Uma abordagem conexionista para anotação de papéis semânticos

Erick Rocha Fonseca

SERVIÇO DE PÓS-GRADUAÇÃO DO ICMC-USP

Data de Depósito:

Assinatura:_____

Uma abordagem conexionista para anotação de papéis semânticos

Erick Rocha Fonseca

Orientador: **Prof. Dr. João Luís Garcia Rosa**

Dissertação apresentada ao Instituto de Ciências Matemáticas e de Computação - ICMC-USP, como parte dos requisitos para obtenção do título de Mestre em Ciências - Ciências de Computação e Matemática Computacional. *VERSÃO REVISADA*

USP – São Carlos
Junho de 2013

Ficha catalográfica elaborada pela Biblioteca Prof. Achille Bassi
e Seção Técnica de Informática, ICMC/USP,
com os dados fornecidos pelo(a) autor(a)

F676a

Fonseca, Erick Rocha
Uma abordagem conexionista para anotação de papéis semânticos / Erick Rocha Fonseca; orientador João Luís Garcia Rosa. -- São Carlos, 2013.
65 p.

Dissertação (Mestrado - Programa de Pós-Graduação em Ciências de Computação e Matemática Computacional) -- Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, 2013.

1. Processamento de línguas naturais. 2. Redes neurais. 3. Aprendizado de máquina. I. Rosa, João Luís Garcia, orient. II. Título.

Agradecimentos

Agradeço à minha família, por todo o apoio que me deram em todos os sentidos, desde muito antes de decidir realizar o mestrado ou de pisar numa universidade.

Agradeço aos amigos que conheci em São Carlos, por todo o companheirismo, convivência, diversão, troca de experiências e aprendizagens.

Agradeço também a todos os outros amigos, que pude ver com menos frequência nos últimos dois anos, mas que também muito contribuíram na minha vida.

Agradeço ao professor João Luís Rosa, que me acompanhou durante o mestrado e me proporcionou a oportunidade de realizar este trabalho, além de assistir às suas aulas deveras interessantes.

Agradeço a todos os outros grandes professores da USP, com os quais também pude aprender muito.

Agradeço às diversas pessoas desconhecidas que me ajudaram na Internet em questões técnicas.

Agradeço à CAPES e à FAPESP pelo apoio financeiro durante o mestrado.

"Às pessoas de agora não lhes passa pela cabeça o trabalho que deram a criar estes vocábulos, em primeiro lugar, e quem sabe se não terá sido, de tudo, o mais difícil, foi preciso perceber que havia necessidade deles, depois houve que chegar a um consenso sobre seus significados imediatos, e finalmente, tarefa que nunca viria a concluir-se por completo, imaginar as consequências que poderiam advir, a médio e a longo prazo, dos ditos efeitos e dos ditos vocábulos."
— José Saramago
O Homem Duplicado

"*What's in a name? That which we call a rose by any other name would smell as sweet.*"
— William Shakespeare
Romeo and Juliet

Resumo

A anotação de papéis semânticos (APS) é uma subárea do Processamento de Línguas Naturais (PLN) que começou a ser explorada para a língua inglesa a partir de 2002. Seu objetivo é detectar estruturas de predicador e argumentos em sentenças escritas, que correspondem a descrições de eventos (normalmente feitas por verbos); seus participantes, como agente e paciente; e circunstâncias, como tempo, local, etc. Diversas aplicações de PLN, como tradução automática e recuperação de informação, têm obtido melhorias em seu desempenho ao empregar a APS como uma etapa de pré-processamento.

Para a língua portuguesa, os avanços na pesquisa de APS são ainda muito incipientes. Dado que a grande maioria dos trabalhos encontrados na literatura desta área emprega aprendizado de máquina supervisionado, um fator limitante tem sido a ausência de dados rotulados em português, problema que apenas recentemente foi parcialmente resolvido com a criação do PropBank-Br. Este recurso segue o modelo de anotação usado no PropBank, o principal conjunto de dados rotulados empregado na tarefa de APS para a língua inglesa. Ainda assim, o PropBank-Br contém menos de um décimo do total de instâncias de dados presentes no PropBank original.

Outro ponto a ser observado é que a abordagem mais comum para a APS baseia-se na extração de uma grande quantidade de informação linguística das sentenças de entrada para ser usada por classificadores automáticos. Tal abordagem mostra-se extremamente dependente de outras ferramentas de PLN, característica particularmente indesejável no caso da língua portuguesa, que não possui muitos recursos livremente disponíveis. Em contrapartida, uma outra abordagem bem sucedida encontrada na literatura abre mão do uso de conhecimento linguístico explícito e associa palavras a sequências numéricas, cujos valores são ajustados durante o treinamento de uma rede neural artificial. Estas sequências são então empregadas pela rede para realizar a APS, e podem servir também para outras tarefas de PLN.

O presente trabalho seguiu o segundo método descrito acima. Foram implementadas alterações nesse método que permitiram um ganho de desempenho em comparação com sua versão original quando testada no PropBank-Br. A versão final do sistema desenvolvido está pronta para uso e poderá auxiliar pesquisas de PLN em português.

Palavras-chave: Processamento de língua natural, redes neurais, papéis semânticos

Abstract

Semantic Role Labeling (SRL) is a subfield of Natural Language Processing (NLP) which began to be explored for English in 2002. Its goal is to detect structures of predicate and arguments in written sentences, which correspond to descriptions of events (usually made by verbs); its participants, such as agents and patients; and circumstances, such as time, place, etc. Many NLP applications, as machine translation and information retrieval, have achieved performance gains by applying SRL as a pre-processing step.

For Portuguese, advances in SRL research are still in very early stages. Given that the majority of works found in the literature of this area employ supervised machine learning, a limiting factor has been the absence of labeled data in Portuguese, a problem that only recently was partially solved with the creation of PropBank-Br. This resource follows the annotation model used in PropBank, the main labeled data set employed in the SRL task for English. Even then, PropBank-Br contains less than one tenth of the data instances present in the original PropBank.

Another point to be observed is that the most common approach to SRL is based on the extraction of a great amount of information from the input sentences to be used by automatic classifiers. Such approach is extremely dependent on other NLP tools, a particularly undesirable feature in the case of Portuguese, which does not have many freely available resources. On the other hand, another succesful approach found in the literature forgoes the use of explicit linguistic knowledge and associates words to numeric sequences, whose values are adjusted during the training of an artificial neural network. These sequences are then employed by the network in order to perform SRL, and can also be useful for other NLP tasks.

This work followed the second method described above. Modifications on this method were implemented and allowed for a performance gain in comparison with its original version when tested on PropBank-Br. The final version of the developed system is ready for use and will be able to help NLP research in Portuguese.

Keywords: Natural language processing, neural networks, semantic roles

Sumário

Resumo v
Abstract vii

Lista de Figuras **xi**

Lista de Tabelas **xiii**

Lista de Abreviações **xvi**

1 Introdução **1**
1.1 Contextualização 1
1.2 Objetivos 4
1.3 Organização da Dissertação 4

2 Papéis Semânticos **5**
2.1 Conceitos Linguísticos 5
2.2 O PropBank 8
2.3 Considerações Finais 11

3 Aprendizado de Máquina e Anotação de Papéis Semânticos **15**
3.1 Aprendizado de Máquina 15
3.1.1 Redes Neurais Artificiais 16
3.2 Sistemas de APS 17
3.2.1 Sistemas de Classificadores de Constituintes 18
3.2.2 Sistemas Baseados em Regras 22
3.2.3 Sistemas de Classificadores de Palavras 22
3.3 O Problema do Domínio 28
3.4 Considerações Finais 28

4 Representações de Palavras **31**
4.1 Motivação 31
4.2 Modelos Distribucionais 32
4.2.1 *Hyperspace Analogue to Language* 34
4.2.2 *Random Indexing* 34
4.3 Modelos Neurais 35

4.4 Experimentos 36
4.4.1 Pré-Processamento 36
4.4.2 Resultados 38
4.5 Considerações Finais 40

5 Implementação e Resultados **43**
5.1 Implementação 43
5.1.1 Implementação de Dois Estágios 44
5.1.2 Identificação de Predicadores 45
5.2 Resultados 46
5.2.1 Arquitetura de Um Estágio 49
5.2.2 Arquitetura de Dois Estágios 49
5.3 Considerações Finais 53

6 Conclusões **55**

A Exemplos de Anotação Automática **57**

Referências Bibliográficas **65**

Lista de Figuras

2.1 Exemplo de estrutura profunda 6
2.2 Distribuição de argumentos no PropBank-Br 13

3.1 Árvore sintática para a sentença exemplo 19
3.2 Exemplo de obtenção de vetores de palavras 23
3.3 Janela de 3 palavras sendo convertida em vetores numéricos. 24
3.4 Estrutura da rede neural. 24
3.5 Exemplo de etiquetação IOBES para APS 26
3.6 Exemplo de etiquetação de *chunks* usada pelo SENNA. 27

5.1 Exemplo de etiquetação IOB para APS 44
5.2 Exemplo de etiquetação IOBES para delimitação de argumentos 45
5.3 Árvore sintática para a sentença mostrada na Figura 5.4. 48
5.4 Exemplo de etiquetação de *chunks* extraída do PropBank-Br. 48

A.1 Resposta do sistema para sentença com dois predicadores. 58
A.2 Resposta do sistema com verbo de movimento. 58
A.3 Resposta do sistema para sentença sem predicador. 59
A.4 Resposta do sistema com erro na detecção do argumento local. 59

Lista de Tabelas

2.1 Características dos papéis prototípicos 7
2.2 Argumentos dos verbos *aproveitar* e *alugar* no PropBank-Br 9
2.3 Argumentos modificadores usados no PropBank-Br 10
2.4 Estatísticas de argumentos no PropBank-Br 12

3.1 Exemplo de convolução sobre uma sentença 25

4.1 Tabela de co-ocorrência para a sentença *Só sei que nada sei.* 34
4.2 Resultados de semelhança lexical com HAL 39
4.3 Resultados de semelhança lexical com RI 40
4.4 Resultados de semelhança lexical com modelo neural 41

5.1 Valores iniciais para a tabela de transições entre etiquetas 45
5.2 Exemplo de convolução sobre uma sentença para classificar argumentos . . . 46
5.3 Valores de comparação para APS 47
5.4 Resultados da APS com a arquitetura de um estágio 49
5.5 Resultados da identificação de argumentos na arquitetura de dois estágios . . 50
5.6 Resultados da classificação de argumentos na arquitetura de dois estágios . . 50
5.7 Resultados da APS em dois estágios 51
5.8 Resultados da APS sem rótulos de argumentos repetidos 52
5.9 Resultados da APS com predicadores identificados automaticamente 52
5.10 Resultados do sistema final separados por tipo de argumento. 53

Lista de Abreviações

AM Aprendizado de Máquina

APS Anotação de Papéis Semânticos

HAL *Hyperspace Analogue to Language*

IOB *In, Out, Begin* (esquema de anotação de sequências)

IOBES *In, Out, Begin, End, Single* (esquema de anotação de sequências)

IOE *In, Out, End* (esquema de anotação de sequências)

MLP *Multilayer Perceptron*

OOV *Out of vocabulary*

PLN Processamento de Línguas Naturais

RI *Random Indexing*

SVM *Support Vector Machines*

VSM *Vector Space Model*

Capítulo

1

Introdução

1.1 Contextualização

O processamento de línguas naturais (PLN) compreende a aplicação de técnicas computacionais para diversos tipos de tratamento de textos. Esses incluem uma vasta gama de aplicações: correção ortográfica e gramatical, tradução automática, interface de busca para um banco de dados, etc. Inúmeros sistemas bem sucedidos, com maior ou menor grau de complexidade, estão difundidos para esses tipos de aplicação como, por exemplo, o tradutor automático do Google e os corretores de escrita do Microsoft Office.

Dentro do PLN, uma tarefa que vem recebendo bastante atenção nos últimos anos é a Anotação de Papéis Semânticos (APS) (Palmer et al., 2010). Essa tarefa consiste em identificar, em sentenças escritas em língua natural, estruturas que indiquem **quem** fez **o quê** para **quem**, **onde** e **como**. Esta é uma forma de interpretação do texto no nível sentencial – isto é, considerando a leitura de cada sentença isoladamente – e por isso a APS é também chamada de *parsing* semântico de superfície.

Mais especificamente, o objetivo da APS é subdividido em dois: em primeiro lugar, identificar um predicador e seus argumentos. O predicador é normalmente um verbo e seus argumentos costumam ser o sujeito, objetos direto e indireto e adjuntos. Caso o predicador pertença a outra classe gramatical, como um adjetivo ou substantivo, os argumentos podem ser praticamente qualquer outro constituinte da oração.

A segunda etapa da APS é atribuir um rótulo (o papel semântico) à relação estabelecida entre cada par de predicador e argumento. Não há consenso sobre o conjunto de todos os papéis semânticos possíveis (Dowty, 1991), mas é comum falar-se em papéis como AGENTE

(que realiza voluntariamente uma ação), PACIENTE (que sofre o resultado de uma ação), BENEFICIÁRIO (para quem uma dada ação é realizada), DESTINO (para onde algo vai), etc.

Há uma forte correlação entre alguns papéis semânticos e funções gramaticais. Por exemplo, um AGENTE é normalmente realizado como sujeito e um PACIENTE, como objeto direto; por isso, a análise sintática de uma sentença é uma informação valiosa para a determinação de papéis semânticos da mesma.

No entanto, há muitos casos em que somente a sintaxe não fornece informação suficiente. Considere os exemplos a seguir, em que os constituintes de cada sentença são rotulados de acordo com sua relação semântica com o verbo:

(1) a. [$_{\text{AGENTE}}$ O menino] quebrou [$_{\text{PACIENTE}}$ o vaso].

b. [$_{\text{PACIENTE}}$ O vaso] quebrou.

Como se pode notar, o sujeito e o objeto de uma oração podem assumir diferentes papéis semânticos, dependendo do significado expresso pelo verbo. Comparando (1a) e (1b), percebe-se que o mesmo verbo pode atribuir diferentes papéis para o seu sujeito, dependendo do seu significado. No caso, *menino* recebe o papel de AGENTE, enquanto *vaso* o de PACIENTE.

A questão crucial aqui é: considerando os elementos envolvidos na sentença e seus significados individuais, como determinar qual a relação que se estabelece entre eles? Um sistema que consiga responder esta pergunta de forma satisfatória não é uma aplicação voltada para o usuário final, mas para auxiliar outros sistemas de PLN.

Màrquez et al. (2008) apontam que as aplicações de APS em sistemas no mundo real têm sido poucas até então, mas virtualmente todas as áreas do PLN que lidam com semântica podem se beneficiar dela. Em particular, sistemas de extração de informação (Surdeanu et al., 2003) e de perguntas e respostas (Kaisser e Webber, 2005) parecem se beneficiar mais diretamente, mas também há trabalhos que empregam APS para sumarização automática (Melli et al., 2005) e tradução estatística (Wu e Fung, 2009).

Collins (1999, cap. 9) menciona que seria ideal para a tarefa de *parsing* sintático que sintaxe e semântica fossem analisadas simultaneamente. O resultado da análise sintática é normalmente fornecido como dado de entrada para a APS; no entanto, algumas ambiguidades sintáticas poderiam ser facilmente resolvidas com tratamento semântico.

O primeiro trabalho desenvolvido para resolver a tarefa de APS é apresentado por Gildea e Jurafsky (2002), feito para a língua inglesa e empregando aprendizado de máquina. Desde então, novos trabalhos trazendo melhorias têm sido reportados tanto para o inglês como para outras línguas (Xue e Palmer 2004; Haghighi et al. 2005; Pradhan et al. 2008; Xue 2008), mas ainda praticamente não foram explorados em português.

Nos últimos anos, o uso de técnicas de aprendizado de máquina supervisionado para o problema de APS tornou-se viável graças à compilação de grandes *corpora* anotados manualmente com papéis semânticos, como o PropBank (Palmer et al., 2005) e a FrameNet (Baker et al., 1998). Dessa forma, é possível treinar um classificador nessas bases e também usá-las como *benchmarks*. Duas edições da CoNLL (*Conference on Computational Natural Language Learning*), uma das conferências internacionais mais importantes para o PLN, propuseram competições de sistemas de APS, refletindo o crescente interesse da comunidade neste ramo de pesquisa (Carreras e Màrquez, 2004; Carreras e Màrquez, 2005). Nestas competições, chamadas de tarefas compartilhadas (*shared tasks*), foram fornecidos dados de treinamento e teste originários do PropBank para o desenvolvimento de sistemas baseados em aprendizado de máquina.

Um fator impeditivo para o uso destas técnicas no português tem sido a ausência de *corpora* anotados com papéis semânticos. Até a presente data, o único recurso existente é o PropBank-Br[1] (Duran e Aluísio, 2011; Duran e Aluísio, 2012), relativamente pequeno e que ainda se deseja ampliar.

A alternativa ao aprendizado de máquina é o desenvolvimento de um sistema simbólico, com regras pré-determinadas para a realização da APS. Tal abordagem é utilizada em Bick (2007), mas apresenta uma série de dificuldades e limitações que serão discutidas na Seção 3.2.2.

Outra característica marcante da maioria das abordagens para a APS é o uso de uma grande quantidade de atributos para a classificação automática, normalmente provenientes da análise sintática da sentença de entrada. Apesar de contribuir para melhorias nos resultados do sistema, a inclusão de muitos atributos tem um custo considerável em termos de performance, pois requer a execução de outras ferramentas (como um *parser* sintático), além de uma etapa de pós-processamento de suas saídas. Visto da perspectiva da língua portuguesa, esta situação é ainda mais problemática, ao se considerar as limitações no que diz respeito à disponibilidade de ferramentas de PLN.

Em contraste com esta tendência, Collobert e Weston (2008) propuseram uma arquitetura baseada em redes neurais para realizar tanto a APS como outras tarefas de análise sentencial, empregando uma quantidade mínima de atributos. Sua arquitetura se baseia na codificação de palavras como vetores de números reais, sem conhecimento linguístico explícito. Esta abordagem permite que se escape da dependência de um *parser* sintático, e constituiu a maior inspiração para o trabalho descrito aqui.

Neste projeto, foi implementado o modelo de Collobert e Weston, incluindo alterações em sua arquitetura que permitiram um bom ganho de performance em comparação com o modelo original. Foram também investigadas técnicas para indução de representações

[1] Disponível em `http://www.nilc.icmc.usp.br/portlex`

numéricas de palavras, empregadas diretamente para realizar a APS.

