Pontifícia Universidade Católica
do Rio de Janeiro

Eduardo Neves Motta

Indução e Seleção Incrementais de Atributos no Aprendizado Supervisionado

Tese de Doutorado

Tese apresentada ao Programa de Pós–graduação em Informática do Departamento de Informática da PUC–Rio como requisito parcial para obtenção Do título de Doutor em Informática

Orientador: Prof. Ruy Luiz Milidiú

Rio de Janeiro
Setembro de 2014

Pontifícia Universidade Católica
do Rio de Janeiro

Eduardo Neves Motta

Indução e Seleção Incrementais de Atributos no Aprendizado Supervisionado

Tese apresentada ao Programa de Pós–graduação em Informática do Departamento de Informática do Centro Técnico Científico da PUC–Rio como requisito parcial para obtenção do título de Doutor em Informática. Aprovada pela Comissão Examinadora abaixo assinada.

Prof. Ruy Luiz Milidiú
Orientador
Departamento de Informática — PUC–Rio

Prof. Marco Antonio Casanova
Departamento de Informática — PUC-Rio

Profª. Maria Cláudia de Freitas
Departamento de Letras — PUC-Rio

Prof. Geraldo Bonorino Xexéo
COPPE-UFRJ

Dr. Cícero Nogueira dos Santos
IBM Research

Prof. Leandro Guimaraes Marques Alvim
UFRRJ

Prof. José Eugenio Leal
Coordenador Setorial do Centro Técnico Científico — PUC–Rio

Rio de Janeiro, 5 de Setembro de 2014

Eduardo Neves Motta

Bacharel em Física pela Universidade Federal do Rio de Janeiro. Mestre em Informática pela Universidade Federal do Estado do Rio de Janeiro.

Ficha Catalográfica

Motta, Eduardo N.

Indução e Seleção Incrementais de Atributos no Aprendizado Supervisionado / Eduardo Neves Motta; orientador: Ruy Luiz Milidiú. — 2014.

90 f. : il. (color); 30 cm

1. Tese (doutorado) - Pontifícia Universidade Católica do Rio de Janeiro, Departamento de Informática, 2014.

Inclui bibliografia.

1. Informática – Teses. 2. Regularização de domínios e modelos. 3. Perceptron esparso. 4. Seleção e indução de atributos. 5. SVM. I. Milidiú, Ruy L.. II. Pontifícia Universidade Católica do Rio de Janeiro. Departamento de Informática. III. Título.

CDD: 004

Agradecimentos

Ao meu orientador, professor Ruy Milidiú, por todo o incentivo, apoio e transferência de conhecimento para o desenvolvimento deste trabalho.

Ao CNPq e à PUC–Rio, pelos auxílios concedidos, sem os quais este trabalho não poderia ter sido realizado.

A todos os colegas, professores e funcionários do Departamento de Informática da PUC-Rio, pelo companheirismo, aprendizado e auxílio.

Aos amigos do LEARN por compartilharem boa parte desta jornada.

Aos meus pais, irmão e amigos, sempre presentes e incentivadores.

Resumo

Motta, Eduardo N.; Milidiú, Ruy L.. **Indução e Seleção Incrementais de Atributos no Aprendizado Supervisionado**. Rio de Janeiro, 2014. 90p. Tese de Doutorado — Departamento de Informática, Pontifícia Universidade Católica do Rio de Janeiro.

A indução de atributos não lineares a partir de atributos básicos é um modo de obter modelos preditivos mais precisos para problemas de classificação. Entretanto, a indução pode causar o rápido crescimento do número de atributos, resultando usualmente em *overfitting* e em modelos com baixo poder de generalização. Para evitar esta consequência indesejada, técnicas de regularização são aplicadas, para criar um compromisso entre um reduzido conjunto de atributos representativo do domínio e a capacidade de generalização.

Neste trabalho, descrevemos uma abordagem de aprendizado de máquina supervisionado com indução e seleção incrementais de atributos. Esta abordagem integra árvores de decisão, *support vector machines* e seleção de atributos utilizando perceptrons esparsos em um *framework* de aprendizado que chamamos IFIS – *Incremental Feature Induction and Selection*. Usando o IFIS, somos capazes de criar modelos regularizados não lineares de alto desempenho utilizando um algoritmo com modelo linear. Avaliamos o nosso sistema em duas tarefas de processamento de linguagem natural em dois idiomas. Na primeira tarefa, anotação morfossintática, usamos dois corpora, o corpus WSJ em língua inglesa e o Mac-Morpho em Português. Em ambos, alcançamos resultados competitivos com o estado da arte reportado na literatura, alcançando as acurácias de 97,14% e 97,13%, respectivamente.

Na segunda tarefa, análise de dependência, utilizamos o corpus da *CoNLL 2006 Shared Task* em português, ultrapassando os resultados reportados durante aquela competição e alcançando resultados competitivos com o estado da arte para esta tarefa, com a métrica UAS igual a 92,01%.

Com a regularização usando um perceptron esparso, geramos modelos SVM que são até 10 vezes menores, preservando sua acurácia. A redução dos modelos é obtida através da regularização dos domínios dos atributos, que atinge percentuais de até 99%. Com a regularização dos modelos, alcançamos uma redução de até 82% no tamanho físico dos modelos. O tempo de predição do modelo compacto é reduzido em até 84%. A redução dos domínios e modelos permite também melhorar a engenharia de atributos, através da análise dos domínios compactos e da introdução incremental de novos atributos.

Palavras–chave

Regularização de domínios e modelos; Perceptron esparso; Seleção e indução de atributos; SVM.

Abstract

Motta, Eduardo N.; Milidiú, Ruy L. (Advisor). **Supervised Learning Incremental Feature Induction and Selection**. Rio de Janeiro, 2014. 90p. PhD. Thesis — Departamento de Informática, Pontifícia Universidade Católica do Rio de Janeiro.

Non linear feature induction from basic features is a method of generating predictive models with higher precision for classification problems. However, feature induction may rapidly lead to a huge number of features, causing overfitting and models with low predictive power. To prevent this side effect, regularization techniques are employed to obtain a trade-off between a reduced feature set representative of the domain and generalization power. In this work, we describe a supervised machine learning approach that incrementally inducts and selects feature conjunctions derived from base features. This approach integrates decision trees, support vector machines and feature selection using sparse perceptrons in a machine learning framework named IFIS – Incremental Feature Induction and Selection. Using IFIS, we generate regularized non-linear models with high performance using a linear algorithm. We evaluate our system in two natural language processing tasks in two different languages. For the first task, POS tagging, we use two corpora, WSJ corpus for English, and Mac-Morpho for Portuguese. Our results are competitive with the state-of-the-art performance in both, achieving accuracies of 97.14% and 97.13%, respectively. In the second task, Dependency Parsing, we use the CoNLL 2006 Shared Task Portuguese corpus, achieving better results than those reported during that competition and competitive with the state-of-the-art for this task, with UAS score of 92.01%.

Applying model regularization using a sparse perceptron, we obtain SVM models 10 times smaller, while maintaining their accuracies. We achieve model reduction by regularization of feature domains, which can reach 99%. Using the regularized model we achieve model physical size shrinking of up to 82%. The prediction time is cut by up to 84%. Domains and models downsizing also allows enhancing feature engineering, through compact domain analysis and incremental inclusion of new features.

Keywords

Domain and model regularization; Sparse perceptron; Feature induction and selection; SVM.

Sumário

Lista de figuras

Lista de tabelas

Experiment!
Make it your motto day and night.
Experiment,
And it will lead you to the light.
The apple on the top of the tree
Is never too high to achieve,
So take an example from Eve,
Experiment!
Be curious,
Though interfering friends may frown.
Get furious
At each attempt to hold you down.
If this advice you always employ,
The future can offer you infinite joy
And merriment,
Experiment,
And you'll see

Cole Porter
Experiment (1933)

1
Introdução

A criação de modelos de aprendizado de máquina supervisionado depende da engenharia de atributos que representem o conhecimento sobre o domínio de interesse. Para criar atributos, o conhecimento sobre o domínio de interesse é utilizado para manualmente criar atributos básicos que expressam os fenômenos deste domínio e que são úteis para o aprendizado.

Além da criação manual de atributos a partir de conhecimento de domínio, alguns métodos para obter atributos automaticamente tais como a utilização de núcleos (*kernels*) não lineares [1, Capítulo 14], indução de atributos [2, 3] e *deep learning* [4] podem ser empregados.

A indução de atributos não lineares a partir dos atributos básicos é um modo de obter modelos preditivos mais precisos para problemas de classificação. Entretanto, a indução pode levar ao rápido crescimento do número de atributos, resultando em *overfitting* e em modelos com baixo poder de generalização. Para evitar esta consequência indesejada, técnicas de regularização são aplicadas, para criar um compromisso entre um reduzido conjunto de atributos representativo do domínio e a capacidade de generalização.

Neste capítulo, apresentamos a motivação desta tese e introduzimos a arquitetura do *framework* IFIS – *Incremental Feature Induction and Selection*, que implementa e integra o método de indução e seleção incrementais de atributos proposto nesta tese. Além disso, sumarizamos os resultados da aplicação do IFIS em uma das tarefas de processamento de linguagem natural selecionadas para avaliar o IFIS. Destacamos, ainda, as contribuições da tese. Por fim, descrevemos a organização deste documento.

1.1
Motivação

Modelos de aprendizado de máquina são baseados em atributos que representam os fenômenos do domínio. Para gerar modelos capazes de generalizar, são necessários atributos suficientes para representar o domínio do problema, mas não um número excessivo de atributos que cause o risco de *overfitting* [5, Capítulo 4]. O modelo que tem muitos parâmetros permite representar os

dados de treino muito fielmente, mas isso não implica necessariamente em um modelo com bom poder de generalização.

Existe, portanto, um compromisso no número de atributos do modelo, para atender a estas duas restrições.

O gráfico da figura 1.1 ilustra, de modo qualitativo, os problemas de *underfitting* e de *overfitting*. Neste gráfico, o erro de treino, que é uma medida da capacidade do modelo fazer predições sobre o próprio conjunto de treino, é representado pela curva azul. A curva vermelha indica o erro de generalização, isto é, o erro medido em um conjunto de avaliação, que não foi utilizado no treino.

O *underfitting* ocorre quando não há atributos suficientes para modelar o domínio de interesse. Modelos de baixa complexidade não representam o fenômeno de maneira suficiente, têm viés alto, que resulta do elevado erro mesmo no conjunto de dados de treino. Neste caso, o erro de generalização, também é alto, devido à limitação da representação do modelo.

Por outro lado, modelos com muitos atributos e, portanto, com alta complexidade, trazem o risco de *overfitting*. Neste caso, ocorre alta variância, ou seja, os modelos são muito sensíveis ao conjunto de dados de treino. Por causa do elevado número de graus de liberdade, o modelo termina por representar os dados de treino e não o fenômeno subjacente. Isso leva a um baixo erro de treino, mas com elevado erro de generalização, já que o modelo é muito dependente dos dados de treino e não é capaz de fazer predições acuradas para o problema geral.

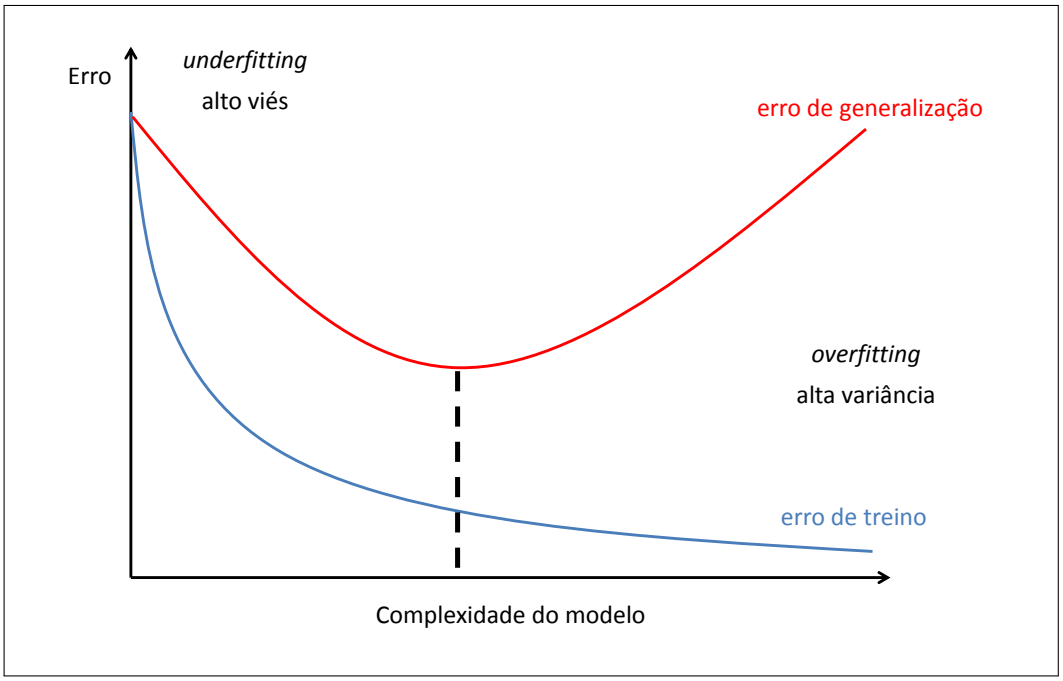

Figura 1.1: *Underfitting* × *Overfitting*.

