First Workshop on Syntactic Analysis of Non-Canonical Language (SANCL)*, volume 59. Citeseer. Citado nas páginas 4, 44, e 47.

Qiu, L., Cao, Y., Nie, Z., Yu, Y., e Rui, Y. (2014). Learning word representation considering proximity and ambiguity. In *AAAI*, páginas 1572–1578. Citado na página 11.

Rodrigues, I. M., Fernandes, E. R., e dos Santos, C. N. (2017). Domain adaptation of POS taggers without handcrafted features. In *2017 International Joint Conference on Neural Networks (IJCNN)*. Aceito para publicação. Citado nas páginas 6 e 64.

Russell, Stuart e Norvig, P. (2009). *Artificial Intelligence: A Modern Approach*. Prentice Hall Press, Upper Saddle River, NJ, USA, 3rd edition. Citado nas páginas 8 e 22.

Russell, S. e Norvig, P. (2009). *Artificial Intelligence: A Modern Approach*. Prentice Hall Press, Upper Saddle River, NJ, USA, 3rd edition. Citado na página 7.

Santos, C. D. e Zadrozny, B. (2014). Learning character-level representations for part-of-speech tagging. In Jebara, T. e Xing, E. P., editors, *Proceedings of the 31st International Conference on Machine Learning (ICML-14)*, páginas 1818–1826. JMLR Workshop and Conference Proceedings. Citado nas páginas 1, 45, e 54.

Schnabel, T. e Schütze, H. (2014). Flors: Fast and simple domain adaptation for part-of-speech tagging. *Transactions of the Association for Computational Linguistics*, 2:15–26. Citado nas páginas 38 e 41.

Shepherd, G. (1998). *The Synaptic Organization of the Brain*. Oxford medical publications. Oxford University Press. Citado na página 8.

Tang, B., Jiang, M., e Xu, H. (2012). Varderbilt's system for sancl2012 shared task. In *Notes of the First Workshop on Syntactic Analysis of NonCanonical Language (SANCL)*. Citado na página 54.

Theano Development Team (2016). Theano: A Python framework for fast computation of mathematical expressions. *arXiv e-prints*, abs/1605.02688. Citado na página 17.

Turian, J., Ratinov, L., e Bengio, Y. (2010). Word representations: A simple and general method for semi-supervised learning. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, ACL '10, páginas 384–394, Stroudsburg, PA, USA. Association for Computational Linguistics. Citado na página 11.

Wang, R., Liu, W., e McDonald, C. (2014). Corpus-independent generic keyphrase extraction using word embedding vectors. In *Software Engineering Research Conference*, pagina 39. Citado nas páginas 11 e 12.

Xu, R., Xu, J., e Wang, X. (2011). Instance level transfer learning for cross lingual opinion analysis. In *Proceedings of the 2Nd Workshop on Computational Approaches to Subjectivity and Sentiment Analysis*, WASSA '11, páginas 182–188, Stroudsburg, PA, USA. Association for Computational Linguistics. Citado na página 37.

Yang, Y. e Eisenstein, J. (2015). Unsupervised multi-domain adaptation with feature embeddings. In *NAACL HLT 2015, The 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Denver, Colorado, USA, May 31 - June 5, 2015*, páginas 672–682. Citado nas páginas 5, 12, 39, 41, 43, 47, 53, 54, e 55.