1.2 Objetivos

O presente trabalho partiu da hipótese de que é possível desenvolver um sistema baseado em aprendizado de máquina para a APS em português sem o uso de atributos provenientes de ferramentas externas, nos moldes de Collobert e Weston (2008). Para tal, é necessário que se tenha representações na forma de vetores numéricos para uma grande quantidade de palavras. Ainda segundo a hipótese, se essas representações forem boas o suficiente, o sistema será capaz de generalizar seu aprendizado para novos dados, incluindo palavras não vistas nos dados de treino de APS.

Neste trabalho, investigaram-se formas de obter representações de palavras, bem como a parametrização ideal de um classificador automático que as empregue na tarefa de APS em língua portuguesa. Como base para treino e teste de classificadores, foi empregado o PropBank-Br (Duran e Aluísio, 2012). O sistema resultante, os principais dados empregados, código fonte e documentação encontram-se disponíveis em `https://github.com/erickrf/nlpnet`.

1.3 Organização da Dissertação

Os capítulos seguintes desta dissertação estão organizados da seguinte forma: no Capítulo 2, são apresentados conceitos linguísticos de papéis semânticos, assim como recursos computacionais preparados para se trabalhar com a tarefa de APS; no Capítulo 3, são revisados trabalhos relacionados e explicadas as diferentes abordagens encontradas na literatura; no Capítulo 4, são mostradas técnicas para geração de representações vetoriais de palavras, sua importância para este projeto e experimentos realizados para obtê-las; no Capítulo 5, é relatada a implementação do sistema proposto para APS e sua avaliação experimental; o Capítulo 6 encerra o texto com as conclusões do trabalho. O Apêndice A apresenta alguns exemplos da execução do sistema desenvolvido.

Capítulo

2

Papéis Semânticos

Este capítulo apresenta os conceitos linguísticos de papéis semânticos e o PropBank-Br, um *corpus* com papéis semânticos anotados para auxiliar pesquisas de PLN nesta área. Por se tratar de um assunto amplamente discutido em Linguística (Levin e Rappaport Hovav, 2005), incluindo muitos pontos que não são relevantes do ponto de vista computacional, a revisão bibliográfica apresentada aqui não será exaustiva, limitando-se aos pontos de maior interesse para o presente trabalho. Considerar-se-ão apenas os predicadores verbais, pois, além de serem o caso mais comum e mais estudado, são os únicos marcados no PropBank-Br.

2.1 Conceitos Linguísticos

Em linhas gerais, o estudo de papéis semânticos tem como objetivo compreender a relação que se estabelece entre sintaxe e semântica em estruturas de predicador e argumentos. Esta não é uma tarefa trivial, já que sentenças sintaticamente semelhantes podem apresentar papéis semânticos diferentes, ou vice-versa, como mostrado nos exemplos (1a) e (1b).

Já por volta do século 4 A.C., a gramática do sânscrito desenvolvida por Pāṇini incluía noções de papéis semânticos. No entanto, o assunto foi muito pouco discutido na Linguística moderna até ser posto novamente em foco por Fillmore (1968).

Nesse trabalho, o autor defende a ideia de que, em um nível abstrato (também chamado estrutura profunda), qualquer oração em língua natural possui uma estrutura composta pelo verbo e seus argumentos, que são realizados como constituintes que podem ter diferentes funções sintáticas. Fillmore classifica os argumentos em seis casos semânticos:

Agentivo Entidade, normalmente animada, instigadora da ação descrita pelo verbo.

Instrumental Força ou objeto envolvido na ação ou estado descrito pelo verbo.

Dativo Entidade, normalmente animada, afetada pelo estado ou ação descrito pelo verbo.

Factitivo Objeto ou ser resultante da ação ou estado descrito pelo verbo, ou compreendida como parte do seu significado.

Locativo Local ou orientação espacial da ação ou estado descrito pelo verbo.

Objetivo Caso semanticamente neutro; o significado de um argumento com este caso é determinado a partir do significado do próprio verbo.

O autor ressalva que esta não é uma lista definitiva, mas que haveria de receber outras adições. A teoria prevê ainda que cada um destes casos só pode ocorrer no máximo uma vez por oração e cada constituinte só pode assumir um caso.

Também segundo Fillmore, modificadores como verbos auxiliares modais, negação, advérbios que modificam o verbo principal, etc., são incluídos na sentença independentemente da estrutura profunda, assim como o tempo verbal.

Na Figura 2.1 é mostrado um exemplo de estrutura profunda que apresenta o verbo *abrir* e três argumentos: *homem*, como AGENTIVO; *porta*, como OBJETIVO e *chave*, como INSTRUMENTAL. A estrutura pode ser realizada por qualquer uma das sentenças de (2a) a (2d).

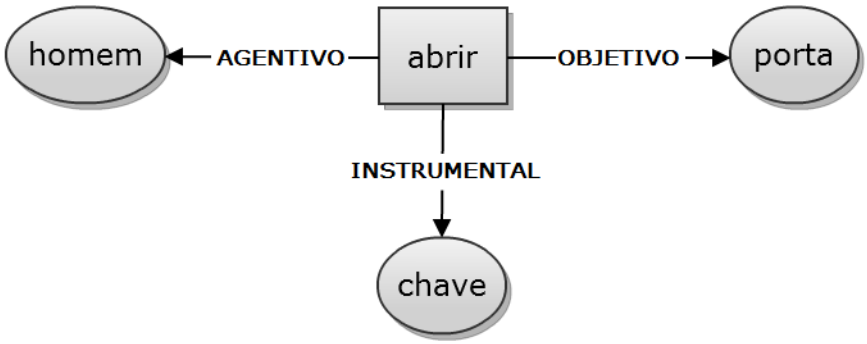

Figura 2.1: Exemplo de estrutura profunda

(2) a. O homem abriu a porta com a chave.

b. A porta foi aberta pelo homem com a chave.

c. O homem não vai abrir a porta com a chave.

d. O homem abre a porta apressadamente com a chave.

Nos anos subsequentes, a definição de um conjunto de papéis universalmente aceito foi motivo para muita discussão entre autores, sem que se chegasse, no entanto, a uma listagem consistente e definitiva. Um grande problema que pode ser apontado, segundo Levin e Rappaport Hovav (2005, cap. 2), é a falta de um método preciso e livre de ambiguidade para se determinar o papel de um determinado argumento.

Dowty (1991) propõe uma alternativa mais flexível à definição de um conjunto de papéis semânticos. Em vez de atribuir papéis de um conjunto pré-determinado aos argumentos de um verbo, o autor atribui acarretamentos lógicos. A ideia dos acarretamentos é que todo verbo exige certas pré-condiçoes de cada um de seus argumentos. Por exemplo, o verbo *procurar* exige que seu sujeito seja um ser consciente que esteja agindo voluntariamente[2]. Já o verbo *construir* exige que seu objeto seja algo que não existia previamente. Para cada verbo, há um conjunto de acarretamentos específicos associados a cada argumento.

A partir daí, ele define os papéis prototípicos de PROTO-AGENTE e PROTO-PACIENTE, cada um relacionado a uma série de acarretamentos, como mostrado na Tabela 2.1. Na prática, argumentos verbais normalmente possuem apenas algumas destas características, ou mesmo características dos dois papéis prototípicos. Dowty explica que os tipos de argumentos estão espalhados ao longo de um contínuo, em vez de se limitarem a um conjunto discreto.

PROTO-AGENTE	PROTO-PACIENTE
Envolvimento voluntário no evento	Sofre mudança de estado
Senciência / Percepção	Alterado incrementalmente
Causa um evento ou mudança de estado em outro participante	Afetado causalmente por outro participante
Movimento (relativo a outro participante)	Estacionário relativo a movimento de outro participante
Existe independente do evento	Não existe idependentemente do evento

Tabela 2.1: Características dos papéis prototípicos

Seguindo este raciocínio, uma listagem de todos os papéis semânticos possíveis deveria considerar cada uma das combinações de acarretamentos produzida por cada verbo da língua. Isso torna-se inviável quando consideramos a quantidade de especificidades definidas por muitos verbos. No entanto, entender o funcionamento dos argumentos verbais de uma forma flexível é a visão mais aceita na Linguística atualmente (Levin e Rappaport Hovav, 2005).

[2]Por voluntariamente, neste contexto, entenda-se controlando suas ações, mesmo que coagido a tal.

Levin (1993) traz um estudo detalhado sobre os padrões de ocorrências de verbos e argumentos para o inglês. Nesse trabalho, a autora analisa mais de 3000 verbos, que são separados em 47 classes principais, de acordo com possíveis realizações sintáticas dos papéis semânticos aceitos pelo verbo. Cada uma é subdivida em dezenas de outras subclasses, que agrupam verbos com comportamentos mais específicos. A ideia dessa classificação é que verbos com comportamento sintático semelhante tendem a possuir significados semelhantes. As classes tornaram-se conhecidas como "classes de Levin"

Os pares (3a) e (3c), acompanhados de formas análogas em português, exemplificam respectivamente as classes 45.1 e 21.1. A primeira inclui verbos como *chip*, *crack* e *smash*; exemplos da segunda são *hack*, *scratch* e *saw*.

(3) a. John broke the vase. / The vase broke.

b. João quebrou o vaso. / O vaso quebrou.

c. John cut the bread. / * The bread cut.

d. João cortou o pão. / * O pão cortou.

Levin observa que verbos de uma mesma classe possuem significados relacionados. A autora associa a cada classe uma característica que descreva o que os seus verbos têm em comum e explique os tipos de realizações sintáticas possíveis para eles. Por exemplo, para a classe do verbo *break*, é associada a ideia de mudança de estado imediata, enquanto que para a do verbo *cut*, é associada a ideia de ação contínua que resulta numa mudança de estado.

Embora não lide diretamente com papéis semânticos, o trabalho de Levin é importante para a área por tratar da conexão, ainda mal compreendida, entre sintaxe e semântica e explicitar padrões de comportamento verbal.

2.2 O PropBank

O PropBank[3] (Palmer et al., 2005) é um recurso linguístico-computacional que apresenta uma camada de anotação com papéis semânticos sobre um subconjunto do Penn Treebank. Dado que o Penn Treebank já possui árvores sintáticas para todas as suas sentenças, o PropBank é implementado na forma de arquivos que indicam, para cada árvore sintática, quais nós não-terminais correspondem a predicadores e quais a seus argumentos. Os textos que compõem o Penn Treebank são oriundos em sua maioria do Wall Street

[3]Mais informações em `http://verbs.colorado.edu/~mpalmer/projects/ace.html`

Journal, publicação jornalística voltada para temas financeiros, e uma pequena parcela do *corpus* Brown, composto por textos literários de diversos autores e períodos.

Similarmente, o PropBank-Br[4] foi construído sobre a porção de textos brasileiros do *corpus* Bosque (Afonso et al., 2002), que também possui anotação sintática corrigida. Os conceitos de argumentos são os mesmos do PropBank original, assim como os rótulos empregados. Os textos que compõem o Bosque foram extraídos do jornal Folha de São Paulo e incluem diversos assuntos. O PropBank-Br é implementado como um arquivo XML que contém a descrição da árvore sintática de cada sentença, bem como quais nós internos correspondem a predicadores e a argumentos.

Outro recurso existente para a APS é a FrameNet (Baker et al., 1998), que segue um modelo inspirado na semântica de Frames de Fillmore (Fillmore, 1985). Também há uma versão em português da FrameNet em construção (Salomão, 2009), no entanto, não foi empregada neste projeto. Segundo Màrquez et al. (2008), trabalhos com APS em inglês usam com mais frequência o PropBank do que a FrameNet.

O modelo de anotação do PropBank define um conjunto de argumentos para cada verbo individualmente e não atribui nomes aos, utilizando, em vez disso, rótulos numerados de **Arg0** a **Arg5**. Cada verbo determina a quantidade de argumentos que podem acompanhá-lo e seus significados [5]. A Tabela 2.2 apresenta, como exemplos, os argumentos dos verbos *aproveitar* e *alugar* do PropBank-Br.

Verbo	**Arg0**	**Arg1**	**Arg2**	**Arg3**	**Arg4**
aproveitar	quem tira proveito	o que é aproveitado	finalidade	–	–
alugar	proprietário	bem alugado	inquilino	valor do alugel	prazo

Tabela 2.2: Argumentos dos verbos *aproveitar* e *alugar* no PropBank-Br

Os papéis definidos, no entanto, não são necessariamente realizados em todas as sentenças em que um verbo ocorre, nem mesmo implicitamente. O PropBank não define argumentos obrigatórios para os verbos, apenas os que *podem* ser encontrados. Observe-se ainda que um verbo com mais de um significado pode definir diferentes conjuntos de argumentos, um para cada sentido.

Além disso, o PropBank define um conjunto de argumentos opcionais comuns a todos os verbos, os **ArgM**, que são expressões indicando tempo, local, modo, etc.[6], e frequentemente

[4]Disponível em `http://www.nilc.icmc.usp.br/portlex`

[5]O PropBank lista apenas verbos como predicadores.

[6]Em Linguística, costuma-se diferenciar *argumento* e *adjunto*, que correspondem, respectivamente, aos **Arg0-5** e **Arg-M**. O termo *argumento* é usado neste trabalho para se referir a ambos, a fim de evitar confusão com a nomenclatura adotada pelo PropBank.

Etiqueta	Significado
ArgM-ADV	Modificador adverbial para a sentença inteira
ArgM-CAU	Causa
ArgM-DIR	Direção
ArgM-DIS	Marcador discursivo
ArgM-EXT	Extensão
ArgM-LOC	Lugar
ArgM-MNR	Maneira
ArgM-NEG	Negação
ArgM-PNC	Propósito
ArgM-PRD	Predicação secundária
ArgM-REC	Recíproco, referenciando outro argumento
ArgM-TMP	Temporal

Tabela 2.3: Argumentos modificadores usados no PropBank-Br

realizados como adjuntos adverbiais. Os rótulos destes argumentos são complementados com um sufixo indicando o seu significado; na Tabela 2.3 são listados todos os **Arg-M** do PropBank-Br. Um exemplo de sentença com papéis semânticos anotados é mostrado em (4).

(4) [$_{\text{ArgM-DIS}}$ Ou seja], **alugam** [$_{\text{Arg1}}$ o imóvel] [$_{\text{Arg4}}$ por um prazo máximo de 90 dias] e ainda podem receber o valor adiantado.

O mais próximo de uma generalização de papéis neste projeto, além dos **ArgM**, são os argumentos **Arg0** e **Arg1**, que correspondem, respectivamente, ao argumento mais próximo do PROTO-AGENTE e do PROTO-PACIENTE de Dowty (1991). Por isso, é importante que os argumentos sejam analisados em conjunto com o predicador para que se possa saber seu significado.

Para comparar o desempenho de diferentes sistemas, é comum usar um subconjunto do *corpus* para testes, deixando o restante como dados de treino para um classificador automático. No caso do PropBank, o padrão é se usar a divisão testes do CoNLL 2005 (Carreras e Màrquez, 2005).

Em seu trabalho com a APS em português, Alva Manchego e Rosa (2012) dividiram o PropBank-Br em cerca de 95% para treino e 5% para teste, aproximadamente a mesma proporção encontrada para o inglês. Do total de 3.308 sentenças, 3.164 são usadas para treino e 144 para teste, fornecendo, respectivamente, 5.537 e 239 proposições (uma proposição é a ocorrência de um predicador e seus argumentos). A mesma divisão foi usada neste trabalho. A quantidade de proposições é menor que a reportada por Duran e Aluísio (2012) devido ao fato de Alva Manchego e Rosa terem descartado sentenças que apresentassem erros na análise sintática. Tais erros foram identificados pela impossibilidade de se construir uma árvore sintática de acordo com a descrição dada no arquivo XML do PropBank-Br.

Para os experimentos deste projeto, foi realizada uma etapa de pré-processamento visando a correção de alguns pequenos erros encontrados no *corpus*, bem como tornar os dados mais tratáveis. As alterações aplicadas são listadas a seguir:

- Corrigidos alguns erros em que o sinal de percentagem era precedido pelo símbolo $, proveniente da saída do *parser* PALAVRAS.
- Corrigidos alguns usos do caracter _ em vez de hífen, como em *quinta_feira*.
- Removida a única ocorrência de um argumento do tipo **Arg5** em todo o *corpus*. Tal decisão foi tomada devido ao fato de ser extramamente difícil que um sistema automatizado pudesse generalizar seu aprendizado a respeito de um tipo de argumento com um único exemplo; além disso, não seria possível que houvesse um **Arg5** nos dados de treino e de teste.
- Alguns argumentos são constituídos por mais de um bloco de palavras não contínuo. Nesses casos, apenas um dos blocos recebe a etiqueta normal do argumento, e os outros a recebem com o prefixo **C-**, conforme exemplificado em 5. Nos experimentos realizados, todos os prefixos **C-** foram removidos.
- Refeitas contrações de palavras como *do*, *no*, *daquele*, etc., que aparecem separadas no *corpus*.

(5) **Tem** $[_{\text{Arg1}}$ sentido$]$ – $[_{\text{ArgM-DIS}}$ aliás$]$, $[_{\text{C-Arg1}}$ muitíssimo sentido$]$.

A Tabela 2.4 e a Figura 2.2 apresentam a distribuição de argumentos no PropBank-Br após a aplicação do pré-processamento descrito acima. É interessante notar que predicadores (denotados pela etiqueta `V`) podem ser compostos por mais de um *token*. Isso se deve a alguns verbos que são necessariamente acompanhados pela partícula *-se*, ou que adquirem significado especial quando usados em certas expressões. Por exemplo, *bater palmas* constitui um único predicador, pois seu significado é diferente do verbo *bater*.

2.3 Considerações Finais

Foram mostrados os conceitos linguísticos de papéis semânticos, bem como as dificuldades inerentes a tentativas de se elaborar uma explicação sistemática deles. Também foi apresentada a estrutura do PropBank-Br, o principal recurso disponível atualmente para a tarefa de APS em português, e que foi empregado no desenvolvimento deste projeto.

	Treino		Teste		Total	
Rótulo	**Argumentos**	***Tokens***	**Argumentos**	***Tokens***	**Argumentos**	***Tokens***
A0	2.940	10.771	124	454	3.064	11.225
A1	4.969	33.111	212	1.267	5.181	34.378
A2	1.065	6.421	38	164	1.103	6.585
A3	112	566	2	5	114	571
A4	74	360	1	2	75	362
AM-ADV	349	1.573	20	102	369	1.675
AM-CAU	155	1.239	1	5	156	1.244
AM-DIR	13	39	2	8	15	47
AM-DIS	283	813	11	30	294	843
AM-EXT	80	183	1	1	81	184
AM-LOC	737	3.511	27	137	764	3.648
AM-MNR	390	1.908	18	47	408	1.955
AM-NEG	318	339	19	19	337	358
AM-PNC	166	1.324	5	53	171	1.377
AM-PRD	186	1.260	6	33	192	1.293
AM-REC	60	60	5	13	65	73
AM-TMP	1.080	4.144	45	174	1.125	4.318
V	5.543	5.771	239	249	5.782	6.020
O	–	57.697	–	2.242	–	59.939
Total*	12.977	131.090	537	5.005	13.514	136.095

* Na contagem total de argumentos, é ignorado o número de predicadores. Na contagem de *tokens*, predicadores são incluídos.