Observe que no ponto indicado pela linha tracejada, o erro de generalização é mínimo. O objetivo do controle de *overfitting* é criar uma restrição à complexidade do modelo, buscando se aproximar deste ponto onde há atributos suficientes para representar o problema, mas não atributos demais a ponto do modelo ser uma mera representação dos dados de treino, caso em que perde poder de generalização.

1.2 O Framework IFIS

O objetivo desta tese é propor uma solução para o problema da indução automática de atributos associado ao da regularização de domínios e, consequentemente, de modelos.

Para atacar este problema, neste trabalho propomos o IFIS, um *framework* para obtenção de atributos e para controle de *overfitting*.

No IFIS, novos atributos são obtidos por indução, a partir de atributos básicos, aqueles definidos manualmente por regras do domínio. A indução consiste em combinar automaticamente dois ou mais atributos em conjunções lógicas que formam novos atributos induzidos não lineares. Por exemplo, se temos os atributos básicos *palavra*=andar e *etiquetaPOS*=verbo, podemos gerar um novo atributo *palavra_etiquetaPOS*=andar_verbo, que representa a ocorrência simultânea destes valores dos atributos.

O controle do crescimento do modelo é feito através da regularização dos domínios dos atributos. Regularizar os domínios dos atributos significa considerar somente valores que são informativos de um atributo categórico. Um exemplo de atributo categórico é a *palavra*, que pode assumir somente um valor para cada posição de um texto. Em cada posição do texto, a palavra pode assumir um valor contido no respectivo léxico, que é o conjunto de palavras distintas. Por exemplo, o léxico do corpus de notícias em língua portuguesa utilizado nesta tese tem cerca de 60.000 palavras distintas. Após a regularização, podemos reduzir o léxico a um tamanho bem menor, em alguns casos a menos de 1% das palavras originais do léxico.

Empregando a combinação de algoritmos, introduzimos o **IFIS**, um *framework* para criar modelos regularizados não lineares de alto desempenho. O IFIS integra os seguintes componentes:

- um algoritmo central de classificação para construção de preditores SVM [6], e que utiliza a implementação LIBLINEAR [7] para aprendizado;
- o EFI [8], um algoritmo para indução de atributos baseado em árvores de decisão[9], e que utiliza a implementação C5.0 [10] para aprendizado;

– um algoritmo de regularização, que utiliza o Perceptron Esparso para seleção de atributos.

A figura 1.2 ilustra a arquitetura e o funcionamento gerais do IFIS. O IFIS

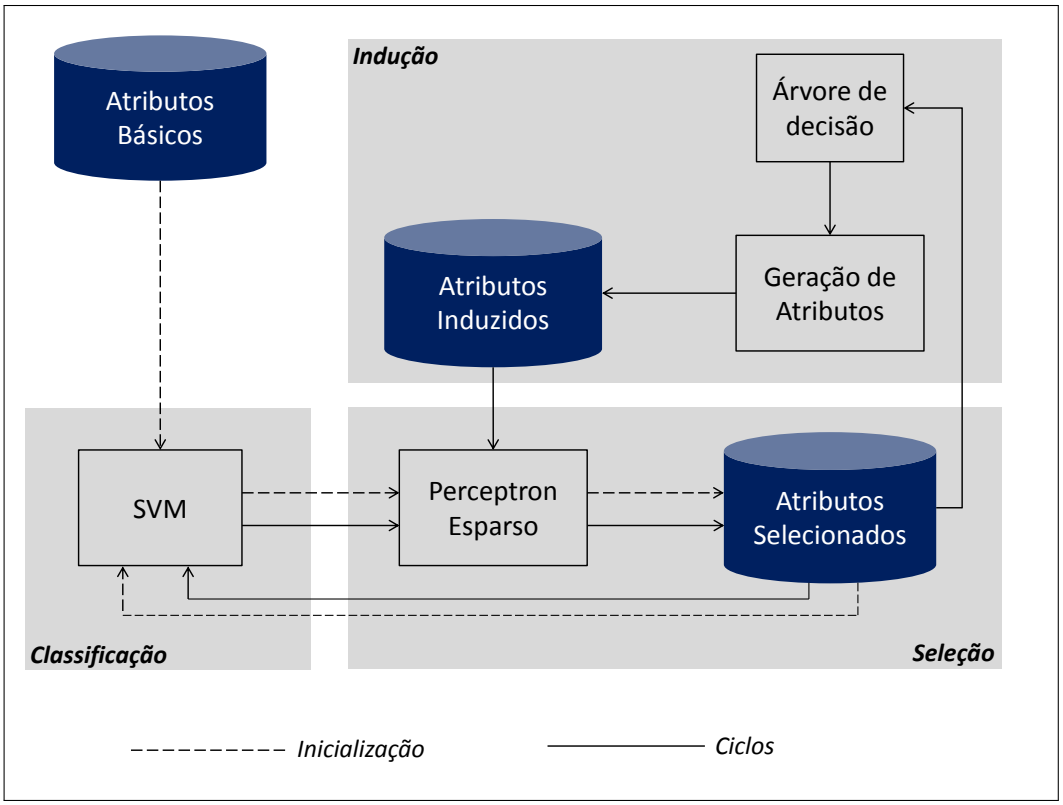

Figura 1.2: Esquema Geral do IFIS –*Incremental Feature Induction and Selection*.

opera em ciclos incrementais, de indução de atributos, sucedida pela seleção de atributos.

Definimos como ciclo 0 o passo de inicialização do IFIS. Neste passo de inicialização, o modelo é obtido somente com os atributos básicos, portanto não há indução de atributos. Este passo é indicado no diagrama da figura 1.2 pela linha tracejada.

No ciclo 1, utilizamos os atributos selecionados no ciclo 0 como entrada para a construção da árvore de decisão. A árvore de decisão nos permite obter gabaritos para a geração dos atributos não lineares induzidos, que são conjunções dos atributos básicos. A partir deste conjunto expandido de atributos induzidos, utilizamos o perceptron esparso para selecionar um subconjunto de atributos. Estes atributos selecionados são usados para obter novo modelo SVM mais compacto.

Os ciclos subsequentes repetem estes passos, sempre utilizando os novos atributos selecionados como entrada para a árvore de decisão, induzindo novos

atributos, que serão selecionados e utilizados pelo algoritmo SVM para gerar o modelo do novo ciclo.

1.3 Resultados Obtidos com o IFIS

Para avaliar o método, aplicamos o IFIS a duas tarefas de processamento de linguagem natural, a saber: anotação morfossintática e análise de dependência.

Na tabela 1.1, apresentamos os resultados da regularização de domínios dos atributos *palavra*, *lema* e o agregado dos outros atributos. Estes dados são referentes à tarefa de análise de dependência, no ciclo 0 do IFIS.

Tabela 1.1: Regularização de Domínios na Tarefa de Análise de Dependência, no Ciclo 0 do IFIS.

Atributo	**Tamanho do Domínio**		**Regularização (%)**
	Original	**Regularizado**	
palavra	13.594	134	99,0
lema	1.817	373	79,5
outros	153.857	7.735	95,0
Total	169.268	8.142	95,2

A coluna *Regularização* indica o percentual de redução do domínio, ou seja, o percentual dos atributos que foram descartados pela regularização. Podemos observar que os atributos *palavra* e *lema* têm o seu domínio reduzido significativamente.

Na tabela 1.2, apresentamos a regularização do modelo da mesma tarefa, no ciclo 1 do IFIS. A regularização do modelo é definida com a redução do número total de atributos do modelo, considerando todos os domínios de todos os atributos. A regularização, eventualmente, pode, além de reduzir o tamanho dos domínios dos atributos e dos modelos, também melhorar a acurácia do modelo, o que ocorreu neste experimento. A acurácia do modelo não regularizado é de 91,89% e sobe para 92,01% após a regularização, uma redução de 1,5% no erro de classificação desta tarefa.

As tabelas 1.3, 1.4 e 1.5 mostram o nosso resultado comparado ao respectivos estados-da-arte para cada tarefa linguística. Observe que as acurácias que obtemos estão próximas aos melhores resultados reportados na literatura. Entretanto, nossos modelos são significativamente menores que os demais, devido à regularização.

Tabela 1.2: Regularização do Modelo da Tarefa de Análise de Dependência, no Ciclo 1 do IFIS.

Original		**Regularizado**		
Núm. de Atributos	Acurácia (%)	Núm. de Atributos	Acurácia (%)	Regularização (%)
351.126	91,89	16.100	92,01	95,4

Tabela 1.3: Acurácia deste trabalho comparada ao estado-da-arte para a tarefa de anotação morfossintática do corpus Mac-Morpho.

Ano	**Sistema**	**Acurácia (%)**
2014	dos Santos & Zadrozny	97,47
2014	IFIS	97,13

Tabela 1.4: Acurácias deste trabalho comparada ao estado-da-arte para a tarefa de anotação morfossintática do corpus WSJ.

Ano	**Sistema**	**Acurácia (%)**
2012	Søgaard	97,50
2014	IFIS	97,14

Tabela 1.5: Desempenho deste trabalho comparado ao estado-da-arte para análise de dependência usando o corpus do CoNLL-2006.

Ano	**Sistema**	**UAS (%)**
2010	Dual Decomposition	93,03
2014	IFIS	92,01

1.4 Contribuições da Tese

As principais contribuições desta tese são cinco, a saber:

- o *framework* IFIS para a construção de preditores compactos, apresentando desempenhos iguais ou melhores que suas versões completas, porém utilizando um número consideravelmente menor de atributos;
- um método para compactar domínios de atributos, que permite aprimorar a etapa de engenharia de atributos;
- uma versão incremental do algoritmo EFI, que permite combinar indução e seleção de atributos;
- desempenho competitivo para as tarefas de anotação morfossintática e geração de árvores de dependência.

– uma variação do algoritmo do perceptron esparso, com a combinação deste com a técnica de *dropout*.

1.5 Organização da Tese

O restante desta tese está organizado da seguinte forma. No capítulo 2, apresentamos os principais trabalhos relacionados à obtenção automática de atributos e controle do *overfitting*. No capítulo 3, apresentamos fundamentos relacionados à indução de atributos e à regularização de domínios. No capítulo 4, descrevemos o *framework* IFIS, seus componentes. Nos capítulos 5 e 6, relatamos os experimentos e apresentamos os resultados da aplicação do IFIS às tarefas de anotação morfossintática e de análise de dependência, respectivamente. Por fim, no capítulo 7, apresentamos um resumo do trabalho, as principais contribuições, e indicamos trabalhos futuros.

2
Trabalhos Relacionados

A motivação desta tese tem duas componentes principais: a geração automática de atributos para obter modelos mais acurados e o controle de complexidade dos modelos para evitar *overfitting*.

Neste capítulo apresentamos as principais abordagens reportadas na literatura para a geração automática de atributos e para o controle de *overfitting*.

2.1
Geração Automática de Atributos

A seguir descrevemos métodos utilizados para geração automática de atributos. São revisados três grupos de métodos: os baseados em núcleo, os que usam indução de atributos e os baseados em *deep learning*.

2.1.1
Métodos baseados em núcleos

O uso de núcleos[1]tem o objetivo de aumentar a capacidade de separação dos classificadores lineares, por meio da introdução de funções de núcleo.

Em 1993, Goyon, Bose & Vapnik propuseram a aplicação de métodos de núcleo ao SVM [11]. Esta ideia é baseada no truque do núcleo (*kernel trick*), introduzido na década de 1960 no âmbito de regressão por Nadaya[12] e Watson[13]. Nesta mesma época Aizerman et. al [14] propuseram ideia semelhante para classificação de padrões.

O truque do núcleo [15] consiste em introduzir uma função no cálculo do produto interno durante o aprendizado do modelo, o que permite operar em um espaço de atributos de alta dimensão, sem ser necessário computar explicitamente os atributos neste espaço. O algoritmo de aprendizado somente calcula pontualmente os produtos internos para cada dado de exemplo. Isso poupa memória durante o treinamento e permite aprender modelos não lineares de maneira eficiente.

Alguns tipos de núcleo utilizados são o polinomial, gaussiano e sigmóide [16].

[1]em inglês, *kernels*.

Métodos baseados em núcleo são amplamente aplicados a diversos problemas de classificação para produzir modelos mais acurados. Entretanto, apresenta algumas desvantagens [17]:

i) Pesos não esparsos

O espaço induzido por funções não-lineares de núcleo é usualmente de alta dimensão. Quando os dados podem ser separados por poucos atributos não lineares, o uso de um número muito grande atributos pode levar a modelos com pesos não esparsos.

ii) Regularização inapropriada

Com algoritmos que usam regularização, como o SVM, a normalização pode ser prejudicada. Isto acontece porque a normalização requer que os valores dos atributos sejam comparáveis. Entretanto, é difícil normalizar os atributos induzidos por núcleos não lineares, porque eles usualmente não têm uma expressão explícita. Como resultado, os atributos não lineares de alta ordem podem sofrer maior penalização pela regularização, já que seus valores são pequenos e demandam pesos grandes para poder influenciar os limites das classes.

iii) Atributos implícitos

Normalmente é difícil derivar expressões explícitas para atributos induzidos pelo núcleo. Isto aumenta a dificuldade de interpretação dos atributos induzidos pelo especialista do domínio.

2.1.2 Indução de atributos

O objetivo de indução de atributos é criar automaticamente combinações não lineares dos atributos básicos para aumentar a acurácia de classificação [17]. Jin & Liu [17] propõem um algoritmo baseado em *boosting* [18] para a geração de atributos induzidos para SVM.

Outra forma de indução de atributos, baseada em entropia, é o EFG [19]. O EFG é baseado na entropia condicional dos atributos básicos e na combinação destes atributos básicos a partir da estrutura de uma árvore de decisão, para formar conjunções que produzem atributos induzido não lineares.

2.1.3 *Deep learning*

O uso de redes neurais no aprendizado de máquina tem uma vasta história de aplicações de sucesso [20].

Mais recentemente, *deep learning* tem sido empregado para o aprendizado de atributos. *Deep learning* é um conjunto de algoritmos de aprendizado

de máquina para aprender modelos multicamadas. É também chamado de aprendizado de representações.

O uso de *deep learning* para aprendizado não supervisionado de atributos tem sido bastante empregado para tarefas de processamento de linguagem natural e tem se mostrado uma abordagem muito promissora [21, 22].

O uso de *word embeddings* é um método não supervisionado para representar o vocabulário em um espaço vetorial, como nos trabalhos de Mikolov *et al.* [23, 24]. O uso desta representação aumenta a acurácia das tarefas de processamento de linguagem natural.

Técnicas de *deep learning* foram recentemente aplicadas a corpora em português, como no trabalho de de Santos & Zadrozny [25]. Neste trabalho os autores propõem o aprendizado de representação no nível de caracteres, que permite derivar atributos morfológicos das palavras automaticamente. Estes atributos melhoram a acurácia da tarefa de anotação morfossintática para inglês e português.

Outro trabalho que utiliza *deep learning* é o de Tang [26]. Neste trabalho é proposto o DLSVM, *Deep Learning Support Vector Machine*, usando um SVM no lugar da camada de função de ativação *softmax* [27].

2.2 Controle de *Overfitting*

O controle de complexidade de modelos é um problema que já é abordado pelo menos desde o século XIII, tendo sido tratado por William de Ockham, que formulou um princípio conhecido como navalha de Occam [28]. A navalha de Occam é um princípio de parcimônia das explicações. O princípio indica que entre um conjunto de hipóteses, ou modelos, a hipótese que explica o fenômeno baseada no menor número de premissas é a mais apropriada.

No âmbito do aprendizado de máquina, os métodos de controle de *overfitting* podem ser divididos em: redução de dimensionalidade, métodos de envelope e regularização.

Estas três abordagens são descritas a seguir.

2.2.1 Redução de Dimensionalidade

Técnicas de redução de dimensionalidade, que buscam obter modelos mais compactos sem perda de precisão são tratados no âmbito da estatística desde o início do século XX [29, 30]. Neste trabalho, Pearson propõe um método para ajuste de curvas a dados experimentais que permite representar os dados com menor número de parâmetros.

Métodos de fatorização de matrizes, como o SVD, *Singular Value Decomposition* [31] são amplamente utilizados para obter atributos que são combinações lineares dos atributos originais e que permitem a melhor reconstrução dos dados originais considerando o erro médio quadrático [32].

Uma maneira de reduzir o número de atributos em redes neurais, chamado *dropout*, é proposta por Hilton *et al.*[33]. O método consiste em omitir randomicamente alguns atributos para que atributos correlacionados possam ser eliminados. Neste trabalho, os autores mostram que o *dropout* permite controlar o *overfitting*, reduzindo o número de atributos redundantes.

Já no contexto de aprendizado de máquina, os métodos de redução de dimensionalidade pode ser divididos em métodos de envelope (*wrapper*) e métodos incorporados (*embedded*). Nos métodos de envelope, o algoritmo de aprendizado de máquina é tratado como uma caixa preta, e a seleção de atributos é feita externamente através de algoritmos de busca no espaço de atributos. Nos métodos incorporados, fazemos modificações na função objetivo de otimização do algoritmo de aprendizado para introduzir penalidades que ajudam a evitar o *overfitting*.

2.2.2 Métodos de envelope

Os métodos de envelope[2]foram popularizados por Kohavi & John [34, 31] A metodologia de envelope é uma maneira simples e poderosa de tratar o problema da seleção de atributos, independente do algoritmo central de aprendizado de máquina.

Na prática, o método de envelope é definido em três etapas: (i) como buscar todos os possíveis subconjuntos de atributos; (ii) como avaliar o desempenho deste subconjunto; e (iii) que classificador central utilizar. Uma busca exaustiva pode ser feita, mas somente se o número de atributos não for muito grande. Entretanto, o problema é NP-difícil [35] e a busca rapidamente se torna computacionalmente intratável. Um amplo espectro de estratégias de busca pode ser usado, como *branch-and-bound*, *simulated annealing* e algoritmos genéticos [34];

Métodos de envelope são considerados "força bruta" porque requerem processamento intensivo e podem ser inviáveis no caso de um número muito grande de atributos.

[2]em inglês, *wrapper*.

2.2.3 Regularização

Regularização é uma técnica empregada em aprendizado de máquina para evitar *overfitting* [36]. Do ponto de vista matemático, a regularização consiste em adicionar um termo de penalização para impedir que os pesos do modelo se ajustem perfeitamente aos dados de treino. Duas estratégias normalmente usadas para introduzir a penalidade na função objetivo são: A primeira, chamada L1, é utilizar a soma da norma dos pesos multiplicado por um fator. A segunda, chamada de L2, é baseada na soma dos quadrados dos pesos, multiplica por um fator [37].

2.3 Considerações Finais

Neste capítulo revisamos as principais abordagens para a geração automática de atributos e para o controle de *overfitting*. No próximo capítulo apresentamos alguns fundamentos que são utilizados na abordagem proposta no IFIS.

3
Fundamentação

Neste capítulo, apresentamos alguns fundamentos que são utilizados na abordagem proposta no IFIS. Na seção 3.1 descrevemos a binarização de atributos categóricos, na seção 3.2 apresentamos a indução de atributos não lineares em classificadores lineares e, por fim, na seção 3.3 discutimos o tratamento de atributos com dimensões elevadas são apresentados a seguir.

3.1
Atributos Categóricos e Atributos Binários

Em processamento de linguagem natural, em geral, os atributos são representados simbolicamente, ou seja, como atributos categóricos. Um exemplo de atributo categórico é *palavra*. Uma sentença em linguagem natural, como o português, é representada como uma sequência de palavras e cada palavra pode assumir um valor de um conjunto chamado de *léxico*. Por exemplo, a sentença *A casa é amarela* é representada como

$$palavra[1] = A$$

$$palavra[2] = casa$$

$$palavra[3] = é$$

$$palavra[4] = amarela$$

A representação categórica é apropriada para determinados tipos de algoritmo de classificação baseados em teoria da informação, como é o caso das árvores de decisão. Entretanto, para algoritmos baseados em modelos matemáticos precisamos representar os atributos categóricos como atributos numéricos. Então, o corpus original de linguagem natural que é representado de forma categórica, precisa ser *binarizado* [38, Seção 10.2] para o processamento pelo SVM e pelo perceptron esparso. Cada par de atributo e valor recebe um índice único de atributo binário. Após a seleção de atributos, os domínios dos atributos categóricos são compactados, de acordo com os atributos binários selecionados. Quando um valor de atributo é descartado pelo perceptron, ele

é codificado como um valor *dummy* na entrada da árvore de decisão. A tabela 3.1 ilustra a transformação dos domínios após regularização dos domínios de palavra e lema para o ciclo 0 da tarefa de análise de dependência em português.

Tabela 3.1: Exemplo de Regularização dos Domínios de Palavra e Lema para o ciclo 0 da tarefa de análise de dependência em português.

Atributo	Valor	Atributo Binário	Selecionado	Atributo	Valor
palavra	conseguem	5	sim	palavra	conseguem
palavra	sempre	716	sim	palavra	sempre
palavra	tenha	3	sim	palavra	tenha
palavra	casa	201	não	palavra	$<dummy>$
palavra	Brasil	43	não	palavra	$<dummy>$
lema	eleger	72	sim	lema	eleger
lema	merecer	137	sim	lema	merecer
lema	renovar	82	sim	lema	renovar
lema	avaliar	78	não	lema	$<dummy>$
lema	prezar	302	não	lema	$<dummy>$

Dado um atributo categórico *palavra* que tem um léxico $\mathcal{L}$ de dimensão $|\mathcal{L}| = n$, a representação binarizada deste atributo é dada por n atributos binários definidos como

$$x_i = \begin{cases} 1 & \text{se } \mathcal{L}_i = \mathcal{L}_j \\ 0 & \text{se } \mathcal{L}_i \neq \mathcal{L}_j \end{cases}$$

onde $0 < i, j <= n$.

3.2 Indução de Atributos e Classificadores Lineares

O uso de classificadores lineares quando os dados não são linearmente separáveis requer a introdução de não linearidade no modelo através de funções de núcleo, como visto na seção 2.1.1 ou através da indução de atributos não lineares, como na seção 2.1.2.

No IFIS, tratamos problemas de classificação não lineares com classificadores lineares, calculando atributos não lineares induzidos a partir de atributos básicos.

Um exemplo do poder da combinação não linear de atributos é a operação XOR, a operação *ou exclusivo* [39]. Os valores dos atributos e de classe da função XOR são mostrados na tabela 3.2. O valor de $y = -1$ corresponde a *falso* e $y = 1$, *verdadeiro*.

Tabela 3.2: Função XOR com seus atributos básicos.

x_1	x_2	$y = x_1 \oplus x_2$
0	0	-1
0	1	1
1	0	1
1	1	-1

Usando somente os atributos originais x_1 e x_2, o problema não é solúvel por um separador linear, como ilustrado na figura 3.1. Não existe reta que separe as classes, como ilustram as linhas tracejadas.