Tabela 2.4: Estatísticas de argumentos no PropBank-Br

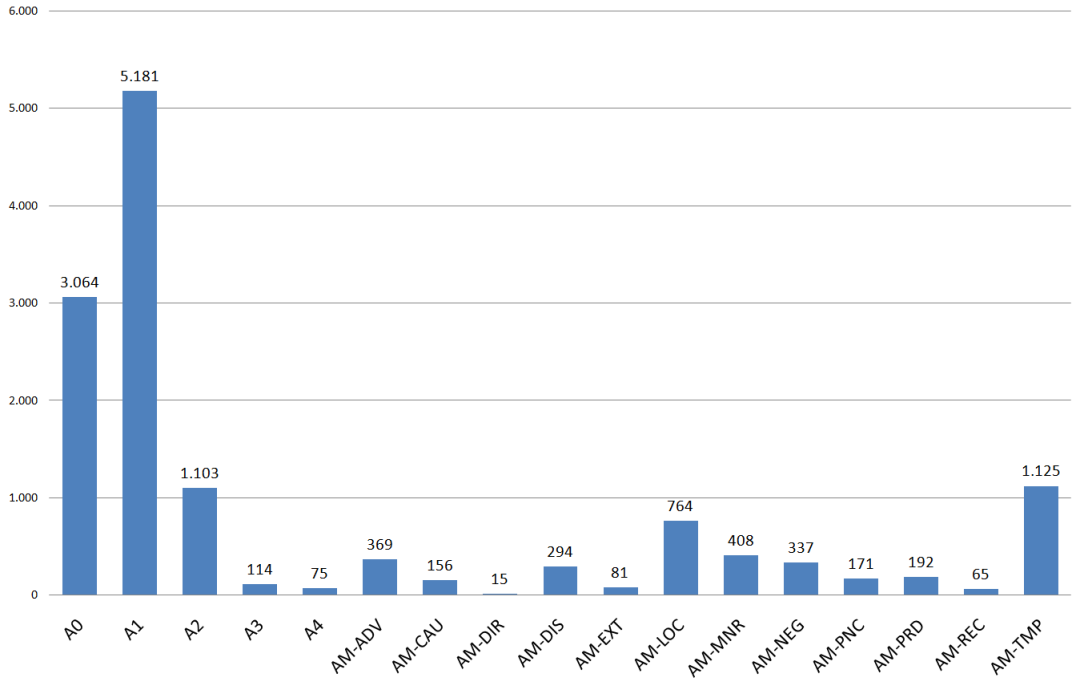

Figura 2.2: Distribuição de argumentos no PropBank-Br

Capítulo

3

Aprendizado de Máquina e Anotação de Papéis Semânticos

Este capítulo apresenta os principais aspectos da tarefa de APS. O aprendizado de máquina, empregado pela quase totalidade dos sistemas que realizam esta tarefa, é tratado aqui. Redes neurais artificiais, uma técnica que emprega aprendizado de máquina e é de especial interesse ao presente trabalho, são também apresentadas. Trabalhos que tratam da APS, que já existem em boa quantidade para o inglês, são analisados. Em especial, é tratado o sistema SENNA (Collobert e Weston, 2008; Collobert et al., 2011), que constituiu a principal inspiração para este projeto.

3.1 Aprendizado de Máquina

Aprendizado de máquina é uma subárea da Inteligência Artificial e da Estatística que consiste, em linhas gerais, do estudo de algoritmos que possibilitem a detecção de padrões em conjuntos de dados. Estes padrões podem servir para dois propósitos: prever alguma informação em dados novos (aprendizado preditivo) ou descrever o comportamento dos dados apresentados (aprendizado descritivo).

O problema de *classificação*, que é de interesse para o presente trabalho, encontra-se na categoria de aprendizado preditivo. Nele, considera-se um conjunto de dados X. Cada elemento $x \in X$ é representado por um vetor de atributos $(x_1, x_2, \ldots, x_n)$ e possui um rótulo y. O valor de cada atributo $x_i, 1 \leq i \leq n$, pode ser um número inteiro, real ou categórico (isto é, podendo assumir apenas valores dentro de uma dada lista, como dias da semana).

O rótulo y, também chamado classe, deve ser discreto; quando este é contínuo, diz-se que se trata um problema de *regressão*.

O objetivo de um algoritmo de classificação é, dado o conjunto de dados X, determinar uma função $\hat{f} : (x_1, x_2, \ldots, x_n) \mapsto y$. Isto é, deseja-se obter um modelo que determine a classe de novos exemplos representados por um vetor de atributos $(x_1, x_2, \ldots, x_n)$. Para poder determinar a classe correspondente a combinações de atributos nunca encontradas antes, a capacidade de generalização é fundamental.

3.1.1 Redes Neurais Artificiais

Redes neurais artificiais são estruturas abstratas que simulam o funcionamento dos neurônios do cérebro humano. Sua estrutura interna pode ser organizada de acordo com diversas arquiteturas, dependendo do tipo de tarefa com que se lida (Haykin, 2009). Para este trabalho, é de interesse a arquitetura conhecida como *Multilayer Perceptron* (MLP), ou Percéptron Multicamadas, que será explicada a seguir.

Redes neurais são constituídas de neurônios artificiais, estruturas abstratas que recebem uma série de valores numéricos como entrada, efetuam operações aritméticas sobre eles e emitem o resultado calculado. Dependendo da arquitetura da rede, a saída de um neurônio pode servir como entrada para outros, ou mesmo para si próprio. Às conexões entre neurônios dá-se o nome de *sinapse*, analogamente às conexões entre neurônios biológicos. As sinapses possuem pesos associados, na forma de valores numéricos, que indicam o quão forte é aquela conexão, e podem ser *excitatórias* (com pesos positivos) ou *inibitórias* (com pesos negativos). Os pesos sinápticos são ajustados durante o treinamento da rede neural.

Na arquitetura MLP, a rede neural é organizada em *camadas* sequenciais, de modo que os neurônios de uma camada tenham suas saídas conectadas às entradas dos neurônios da camada seguinte. Caso haja conexão de todos os neurônios de uma camada com os da próxima, a rede é dita *totalmente conectada*; caso contrário, é chamada de *parcialmente conectada*. Nesse tipo de arquitetura, os dados de entrada da rede são acessados diretamente pelos neurônios da primeira camada, enquanto os neurônios das camadas subsequentes recebem apenas o sinal de saída dos neurônios anteriores.

As camadas de neurônios antes da última são chamadas *camadas ocultas*. Elas permitem a detecção de características nos dados de entrada que seriam impossíveis de serem descobertas com uma única camada. No entanto, para que esta propriedade seja válida, é necessário que a função de ativação dos neurônios de cada camada oculta seja não-linear. Exemplos de funções frequentemente empregadas com este fim são a função logística e a tangente hiperbólica.

A resposta da rede para uma determinada entrada corresponde aos valores de saída dos neurônios de sua última camada. Em problemas de classificação, normalmente há um

neurônio para cada classe do problema, de modo que a classe atribuída aos dados será aquela cujo neurônio tiver o maior valor de saída.

O treinamento da rede se constitui basicamente do ajuste dos pesos sinápticos e emprega o algoritmo de retropropagação (*backpropagation*). Sua ideia central é determinar a taxa de variação do erro da rede em função de seus pesos, utilizando cálculo diferencial. Desta forma, os pesos são ajustados para valores que minimizem o erro nos exemplos encontrados, o que corresponde a caminhar no espaço de possíveis parâmetros da rede na direção do gradiente descendente.

O treinamento de redes neurais é normalmente medido em *épocas*, onde cada época corresponde à execução do algoritmo de retropropagação sobre todos os exemplos de treinamento. Normalmente, o treinamento é feito até que algum critério seja atingido, como passar um determinado número de épocas ou que a taxa de acertos fique acima de um limiar.

A natureza das redes MLP de lidar com dados numéricos e sua capacidade de calcular pequenos ajustes para serem aplicados gradativamente em seus parâmetros é particularmente interessante para os propósitos deste trabalho, como será mostrado na Seção 4.3.

A principal dificuldade do uso de redes neurais é a escolha de bons valores para seus parâmetros, como a taxa de aprendizado (que determina quão rapidamente são ajustados os pesos sinápticos), a quantidade de neurônios e de camadas da rede. Não há métodos para uma escolha adequada de valores que funcionem em qualquer situação; normalmente é necessário que estes sejam determinados experimentalmente.

3.2 Sistemas de APS

Para realizar a tarefa de APS, é preciso que seja definido qual formalismo de anotação será utilizado, isto é, qual o conjunto de papéis para serem atribuídos aos argumentos de predicadores. Virtualmente todos os sistemas feitos para a língua inglesa utilizam ou o modelo de anotação da FrameNet ou o do PropBank, com este último sendo mais dominante nos últimos anos.

A avaliação de desempenho é feita empregando-se os conceitos de precisão e cobertura, como é comum na área de Aprendizado de Máquina. A precisão refere-se à proporção de predições corretas do sistema, dentro do total de predições feitas. A cobertura mede a proporção entre a quantidade de papéis semânticos identificados corretamente e o total que havia para ser identificado. Além destas duas, é comum empregar-se a medida F, calculada como média harmônica entre ambas:

$$\text{medida F} = 2 \cdot \frac{\text{precisão} \cdot \text{cobertura}}{\text{precisão} + \text{cobertura}}$$

A seguir são relatados as principais arquiteturas de sistemas de APS encontradas na literatura.

3.2.1 Sistemas de Classificadores de Constituintes

A maioria dos sistemas de APS reportados na literatura recebe como entrada uma sentença, extrai sua árvore sintática e, para cada predicador presente nela, realiza um processo de duas etapas principais, listadas a seguir (Màrquez et al., 2008). Considere a sentença (6a), extraída do PropBank-Br, e sua árvore sintática mostrada na Figura 3.1 para ilustrar o processo. A anotação é feita em relação ao predicador *estima*.

Delimitar os argumentos do predicador Esta etapa inclui reconhecer quais constituintes da sentença são argumentos do predicador (um constituinte equivale a todas as palavras geradas a partir de um nó não-terminal da árvore sintática). A saída desta primeira etapa corresponde a (6b). Note-se que um argumento pode englobar vários subconstituintes sintáticos, inclusive verbos, como é o caso em *o tempo necessário para classificar as áreas e culturas plantadas em uma extensão de 180 quilômetros*.

Determinar o rótulo de cada argumento Dados os argumentos delimitados, resta atribuir-lhes os rótulos corretos. A saída do anotador corresponde a (6c).

(6) a. Ele **estima** em dez horas o tempo necessário para classificar as áreas e culturas plantadas em uma extensão de 180 quilômetros.

b. [Ele] **estima** [em dez horas] [o tempo necessário para classificar as áreas e culturas plantadas em uma extensão de 180 quilômetros].

c. [$_{\text{Arg0}}$ Ele] **estima** [$_{\text{Arg2}}$ em dez horas] [$_{\text{Arg1}}$ o tempo necessário para classificar as áreas e culturas plantadas em uma extensão de 180 quilômetros].

Para cada um dos dois passos acima, é resolvido um problema de classificação. No primeiro, todos os nós não-terminais da árvore são classificados como argumentos ou não-argumentos. No segundo, é determinado o papel semântico dos constituintes que foram identificados como argumentos. A motivação de um processo dividido em duas etapas é que, em cada uma, um diferente conjunto de atributos fornece melhor desempenho para o classificador (Punyakanok et al., 2005).

Ainda no primeiro passo, o sistema deve impedir que dois argumentos do mesmo predicador possuam palavras que se sobreponham. Na prática, isso significa que se um não-terminal é classificado como argumento, todos os seus descendentes diretos e indiretos de-

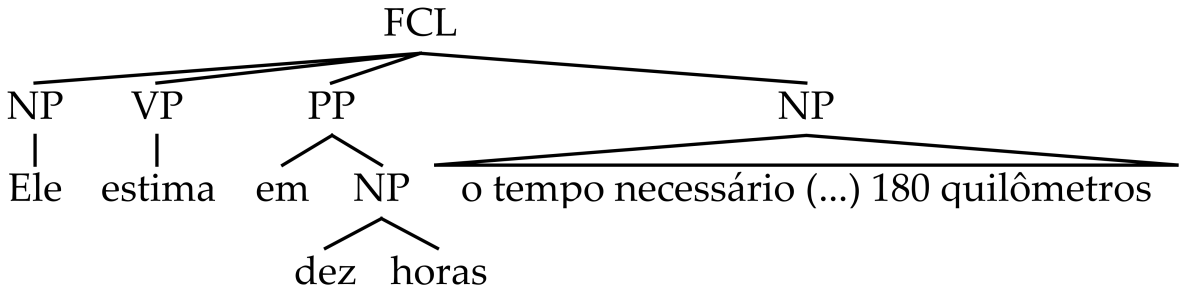

Figura 3.1: Árvore sintática para (6a), segundo o formalismo do *parser* PALAVRAS (Bick, 2000). Note-se que alguns não-terminais foram suprimidos nas subárvores representadas por um triângulo. `FCL` corresponde a uma sentença com verbo finito; `NP` a sintagma nominal; `VP` a sintagma nominal; `PP` a sintagma preposicional.

vem ser classificados como não-argumento[7]. Outras restrições do sistema devem garantir durante o segundo passo que o mesmo papel não seja atribuído mais de uma vez.

Xue e Palmer (2004) introduzem uma etapa de poda antes da identificação dos argumentos, em que são identificados constituintes que claramente não podem se relacionar com o predicador. A árvore sintática é percorrida para se determinar quais constituintes podem ser podados com segurança. Esta etapa tem a vantagem tanto de acelerar o treinamento do classificador como de melhorar seu desempenho, na medida em que permite que haja menos constituintes que possam ser erroneamente tratados como argumentos.

Haghighi et al. (2005) propõem um modelo que considera os *n* resultados mais prováveis na classificação de cada constituinte, tanto no primeiro como no segundo passo. Em seguida, é calculada a probabilidade de cada possível combinação de rótulos de argumentos. O resultado final é dado pela sequência que possuir a maior probabilidade.

Atributos Empregados

Uma decisão fundamental para este tipo de abordagem é o conjunto de atributos que será usado nos classificadores. Gildea e Jurafsky (2002) relatam o trabalho pioneiro de APS, no qual os autores empregaram apenas sete atributos extraídos da árvore sintática da sentença de entrada. Outros propuseram estender este conjunto (Xue e Palmer, 2004; Pradhan et al., 2004); trabalhos mais recentes chegam a contar com mais de 20 atributos, como Pradhan et al. (2008). Segue uma listagem dos atributos frequentemente empregados:

Predicador O predicador com o qual o constituinte se relaciona.

[7]Segundo um ponto de vista *top-down*. Uma abordagem *bottom-up* também é possível, funcionando de forma análoga: se um não-terminal é classificado como argumento, todos os seus ancestrais na árvore devem ser classificados como não-argumento

Tipo de Sintagma A categoria sintática (sintagma nominal, preposicional, adverbial, etc.) do constituinte.

Caminho na Árvore O caminho que precisa ser percorrido na árvore sintática para se chegar do predicador ao constituinte em questão. Este atributo é normalmente representado como uma cadeia de caracteres contendo os nós percorridos e a direção do percurso nó a nó. Diversos outros atributos podem ser definidos em função deste, como caminhos que consideram apenas alguns tipos de nós.

Posição Indica se o constituinte aparece antes ou depois do predicador na sentença.

Voz Indica se a sentença está na voz ativa ou passiva.

Núcleo do Sintagma A palavra que corresponde ao núcleo do sintagma. Também pode ser observada a classe gramatical desta palavra.

Primeira e Última Palavras do Sintagma A primeira e a última palavra do constituinte. Também podem ser consideradas suas classes gramaticais.

Subcategorização A regra de produção gramatical que gerou o predicador.

Categoria Principal Este atributo indica se, ao se percorrer a árvore desde o constituinte até a raiz, é encontrado primeiro um nó do tipo S (sentença) ou do tipo SV (sintagma verbal). Normalmente, o sujeito terá o valor S neste atributo, enquanto objetos direto e indireto terão o valor SV.

Tipo de Entidade Nomeada Indica se o núcleo do sintagma corresponde a uma entidade nomeada (como um nome de pessoa, lugar, instituição, etc.). Em caso positivo, é comum indicar-se qual tipo de entidade.

Outra questão de interesse é o tipo de classificador empregado. Gildea e Jurafsky (2002) usaram um modelo simples chamado *backoff lattice* que, para classificar um novo dado, busca em seus exemplos de treino por instâncias com o mesmo valor para a maior quantidade possível de atributos. Trabalhos subsequentes obtiveram melhores resultados com outros classificadores: segundo Palmer et al. (2010), os tipos de classificadores mais comuns para esta tarefa são o SVM (*support vector machines* – máquinas de vetores de suporte) (Cortes e Vapnik, 1995) e Máxima Entropia (Berger et al., 1996).

Atualmente, o sistema de Koomen et al. (2005), seguindo este tipo de arquitetura, detém o estado da arte da tarefa de APS para o inglês, alcançando uma pontuação de 77,92 na medida F, quando avaliado no conjunto de teste do CoNLL 2005. O sistema combina o resultado de seis classificadores de argumentos em uma etapa de inferência. Cada um dos

classificadores emprega uma árvore sintática diferente (a saber, uma gerada pelo *parser* de Collins (Collins, 1999) e as cinco melhores árvores do *parser* de Charniak (Charniak e Johnson, 2005)) e associa um fator de confiança (ou probabilidade) a cada papel atribuído a um constituinte. Na etapa de inferência, é escolhida como saída final do sistema a atribuição de papéis semânticos que maximize a soma das respectivas probabilidades calculadas por cada classificador.