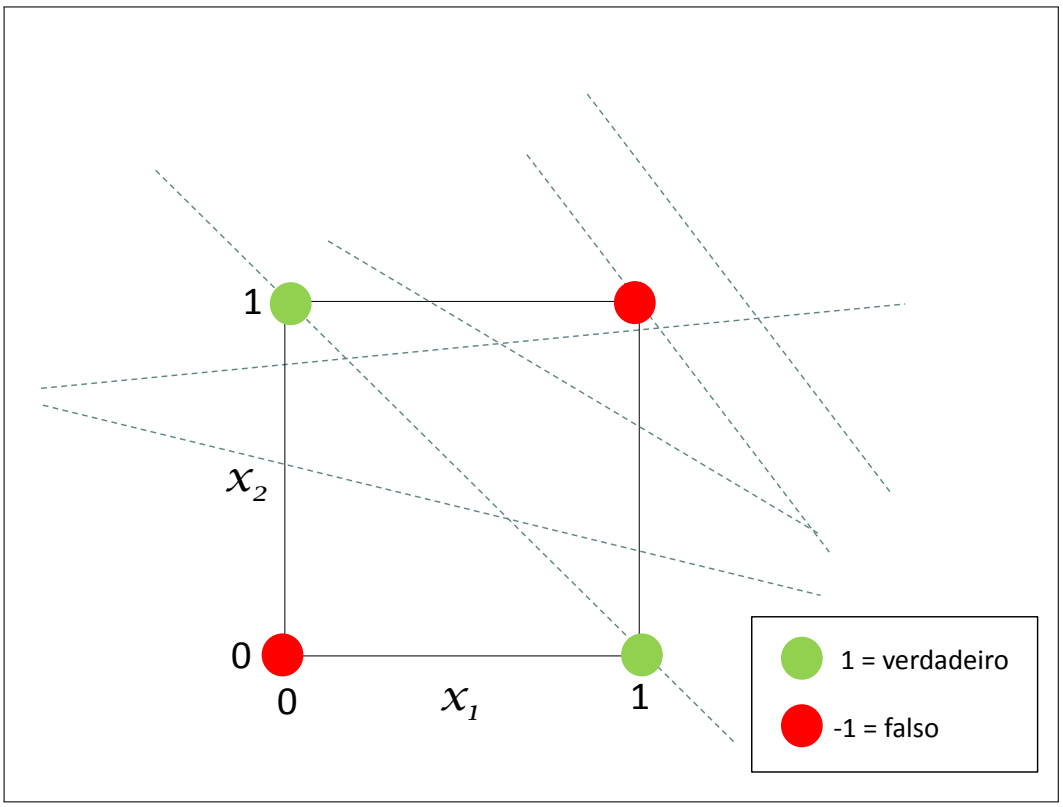

Figura 3.1: Função XOR com atributos originais. y=falso é indicado por um ponto vermelho e y=verdadeiro é indicado por um ponto verde.

Ao introduzirmos um novo atributo derivado que é a conjunção $x_3 = x_1 \wedge x_2$, mostrado na tabela 3.3, podemos separar as classes, como ilustrado na figura 3.2

O atributo x_3 está fora do plano definido por x_1 e x_2, o que torna possível ter um plano de margem máxima separando linearmente os exemplos.

Note que o plano sombreado permite separar os valores falsos, acima do plano, dos verdadeiros, abaixo do plano. Ou seja, o problema passa a ser solúvel com um separador linear de três dimensões.

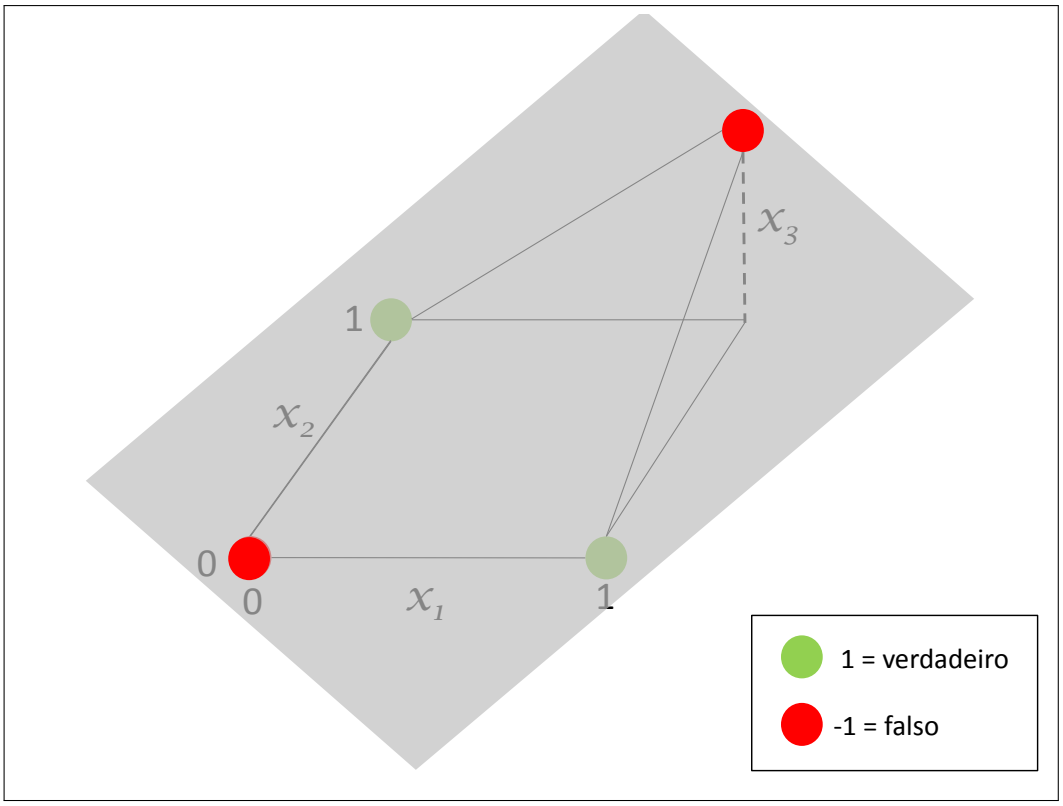

Figura 3.2: Função XOR com atributo induzido.

Tabela 3.3: Função XOR com atributo induzido.

x_1	x_2	x_3	$y = x_1 \oplus x_2$
0	0	0	-1
0	1	0	1
1	0	0	1
1	1	1	-1

3.3 Atributos com Alta Dimensão

No IFIS utilizamos um algoritmo de árvore de decisão para realizar a indução dos atributos não lineares. As árvores de decisão são sensíveis a atributos que têm cardinalidade elevada, como palavra e lema [40]. Por exemplo, o léxico do corpus WSJ tem cerca de 40.000 palavras e o do corpus Mac-Morpho cerca de 60.000 palavras. O cálculo de ganho de informação para estes atributos com alta dimensão sofre uma distorção [41, 40], sendo necessário controlar a cardinalidade destes atributos.

A solução mais comum para reduzir o tamanho do léxico é escolher as 200 palavras mais frequentes, reduzindo a cardinalidade deste atributo.

Nas representações esparsas, como o SVM, este problema é amenizado. Todavia, se o atributo palavra tem um léxico com tamanho 40.000 e desejamos

representar uma janela de contexto com tamanho 7, são criados 280.000 atributos binários, para representar cada possível palavra em cada posição da janela. Este elevado número de atributos pode causar *overfitting* no SVM.

No IFIS podemos fazer isto de dois modos. A primeira maneira é selecionar os N primeiros atributos, de acordo com a contagem do vetor U, ordenado de forma decrescente. Tipicamente este limite é da ordem de $N = 200$.

Outra forma que exploramos neste trabalho é uma forma implícita, pela escolha conveniente dos parâmetros do perceptron esparso, o número de valores selecionados para palavra, por exemplo, pode ser pequeno o suficiente para que não seja necessário aplicar o primeiro método.

3.4 Considerações Finais

Neste capítulo apresentamos três fundamentos relacionados a esta tese. A conversão de atributos categóricos em atributos binários, que no IFIS é utilizada para podermos trabalhar com algoritmos lineares como o SVM e o perceptron esparso e também com algoritmos que são baseados em categorias, como a árvore de decisão. Discutimos a indução de atributos não lineares a partir de atributos básicos, de modo a introduzir não linearidade nos modelos, mesmo usando um algoritmo central que é linear, como o SVM. Por fim, tratamos da questão de atributos com alta cardinalidade, como é o caso das palavras e lemas.

4
IFIS - Indução e Seleção Incrementais de Atributos

Neste capítulo, descrevemos a solução proposta para as questões de indução e regularização de domínios, o *framework* IFIS, descrevemos seus componentes e seu funcionamento. Em seguida, destacamos a integração entre seus componentes. Por fim, descrevemos a parametrização do IFIS.

4.1
O *Framework* IFIS

O **IFIS** é um *framework* para criar modelos regularizados não lineares de alto desempenho. A figura 4.1 ilustra a arquitetura e o funcionamento do IFIS. No primeiro ciclo, utilizamos os atributos básicos como entrada para a construção da árvore de decisão e para a gerar o modelo SVM. A árvore de decisão nos permite obter gabaritos para a geração dos atributos não lineares induzidos, que são conjunções dos atributos básicos. A partir deste conjunto expandido de atributos induzidos, utilizamos o perceptron esparso para selecionar um subconjunto de atributos. Estes atributos selecionados são usados para obter novo modelo SVM mais compacto. Os ciclos subsequentes repetem estes passos descritos, porém utilizando os novos atributos selecionados como entrada para a árvore de decisão e para o algoritmo SVM.

O IFIS opera em ciclos. O ciclo 0 funciona como inicialização do processo, não havendo indução de atributos, somente a regularização para obtenção do modelo regularizado.

O ciclo 0 é definido pelo modelo SVM gerado somente com os atributos básicos. Este modelo é regularizado através da seleção de atributos com o perceptron esparso. Os atributos selecionados servem, então, para a criação de novo modelo SVM, correspondente ao ciclo 0 regularizado.

Como exemplo, na tabela 4.1, apresentamos a regularização obtida no ciclo 0 da tarefa de anotação morfossintática.

Observe que a regularização é de 68,6%, ou seja, este percentual dos atributos originais é descartado após a regularização. Ainda assim, a acurácia do modelo sofre somente uma pequena redução de 0,02%. Os atributos selecionados no ciclo 0 são utilizados como entrada para o ciclo 1.

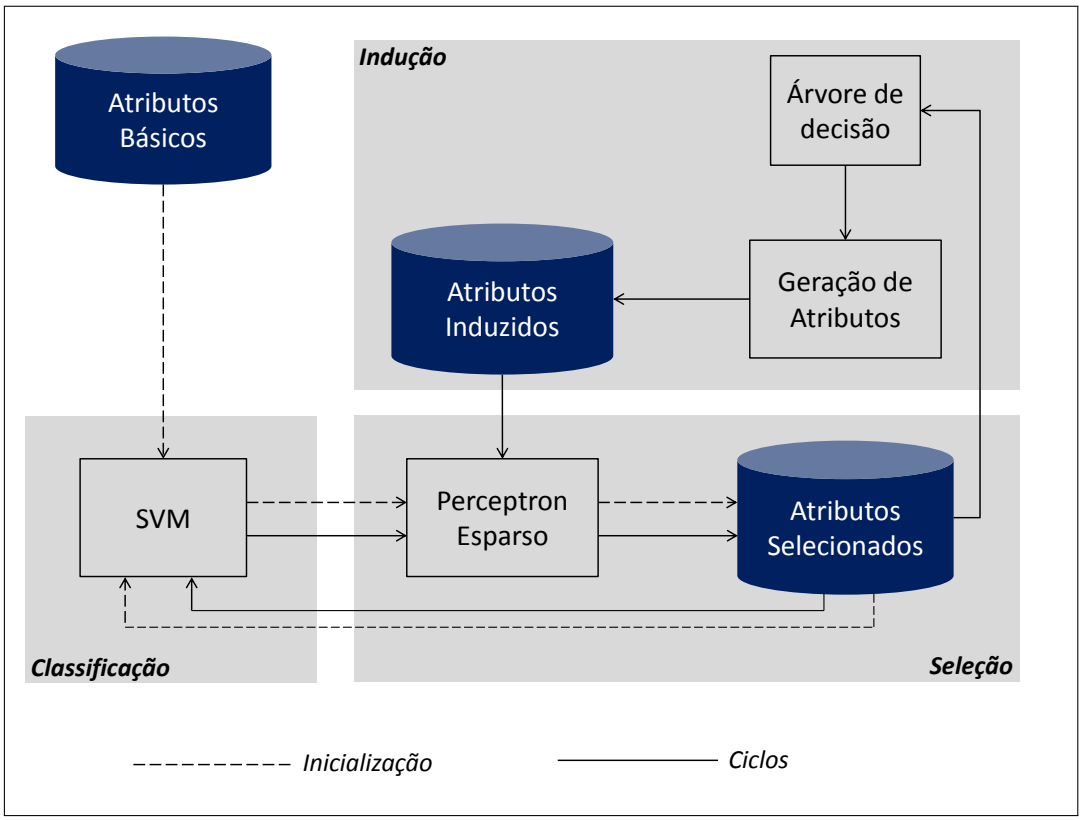

Figura 4.1: Esquema do IFIS –*Incremental Feature Induction and Selection.*

Tabela 4.1: Regularização dos Ciclos 0 e 1 na tarefa de anotação morfossintática do WSJ.

Original		**Regularizado**		
Núm. de Atributos	Acurácia (%)	Núm. de Atributos	Acurácia (%)	Regularização (%)
205.848	97,14	64.588	97,12	68,6

No algoritmo 4.1, apresentamos o pseudo-código do IFIS.

Os atributos originais de entrada são, em geral, categóricos. A entrada da árvore de decisão é categórica. A entrada do perceptron e do SVM é binária. Todas os atributos categóricos são convertidos para atributos binários.

A saída do SVM já elimina alguns atributos, que são aqueles que na matriz de pesos $\mathbf{w}$ têm todos os pesos zerados, qualquer que seja a classe.

4.2 Componentes do IFIS

A seguir descrevemos cada componente do *framework* IFIS.

Algoritmo 4.1 Algoritmo do IFIS

Entrada: $\mathcal{D}$ Dataset binário de treino
$CICLOS$ Número de ciclos

Saída: $\mathbf{w}$ Modelo compacto

$\mathbf{w} \leftarrow LIBLINEAR(\mathcal{D})$
$AtributosSelecionados \leftarrow PerceptronEsparso(\mathcal{D}, \mathbf{w})$
$\mathcal{D} \leftarrow CompactaDataset(\mathcal{D}, AtributosSelecionados)$
$\mathbf{w} \leftarrow LIBLINEAR(\mathcal{D})$
$c \leftarrow 0$
while $c < CICLOS$ **do**
  $\mathcal{D}_{categórico} \leftarrow ConverteParaCategórico(\mathcal{D})$
  $ÁrvoreDecisão \leftarrow C5.0(\mathcal{D}_{categórico})$
  $Gabaritos \leftarrow ExtraiGabaritos(ÁrvoreDecisão)$
  $\mathcal{D} \leftarrow AdicionaAtributosInduzidos(\mathcal{D}, Gabaritos)$
  $AtributosSelecionados \leftarrow PerceptronEsparso(\mathcal{D}, \mathbf{w})$
  $\mathcal{D} \leftarrow CompactaDataset(\mathcal{D}, AtributosSelecionados)$
  $\mathbf{w} \leftarrow LIBLINEAR(\mathcal{D})$
  $c \leftarrow c + 1$
end while
return $\mathbf{w}$

4.2.1 *Support Vector Machine*

A não linearidade no IFIS é introduzida pelos atributos induzidos, que são conjunções de atributos básicos, portanto não lineares em relação a eles. Portanto, no IFIS todos os modelos SVM são lineares. Assim, utilizamos a implementação **LIBLINEAR** [7] que tem desempenho muito superior a implementações anteriores como o **LIBSVM** [42].

Dois aspectos importantes do SVM são descritos a seguir. O SVM é um separador linear de *margem máxima*. Isto significa que o modelo do classificador maximiza a separação entre as classes, o que favorece a o poder de generalização do modelo gerado pelo SVM. Os dados de cada classe que estão mais próximos da margem são chamados de vetores de suporte.

Outro aspecto é a regularização, que introduz a tolerância a erros, incluindo nos vetores de suporte pontos que estão além da margem. Este classificador então é chamado de classificador de margem suave.

A figura 4.2 ilustra estes dois aspectos. A margem é mostrada como a linha tracejada e os vetores de suporte estão circulados, com a indicação correspondente. Observe o ponto verde abaixo da margem é um caso de tolerância a erro, da margem suave.

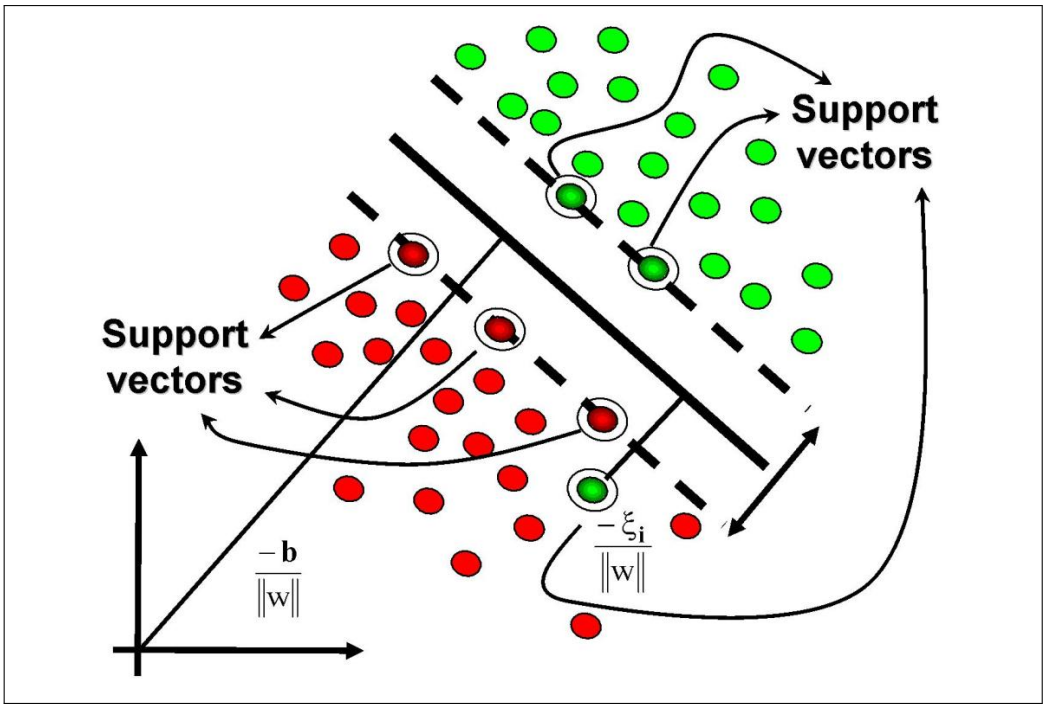

Figura 4.2: Margem e vetores de suporte do SVM.

4.2.2 EFI - *Entropy-Guided Feature Induction*

O EFI [8] é um algoritmo para indução de atributos, guiado por entropia, baseado em árvores de decisão [9] Utilizamos a implementação C5.0 [10], que é uma evolução do algoritmo C4.5 [43], com desempenho bastante superior. A figura 4.3 ilustra um segmento de árvore de decisão. Este exemplo foi extraído do problema de anotação morfossintática em português, no corpus Mac-Morpho.

Os gabaritos são extraídos da árvore desconsiderando as folhas e as etiquetas dos arcos, mantendo a estrutura da árvore e os atributos usados nos nós de decisão. A figura 4.4 ilustra a geração dos gabaritos.

A geração dos gabaritos é feita para um comprimento mínimo e um comprimento máximo dos gabaritos. Isto determina até que nível a árvore é percorrida.

4.2.3 Perceptron esparso

O algoritmo do perceptron esparso [44] proposto por Goldberg & Elhadad é descrito a seguir. Esta variação do algoritmo original do perceptron [45]

> "é baseada na intuição que somente atributos relevantes devem permanecer no modelo final e que todos os atributos são irrelevantes a princípio. Para que um atributo seja considerado relevante, ele

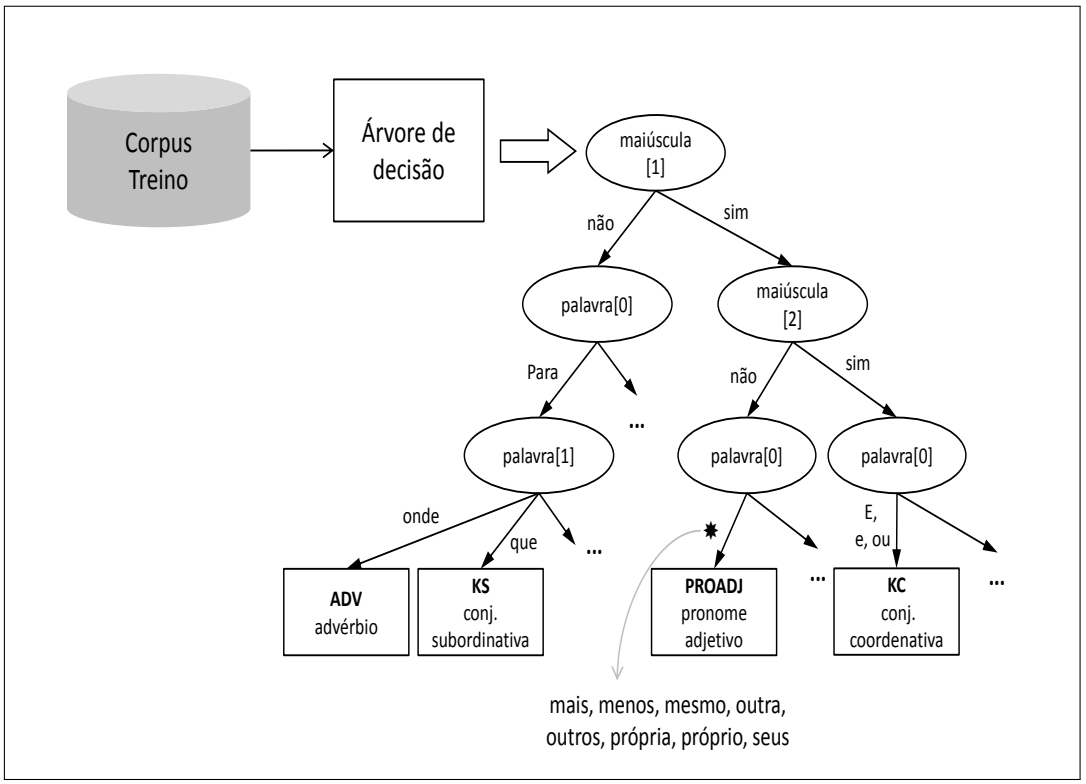

Figura 4.3: Exemplo de Árvore de Decisão Gerada no Anotador Morfossintático do Mac-Morpho.

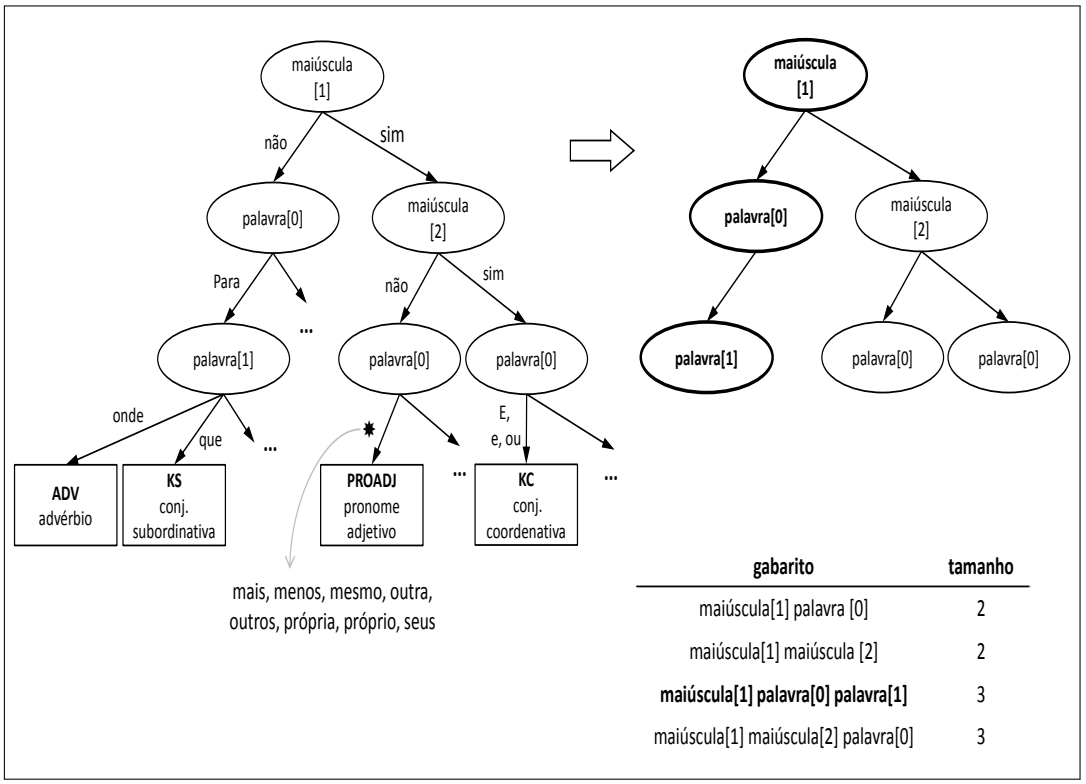

Figura 4.4: Exemplo de Gabaritos Gerados no Anotador Morfossintático do Mac-Morpho.

precisa participar em pelo menos L atualizações na matriz de pesos do perceptron."