O sistema desenvolvido por Amancio (2011) lida com a tarefa de APS em língua portuguesa, com o objetivo de facilitar o entendimento de textos por analfabetos funcionais. Por isso, em vez de atribuir rótulos como os do PropBank ou da FrameNet para argumentos verbais, o sistema gera perguntas como **quem?**, **para quem?**, **onde?** e **como?**, e suas respectivas respostas, para cada verbo. Segundo o autor, fornecer estas perguntas e respostas ajuda a leitura. Amancio utilizou um *corpus* anotado com estes tipos de pergunta para treinar um classificador baseado em uma variante do algoritmo SVM, empregando atributos semelhantes aos descritos aqui. O sistema obtém uma pontuação de 79 na medida F para a tarefa de identificação e classificação de perguntas. Apesar de ser um valor bastante alto quando comparado aos resultados da APS em inglês, deve-se levar em conta que, pela natureza de seu trabalho, Amancio empregou um *corpus* que passara por um processo de simplificação textual (com ocorrências mínimas de orações subordinadas e relativas, por exemplo). Textos que ocorrem naturalmente tendem a ser mais difíceis de se lidar, devido à maior complexidade sintática.

Outro trabalho relevante é o de Alva Manchego e Rosa (2012), que apresentam um *benchmark* para a tarefa de APS em português do Brasil, baseado no PropBank-Br. O *benchmark* inclui um *baseline* baseado em um pequeno conjunto de regras simples que atribuem papéis baseados nas funções sintáticas dos constituintes. Estas regras cobrem os casos mais recorrentes, como sujeito → A0 e objeto direto → A1, obtendo uma medida F de 50,1% na porção de testes do PropBank-Br e 48,6% sobre todo o *corpus*. No entanto, é importante lembrar que este *baseline* foi computado com as árvores sintáticas corrigidas presentes no PropBank-Br; não há resultados com *parses* automáticos. Em contrapartida, o sistema desenvolvido neste projeto não necessitou deste tipo de conhecimento, e está pronto para o uso com novas sentenças sem a necessidade de um *parser*.

Em sua dissertação de mestrado (Alva Manchego, 2013), Alva Manchego apresenta um classificador automático que usa algoritmo de máxima entropia e alcança uma pontuação na medida F de 82,3. Além disso, são estudadas técnicas de aprendizado semissupervisionado para contornar o problema da escassez de dados rotulados para APS em português. Porém, devido ao fato de o sistema depender de árvores sintáticas no padrão do PropBank-Br, que não são geradas por um *parser* automático, Alva Manchego não pôde usar seu sistema resultante com mais textos.

Por fim, Sequeira et al. (2012) propõem um projeto também voltado para a identificação de papéis semânticos em português, mas estão preocupados somente com sujeitos, relações (verbos de conteúdo) e objetos, e portanto analisam apenas árvores sintáticas. Esse trabalho ainda não possui resultados publicados.

3.2.2 Sistemas Baseados em Regras

Uma alternativa ao aprendizado de máquina é a elaboração de um sistema que conte com regras que indiquem como realizar a anotação. Essas regras precisariam prever todas as diferentes formas em que os papéis semânticos podem se distribuir pela sentença. Dado que não se tem uma resposta definitiva para prever sistematicamente a ocorrência de papéis semânticos, esta abordagem precisa se valer de heurísticas.

Uma grande desvantagem deste método é que cada regra precisa ser criada manualmente e o sistema não pode se beneficiar com um eventual aumento no tamanho do *corpus* de treino. Por outro lado, é possível contornar a escassez de *corpora* anotados com papéis semânticos, como é o caso em português.

Um trabalho que se vale desta estratégia, para a língua portuguesa, é descrito em Bick (2007). O sistema conta com uma gramática com mais de 500 regras e funciona como uma camada a mais para o *parser* sintático PALAVRAS (Bick, 2000), do mesmo autor, que possui licença de uso proprietária.

Bick, no entanto, emprega um conjunto próprio e muito amplo de rótulos de papéis semânticos. Alguns deles escapam da ideia de argumento de um predicador, como é o caso de VOCATIVO, por exemplo, um dos papéis usados pelo autor. O sistema foi testado em uma porção de 2.500 palavras do *corpus* Floresta Sintá(c)tica (Bick et al., 2007) e obteve precisão de 90,5% e cobertura de 86,6%. Note-se que ainda não existe uma versão deste *corpus* com anotação de papéis semânticos revisada e disponibilizada. É um resultado bastante expressivo; no entanto, o conjunto de papéis atribuídos e a pouca quantidade de dados de testes pode tê-lo influenciado positivamente.

3.2.3 Sistemas de Classificadores de Palavras

Uma terceira abordagem que pode ser identificada em trabalhos de APS consiste em encarar o problema como a classificação de cada palavra individualmente. No caso, para cada predicador da sentença, as palavras são classificadas como o próprio predicador, parte de um argumento (e, em caso positivo, a qual) ou não relacionada. O presente trabalho visa seguir este tipo de abordagem (Fonseca e Rosa, 2012).

O sistema SENNA[8] (Collobert e Weston, 2008; Collobert et al., 2011; Collobert, 2011) é

[8]Disponível em `http://ml.nec-labs.com/senna/`

o principal representante dessa linha. Além da APS, o SENNA pode ser usado para resolver outras tarefas de PLN, como a etiquetação morfossintática (*part-of-speech tagging*), reconhecimento de entidades nomeadas e *parsing* sintático completo ou de superfície (*chunking*).

Em uma única etapa, o sistema identifica quais palavras formam argumentos verbais e as classifica. Os autores evitam o uso de atributos provenientes de análises linguísticas do texto de entrada, como os que são normalmente encontrados na abordagem de dois estágios (Seção 3.2.1). Em lugar deles, empregam representações radicalmente diferentes para as sentenças.

Cada palavra presente no vocabulário do sistema tem um vetor de números reais de tamanho fixo associado (treinado separadamente, de forma explicada na Seção 4.3). Assim, as palavras são vistas como índices para seus vetores correspondentes.

Este tipo de representação também pode incluir atributos discretos, como presença de letras maiúsculas ou classe gramatical. Para cada atributo, deve haver uma listagem com todos os seus possíveis valores, e para cada um deles, um vetor numérico associado. A representação final é obtida concatenando-se o vetor da palavra com os de cada atributo.

A Figura 3.2 ilustra um caso em que o sistema possui uma tabela onde são armazenados os vetores correspondentes às palavras e outra onde estão os que codificam a diferentes tipos de combinação de letras maiúsculas e minúsculas. São exemplificadas as representações finais de algumas palavras. Os valores foram determinados arbitrariamente e servem apenas de exemplo.

Palavra	**Vetor**		
o	0,97	-0,34	0,16
caixa	-0,81	0,09	-0,21
cidade	0,49	0,82	0,63
⋮		⋮	

Padrão	**Vetor**	
Todas minúsculas (abc)	0,04	0,72
Inicial (ou única) maiúscula (A, Abc)	-0,59	0,18
Todas maiúsculas (ABC)	0,94	0,51
Outra combinação (AbC)	-0,12	-0,65

Ocorrência	**Vetor Final**				
O	0,97	-0,34	0,16	-0,59	0,18
caixa	-0,81	0,09	-0,21	0,04	0,72
CIDADE	0,49	0,82	0,63	0,94	0,51
CiDaDe	0,49	0,82	0,63	-0,12	-0,65

Figura 3.2: Exemplo de obtenção de vetores de palavras

Para algumas tarefas mais simples de PLN, como a etiquetação morfossintática, é viável que o sistema observe apenas o contexto local de cada palavra. Nessas tarefas, cada palavra da sentença é fornecida ao sistema para ser etiquetada de cada vez, acompanhada de k palavras à sua direita e à sua esquerda[9]. Elas são convertidas em seus vetores numéricos,

[9]Para o caso em que a palavra se encontra próxima do início ou do fim da sentença, é usado um símbolo

concatenadas e usadas como entrada para uma rede neural artificial MLP com uma camada oculta. É mostrada na Figura 3.3 uma janela de 3 palavras sendo convertida em vetores numéricos (novamente, os valores servem apenas de exemplo).

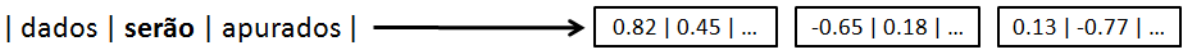

Figura 3.3: Janela de 3 palavras sendo convertida em vetores numéricos.

No caso da APS, é importante o sistema analise toda a sentença, e não apenas uma janela de tamanho limitado, dado que pode haver relações de predicador e argumentos a longa distância. No entanto, sentenças em língua natural têm comprimento variável, enquanto redes neurais convencionais lidam com entradas de tamanho fixo. Para contornar essa limitação, o SENNA emprega uma rede neural com uma camada de *convolução* após a camada de entrada (ilustrada na Figura 3.4), que funciona da seguinte forma.

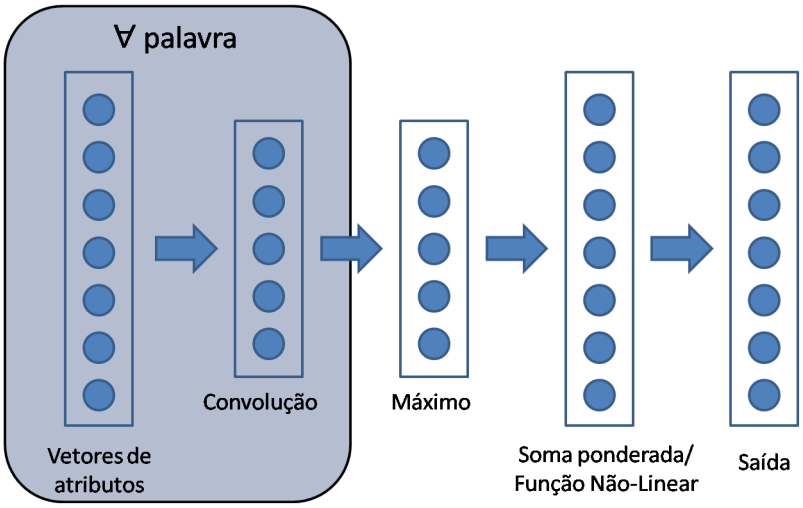

Figura 3.4: Estrutura da rede neural. A primeira camada possui os valores dos vetores numéricos das palavras da janela de entrada. A segunda camada recebe cada janela, calculando a soma ponderada em cada neurônio e armazenando o resultado. A camada de máximo apenas extrai o maior valor armazenado por cada neurônio de convolução e aplica uma função sigmoide. As duas últimas camadas executam operações padrão de redes neurais.

Para classificar cada palavra, a rede recebe todas as janelas de palavras da sentença, uma a uma. A camada de convolução armazena em cada neurônio o resultado calculado e não emite uma saída até que tenha recebido todas as entradas. Quando isso acontece, cada neurônio emite o maior valor que calculou ao longo de todas as janelas de palavras.

especial para representar uma posição fora dos limites da sentença.

Palavra 1	DA	DP	Palavra 2	DA	DP	Palavra 3	DA	DP
`*PADDING*`	-1	-2	Ele	0	-1	estima	1	0
Ele	0	-1	estima	1	0	em	2	1
estima	1	0	em	2	1	dez	3	2
em	2	1	dez	3	2	horas	4	3
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮

DA: Distância à palavra alvo
DP: Distância ao predicador

Tabela 3.1: Exemplo do funcionamento da convolução na rede neural para classificar a palavra *Ele* em relação ao predicador *estima*. `*PADDING*` indica uma posição fora dos limites da sentença. Cada linha da tabela representa uma entrada para o sistema, que precisa ser convertida em vetores numéricos. Após a rede ter recebido e processado todas as janelas, os neurônios da camada de convolução emitem a saída relativa ao par *Ele – estima*. Para classificar as outras palavras, é necessário repetir o processo variando os valores de distância.

Desta forma, cada neurônio na camada de convolução está capturando algum atributo latente ao longo de todos os dados de entrada. Após a camada de convolução, a rede possui uma camada oculta convencional e, por fim, a camada de saída. Para indicar qual palavra se pretende classificar, e com respeito a qual predicador, o vetor de cada palavra é acrescido de atributos correspondendo à distância (medida em quantidade de palavras) à palavra e ao predicado de interesse. Note-se que, caso estes não fossem usados, o resultado da convolução (e de toda a rede neural) seria o mesmo para todas as palavras.

Na Tabela 3.1 é exemplificado o funcionamento da convolução sobre a Sentença 7 para classificar a palavra *Ele* em relação ao predicador *estima*, com uma janela de tamanho 3. Cada linha da tabela corresponde a uma janela de palavras. Para cada entrada, as palavras são convertidas em seus vetores numéricos (incluindo os vetores correspondentes às distâncias), concatenadas, e fornecidas à rede neural. Após todas as janelas terem sido processadas, cada neurônio da camada de convolução emite como saída o maior valor calculado ao longo de todas as entradas.

(7) Ele **estima** em dez horas o tempo necessário para classificar as áreas e culturas plantadas em uma extensão de 180 quilômetros.

Para classificar as outras palavras da sentença, o processo deverá ser repetido, alterando apenas as distâncias relativas. É possível calcular inicialmente uma matriz contendo o resultado da convolução sobre todas as janelas de palavras sem considerar os vetores de distância, e reaproveitá-la para maior eficiência computacional. Em seguida, a cada janela, é necessário apenas acrescentar os valores correspondentes à distância ponderados pelos pesos de cada neurônio.

Note-se que, ao se classificar cada palavra individualmente, o sistema não aproveita o conhecimento de que cada argumento é formado por um bloco de palavras (constituinte), podendo atribuir rótulos alternados para cada *token*. Para incorporar essa noção, o SENNA usa o esquema IOBES (acrônimo de *In*, *Out*, *Begin*, *End*, *Single*): a etiqueta atribuída a cada *token* é formada pela combinação de um papel semântico e uma letra indicando se se trata do início do argumento, uma posição dentro do argumento, o final do argumento, o único *token* do argumento, ou se o *token* não faz parte de argumento algum. A seguir é mostrado como a Sentença 7 deve ser anotada no formato IOBES.

S-A0	**S-V**	**B-A2**	**I-A2**	**E-A2**	**B-A1**	**I-A1**	**I-A1**	**I-A1**	**I-A1**	**I-A1**
Ele	**estima**	em	dez	horas	o	tempo	necessário	para	classificar	as
I-A1	**I-A1**	**I-A1**	**I-A1**	**I-A1**	**I-A1**	**I-A1**	**I-A1**	**I-A1**	**E-A1**	**O**
áreas	e	culturas	plantadas	em	uma	extensão	de	180	quilômetros	.

Figura 3.5: Exemplo de etiquetação IOBES para APS

No lugar do IOBES, também é possível usar outros esquemas mais sucintos, como IOB (*In*, *Out*, *Begin*) ou IOE (*In*, *Out*, *End*), que empregam um conjunto reduzido de etiquetas e possuem a mesma capacidade do IOBES de delimitar sequências de *tokens*[10]. No entanto, Collobert et al. informam terem obtido melhores resultados com o IOBES, mesmo com o aumento da quantidade de etiquetas (com IOB ou IOE, existem duas etiquetas para cada papel semântico, enquanto que com IOBES, existem quatro).

Paralelamente à rede neural, uma matriz com a pontuação da passagem de cada etiqueta para outra é treinada, permitindo que a atribuição de papéis seja feita baseada na melhor combinação entre a pontuação *a priori* da rede para cada *token* e a matriz de transições. A pontuação total de uma sequência de etiquetas $y_1 \dots y_n$ para uma sentença composta pelos *tokens* $x_1 \dots x_n$ é obtida da seguinte forma. Para cada *token* x_i, somam-se a pontuação da rede para a etiqueta y_i e a transição de y_{n-1} para y_n (é necessário que a tabela de transições possua uma pontuação para iniciar a sentença com cada etiqueta).

Formalmente, sendo x a sentença de entrada, y a sequência de tags, A a matriz de transições, $f_{y_i,j}$ a saída da rede para a etiqueta i no *token* j, e θ os parâmetros da rede, tem-se o seguinte:

$$s(x, y, \theta) = \sum_{i=1}^{n} (A_{y_{i-1},y_i} + f_{y_i,i}) \tag{3.1}$$

Para facilitar o treinamento, transições impossíveis podem ser inicializadas com valores extremamente baixos. Estas incluem a transição de uma etiqueta B para qualquer outra que

[10]No esquema IOB, os códigos E e S do IOBES são substituídos, respectivamente, por I e B. No IOE, B e S são substituídos por I e E.

não seja I ou E do mesmo argumento; ou da etiqueta O para qualquer uma que não seja O, B ou S. O treinamento da rede é feito de modo a maximizar a pontuação da sequência de etiquetas correta de cada proposição em detrimento de todas as outras.

Para encontrar a sequência com a maior pontuação, é empregado o algoritmo de Viterbi (Viterbi, 1967). Este algoritmo é baseado em programação dinâmica, funcionando da seguinte forma. No primeiro passo, é armazenada a pontuação para cada etiqueta y ser atribuída ao primeiro elemento x_0 da sequência – o que corresponde, no caso do SENNA, à soma da pontuação *a priori* emitida pela rede para o par (x_0, y) e o valor da matriz para a transição entre começo da sentença e a etiqueta y.

A seguir, o algoritmo calcula a pontuação para cada etiqueta y ser atribuída ao próximo *token* x_i. Esta é determinada pela soma da saída da rede para (x_i, y), a pontuação da transição a partir da etiqueta do *token* x_{i-1} e a pontuação obtida até o *token* x_{i-1}. Como a etiqueta na posição x_{i-1} ainda não foi determinada, todas as possibilidades são consideradas e apenas a de maior pontuação é armazenada. Neste momento, fixa-se a etiqueta para o *token* x_{i-1}, mas não para x_i. Este processo é repetido até o fim da sequência, quando a etiqueta com a maior pontuação no último *token* é selecionada.

A versão do SENNA que alcança os melhores resultados reportados, e disponibilizada pelos autores, inclui nas representações de palavras apenas a presença de letras maiúsculas e a análise de *chunks* sintáticos, gerada por outro módulo do próprio sistema. Os *chunks* são codificados usando o esquema IOBES e categorias comuns para o inglês, conforme mostrado na Figura 3.6. Para cada possível etiqueta, há um vetor de atributos.

B-NP	**E-NP**	**S-VP**	**S-PP**	**B-NP**	**E-NP**	**O**
The	cat	sat	on	the	mat	.

Figura 3.6: Exemplo de etiquetação de *chunks* usada pelo SENNA. NP significa sintagma nominal; VP, sintagma verbal e PP, sintagma preposicional.

Graças a essa simplicidade no uso de atributos de entrada, o tempo de execução do sistema é reduzido em comparação a outros que alcançam resultados do estado da arte. A medida F para o desempenho do SENNA no conjunto de teste do CoNLL 2005 é de 75,49.

Um trabalho que segue em linha semelhante é o de Rosa e Adán-Coello (2010), também baseado em uma rede neural artificial. Neste trabalho, os autores representam palavras por seus traços semânticos, que são definidos como um conjunto de atributos categóricos. Há um conjunto para substantivos e outro para verbos, com dois a quatro possíveis valores para cada atributo. Exemplos de traços semânticos de substantivos são “tamanho” (pequeno, médio, grande ou indeterminado) e “forma” (redondo, angular, irregular ou indeterminado); e de verbos são “social”, “controle da ação”, “estado psicológico” (todos admitindo os valores *sim* ou *não*).

O sistema substitui cada palavra da sentença de entrada por um vetor de valores binários que indicam os valores de seus traços, e determina o papel semântico de cada uma. No entanto, o sistema não foi produzido para lidar com um *corpus* real, mas com um gerado automaticamente e com a sintaxe de cada sentença prevista.

3.3 O Problema do Domínio

Um ponto importante a ser lembrado é o domínio textual onde se emprega o classificador. Classificadores baseados em aprendizado de máquina treinados em um determinado domínio, como o jornalístico, tendem a apresentar uma queda de desempenho ao serem testados em outros.

Pradhan et al. (2008) exploram esse problema, comparando resultados da APS feita com diferentes combinações de conjuntos de treino e teste. No caso, foram usados o Wall Street Journal (domínio jornalístico) e o *corpus* Brown (domínio literário), as duas coleções que compõem o PropBank em inglês. Os autores constatam a queda de desempenho associada ao domínio, além de que o literário parece apresentar maiores dificuldades do que o jornalístico. Segundo os autores, isso se deve principalmente à maior heterogeneidade dos textos e ao fato de muitos predicadores possuírem mais de um sentido.

Kundu e Roth (2011) lidam com esse problema fazendo adaptações no texto a ser anotado. As adaptações incluem a troca de palavras ausentes nos dados de treino por sinônimos e simplificação da estrutura das sentenças (com remoção de adjetivos, por exemplo). Os autores mostram resultados da APS feita sobre o *corpus* Brown com um modelo treinado apenas com dados do Wall Street Journal, alcançando uma medida F de 70,5. Esse valor é superior a outros resultados da literatura treinados da mesma forma.

No caso do PropBank-Br, constituído apenas de textos jornalísticos, é de se esperar que haja o problema do domínio. No entanto, não se pode verificá-lo quantitativamente, dada a ausência de textos de outros domínios em português anotados com papéis semânticos. Essa é uma das preocupações futuras que se deve ter ao expandir o recurso.

3.4 Considerações Finais

Foram apresentadas diferentes abordagens usadas por sistemas que lidam com a tarefa de APS. Para os sistemas baseados em constituintes, que são a grande maioria dos trabalhos na área, foram detalhadas as principais técnicas introduzidas por diferentes autores. No caso da abordagem baseada em palavras, foi explicado o funcionamento do sistema SENNA, que constitui a principal inspiração para o trabalho aqui realizado.

Percebe-se que, enquanto o problema é intensamente pesquisado na língua inglesa, para

o português a pesquisa ainda é muito pequena. Dos poucos trabalhos voltados para o português, o de Bick (2007) foi implementado com um sistema baseado em regras gramaticais e de forma totalmente diferente do padrão do PropBank. Os de Amancio (2011), Sequeira et al. (2012) e Alva Manchego (2013) seguem abordagens baseadas em aprendizado de máquina, embora somente o último lide com o PropBank-Br, de modo a ser o único com o qual se possam comparar os resultados do presente projeto.

Capítulo

4

Representações de Palavras

Neste capítulo, são mostradas técnicas para obtenção de representações de palavras em forma de vetores numéricos, chamadas de modelos de espaço vetorial (*Vector Space Models* — VSM). Também são apresentados experimentos realizados com diferentes técnicas, e os resultados obtidos. Todos os métodos aqui descritos baseiam-se na observação de grandes porções de texto, a partir das quais são modeladas as representações. O propósito de se obter representações de palavras é usá-las como parte do sistema de APS, conforme explicado na Seção 3.2.3.

4.1 Motivação

Ao se considerar a APS (bem como outras tarefas de PLN) como puramente uma questão de AM, a preocupação reside apenas em fazer um classificador automático obter bons resultados com os dados que se possui – independentemente de como estes dados foram obtidos. No caso da APS, como mostrado na Seção 3.2.1, é comum se usar como dados de entrada classes gramaticais de palavras, caminhos na árvore sintática, núcleo de sintagma, tipos de entidade nomeada, entre outros.

No entanto, ao se tratar estes problemas na prática, há também a preocupação com fatores externos ao AM. Sobretudo, é preciso determinar a forma com que se obtêm os valores dos atributos que serão empregados pelo classificador, já que, a rigor, os dados de entrada são apenas sentenças em língua natural. Ao se empregar atributos provenientes de uma análise linguística sobre as sentenças, há a dependência de outras ferramentas de PLN, o que pode implicar em desvantagens como propagação de erro e maior custo

computacional. Além disso, o *parser* sintático para português com melhor performance conhecida atualmente é o PALAVRAS, que possui licença de uso proprietária. Somada a esses problemas de ordem técnica, há também a dificuldade teórica de se definir quais são os atributos relevantes para um dado problema.

A arquitetura do sistema desenvolvido neste projeto segue o modelo de Collobert e Weston (2008), que dispensa o uso de atributos definidos manualmente (embora ainda o permita). Em vez disso, são usados como entrada vetores de números reais que representam palavras, conforme mostrado na Seção 3.2.3. Desta forma, mesmo sem se explicitar o que cada componente do vetor representa, a rede neural é capaz de encontrar propriedades latentes deles (Bengio, 2009) para realizar tarefas de PLN.

Embora o modelo citado possa inicializar os vetores de forma aleatória e ajustá-los durante o treinamento de tarefas como a APS, Collobert et al. (2011) mostram um ganho de desempenho considerável ao usar valores previamente treinados com um modelo de língua. A forma como estes vetores são induzidos é mostrada na Seção 4.3, e usa a técnica de AM conhecida como *bootstrapping*, que consiste em aproveitar grandes quantidades de dados não-rotulados (no caso, texto puro) para melhorar o desempenho de um classificador automático.

Adicionalmente, dado que palavras com significados semelhantes possuam vetores com valores semelhantes, o impacto de palavras não vistas durante o treinamento supervisionado (referidas como OOV – *out of vocabulary*) é reduzido, se houver representações vetoriais para elas. Esta condição é facilmente satisfeita, pois a indução de vetores é feita sobre textos sem nenhum tipo de anotação, que estão disponíveis em grande quantidade, permitindo gerar representações para um vasto vocabulário.

Durante o curso deste projeto, tentou-se induzir representações vetoriais para português seguindo o mesmo método de Collobert e Weston; no entanto, não se obtiveram bons resultados, como será mostrado na Seção 4.4. Por isso, foram analisadas outras técnicas para geração de modelos vetoriais, mostradas na Seção 4.2.

4.2 Modelos Distribucionais

Modelos distribucionais são inspirados no raciocínio de que palavras que ocorrem nos mesmos contextos tendem a ter significados semelhantes, ideia que é conhecida como a hipótese distribucional. Karlgren e Sahlgren (2001) abordam motivações filosóficas para o uso deste tipo de técnica, sobretudo devido à observação feita por filósofos e linguistas de que o significado de uma palavra é determinado por meio de como ela é usada, isto é, acompanhada de quais outras palavras e em quais contextos.

Este raciocínio é facilmente exemplificado. Dado um contexto, como por exemplo *via-*

jar para, palavras que apareçam em seguida a ele tendem a ser nomes de lugares, cidades, países, etc. A observação de diversas ocorrências deste contexto permite identificar a similaridade entre palavras que o sigam.

O funcionamento de modelos distribucionais se baseia em criar uma tabela com a contagem de ocorrências de palavras em cada documento de uma coleção. A tabela, portanto, tem dimensões $W \times D$, onde W é o tamanho do vocabulário e D a quantidade de documentos. Originalmente, essa técnica foi empregada na área de recuperação da informação (Turney e Pantel, 2010), com o objetivo de encontrar documentos compatíveis com uma determinada consulta.

De acordo com a tabela, documentos podem ser vistos como pontos em um espaço W-dimensional (ou possivelmente W'-dimensional, com $W' \ll W$, aplicando alguma técnica de redução de dimensionalidade, dado que W normalmente corresponde a pelo menos algumas dezenas de milhares). Consultas por termos são também convertidas em pontos nesse espaço e comparadas com os documentos disponíveis, usando alguma medida geométrica como distância euclidiana (mostrada na Equação 4.1) ou cosseno entre vetores (mostrado na Equação 4.2).

$$d(\mathbf{u}, \mathbf{v}) = \sqrt{(u_1 - v_1)^2 + (u_2 - v_2)^2 + \cdots + (u_n - v_n)^2} \tag{4.1}$$

$$cosseno(\mathbf{u}, \mathbf{v}) = \frac{\mathbf{u} \cdot \mathbf{v}}{|\mathbf{u}||\mathbf{v}|} = \frac{\sum_{i=1}^{n} u_i \times v_i}{\sqrt{\sum_{i=1}^{n} u_i^2} \times \sqrt{\sum_{i=1}^{n} v_i^2}} \tag{4.2}$$

Analogamente, palavras também podem ser vistas como pontos em um espaço D-dimensional (ou D'-dimensional, com a mesma analogia). Turney e Pantel (2010) discutem diversas parametrizações para este tipo de modelo, como funções para serem aplicadas no resultado das contagens e técnicas de redução de dimensionalidade. Além disso, em vez de contagem de ocorrências por *documentos*, pode-se falar mais genericamente de *contextos*, que podem ser tanto um documento quanto um parágrafo ou mesmo uma janela de palavras de tamanho fixo.

Sahlgren (2008) analisa os fundamentos teóricos relacionados às contagens de ocorrências em diferentes tipos de contexto e os resultados que podem fornecer. Segundo o autor, o único conhecimento que pode ser obtido por meio da análise distribucional é a similaridade lexical, compreendida como a semelhança geométrica entre seus vetores. No entanto, Sahlgren discerne dois tipos de semelhança que este tipo de análise pode descobrir: *sintagmáticas* (correspondendo a palavras que co-ocorrem próximas uma da outra) e *paradigmáticas* (correspondendo a palavras que podem ocorrer em um mesmo contexto).

Um exemplo de relação sintagmática é a que existe entre as palavras na expressão *mesa de madeira*, que formam uma sequência aceitável, enquanto a semelhança paradigmática

pode ser encontrada entre as palavras *porta*, *cadeira* e *mesa*, que poderiam ocupar o mesmo lugar na expressão. Para o presente trabalho, são interessantes os modelos que capturam semelhanças paradigmáticas, visto que se adotou uma abordagem de APS que analisa uma palavra de cada vez.

Modelos distribucionais que buscam exprimir relações sintagmáticas trabalham com grandes contextos, como documentos ou pelo menos sentenças completas. Já os modelos que exprimem relações paradigmáticas analisam contextos menores, sendo uma janela de 10 ou menos palavras uma escolha frequente.

4.2.1 *Hyperspace Analogue to Language*

O *Hyperspace Analogue to Language* (HAL) (Lund e Burgess, 1996) é um modelo distribucional que contabiliza co-ocorrências de palavras dentro de uma pequena janela deslizante. O tamanho da janela normalmente usado é de poucas palavras, entre 2 a 10. Para contabilizar as co-ocorrências, é usada uma matriz M de dimensões $W \times W$, com W sendo o tamanho do vocabulário. Cada célula M_{ij} armazena a quantidade de vezes que a i-ésima palavra foi encontrada antes da j-ésima.[11] Desta forma, cada linha i contém a contagem referente às ocorrências *após* a palavra i, e a coluna j, as ocorrências *antes* da palavra j. A Tabela 4.1 contém as contagens para a sentença *"Só sei que nada sei."*, com uma janela de tamanho 2.

O vetor de atributos de cada palavra é obtido ao se concatenar a sua linha na tabela com a sua coluna, correspondendo a um vetor de tamanho $2 \times W$. Os autores do método indicam, no entanto, que boas representações podem ser obtidas com apenas as 100 a 200 dimensões com maior variação dentre os vetores.

	só	sei	que	nada	.
só	0	1	1	0	0
sei	0	0	1	1	1
que	0	1	0	1	0
nada	0	1	0	0	1
.	0	0	0	0	0

Tabela 4.1: Tabela de co-ocorrência para a sentença *Só sei que nada sei.*

4.2.2 *Random Indexing*

O *Random Indexing* (RI) (Sahlgren, 2005) é uma outra técnica para gerar modelos distribucionais, motivada pelo propósito de ser uma alternativa computacionalmente barata

[11] Note-se que a escolha de M_{ij} para contabilizar as ocorrências da palavra i antes da palavra j, e não o contrário, é arbitrária e não faz diferença para o modelo.

em comparação a outros métodos bem difundidos de VSM, como o HAL e o *Latent Semantic Analysis* (Landauer e Dumais, 1997). Para este fim, o *Random Indexing* dispensa a etapa de redução de dimensionalidade, que é a mais custosa em outros modelos.

O método funciona da seguinte forma. Inicialmente, é associado um vetor de índice (*index vector*) a cada palavra do vocabulário. Estes vetores contêm apenas -1, +1 e 0 distribuídos aleatoriamente, sendo bastante esparsos (maioria dos valores nulos) e com dimensionalidade da ordem de milhares. Em seguida, um *corpus* é analisado, e a cada ocorrência de uma palavra i próxima de uma palavra j, o vetor de i é adicionado ao de j.

Sahlgren indica que foram obtidos bons resultados em diferentes aplicações de PLN que empregaram esta técnica. A principal desvantagem encontrada é a necessidade do uso de vetores de dimensões muito grandes, que são computacionalmente custosos para serem processados em outras aplicações (como nos sistemas de APS abordados neste trabalho). Ainda assim, há a possibilidade de se empregar uma técnica externa para reduzir a dimensionalidade dos vetores.

4.3 Modelos Neurais

Uma outra técnica encontrada na literatura para induzir representações numéricas para palavras é o uso de um modelo de língua baseado em redes neurais artificiais. Este tipo de abordagem é particularmente interessante para o presente trabalho por ter sido empregado para gerar as representações usadas pelo sistema SENNA (Collobert e Weston, 2008).

A função de um modelo de língua é atribuir uma pontuação a sequências de palavras, indicando o quão aceitáveis elas são. Modelos de língua frequentemente lidam com probabilidades de ocorrência; mas para a indução de representações de palavras, são empregados modelos discriminativos. O funcionamento do modelo de Collobert e Weston emprega uma arquitetura semelhante à das redes neurais usadas em outras tarefas de PLN: dada uma janela de palavras, estas são convertidas em seus vetores numéricos e fornecidas à rede neural. Essa rede possui um único neurônio em sua camada de saída, que indica a pontuação atribuída à sequência de entrada.

O treinamento da rede é feito de forma a privilegiar sequências vistas em textos reais em detrimento de alterações aleatórias feitas sobre elas. Dada uma janela de palavras w extraída do *corpus* de treino, é gerada uma nova janela w' pela substituição da palavra central de w por uma outra escolhida aleatoriamente do vocabulário. Deseja-se que a pontuação emitida para w seja maior que a de w' por uma dada margem; dessa forma, o erro da rede é calculado conforme a fórmula a seguir:

$$e(w, w') = max(0, 1 - f(w) + f(w')) \tag{4.3}$$

onde $f(\cdot)$ corresponde à saída da rede. Os pesos da rede são ajustados por meio do algoritmo de retropropagação, e as alteração são aplicadas também às representações de palavras.

Este modelo foi empregado por outros autores com algumas variações, como gerar o exemplo negativo w' por meio de uma alteração da última palavra em vez da palavra central (Turian et al., 2010) ou a incorporação de informação sobre o contexto global do documento (Huang et al., 2012).

O tipo de semelhança de significado capturado por este tipo de modelo é o sintagmático, dado que se treina uma rede neural para distinguir quais palavras são aceitáveis para ocupar uma determinada posição dentro de uma janela de tamanho pequeno.

4.4 Experimentos

A avaliação de VSMs é tradicionalmente feita avaliando sua capacidade de identificar palavras semelhantes. Para os propósitos do presente trabalho, é interessante que se possa identificar semelhanças *paradigmáticas* entre palavras. Foram escolhidas palavras aleatoriamente entre as mais comuns do PropBank-Br para avaliar essa capacidade nos vetores induzidos. Acredita-se que os resultados aqui apresentados possam servir para outros trabalhos de PLN em língua portuguesa, independente de se tratar de uma tarefa relacionada à APS.

Todos os modelos apresentados a seguir foram obtidos a partir de um *corpus* composto pela versão em português da Wikipédia[12] e pelo *corpus* PLN-BR (Bruckschen et al., 2008), que contém textos jornalísticos extraídos do jornal Folha de São Paulo. A escolha destes *corpora* se deu, primeiramente, pelo seu tamanho: a Wikipédia possui cerca de 740 mil artigos, e o PLN-BR, cerca de 100 mil. A Wikipédia é particularmente interessante por tratar de uma enorme variedade de assuntos dentro do conhecimento humano, e, com isso, empregar um vasto vocabulário. Já o PLN-BR, por ser composto de textos jornalísticos, apresenta um vocabulário mais próximo daquele do PropBank-Br.

4.4.1 Pré-Processamento

Ambos os *corpora* passaram pelo mesmo processo de tokenização, e tiveram todas as ocorrências de números substituídas por um símbolo especial `__NUMBER__`, como é prática comum em trabalhos com VSM. Além disso, todas as ocorrências de diferentes tipos de aspas (como “, ”, « e ») foram mapeadas para um único símbolo ". Por fim, verbos com ênclise foram separados dos pronomes que os acompanham e mantidos com o hífen. Por exemplo,

[12]Disponível em `http://dumps.wikimedia.org/ptwiki`

encontrá-lo foi transformado em dois *tokens*, *encontrá-* e *lo*. A separação é motivada por verbo e pronome serem duas unidades distintas, o que tem impacto direto na APS: quando o verbo for um predicador, normalmente o pronome enclítico assume um papel semântico. A escolha de manter o hífen no fim do verbo vem em conformidade com o padrão encontrado no PropBank-Br. Mesóclises não foram tratadas, dada a raridade do fenômeno em português do Brasil e sua ausência no PropBank-Br.