Para desconsiderar os atributos que ainda não foram ativados, mantemos um vetor U com o número de atualizações em que cada atributo participou. No cálculo da ativação de cada classe, o vetor U funciona como uma máscara, já que o produto interno desconsidera os atributos que ainda não atingiram o limiar L. O vetor U é comum a todas as classes, isto é, é incrementado sempre que um atributo causa correção do perceptron, independentemente da classe. A atualização da matriz de pesos $\mathbf{w}$ é feita como no perceptron original, independentemente do vetor U. Isto faz com que, quando um atributo atinge o limiar, ele seja considerado com toda a história de atualização da matriz $\mathbf{w}$.

Neste trabalho usamos um perceptron multiclasse, que tem um vetor de pesos para cada classe, compondo uma matriz de pesos $\mathbf{w}$. O pseudo código do perceptron multiclasse original é descrito no algoritmo 4.2.

Algoritmo 4.2 Perceptron Multiclasse

Entrada: $\mathcal{D}$ Dataset de treino
$\acute{E}POCAS$ Número de épocas

Saída: $\mathbf{w}$ Modelo representado pela matriz de pesos

```
1: w ← 0
2: t ← 0
3: while t < ÉPOCAS do
4:    for each(x, y) ∈ D do
5:       ŷ ← arg max_{1≤k≤C}{w_k.x}
6:       if ŷ ≠ y then
7:          w_y ← w_y + x
8:          w_ŷ ← w_ŷ − x
9:       end if
10:   end for
11:   t ← t + 1
12: end while
13: return w
```

Introduzimos uma modificação, que consiste em inicializar a matriz $\mathbf{w}$ com os pesos calculados pelo SVM. Isso reduz os erros no início do treinamento do perceptron.

No algoritmo 4.3, apresentamos o pseudo-código do perceptron esparso multiclasse. No ciclo 0, o parâmetro de entrada $\mathbf{w}_0$ recebe a matriz de pesos do modelo SVM. Nos ciclos subsequentes, o parâmetro de entrada $\mathbf{w}_0$ recebe a matriz de pesos do modelo SVM compacto do ciclo anterior. O parâmetro de entrada U_0 recebe os atributos ativos no ciclo anterior.

Algoritmo 4.3 Perceptron Multiclasse Esparso

Entrada:	$\mathcal{D}$	Dataset de treino
	$\mathbf{w}_0$	Pesos iniciais
	U_0	Contagens iniciais
	$ÉPOCAS$	Número de épocas
	L	Limiar de seleção
	$Reset\,Contagens$	Flag que indica se U deve ser zerado a cada época
Saída:	$\mathbf{K} = \{k \mid u_k \geq L\}$	Atributos selecionados

1: $\mathbf{w} \leftarrow \mathbf{w}_0$
2: $\mathbf{U} \leftarrow \mathbf{U}_0$
3: $t \leftarrow 0$
4: **while** $t < ÉPOCAS$ **do**
5: **for each** $(x, y) \in \mathcal{D}$ **do**
6: $\hat{y} \leftarrow \arg\max_{1 \leq k \leq C} \{w_k[\mathbf{U} \geq L].\mathbf{x}[\mathbf{U} \geq L]\}$
7: **if** $\hat{y} \neq y$ **then**
8: $\mathbf{w}_y \leftarrow \mathbf{w}_y + \mathbf{x}$
9: $\mathbf{w}_{\hat{y}} \leftarrow \mathbf{w}_{\hat{y}} - \mathbf{x}$
10: **for each** i **s.t.** $x_{ik} \neq 0$ **do**
11: $U_i \leftarrow U_i + 1$
12: **end for**
13: **end if**
14: **end for**
15: **if** $Reset\,Contagens$ **then**
16: **for each** i **s.t.** $U_i < L$ **do**
17: $U_i \leftarrow 0$
18: **end for**
19: **end if**
20: $t \leftarrow t + 1$
21: **end while**
22: $\mathbf{K} \leftarrow \{k \mid U_k \geq L\}$
23: **return** $\mathbf{K}$

Observe, na linha 6, que para a predição somente participam os atributos que já estão ativos, ou seja, que têm contagem no vetor U maior ou igual ao limiar L.

Nossa implementação do perceptron esparso multiclasse é uma extensão do trabalho de Herszterg *et al.*. É desenvolvida em C++11, com paralelização multi-CPU, que utiliza a API OpenMP [46].

4.3 Impacto da regularização

A seguir, discutimos a sensibilidade ao grau de regularização dos modelos.

O principal efeito da regularização é a redução do número de atributos do modelo. No gráfico da figura 4.5, a abcissa corresponde ao limiar do perceptron

esparso e a ordenada é o número de atributos selecionados pelo perceptron esparso. Neste gráfico podemos observar a redução do número de atributos à medida que o limiar do perceptron aumenta.

A medida de regularização que adotamos é o percentual de atributos que são descartados pela regularização.

No gráfico da figura 4.6, a abcissa corresponde ao limiar do perceptron esparso e a ordenada é a regularização, ou seja, o percentual de atributos descartados.

A sensibilidade da acurácia medida no conjunto de treino é mostrada no gráfico da figura 4.7. A sensibilidade da acurácia medida no conjunto de desenvolvimento é mostrada no gráfico da figura 4.8.

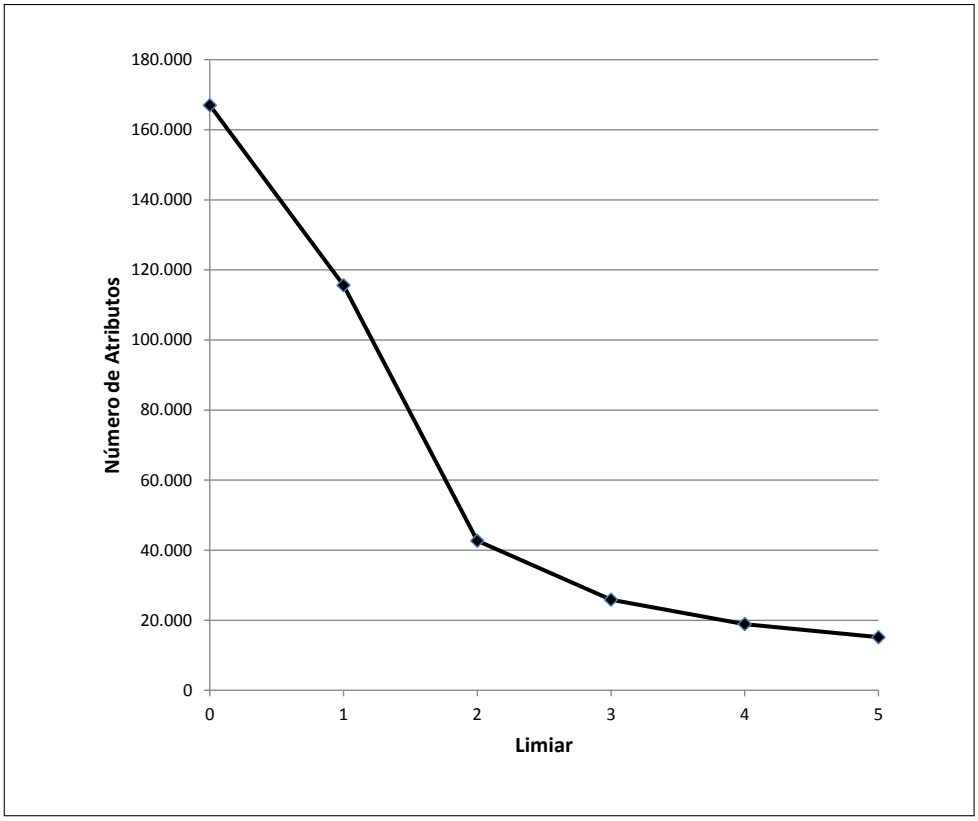

Figura 4.5: Impacto da Regularização no Número de Atributos.

4.4 Integração entre os componentes

A parte do *framework* IFIS que orquestra toda a execução e integra os diversos componentes foi desenvolvida em Python. A seguir, descrevemos e a interdependência entre os componentes.

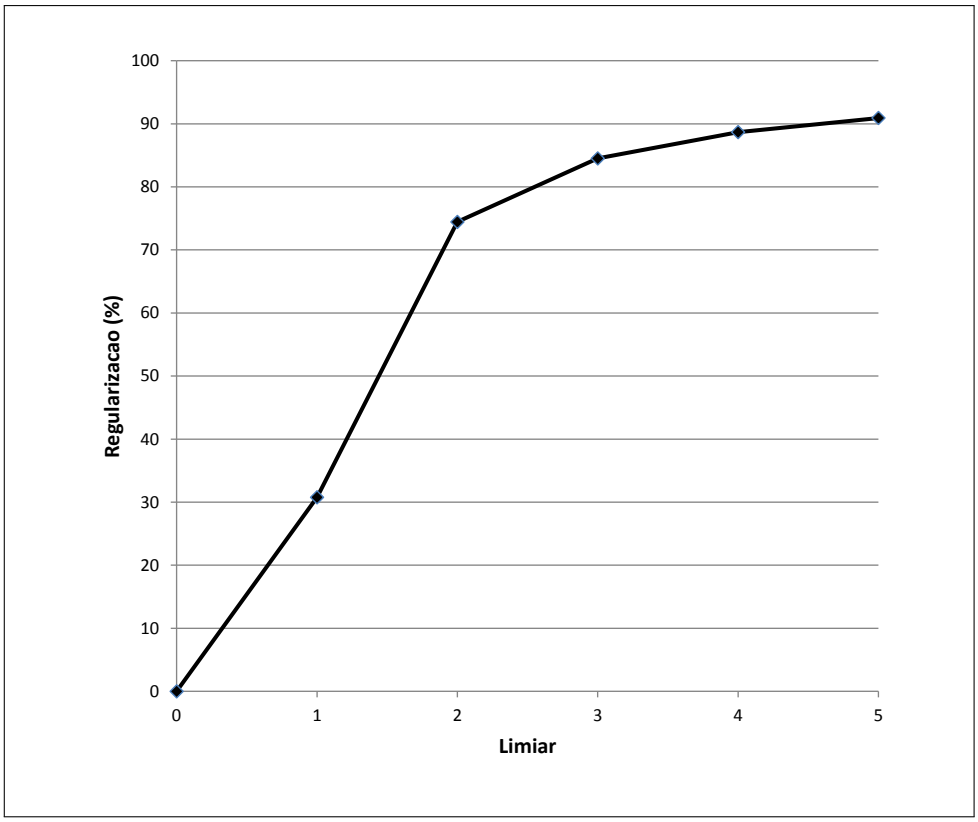

Figura 4.6: Redução Percentual do Modelo.