Com os artigos provenientes da Wikipédia, além da remoção da marcação básica MediaWiki (para indicar formatação com negrito, itálico, links, etc.), foi realizado o seguinte pré-processamento, visando minimizar o conteúdo que não fosse texto simples:

- Remoção de páginas de desambiguação ou redirecionamento.
- Remoção das seções "Ver Também", "Bibliografia" e "Ligações Externas".
- Correção de sinais de pontuação repetidos, como duas vírgulas ou aspas. Este problema se mostrou bastante recorrente nos textos.
- Remoção de linhas com uma ou duas palavras (frequentemente encontradas em listas).
- Remoção de sentenças contendo templates. Templates são um tipo de marcação especial da Wikipédia para gerar conteúdo com algum padrão. O resultado produzido por um template normalmente faz parte do conteúdo da sentença onde aparece. Por exemplo, o template `{{PEPB|dodó|dodô}}` gera o texto "dodó$^{\text{(português europeu)}}$ ou dodô$^{\text{(português brasileiro)}}$". Como o comportamento de cada template é definido separadamente, e existem centenas deles, optou-se por simplesmente remover sentenças que os contivessem.
- Remoção de sentenças que, após a primeira etapa de limpeza de marcação, contivessem as sequências de caracteres `{{`, `}}`, `[[` ou `]]`. A presença destes normalmente é devida a algum template que o script de pré-processamento não foi capaz de reconhecer corretamente.
- Remoção de páginas de dias do ano, que contêm majoritariamente listas e templates.
- Remoção do conteúdo subscrito e superscrito, demarcado pelas tags `<small>` e `<super>`, respectivamente (o resto das sentenças onde apareciam foi mantido).

Para gerar os vetores com o modelo HAL e o RI, foi usado o software Semantic Vectors[13]. Para gerar os vetores com modelos neurais, foi usado o código de Huang et al. (2012)[14].

[13]Disponível em `http://code.google.com/p/semanticvectors/`
[14]Disponível em `http://ai.stanford.edu/~ehhuang/`

Foram gerados vetores para todos os *tokens* com pelo menos 40 ocorrências na combinação dos dois *corpora*, sem diferenciar letras maiúsculas e minúsculas. Ao contrário do processo descrito em Fonseca e Rosa (2012), não foi aplicada lematização às palavras, devido a dificuldades práticas. Havia a possibilidade de se empregar o módulo de lematização do *parser* PALAVRAS; nesse caso, seria necessário também lematizar todas as palavras encontradas por um sistema que usasse as representações (já que não haveria vetores para formas flexionadas). No caso do PALAVRAS, isso criaria dependência de um sistema externo com licença proprietária, o que se quis evitar.

Outra alternativa para a lematização, também descartada, foi o sistema desenvolvido no NILC por Erick Maziero[15]. Esse lematizador determina inicialmente a classe gramatical de cada palavra e busca seu lema em uma base de dados. No entanto, quando a mesma palavra pode ter mais de um lema com a mesma classe gramatical (como *foi*, que pode ser uma forma flexionada dos verbos *ser* ou *ir*), o sistema simplesmente retorna todas as possibilidades. Tal comportamento introduz a necessidade de uma etapa de desambiguação, que se optou por não tratar neste projeto.

4.4.2 Resultados

As seguintes parametrizações dos modelos foram empregadas. Para o HAL e o modelo neural, foram usados vetores com 50 dimensões; para o RI, 2000 dimensões. Para o HAL e o RI, foi usada uma janela de 3 *tokens* à direita e 3 à esquerda (3+3), para o modelo neural, apenas 10 à esquerda. A escolha de janelas se deve a bons resultados reportados com estes tamanhos na literatura. O modelo neural contou com 100 neurônios na camada oculta, número também comum em trabalhos relacionados.

Foram escolhidas aleatoriamente 10 palavras do PropBank-Br com pelo menos 10 ocorrências. Para cada uma delas, são mostradas nas tabelas 4.2, 4.3 e 4.4 as 10 palavras mais semelhantes segundo cada tipo de VSM. A semelhança entre palavras é medida como o cosseno entre seus vetores (Equação 4.2).

A primeira conclusão tirada da observação dos resultados é que o modelo neural gerou representações que capturaram menos palavras semelhantes. Por exemplo, foi o único modelo a não encontrar outros pronomes substantivos semelhantes a *ela*, e o que mais teve variação nas classes gramaticais e flexões das palavras retornadas em geral. Nota-se também que, ao contrário do esperado, as semelhanças capturadas pelo modelo nem sempre refletem relações sintagmáticas, mas palavras que estão relacionadas a determinados assuntos, como *inflação/investimentos* e *profissionais/pessoal*. Uma das razões para as representações não terem sido tão boas provavelmente é o número de iterações do algoritmo: o modelo funciona continuamente selecionando janelas de texto e executando a rede neural

[15] Disponível em `http://www2.icmc.usp.br/~taspardo/sucinto/resources.html`

ela	grupos	melhor	profissionais	porém
ele	blocos	maior	cantores	enfim
eles	sistemas	cara	gigantes	evidentemente
também	conjuntos	propósito	músicos	obviamente
esta	núcleos	pior	objetos	aliás
ainda	projetos	atendente	jovens	ora
normalmente	sites	montante	intérpretes	portanto
essa	depósitos	pedido	professores	paradoxalmente
embora	modelos	crescente	objectos	aparentemente
hsinbyushin	protocolos	soldo	itens	contraditoriamente
urge	programas	personagem	auxiliares	convenhamos

crescimento	inflação	sérvios	responsável	necessidade
deslocamento	violência	monarcas	responsavel	possibilidade
tamanho	respiração	índios	reponsável	inexistência
cálculo	música	republicanos	bancado	noção
conteúdo	sabedoria	espanhóis	gerido	impossibilidade
amadurecimento	psicoterapia	gregos	patrocinada	ingestão
rendimento	magia	cossacos	financiado	condição
credenciamento	censura	censores	gerenciado	escassez
valor	pintura	padres	influenciada	incumbência
controle	impunidade	bolcheviques	patrocinado	disponibilidade
comportamento	oração	franceses	impulsionada	chance

Tabela 4.2: Resultados de semelhança lexical com HAL

sobre elas, sem um critério de parada definido. Collobert et al. (2011) relatam semanas de treinamento para as representações que empregaram; possivelmente, o modelo empregado no presente projeto terá representações melhores após mais iterações.

Quanto aos outros métodos, não se chegou a uma conclusão definitiva. Ambos foram capazes de agrupar palavras com significado e uso semelhantes, embora também tenham incluído outras mais discrepantes. Percebe-se, ainda, a influência de erros de grafia recorrentes no *corpus*: a palavra *responsável* aparece também grafada como *reponsável* e *responsavel*. Algumas regularidades são também notadas nos resultados do HAL e do RI: todas as palavras retornadas para os substantivos *grupos*, *crescimento*, *inflação*, *sérvios* e *necessidade* apresentam o mesmo gênero e número que a palavra comparada, e todos as palavras retornadas para *responsável* (seja como adjetivo ou substantivo) são comumente seguidas pela preposição *por*.

No entanto, uma diferença sensível entre ambos é o número de dimensões dos vetores: 50 do HAL contra 2000 do RI, o que significa que os vetores gerados pelo primeiro serão muito mais baratos computacionalmente de serem tratados. Ainda que se possa aplicar técnicas de redução de dimensionalidade aos vetores do RI, tal processo poderia fazer com que os vetores percam sua capacidade expressiva, motivo pelo qual se decidiu empregar

ela	grupos	melhor	profissionais	porém
ele	blocos	material	aviões	aliás
ali	núcleos	sacrifício	barcos	enfim
aí	ciclos	modelo	objetos	paradoxalmente
aqui	processos	critério	pescadores	evidentemente
ainda	agrupamentos	poder	carros	afinal
neles	projectos	deus	aparelhos	mas
eles	filmes	padrão	voluntários	obviamente
nela	projetos	abate	professores	entretanto
nele	satélites	equipamento	animais	porem
não	canais	aprendizado	escravos	ironicamente

crescimento	inflação	sérvios	responsável	necessidade
deslocamento	marca	franceses	responsavel	impossibilidade
valor	concorrência	russos	patrocinado	possibilidade
bloqueio	inadimplência	ingleses	motivado	ocorrência
estudo	colheita	muçulmanos	reponsável	inexistência
rendimento	empresa	búlgaros	impulsionado	noção
perfil	equipe	suecos	operado	probabilidade
saque	paralisação	romenos	estimulado	existência
peso	salvação	republicanos	financiado	cobrança
tamanho	tarifa	holandeses	bancado	emissão
desenvolvimento	matança	índios	motivada	incapacidade

Tabela 4.3: Resultados de semelhança lexical com RI

vetores gerados pelo HAL nos experimentos com APS. Contudo, não se pode concluir definitivamente que os vetores do HAL sejam melhores para a tarefa; esta comparação poderia ser investigado mais a fundo em outros trabalhos.

4.5 Considerações Finais

Foram apresentados, neste capítulo, a motivação para se induzir representações numéricas para palavras, bem como diferentes técnicas encontradas na literatura para gerá-las. Foram feitos experimentos com três dos modelos sobre um *corpus* composto de textos em língua portuguesa.

Apesar de resultados satisfatórios, é interessante lembrar que a língua portuguesa apresenta uma grande quantidade de formas flexionadas, especialmente para verbos. Esse fato leva a uma maior esparsidade de formas únicas no *corpus*, o que prejudica a qualidade dos vetores gerados. O uso de um lematizador permanece como opção interessante para a indução de novos vetores.

Baseado nos resultados experimentais obtidos, preferiu-se empregar no sistema de APS os vetores gerados pelo HAL, já que são capazes de capturar melhor as semelhanças sin-

ela	grupos	melhor	profissionais	porém
preparar	alguns	boa	técnica	todas
ao	partir	segunda	técnico	fato
seu	regiões	sucesso	dessa	assim
um	maioria	duas	técnicas	entretanto
(	várias	só	falta	enquanto
mas	número	fazendo	quadro	fazendo
para	maior	quase	pessoal	tanto
por	locais	final	melhores	apesar
relacionar	outras	antes	desenvolver	outras
depois	diferentes	vez	prática	essa

crescimento	inflação	sérvios	responsável	necessidade
expansão	reajuste	slobodan	receber	necessário
central	tributária	croatas	principais	necessária
atividade	poupança	sérvio	maiores	portanto
econômicas	juros	sérvia	importante	responsabilidade
alimentos	empresários	ex-iugoslávia	dessa	processo
regiões	endividamento	albanesa	ajuda	termos
maioria	preços	croata	objetivo	dessa
agricultura	investimentos	sérvias	ação	tipo
principalmente	dólar	albânia	tentativa	etc
aumento	gastos	bósnios	livre	criar

Tabela 4.4: Resultados de semelhança lexical com modelo neural

tagmáticas em poucas dimensões. Dessa forma, a rede neural responsável pela anotação de papéis semânticos encontrará mais facilmente padrões nos seus dados de entrada, bem como terá maior facilidade de lidar com palavras OOV. Os vetores gerados pelo HAL foram empregados na maior parte dos experimentos para resolver a tarefa de APS, como será mostrado no capítulo a seguir.

Capítulo

5

Implementação e Resultados

Neste capítulo, é apresentado o sistema de APS desenvolvido e o pré-processamento realizado no PropBank-Br. São detalhadas as adaptações feitas para se empregar uma arquitetura de dois estágios em lugar do modelo original do SENNA. São apresentados os resultados de experimentos com o PropBank-Br, incluindo testes de certas características particulares do sistema, e discutem-se os resultados obtidos.

5.1 Implementação

O sistema de APS foi totalmente desenvolvido em Python e Cython, um superconjunto de Python que produz extensões em C. Esta última foi usada para a execução das redes neurais, computacionalmente custosa, em conjunto com a biblioteca NumPy. A linguagem Python permite a implementação de diversos tipos de pré-processamento (principalmente operações com *strings*) de forma bastante fácil, e tem suporte em diversas plataformas.

O funcionamento geral do classificador é o mesmo apresentado pelo sistema SENNA (Seção 3.2.3). Cada *token* lido é convertido em um vetor de números reais e fornecido para uma rede neural. Os vetores usados foram obtidos pelo HAL, conforme descrito no Capítulo 4. A rede emprega uma camada oculta de convolução após a entrada, podendo ser seguida por uma camada convencional (foram feitos experimentos com e sem).

Na tarefa de APS, foram feitos experimentos com duas arquiteturas, a saber:

1. Uma única rede neural que delimita os argumentos e atribui seus rótulos (mesma estratégia usada pelo SENNA). Para reconhecer os limites dos argumentos, foi usado o esquema IOB, em contraste com o IOBES do SENNA. A escolha se deu pelo fato

de que o IOBES implicaria em uma maior esparsidade dos dados, já que haveria quatro etiquetas distintas para cada papel semântico. Dada a quantidade de dados do PropBank-Br, este é um fator crítico. Um exemplo da APS com o esquema IOB é mostrado na Figura 5.1.

B-A0	**B-V**	**B-A2**	**I-A2**	**I-A2**	**B-A1**	**I-A1**	**I-A1**	**I-A1**	**I-A1**	**I-A1**
Ele	**estima**	em	dez	horas	o	tempo	necessário	para	classificar	as
I-A1	**I-A1**	**I-A1**	**I-A1**	**I-A1**	**I-A1**	**I-A1**	**I-A1**	**I-A1**	**I-A1**	**O**
áreas	e	culturas	plantadas	em	uma	extensão	de	180	quilômetros	.

Figura 5.1: Exemplo de etiquetação IOB para APS

2. Uma rede neural que identifica limites de argumentos e outra que os classifica. Essa arquitetura é inspirada na estratégia da maioria dos trabalhos encontrados na literatura, conforme mostrado na Seção 3.2.1, mas se vale do princípio do SENNA de trabalhar no nível de palavras, e não de constituintes. Esta arquitetura obteve melhores resultados, e será explicada em mais detalhes a seguir.

5.1.1 Implementação de Dois Estágios

A motivação para a implementação de um processo de dois estágios é a mesma dos sistemas baseados em classificar constituintes: os atributos com maior capacidade preditiva para identificação e classificação de argumentos não são necessariamente os mesmos. Mesmo que não se empregue explicitamente atributos originários da análise sintática, as conexões da rede neural codificam algum tipo de conhecimento linguístico sobre os dados.

Para a identificação de argumentos, os *tokens* da sentença são classificados com as cinco etiquetas IOBES, sem combiná-las com nenhum papel. Uma rede neural atribui a pontuação *a priori* de cada *token* receber cada etiqueta, e uma matriz contém a pontuação para cada transição entre etiquetas. A sequência final é obtida através do algoritmo de Viterbi (Viterbi, 1967). A matriz de transições possui o valor 0 para cada transição possível e um valor negativo para cada transição impossível, grande o suficiente para inviabilizar qualquer caminho que passe por ela. A Tabela 5.1 contém uma representação da matriz.

Na classificação de argumentos, uma segunda rede neural recebe a sentença e classifica cada argumento separadamente. Não é necessária uma tabela de transições, pois não há sequências de etiquetas inválidas. É importante notar que, ao contrário da rede neural da arquitetura de uma única etapa e daquela da etapa de identificação, esta rede não classifica *tokens* separadamente, mas *argumentos* inteiros, mesmo que seus dados de entrada sejam um *token* de cada vez. Por isso, há uma diferença na forma em que as representações de

	I	**O**	**B**	**E**	**S**
I	0	$-\infty$	$-\infty$	0	$-\infty$
O	$-\infty$	0	0	$-\infty$	0
B	0	$-\infty$	$-\infty$	0	$-\infty$
E	$-\infty$	0	0	$-\infty$	0
S	$-\infty$	0	0	$-\infty$	0
$*$	$-\infty$	0	0	$-\infty$	0

Tabela 5.1: Valores iniciais para a tabela de transições entre etiquetas IOBES. O símbolo $*$ indica o começo da sentença. O valor da célula (x, y) corresponde à transição de x para y.

palavras são complementadas com vetores relativos à distância: em vez de se calcular a distância à palavra alvo, deve-se calculá-la em relação ao argumento alvo.

A adaptação é bastante semelhante ao original: é atribuída distância 0 para todos os *tokens* dentro do argumento alvo; para *tokens* à esquerda dele, é atribuída a distância ao primeira *token* do argumento; e para palavras à direita, a distância ao último *token*. O cálculo da distância ao predicador permanece inalterado.

A seguir, é mostrado um exemplo do funcionamento desta arquitetura para a Sentença 8. No primeiro estágio, os *tokens* são delimitados em argumentos, gerando uma saída como mostrado na Figura 5.2. No segundo estágio, ao classificar cada argumento, a rede neural recebe cada janela de palavras de uma vez, conforme mostrado na Tabela 5.2. Na tabela, cada *token* é acompanhado de uma janela de um outro *token* à sua esquerda e à sua direita.

(8) Ele **estima** em dez horas o tempo necessário para classificar as áreas e culturas plantadas em uma extensão de 180 quilômetros.

S	**S**	**B**	**I**	**E**	**B**	**I**	**I**	**I**	**I**	**I**
Ele	**estima**	em	dez	horas	o	tempo	necessário	para	classificar	as
I	**I**	**I**	**I**	**I**	**I**	**I**	**I**	**I**	**E**	**O**
áreas	e	culturas	plantadas	em	uma	extensão	de	180	quilômetros	.

Figura 5.2: Exemplo de etiquetação IOBES para delimitação de argumentos

5.1.2 Identificação de Predicadores

Identificar quais são os predicadores em uma dada sentença é necessário para ambas as abordagens acima. Conforme explicado na Seção 3.2.3, as redes neurais precisam saber a distância entre cada *token* e o predicador.

Palavra 1	DA	DP	Palavra 2	DA	DP	Palavra 3	DA	DP
PADDING	-3	-2	Ele	-2	-1	estima	-1	0
Ele	-2	-1	estima	-1	0	em	0	1
estima	-1	0	em	0	1	dez	0	2
em	0	1	dez	0	2	horas	0	3
dez	0	2	horas	0	3	o	1	4
horas	0	3	o	1	4	tempo	2	5
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮

DA: Distância ao argumento alvo
DP: Distância ao predicador

Tabela 5.2: Exemplo do funcionamento da convolução na rede neural para classificar o argumento *em dez horas* em relação ao predicador *estima*. Note-se que nos três *tokens* que constituem o argumento, a distância ao alvo é zero. Assim como na rede de uma etapa, *PADDING* indica uma posição fora dos limites da sentença e cada linha da tabela representa uma entrada para o sistema, que precisa ser convertida em vetores numéricos. Após a rede ter recebido e processado todas as janelas, cada neurônio da camada de convolução emite o valor máximo que produziu. Para classificar os outros dois argumentos, é necessário repetir o processo variando os valores de distância.

No entanto, esta não é uma tarefa trivial, dado que nem todos os verbos são predicadores no PropBank-Br (embora a recíproca seja verdadeira): verbos auxiliares e modais não são considerados predicadores, assim como o verbo *ser*. Mais dificuldades surgem do fato de verbos como *ter* serem frequentemente usados tanto como auxiliares quanto como verbos principais. Além disso, os verbos *ser* e *ir* possuem formas idênticas no pretérito perfeito do indicativo e no imperfeito do subjuntivo.

Acredita-se que examinar o contexto local de cada *token* seja suficiente para determinar se o mesmo é um predicador ou não. Para esse fim, foi empregada uma rede neural MLP que recebe uma janela de *tokens* para classificar o seu elemento central; após experimentos, verificou-se que uma janela de tamanho 7 fornecia melhores resultados. Esta rede não realiza convolução, dado que sua entrada tem tamanho fixo, mas ajusta as representação vetoriais de entradas da mesma forma que as outras redes.

5.2 Resultados

Nesta seção são apresentados os resultados obtidos na APS. A título de comparação, são apresentados na Tabela 5.3 os valores do *baseline* de Alva Manchego e Rosa (2012) e do classificador automático de Alva Manchego (2013). No entanto, deve-se ter em mente que estes resultados foram obtidos aproveitando a anotação sintática corrigida (incluindo árvores completas) presente no PropBank-Br, o que não corresponde a um cenário realista.

Além disso, o *baseline* identifica apenas os argumentos do tipo **Arg0**, **Arg1** e **ArgM-NEG** (negação).

Sistema	**Precisão**	**Cobertura**	**Medida F**	**Acurácia**
baseline	64,6%	40,9%	50,1	–
Classificador supervisionado	83,0%	81,7%	82,3	–
Identificação	94,9%	93,7%	94,3	–
Classificação	–	–	–	85,5%

Tabela 5.3: Valores de comparação para APS. As duas últimas linhas da tabela se referem às etapas de identificação de classificação de argumentos do classificador supervisionado.

Nos experimentos realizados, foram testadas diferentes parametrizações variando o tamanho da janela de palavras de entrada, quantidade de neurônios da rede e taxas de aprendizado. Aqui são reportados os melhores resultados obtidos em cada arquitetura, e também alguns abaixo do ótimo, visando evidenciar o impacto na performance de algumas variações de configurações.

Quanto ao tamanho da janela, exceto pela rede responsável pela detecção de predicadores, os melhores resultados foram obtidos com três palavras, de modo que todas as configurações apresentadas aqui usaram esse tamanho. Para cada palavra, a camada de entrada da rede recebe 50 valores de sua representação vetorial, mais 5 da representação de letras maiúsculas, 5 para a distância ao predicador, 5 para a distância ao *token* alvo e 5 para as etiquetas de *chunking*, quando usadas. Dessa forma, há 195 ou 210 neurônios na camada de entrada. No caso da detecção de predicadores, é usada uma janela de tamanho 7 contendo apenas as representações vetoriais das palavras e de letras maiúsculas, totalizando 385 neurônios.

Em todos os experimentos, as representações de palavras foram iniciadas com os valores obtidos pelo HAL e alteradas pela retropropagação da rede neural, exceto onde indicado o contrário. Os vetores que codificam presença de letras maiúsculas, exemplificados na Seção 3.2.3, também foram usados.

Experimentou-se também avaliar o benefício trazido ao se agregar mais conhecimento linguístico sobre as sentenças de entrada na forma de *chunks* sintáticos. Essas informações, quando usadas, são provenientes da anotação corrigida presente no PropBank-Br; no entanto, como ela não delimita *chunks* explicitamente, foi usado um conjunto de regras para extraí-los das árvores sintáticas, listadas a seguir. As etiquetas usadas para *chunk* foram formadas pelo esquema IOB e as seguintes categorias: `NP` (substantivo como núcleo), `VP` (verbo como núcleo), `CL` (introdução de uma oração, normalmente tem uma conjunção como núcleo), `ADJP` (adjetivo como núcleo), `ADVP` (advérbio como núcleo) e `O` (outros).

- A príncipio, a categoria atribuída é o tipo de nó não-terminal mais baixo da árvore sintática ao qual o *token* está ligado.

- Se a categoria não estiver entre as cinco reconhecidas, é atribuída a etiqueta `O`. Este é o caso, por exemplo, de interjeições e sinais de pontuação.

- Se o nó não-terminal for uma subárvore direta de outro nó com maior prioridade, é usada a categoria do nó pai (regra verificada recursivamente, até se chegar ao ancestral com maior prioridade). A prioridade de categorias é a seguinte (em ordem decrescente): `PP` → `NP` → `ADJP` → `ADVP`. Seguindo o mesmo raciocínio, um `VP` deve ter prioridade sobre um `ADVP`; no entanto, nas árvores sintáticas do PropBank-Br, um `ADVP` nunca é uma subárvore de um `VP`.

Um exemplo de sentença com etiquetas de *chunks* é mostrado na Figura 5.4 e sua árvore sintática na Figura 5.3.

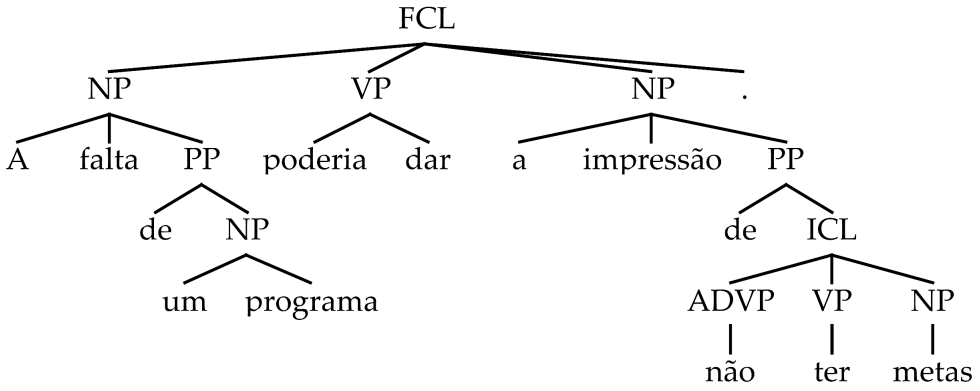

Figura 5.3: Árvore sintática para a sentença mostrada na Figura 5.4. `FCL` e `ICL` indicam dois tipos de oração; respectivamente, oração com verbo finito e infinito.

B-NP	**I-NP**	**B-PP**	**I-PP**	**I-PP**	**B-VP**	**I-VP**
A	falta	de	um	programa	poderia	dar
B-NP	**I-NP**	**B-PP**	**B-ADVP**	**B-VP**	**B-NP**	**O**
a	impressão	de	não	ter	metas	.

Figura 5.4: Exemplo de etiquetação de *chunks* extraída do PropBank-Br.

Exceto onde indicado o contrário, os predicadores de cada sentença também foram lidos dos arquivos do PropBank-Br. Isto foi feito para que se pudesse avaliar o desempenho do sistema sem a propagação de erros. Apenas as melhores configurações foram avaliadas com predicadores determinados automaticamente, conforme mostrado nos últimos experimentos.

É importante salientar também que o processo de treinamento das redes neurais para a APS é bastante custoso computacionalmente. O treinamento de cada modelo pode durar de cerca de 10 a 30 horas, dependendo da quantidade de conexões da rede. Portanto,

como se desejava avaliar diversas variantes do sistema, optou-se por usar somente a versão padrão do algoritmo de retropropagação. O uso de outro algoritmo de treinamento com as configurações que obtiveram os melhores resultados é uma investigação interessante que pode ser explorada futuramente.

5.2.1 Arquitetura de Um Estágio

Na Tabela 5.4 são apresentados os melhores resultados obtidos pela arquitetura de um estágio em diferentes configurações. A primeira linha corresponde ao sistema em sua configuração padrão, alcançando uma pontuação na medida F superior ao *baseline* em cerca de 12,2 pontos. A segunda linha mostra o melhor resultado obtido por uma rede neural que não emprega a camada oculta convencional após a convolução. A diferença de cerca de 2,2 pontos na medida F em relação ao primeiro modelo evidencia sua importância. Já na terceira linha, pode-se ver o ganho de performance trazido pelo conhecimento linguístico agregado. Mesmo assim, a medida F ainda está cerca de 15,4 pontos abaixo daquela do sistema supervisionado de Alva Manchego e Rosa.

Descrição	**Precisão**	**Cobertura**	**Medida F**
Configuração padrão	64,41%	60,34%	62,31
Sem camada oculta	63,30%	57,17%	60,08
Uso de *chunks* sintáticos	68,35%	65,55%	66,92
Vetores aleatórios	58,48%	55,87%	57,14

Tabela 5.4: Resultados da APS com a arquitetura de um estágio.

Por último, vê-se a diferença de desempenho causada pelo uso de representações de palavras geradas aleatoriamente. Embora estas sejam ajustadas ao longo do treinamento, não é suficiente para que o sistema atinja bons níveis de performance, ficando mais de 5 pontos abaixo do modelo que emprega representações do HAL.

5.2.2 Arquitetura de Dois Estágios

Na Tabela 5.5 são apresentados os resultados obtidos pela rede neural responsável pela delimitação de argumentos na arquitetura de dois estágios. Na primeira linha, é apresentado o melhor resultado encontrado pelo sistema sem usar nenhum tipo de conhecimento linguístico explícito. Sua pontuação na medida F encontra-se quase 20 pontos abaixo do sistema de Alva Manchego, uma diferença bastante significativa. Esta pode ser atribuída ao fato de que, para identificar argumentos eficientemente, o sistema precisa aprender certas regularidades linguísticas encontradas no texto (como os limites de cada sintagma), enquanto o classificador de Alva Manchego as abstrai da árvore sintática. O tamanho do

PropBank-Br parece ser insuficiente para que a rede neural desenvolva bem esse tipo de conhecimento. De fato, ao se empregar *chunks* sintáticos, o desempenho do sistema melhora em quase 5 pontos na medida F.

Descrição	**Precisão**	**Cobertura**	**F1**
Configuração padrão	76,42%	72,44%	74,38
Sem camada oculta	72,80%	70,76%	71,77
Uso de *chunks* sintáticos	80,15%	78,96%	79,17

Tabela 5.5: Resultados da identificação de argumentos na arquitetura de dois estágios

A segunda linha da tabela reitera a constatação da importância da camada oculta após a convolução. Sem esta, os melhores resultados obtidos ficaram 2,6 pontos abaixo da medida F do primeiro modelo.

Para a classificação de argumentos, os resultados obtidos são apresentados na Tabela 5.6. Os testes foram feitos fornecendo-se a cada rede neural as sentenças com os argumentos corretamente delimitados, e a acurácia corresponde à proporção de argumentos que receberam o rótulo correto.

A configuração com a melhor performance teve uma acurácia cerca de 2,5% superior àquela do sistema de Alva Manchego, sem aproveitar nenhuma anotação linguística nos dados. Isso pode ser atribuído ao fato de que, para a classificação de argumentos, a estrutura sintática da sentença não é tão importante quanto atributos que reflitam o significado das palavras do constituinte. Também por esse motivo, não se optou incluir dados de *chunks* sintáticos no treinamento. É interessante notar também que a rede sem camada oculta após a convolução foi a que obteve melhores resultados. Aparentemente, por esta ser uma tarefa mais fácil, a camada extra adiciona uma complexidade desnecessária à rede.

Descrição	**Acurácia**
Configuração padrão	87,50%
Sem camada oculta	88,14%

Tabela 5.6: Resultados da classificação de argumentos na arquitetura de dois estágios

Combinando os melhores modelos treinados para cada etapa, obteve-se um classificador de dois passos. Seus resultados são apresentados na Tabela 5.7. Em comparação com a arquitetura de um estágio, pode-se ver uma melhora de aproximadamente 2,8 pontos na medida F na configuração padrão e de 1 ponto ao se usar conhecimento linguístico explícito. Estes resultados corroboram a vantagem de se separar as duas etapas da APS.

Todos os resultados apresentados até aqui foram obtidos por sistemas sem restrições quanto a rótulos de argumentos repetidos na mesma sentença. Segundo a teoria de papéis semânticos, cada papel central (os argumentos numerados) só pode ser atribuído a, no máximo, um elemento da estrutura profunda da proposição e, embora possa haver repetição

Descrição	Precisão	Cobertura	Medida F
Configuração padrão	67,06%	63,31%	65,13
Uso de *chunks* sintáticos	68,97%	67,04%	67,99

Tabela 5.7: Resultados da APS em dois estágios

de argumentos modificadores (os **ArgM**), tais casos são raros. Em 9 (sentença extraída do PropBank-Br), é mostrado um exemplo de sentença com um papel repetido. Além disso, ao se remover as ocorrências dos rótulos prefixados por **C-**, conforme mostrado na Seção 2.2, passou a haver também repetição de argumentos numerados, como exemplificado em 10. Essas foram as motivações para se permitir esse tipo de comportamento do sistema.

(9) Ele fala que tem idéias e não um programa, mas [$_{\text{ArgM-TMP}}$ muitas vezes], [$_{\text{ArgM-TMP}}$ quando perguntado] [$_{\text{Arg0}}$ ele] **mostra** [$_{\text{Arg1}}$ que também não tem nem idéia].

(10) **Tem** [$_{\text{Arg1}}$ sentido] – [$_{\text{ArgM-DIS}}$ aliás], [$_{\text{Arg1}}$ muitíssimo sentido].

Por outro lado, este fenômeno é bastante raro, ocorrendo em pouco mais de 3% das proposições do PropBank-Br; e a possibilidade de repetir rótulos de argumentos permite que o sistema cometa erros como mostrado em (11) (onde o segundo argumento é, na verdade, um **Arg2**). Portanto, buscou-se avaliar o sistema com restrições quanto à repetição de papéis, que funcionam da seguinte forma.

(11) $*$ [$_{\text{Arg1}}$ Nem todos] passam [$_{\text{Arg1}}$ pelas duas fases].

A príncipio, o rótulo de cada argumento é determinado pelo neurônio de saída que tiver o valor mais alto. Quando dois ou mais argumentos forem candidatos a um mesmo rótulo y_i, são determinadas as probabilidades de cada um pela função softmax (descrita na Equação 5.1) aplicada sobre a pontuação de todos os neurônios de saída. O argumento que tiver a maior probabilidade de possuir aquele rótulo o recebe, e os demais deverão ter outro atribuído. É possível que, após esta segunda atribuição, ainda haja conflitos; neste caso, o processo é repetido quantas vezes forem necessárias.

$$P(y_i) = \frac{exp(y_i)}{\sum_j exp(y_j)} \tag{5.1}$$

No entanto, os resultados do sistema sob essa configuração (mostrados na Tabela 5.8) apresentam uma pequena queda no desempenho. Após examinar os resultados, constatou-se que os erros devidos à repetição de argumentos ocorrem em quantidade desprezível. Considerando-se que esta restrição sobre o sistema pode piorar seu desempenho, ja que

Descrição	Precisão	Cobertura	Medida F
Configuração padrão	66,67%	62,94%	64,75
Uso de *chunks* sintáticos	68,58%	66,67%	67,61

Tabela 5.8: Resultados da APS sem rótulos de argumentos repetidos

impede os casos em que um rótulo é repetido de serem identificados corretamente, optou-se por não usá-la. Ainda assim, a implementação disponibilizada do sistema oferece a possibilidade de ativá-la.

Por fim, apresentam-se os resultados obtidos ao se identificar os predicadores automaticamente. Este experimento corresponde ao cenário do mundo real onde a única entrada do sistema é o texto para ser etiquetado. A rede responsável por identificar predicadores é executada, e a informação sobre quais *tokens* são predicadores é repassada às outras duas. Os resultados são mostrados na Tabela 5.9

Descrição	Precisão	Cobertura	F1
Identificação de predicadores	88,62%	91,21%	89,90
APS sem *chunks*	66,95%	58,10%	62,21
APS com *chunks*	69,01%	62,20%	65,43

Tabela 5.9: Resultados da APS com predicadores identificados automaticamente

Como Alva Manchego (2013) não detecta predicadores automaticamente, não há outro sistema com o qual se possa comparar os resultados acima. No entanto, acredita-se que uma medida F abaixo de 90 possa ser bem melhorada; em especial, através do uso de etiquetas de classes gramaticais que diferenciem verbos auxiliares dos principais. Como este não é o caso do PropBank-Br, não foram usadas suas etiquetas. Ideias interessantes para melhorar o identificador seria consultar uma lista de verbos ou treinar um identificador de classes gramaticais no *corpus* MacMorpho (Aluísio et al., 2003), que faz esse tipo de distinção.

Os resultados do sistema final com detecção automática de predicadores são mostrados na Tabela 5.10, separados por tipo de argumento. Como esperado, a performance do sistema foi melhor para os argumentos mais comuns. Além destes, também se teve muito bons resultados para os argumentos de negação (**ArgM-NEG**). Esses são na maioria das vezes apenas a palavra *não*, o que os torna facilmente detectáveis.

Os resultados para predicadores (item V na Tabela 5.10) não são os mesmos da Tabela 5.9 devido aos predicadores com mais de uma palavra. O módulo identificador de predicadores precisa detectar apenas a palavra principal de cada proposição (como *bater* em *bater palmas*), enquanto o resultado final do sistema inclui todos os *tokens*.

As dimensões das redes usadas no sistema final são as seguintes. A rede neural responsável pela identificação de predicadores tem apenas uma camada oculta com 100 neurônios.

A rede que delimita os argumentos tem 200 neurônios na camada de convolução e 200 na camada oculta que a segue. A rede que classifica os argumentos tem 250 neurônios na camada de convolução.

Rótulo	Precisão	Cobertura	Medida F
A0	76.72%	71.77%	74.17
A1	65.24%	64.62%	64.93
A2	40.00%	31.58%	35.29
A3	0.00%	0.00%	0.00
A4	0.00%	0.00%	0.00
AM-ADV	62.50%	25.00%	35.71
AM-CAU	25.00%	100.00%	40.00
AM-DIR	0.00%	0.00%	0.00
AM-DIS	50.00%	54.55%	52.17
AM-EXT	0.00%	0.00%	0.00
AM-LOC	63.64%	51.85%	57.14
AM-MNR	44.44%	22.22%	29.63
AM-NEG	100.00%	89.47%	94.44
AM-PNC	0.00%	0.00%	0.00
AM-PRD	0.00%	0.00%	0.00
AM-REC	100.00%	40.00%	57.14
AM-TMP	75.76%	55.56%	64.10
V	95.85%	87.03%	91.23

Tabela 5.10: Resultados do sistema final separados por tipo de argumento.

5.3 Considerações Finais

Neste capítulo, apresentou-se a implementação do sistema de APS, principal objetivo deste projeto e a avaliação de sua perfomance sob diferentes parametrizações. Foram incorporadas alterações no modelo original do SENNA, dividindo o funcionamento em uma etapa para identificação e outra para classificação de argumentos. Com esta nova arquitetura, foi possível chegar a resultados melhores do que com o modelo de uma única etapa.