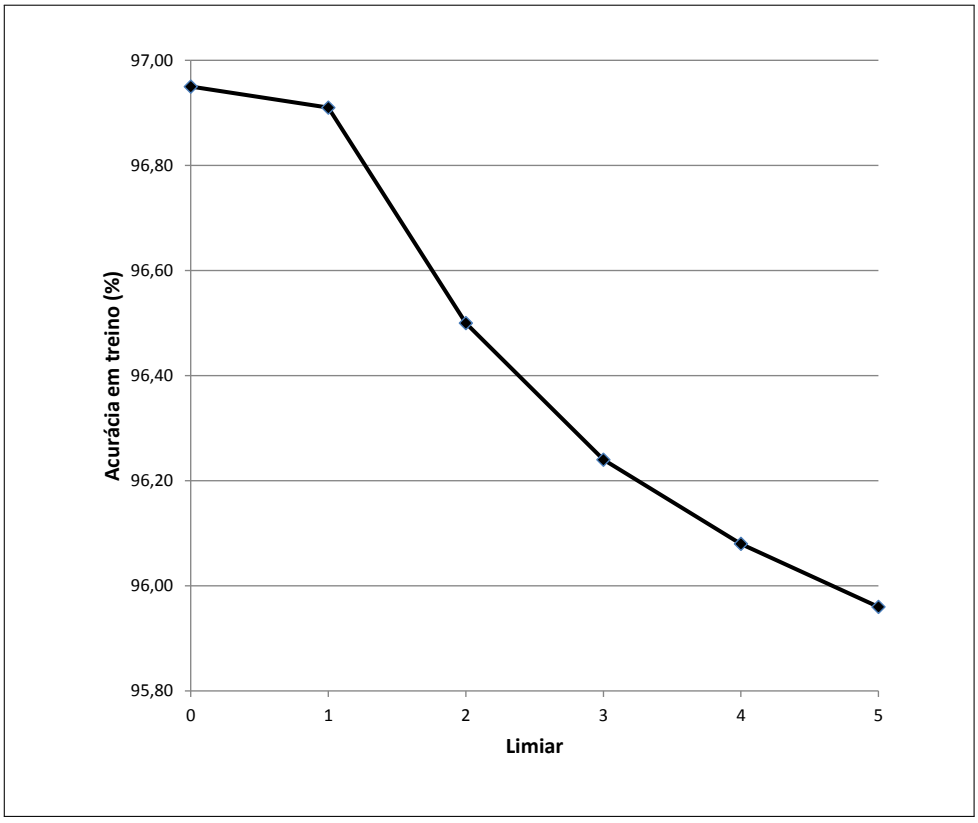

Figura 4.7: Sensibilidade da Acurácia em treino.

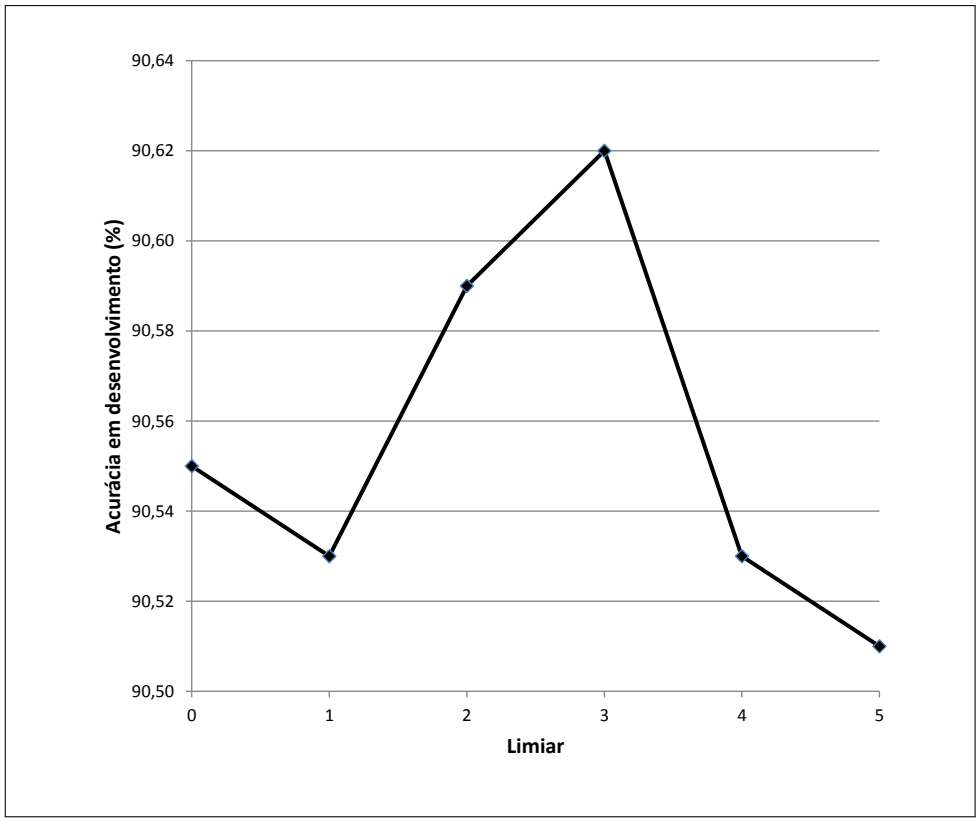

Figura 4.8: Sensibilidade da Acurácia em desenvolvimento.

4.4.1 Inicialização da matriz de pesos w

No ciclo 0, a matriz **w** do perceptron recebe a matriz de pesos do SVM gerada no ciclo 0 com todas os atributos. A partir do ciclo 1, a matriz **w** do perceptron é inicializada como os pesos do modelo SVM do ciclo anterior. Com esta modificação no perceptron, a taxa de erros no início do aprendizado é menor, selecionado, então, atributos mais informativos para a regularização.

4.4.2 Inicialização do vetor U

A cada ciclo, os atributos que foram selecionados permanecem em todo processamento posterior. Para obter este comportamento, a partir do ciclo 1 o vetor U é inicializado com um $inteiro > L$ para todos os atributos regularizados no ciclo 0, de tal forma que o perceptron já inicia o aprendizado com estes atributos ativos.

No ciclo 0, o parâmetro de entrada U_0 recebe $\mathbf{0}$.

4.5 Parametrização dos experimentos

Cada experimento processado pelo IFIS tem um perfil de configuração que especifica todos os parâmetros necessários. Estes parâmetros são descritos a seguir.

O parâmetro de *regularização C do algoritmo SVM* controla a tolerância a erros no treino, afetando a margem. Para valores grandes de C, o SVM selecionará um hiperplano de margem estreita, reduzindo o número de erros em treino. Por outro lado, valores pequenos de C geram um hiperplano com margem maior, mesmo que cause mais erros em treino. O valor padrão é $C = 1$.

O parâmetro *CF da árvore de decisão* afeta a severidade da poda da árvore de decisão. Valores menores que o padrão de 25% causam poda mais intensa, enquanto valores maiores resultam em menos poda.

O parâmetro **L** determina o limiar de seleção do perceptron esparso. Quanto mais alto o valor de **L**, menos atributos são selecionados pelo perceptron esparso.

O parâmetro *ÉPOCAS* especifica o número de vezes que o conjunto de treino é processado pelo perceptron esparso.

A extração de gabaritos da árvore de decisão é definida por um comprimento mínimo e um comprimento máximo dos caminhos percorridos na árvore. Tamanhos típicos de gabaritos variam de dois a quatro [47, 19].

4.6 Extensão do Algoritmo do Perceptron Esparso

Nesta seção, apresentamos uma modificação no algoritmo do perceptron esparso, introduzindo o *dropout*. O *dropout* tem o objetivo de promover maior regularização através da eliminação de atributos correlacionados. Isto também permite maior controle sobre o número de atributos selecionados, além de trazer a possibilidade de geração de comitês de seleção.

Introduzimos uma modificação no algoritmo do perceptron esparso, chamada *dropout*. Esta modificação é uma adaptação ao perceptron esparso da ideia de *dropout* em redes neurais, proposta no trabalho de Hinton *et. al* [33].

A modificação consiste em nem sempre fazer o incremento do vetor de atualizações U, mesmo que um atributo participe de uma correção da matriz de pesos. O incremento corresponde à linha 11 do pseudo código do perceptron esparso, descrito no algoritmo original 4.3. Em vez de sempre fazer o incremento, consideramos a possibilidade de abandonar (*dropout*) este atributo na atualização, de acordo com uma probabilidade P dada.

O pseudo código modificado é mostrado no algoritmo 4.4. Um novo parâmetro de entrada é a probabilidade $P_{dropout}$ de o atributo ser desconsiderado no incremento. Observe que a cada atualização, um novo sorteio é feito, permitindo que o mesmo atributo participe em algumas atualizações mas não em outras.

Com a introdução do *dropout*, podemos fazer uma seleção mais fina do número de atributos, através da variação da probabilidade $P_{dropout}$. No algoritmo original, como o limiar é um inteiro, não temos uma seleção do número de atributos tão precisa.

A combinação do algoritmo do perceptron estruturado com a técnica de *dropout* é uma contribuição original deste trabalho.

4.7 Considerações Finais

A combinação do SVM, um algoritmo de classificação linear de margem máxima, com o EFI, um método de indução de atributos não lineares baseado em entropia, com o perceptron esparso, que permite selecionar os atributos mais informativos, permite obter modelos compactos com atributos não lineares.

Nesta implementação do IFIS, utilizamos o SVM como algoritmo de classificação central, mas outros algoritmos podem ser utilizados neste *framework*. Por exemplo, algoritmos estruturados como o ESL – *Entropy-Guided Structure Learning* [3, 19] ou o *Structural Support Vector Machines* [48] podem substituir o SVM. Neste caso, uma implementação estruturada do perceptron esparso substitui o perceptron multiclasse.

Nos dois capítulos a seguir, descrevemos a aplicação do IFIS às tarefas de anotação morfossintática e de análise de dependência, e apresentamos os resultados obtidos.

Algoritmo 4.4 Perceptron Esparso com *Dropout*

	$\mathcal{D}$	Dataset de treino
	$\mathbf{w}_0$	Pesos iniciais
	U_0	Contagens iniciais
Entrada:	$\acute{E}POCAS$	Número de épocas
	L	Limiar de seleção
	$Reiniciar\,Contagens$	Indica se U deve ser zerado a cada época
	$P_{dropout}$	Probabilidade de *dropout*
Saída:	$\mathbf{K} = \{k \mid u_k \geq L\}$	Atributos selecionados

$\mathbf{w} \leftarrow \mathbf{w}_0$
$\mathbf{U} \leftarrow \mathbf{U}_0$
$t \leftarrow 0$
while $t < \acute{E}POCAS$ **do**
  for each $(x, y) \in \mathcal{D}$ **do**
    $\hat{y} \leftarrow \arg\max_{1 \leq k \leq C} \{w_k[\mathbf{U} \geq L].\mathbf{x}[\mathbf{U} \geq L]\}$
    if $\hat{y} \neq y$ **then**
      $\mathbf{w}_y \leftarrow \mathbf{w}_y + \mathbf{x}$
      $\mathbf{w}_{\hat{y}} \leftarrow \mathbf{w}_{\hat{y}} - \mathbf{x}$
      for each i **s.t.** $x_{ik} \neq 0$ **do**
        $r \leftarrow random[0, 1)$
        if $r \geq P_{dropout}$ **then**
          $U_i \leftarrow U_i + 1$
        end if
      end for
    end if
  end for
  if $Reiniciar\,Contagens$ **then**
    for each i **s.t.** $U_i < L$ **do**
      $U_i \leftarrow 0$
    end for
  end if
  $t \leftarrow t + 1$
end while
$\mathbf{K} \leftarrow \{k \mid U_k \geq L\}$
return $\mathbf{K}$

5
Anotação Morfossintática

Neste capítulo apresentamos os experimentos de aplicação do IFIS à tarefa de anotação morfossintática. A seguir descrevemos a tarefa, o corpus e a modelagem utilizados, os resultados experimentais alcançados, a regularização dos léxicos e, por fim, tecemos considerações sobre os experimentos de anotação morfossintática.

5.1
Introdução

A tarefa de anotação morfossintática consiste em atribuir uma etiqueta com informações sintáticas e morfológicas associadas à cada palavra no contexto de uma sentença [49]. Esta etiqueta é chamada de etiqueta POS[1]. É uma tarefa essencial em processamento de linguagem natural, cuja saída serve de entrada para várias outras tarefas fundamentais. Por isso, a acurácia desta tarefa impacta toda a cadeia de processamento linguístico.

Na tabela 5.1, mostramos um exemplo de sentença anotada com as etiquetas do corpus Mac-Morpho, acrescentando uma breve descrição da etiqueta na terceira coluna.