Também foi mostrada a importância do uso de boas representações vetoriais para palavras e de uma camada oculta após a convolução da rede neural em algumas tarefas. Foram feitas comparações com uma variante do sistema que emprega conhecimento linguístico explícito, para se avaliar o possível ganho de performance. Por fim, experimentou-se também impedir o sistema de classificar mais de um argumento com o mesmo rótulo, o que não melhorou seu desempenho.

Graças à sua independência de ferramentas externas, especialmente quanto a um *parser* sintático, o sistema apresentado aqui poderá ser facilmente utilizado por outros pesquisadores; espera-se assim contribuir para o PLN em língua portuguesa.

A performance do sistema aqui apresentado ficou significativamente aquém daquela do trabalho de Alva Manchego (2013). O gargalo, conforme apontam os resultados, é a fase de identificação de argumentos, enquanto a classificação obteve os melhores resultados para a língua portuguesa até então.

Os autores do sistema SENNA informam que ele alcança uma performance de 74,15% no PropBank sem usar qualquer tipo de conhecimento linguístico explícito (Collobert et al., 2011), o que já é bastante próximo do estado da arte em inglês. Esse resultado indica que, havendo dados de treinamento em uma grande quantidade, o esse tipo de arquitetura é capaz de compensar a ausência de anotação sintática. Acredita-se que para melhorar o desempenho do sistema aqui descrito seja imprescindível que se aumente o tamanho do PropBank-Br.

Capítulo

6

Conclusões

Este projeto teve como objetivo desenvolver um sistema para APS em língua portuguesa, evitando a dependência de um *parser* sintático e outras ferramentas externas de PLN. Foi seguida a arquitetura do sistema SENNA, que obteve bons resultados na língua inglesa. Como objetivo parcial, introduzido pela escolha dessa arquitetura, estava a geração de representações para palavras na forma de vetores de números reais.

Para tanto, a revisão da literatura incluiu trabalhos sobre papéis semânticos (tanto aspectos teóricos quanto a automazição da tarefa) e representações numéricas de palavras. Foi compilado um grande *corpus* de domínio geral, a partir da Wikipédia e do *corpus* PLN-BR, sobre o qual foram feitos experimentos com VSMs e escolheu-se um modelo para ser usado na APS.

Por fim, foi implementado o sistema de APS, incorporando uma divisão entre identificação e classificação de argumentos, ausente no SENNA. Diferentes parametrizações do sistema foram testadas, e os resultados obtidos discutidos aqui. O sistema desenvolvido é o único atualmente que se encontra pronto e disponível para realizar APS em portugues, seguindo o estilo do PropBank-Br.

A performance alcançada pelo sistema ainda está bastante abaixo do estado da arte em inglês, ou mesmo de outro sistema para português que emprega informações produzidas por ferramentas externas. Os resultados de experimentos indicam que o gargalo da performance encontra-se na fase de identificação de argumentos, e um aumento da quantidade de dados de treino melhoraria seu funcionamento, na medida em que o sistema aprenderia melhor a reconhecer os limites de constituintes sintáticos.

Há alguns pontos que poderiam ser melhor investigados em trabalhos futuros, listados

a seguir.

- Uso de outras técnicas de VSM, como o modelo neural, e/ou o uso de formas lematizadas das palavras.
- Aproveitar representações vetoriais de palavras treinadas em outras tarefas de PLN, como classificação morfossintática.
- Uso de outros tipos de estrutura interna nas redes neurais, ou mesmo outros classificadores.
- Agrupar verbos de acordo com o conjunto de argumentos que podem receber e treinar um classificador para cada grupo. No entanto, essa abordagem provavelmente necessitaria de um aumento do tamanho atual do PropBank-Br, pois cada classificador teria uma quantidade ainda menor de dados de treino.
- Realizar a APS considerando apenas os argumentos mais facilmente identificados pelo sistema. A anotação seria menos completa, mas é de se esperar que a performance do sistema melhore, dado que haveria menos complexidade no problema.
- Uso de técnicas de aprendizado semissupervisionado para contornar a escassez de dados anotados. Estas poderiam ser facilmente aplicadas, ja que o sistema está pronto para uso e sem dependências de ferramentas externas.

Apêndice

A

Exemplos de Anotação Automática

A seguir, são mostrados exemplos da saída do sistema desenvolvido para algumas sentenças. Todos as saídas foram produzidas pela configuração de dois estágios que obteve os melhores resultados sem conhecimento explícito e usando detecção automática de predicadores. As sentenças fornecidas como entrada foram extraídas do conjunto de testes do PropBank-Br, exceto pela que foi usada na Figura A.3 (sentença sem predicador).

As saídas são organizadas da seguinte forma. Cada linha diz respeito a um *token*, que é mostrado na primeira coluna. A segunda coluna repete o *token* caso ele seja um predicador e mostra apenas o símbolo – caso contrário. Em seguida, para cada predicador presente na sentença, há uma coluna com o rótulo do *token*.

Na Figura A.1 é mostrada a resposta do sistema para a sentença *Ele estima em dez horas o tempo necessário para classificar as áreas e culturas plantadas em uma extensão de 180 quilômetros*. O sistema reconheceu os dois predicadores e produziu a anotação correta com respeito a ambos.

Já na Figura A.2 é mostrado um caso incomum, em que o sujeito do verbo assume o papel de **Arg1** em vez de **Arg0** (conforme ocorre com verbos de movimento). Como discutido no Capítulo 5, tais casos podem confundir o sistema, já que o padrão encontrado na maioria das sentenças é um **Arg0** antes do verbo e um **Arg1** após. Ainda assim, o sistema foi capaz de emitir a resposta correta.

A resposta para uma sentença sem predicadores é mostrada na Figura A.3. Na Figura A.4, é apresentada uma saída onde o sistema cometeu dois erros. O **Arg1**, na realidade, é composto pelas palavras *aulas de alfabetização para jovens de 14 a 22 anos*. Não há um **Arg2** na sentença; no entanto, constituintes iniciados pela preposição *para* frequente-

Ele	-	S-A0	O
estima	estima	S-V	O
em	-	B-A2	O
dez	-	I-A2	O
horas	-	E-A2	O
o	-	B-A1	O
tempo	-	I-A1	O
necessário	-	I-A1	O
para	-	I-A1	O
classificar	classificar	I-A1	S-V
as	-	I-A1	B-A1
áreas	-	I-A1	I-A1
e	-	I-A1	I-A1
culturas	-	I-A1	I-A1
plantadas	-	I-A1	I-A1
em	-	I-A1	I-A1
uma	-	I-A1	I-A1
extensão	-	I-A1	I-A1
de	-	I-A1	I-A1
180	-	I-A1	I-A1
quilômetros	-	E-A1	E-A1
.	-	O	O

Figura A.1: Resposta do sistema para sentença com dois predicadores.

Nem	-	B-A1
todos	-	E-A1
passam	passam	S-V
pelas	-	B-A2
duas	-	I-A2
fases	-	E-A2
.	-	O

Figura A.2: Resposta do sistema com verbo de movimento.

```
Isso                    -
é                       -
verdade                 -
.                       -
```

Figura A.3: Resposta do sistema para sentença sem predicador.

```
À                   -                   B-AM-TMP
noite               -                   E-AM-TMP
,                   -                   O
o                   -                   B-A0
projeto             -                   E-A0
prevê               prevê               S-V
aulas               -                   B-A1
de                  -                   I-A1
alfabetização       -                   E-A1
para                -                   B-A2
jovens              -                   I-A2
de                  -                   I-A2
14                  -                   I-A2
a                   -                   I-A2
22                  -                   I-A2
anos                -                   I-A2
nos                 -                   I-A2
Cieps               -                   E-A2
.                   -                   O
```

Figura A.4: Resposta do sistema com erro na detecção do argumento local.

mente recebem esse rótulo, o que parece ter confundido o sistema. Além disso, as palavras *nos Cieps* constituem um argumento de local, e deveriam ser rotuladas com **ArgM-LOC**. Esse tipo de erro indica a dificuldade do sistema em tratar corretamente argumentos distantes do predicador.

Referências Bibliográficas

Afonso, S., Bick, E., Haber, R., e Santos, D. (2002). Floresta sintá(c)tica: a treebank for Portuguese. In *Proceedings of the Third International Conference on Language Resources and Evaluation (LREC 2002)*, páginas 1698–1703.

Aluísio, S., Pelizzoni, J., Marchi, A. R., de Oliveira, L., Manenti, R., e Marquiafável, V. (2003). An account of the challenge of tagging a reference corpus for brazilian portuguese. In *Proceedings of the 6th international conference on Computational processing of the Portuguese language*, PROPOR'03, páginas 110–117, Berlin, Heidelberg. Springer-Verlag.

Alva Manchego, F. E. (2013). Anotação automática semissupervisionada de papéis semânticos para o português do Brasil. Dissertação de mestrado, Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo.

Alva Manchego, F. E. e Rosa, J. L. G. (2012). Semantic Role Labeling for Brazilian Portuguese: A Benchmark. In *IBERAMIA 2012*, volume 7637 of *Lecture Notes in Artificial Intelligence*, páginas 481–490. Springer.

Amancio, M. A. (2011). Elaboração textual via definição de entidades mencionadas e de perguntas relacionadas aos verbos em textos simplificados do português. Dissertação de Mestrado, ICMC – USP, São Carlos, SP.

Baker, C. F., Fillmore, C. J., e Lowe, J. B. (1998). The Berkeley FrameNet project. In *Proceedings of the 17th international conference on Computational linguistics - Volume 1*, COLING '98, páginas 86–90.

Bengio, Y. (2009). Learning deep architectures for AI. *Foundations and Trends in Machine Learning*, 2(1):1–127.

Berger, A. L., Pietra, S. A. D., e Pietra, V. J. D. (1996). A Maximum Entropy Approach to Natural Language Processing. *Computational Linguistics*, 22(1):39–71.

Bick, E. (2000). *The parsing system PALAVRAS: automatic grammatical analysis of Portuguese in a constraint grammar framework*. Tese de Doutorado, Department of Linguistics – Aarhus University.

Bick, E. (2007). Automatic semantic role annotation for Portuguese. In *Proceedings of TIL 2007-5th Workshop on Information and Human Language Technology*, páginas 1713–1716.

Bick, E., Santos, D., Afonso, S., e Marchi, R. (2007). Floresta sintá(c)tica: Ficção ou realidade? In *Avaliação conjunta: um novo paradigma no processamento computacional da língua portuguesa*, páginas 291–300. IST Press.

Bruckschen, M., Muniz, F., Souza, J. G. C., Fuchs, J. T., Infante, K., Muniz, M., Gonçalves, P. N., Vieira, R., e Aluísio, S. M. (2008). Anotação Lingüística em XML do Corpus PLN-BR. Relatório técnico.

Carreras, X. e Màrquez, L. (2004). Introduction to the CoNLL-2004 shared task: Semantic role labeling. In Ng, H. T. e Riloff, E., editors, *HLT-NAACL 2004 Workshop: Eighth Conference on Computational Natural Language Learning (CoNLL-2004)*, páginas 89–97.

Carreras, X. e Màrquez, L. (2005). Introduction to the CoNLL-2005 shared task: semantic role labeling. In *Proceedings of the Ninth Conference on Computational Natural Language Learning*, CONLL '05, páginas 152–164.

Charniak, E. e Johnson, M. (2005). Coarse-to-fine n-best parsing and MaxEnt discriminative reranking. In *Proceedings of the 43rd Annual Meeting of the ACL*, páginas 173–180.

Collins, M. J. (1999). *Head-driven statistical models for natural language parsing*. Tese de Doutorado, University of Pennsylvania, Philadelphia, USA.

Collobert, R. (2011). Deep learning for efficient discriminative parsing. In *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, volume 15, páginas 224–232.

Collobert, R. e Weston, J. (2008). A unified architecture for natural language processing: Deep neural networks with multitask learning. In *International Conference on Machine Learning, ICML*.

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

Cortes, C. e Vapnik, V. (1995). Support-vector networks. *Machine Learning*, 20:273–297.

Dowty, D. (1991). Thematic proto-roles and argument selection. *Language*, 67(3):547–619.

Duran, M. S. e Aluísio, S. M. (2011). Propbank-Br: a Brazilian Portuguese corpus annotated with semantic role labels. In *Proceedings of STIL 2011 8th Brazilian Symposium in Information and Human Language Technology*.

Duran, M. S. e Aluísio, S. M. (2012). Propbank-br: a brazilian treebank annotated with semantic role labels. In Calzolari, N., Choukri, K., Declerck, T., Doğan, M. U., Maegaard, B., Mariani, J., Odijk, J., e Piperidis, S., editors, *Proceedings of the Eight International Conference on Language Resources and Evaluation (LREC'12)*, Istanbul, Turkey. European Language Resources Association (ELRA).

Fillmore, C. J. (1968). The case for case. In Bach, E. W. e Harms, R. T., editors, *Universals in Linguistic Theory*, páginas 1–88. Holt, Rinehart & Winston, New York.

Fillmore, C. J. (1985). Frames and the semantics of understanding. *Quaderni di semantica*, 6(2):222–254.

Fonseca, E. R. e Rosa, J. L. G. (2012). An Architecture for Semantic Role Labeling on Portuguese. In Caseli, H., Villavicencio, A., Teixeira, A., e Perdigão, F., editors, *PROPOR 2012 – International Conference on Computational Processing of Portuguese*, volume 7243 of *Lecture Notes in Computer Science*, páginas 204–209. Springer.

Gildea, D. e Jurafsky, D. (2002). Automatic labeling of semantic roles. *Computational Linguistics*, 28(3):245–288.

Haghighi, A., Toutanova, K., e Manning, C. D. (2005). Joint learning improves semantic role labeling. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, páginas 589–596.

Haykin, S. (2009). *Neural Networks and Neural Machines*. Prentice Hall.

Huang, E. H., Socher, R., Manning, C. D., e Ng., A. Y. (2012). Improving Word Representations via Global Context and Multiple Word Prototypes. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics*, páginas 873–882.

Kaisser, M. e Webber, B. (2005). Question answering based on semantic roles. In *Proceedings of the 5th Workshop on Important Unresolved Matters*, páginas 41–48.

Karlgren, J. e Sahlgren, M. (2001). *From Words to Understanding*, páginas 294 – 308. CSLI Publications.

Koomen, P., Punyakanok, V., Roth, D., e Yih, W. (2005). Generalized Inference with Multiple Semantic Role Labeling Systems. In *Proceedings of the 9th Conference on Computational Natural Language Learning (CoNLL)*, páginas 181–184.

Kundu, G. e Roth, D. (2011). Adapting Text instead of the Model: An Open Domain Approach. In *Proceedings of the Fifteenth Conference on Computational Natural Language Learning*, páginas 229–237. Association for Computational Linguistics.

Landauer, T. K. e Dumais, S. T. (1997). A Solution to Plato's Problem: The Latent Semantic Analysis Theory of Acquisition, Induction, and Representation of Knowledge. *Psychological Review*, 104(2):211–240.

Levin, B. (1993). *English verb classes and alternations: A preliminary investigation*. University Of Chicago Press.

Levin, B. e Rappaport Hovav, M. (2005). *Argument Realization*. Research Surveys in Linguistics. Cambridge University Press.

Lund, K. e Burgess, C. (1996). Producing high-dimensional semantic spaces from lexical co-occurrence. *Behavior Research Methods, Instruments, & Computers*, 28(2):203–208.

Màrquez, L., Carreras, X., Litkowski, K. C., e Stevenson, S. (2008). Semantic role labeling: An introduction to the special issue. *Computational Linguistics*, 34(2):145–159.

Melli, G., Wang, Y., Liu, Y., Kashani, M. M., Shi, Z., Gu, B., Sarkar, A., e Popowich, F. (2005). Description of SQUASH, the SFU Question Answering Summary Handler for the DUC-2005 Summarization Task. In *Proceedings of the HLT/EMNLP Document Understanding Workshop (DUC)*.

Palmer, M., Gildea, D., e Xue, N. (2010). *Semantic Role Labeling*. Synthesis Lectures on Human Language Technologies. Morgan & Claypool.

Palmer, M., Kingsbury, P., e Gildea, D. (2005). The Proposition Bank: An annotated corpus of semantic roles. *Computational Linguistics*, 31(1):71–106.

Pradhan, S., Ward, W., Hacioglu, K., e Martin, J. H. (2004). Shallow semantic parsing using support vector machines. In *HLT-NAACL 2004: Main Proceedings*, páginas 233–240.

Pradhan, S., Ward, W., e Martin, J. H. (2008). Towards robust semantic role labeling. *Computational Linguistics*, 34(2):289–310.

Punyakanok, V., Roth, D., e tau Yih, W. (2005). The necessity of syntactic parsing for semantic role labeling. In *Proceedings of the International Joint Conference on Artificial Intelligence*, páginas 1117–1123.

Rosa, J. L. G. e Adán-Coello, J. M. (2010). Biologically plausible connectionist prediction of natural language thematic relations. *Journal of universal computer science*, 16:3245–3277.

Sahlgren, M. (2005). An Introduction to Random Indexing. In *Methods and Applications of Semantic Indexing Workshop at the 7th International Conference on Terminology and Knowledge Engineering*.

Sahlgren, M. (2008). The distributional hypothesis. *Italian Journal of Linguistics*, 20(1):33–52.

Salomão, M. M. M. (2009). FrameNet Brasil: Um trabalho em progresso. *Calidoscópio*, 7(3):171–182.

Sequeira, J., Gonçalves, T., e Quaresma, P. (2012). Semantic role labeling for portuguese – a preliminary approach –. In Caseli, H., Villavicencio, A., Teixeira, A., e Perdigão, F., editors, *Computational Processing of the Portuguese Language*, volume 7243 of *Lecture Notes in Computer Science*, páginas 193–203. Springer Berlin / Heidelberg.

Surdeanu, M., Harabagiu, S., Williams, J., e Aarseth, P. (2003). Using predicate-argument structures for information extraction. In *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics - Volume 1*, páginas 8–15.

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*, páginas 384–394.

Turney, P. D. e Pantel, P. (2010). From Frequency to Meaning: Vector Space Models of Semantics. *Journal of Artificial Intelligence Research*, 37:141–188.

Viterbi, A. (1967). Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. *IEEE Transactions on Information Theory*, 13(2):260–269.

Wu, D. e Fung, P. (2009). Semantic Roles for SMT: A Hybrid Two-Pass Model. In *Proceedings of NAACL HLT 2009: Short Papers*, páginas 13–16.

Xue, N. (2008). Labeling Chinese Predicates with Semantic Roles. *Computational Linguistics*, 34(2):225–255.

Xue, N. e Palmer, M. (2004). Calibrating features for semantic role labeling. In *Proceedings of EMNLP, 2004*, páginas 88–94.