Na tabela 5.2, apresentamos para os dois corpora os resultados dos BLS estatísticos, que atribuem para cada palavra a correspondente etiqueta morfossintática mais frequente no corpus de treino. Observe como estes preditores simples já atingem acurácias bastante altas.

Nossos resultados para o corpus Mac-Morpho são superiores ao reportado por Fernandes [19], até recentemente o melhor resultado para este corpus. Recentemente, o estado-da-arte foi superado pelo trabalho de dos Santos & Zadrozny [50], que aplica *deep learning* para aprendizado de novos atributos, utilizando textos externos ao corpus para o aprendizado não supervisionado.

Na tabela 5.3, comparamos nossos resultados, com o estado-da-arte e o melhor resultado anterior, do trabalho de Fernandes. Neste trabalho, Fernandes utiliza um algoritmo estruturado para resolver a tarefa no nível da sentença

[1] sigla da expressão em inglês *Part-of-Speech*.

Tabela 5.1: Exemplo de anotação morfossintática do corpus Mac-Morpho.

Palavra	**Etiqueta**	**Significado**
Houve	V	Verbo
,	,	Pontuação
entretanto	KC	Conjunção coordenativa
,	,	Pontuação
reação	N	Nome
negativa	ADJ	Adjetivo
de	PREP	Preposição
os	ART	Artigo
parlamentares	N	Nome
.	.	Pontuação

Tabela 5.2: Acurácias dos BLS para os corpora Mac-Morpho e WSJ.

Corpus	**Acurácia (%)**
Mac-Morpho	90,72
WSJ	91,76

e não por *token*. No trabalho de dos Santos *et. al* [51], o algoritmo ETL é empregado também em um classificador de *tokens*.

Para o corpus WSJ, a acurácia alcançada é competitiva com os resultados reportados na literatura. Em 2002, Collins [52] propôs um método baseado em perceptron que obteve acurácia de 97,11%. O estado-da-arte reportado por Søgaard [53] é de 97,50%, em um método semisupervisionado, que utiliza dados externos ao corpus. O melhor anotador morfossintático que não usa dados externos ao corpus é o de Shen *et. al* [54], que alcança 97,33%. Dos Santos & Zadrozny propõem um anotador que utiliza textos externos para o aprendizado não supervisionado de atributos, obtendo acurácia de 97,32%. Comparando o número de atributos, obtemos uma redução expressiva em relação ao trabalho de Toutanova *et. al* [55], que usa 460 mil atributos. Neste trabalho, utilizamos somente cerca de 14% desta quantidade. Na tabela 5.4, comparamos nossos resultados com os sistemas citados.

A seguir, detalhamos a aplicação do IFIS à tarefa de anotação morfossintática, em Português, utilizando o corpus Mac-Morpho e em Inglês, utilizando o corpus de textos do *Wall Street Journal*. Na seção 5.2, descrevemos os dois corpora utilizados nos experimentos. Na seção 5.3, apresentamos a modelagem da tarefa de anotação morfossintática. Na seção 5.4, apresentamos

Tabela 5.3: Acurácias dos sistemas para o corpus Mac-Morpho.

Ano	Sistema	Acurácia (%)
2014	dos Santos & Zadrozny	97,47
2014	Este trabalho	97,13
2012	Fernandes	97,12
2008	dos Santos *et. al*	96,75
	BLS	90,72

Tabela 5.4: Acurácias dos sistemas para o corpus WSJ.

Ano	Sistema	Acurácia (%)	Acurácia nas palavras não vistas (%)
2012	Søgaard	97,50	Não disponível
2007	Shen *et al.*	97,33	Não disponível
2014	dos Santos & Zadrozny	97,32	89,86
2003	Toutanova *et al.*	97,24	89,04
2014	Este trabalho	97,14	88,90
2002	Collins	97,11	Não disponível
	BLS	91,76	16,85

os resultados experimentais. Na seção 5.5, apresentamos detalhes da regularização do léxico. Por fim, na seção 5.8, apresentamos nossas conclusões sobre a aplicação do IFIS à tarefa de anotação morfossintática.

5.2 Corpora

Utilizamos dois corpora, que são os padrões de fato para esta tarefa nos correspondentes idiomas, a saber: o Mac-Morpho para o português e o WSJ para o inglês. O Mac-Morpho [56] é um corpus formado por artigos jornalísticos do ano de 1994, retirados da Folha de São Paulo. O corpus WSJ é composto de textos do *Wall Street Journal* [57], que inclui textos jornalísticos do ano de 1989.

O conjunto de etiquetas de POS do corpus Mac-Morpho tem tamanho 41, sendo 19 sinais de pontuação. O tamanho de seu léxico é de 61.002 palavras. O número de etiquetas de POS do corpus WSJ é de 45, sendo 9 sinais de pontuação. O tamanho de seu léxico é de 43.210 palavras.

A partição do corpus Mac-Morpho utilizada para treino e teste é a mesma utilizada por dos Santos [51], conforme descrito na tabela 5.5. Para calibração

Tabela 5.5: Partição do corpus Mac-Morpho.

Dataset	Sentenças	Palavras	Palavras Não-Vistas
Treino	44.233	1.067.671	0
Desenvolvimento	4.423	102.055	3.212
Teste	9.141	213.794	8.927

de parâmetros criamos um conjunto de desenvolvimento com 10% do conjunto de treino, selecionados aleatoriamente.

A partição do corpus WSJ em treino, desenvolvimento e teste é a mesma proposta por Collins [52], sendo descrita na tabela 5.6.

Tabela 5.6: Partição do corpus WSJ.

Dataset	Seções	Sentenças	Palavras	Palavras Não-Vistas
Treino	0-18	38.219	912.344	0
Desenvolvimento	19-21	5.527	131.768	4.467
Teste	22-24	5.462	129.654	3.649

Em ambas tabelas está indicada a quantidade de palavras que não ocorrem no conjunto de treino mas parecem nos conjuntos de desenvolvimento e teste.

5.3 Modelagem

A anotação morfossintática é tratada como uma tarefa de classificação de *tokens*. Para tal, construímos um preditor para as etiquetas POS de cada palavra.

Cada *token* dá origem a um exemplo para o aprendizado supervisionado. O contexto de cada *token* é representado por atributos calculados para a vizinhança do *token*. A figura 5.1 mostra a geração de um exemplo para o *token* 'reação', como janela de tamanho 7, ou seja, a posição central, três posições à esquerda e três posições à direita. Estão exemplificados os atributos de palavra e POS.

A seguir, descrevemos a Engenharia de Atributos, destacando os atributos básicos. Além disso, apresentamos a decomposição da predição em duas etapas e o tratamento de palavras desconhecidas.

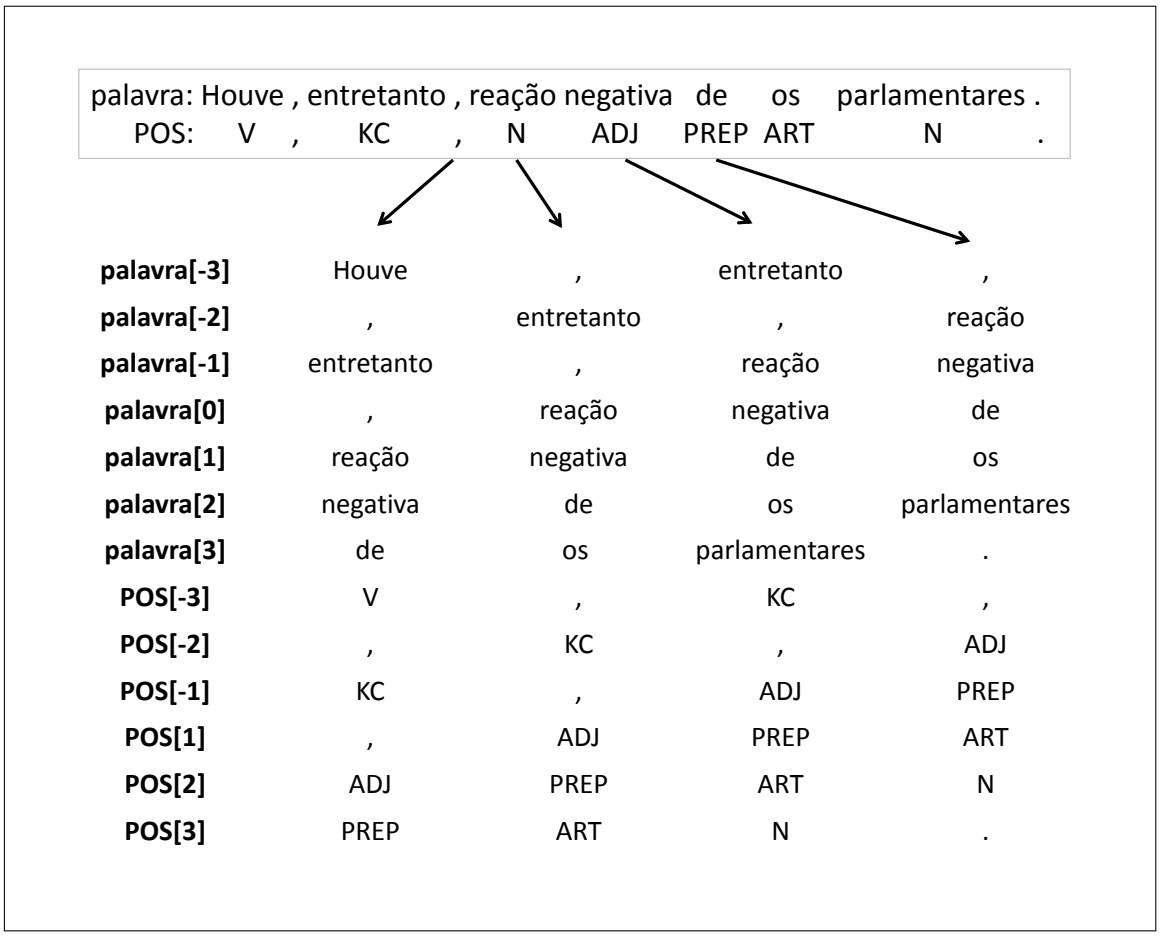

Figura 5.1: Exemplo de Geração de Atributos de Contexto.

5.3.1 Atributos Básicos

Os atributos básicos utilizados na modelagem são descritos na tabela 5.7. Estes atributos são utilizados em ambos os corpora, e são calculados *token* a *token* em uma janela de contexto de tamanho 7, contendo a palavra corrente, que é denominada de posição 0, três palavras à esquerda e três palavras à direita. O atributo *POS BLS* é calculado como a etiqueta morfossintática mais frequente para cada palavra no corpus de treino, obtida por um *baseline system* (BLS) estatístico. Usamos este atributo em todas os *tokens* da janela, menos na posição 0. Na determinação do atributo *sufixo*, o *stem* é obtido com o *stemmer* de Porter [58] para o inglês e o *stemmer* de Orengo [59] para o português.

Adicionalmente, para o corpus Mac-Morpho, utilizamos também os atributos descritos na tabela 5.8.

Tabela 5.7: Atributos usados para os corpora Mac-Morpho e *Wall Street Journal*.

Atributo	**Contexto**	**Descrição**
token	Janela toda	Palavra ou símbolo de pontuação
POS BLS	Posições -3,-2,-1,1,2,3	etiqueta POS estimada pelo BLS
sufixo	Janela toda	É a palavra com o *stem* removido do início.
sóMaiúsculas	Janela toda	Indica que a palavra contém somente letras maiúsculas
sóMinúsculas	Janela toda	Indica que a palavra contém somente letras minúsculas
éTítulo	Janela toda	Indica que a palavra tem somente a primeira letra maiúscula
temNúmero	Janela toda	Indica se a palavra contém números
temHífen	Janela toda	Indica se a palavra tem hífen
comprimento1-5	Posição 0	Indicadores binários que a palavra tem comprimento 1, 2, 3, 4 ou 5
prefixo1-5	Posição 0	Prefixos da palavra de tamanho 1 a 5
sufixo1-5	Posição 0	Sufixos da palavra de tamanho 1 a 5

5.3.2 Etapas

Os modelos são gerados em duas etapas, com o objetivo de refinar a predição de POS. O diagrama da figura 5.2 ilustra as duas etapas, que comentamos a seguir.

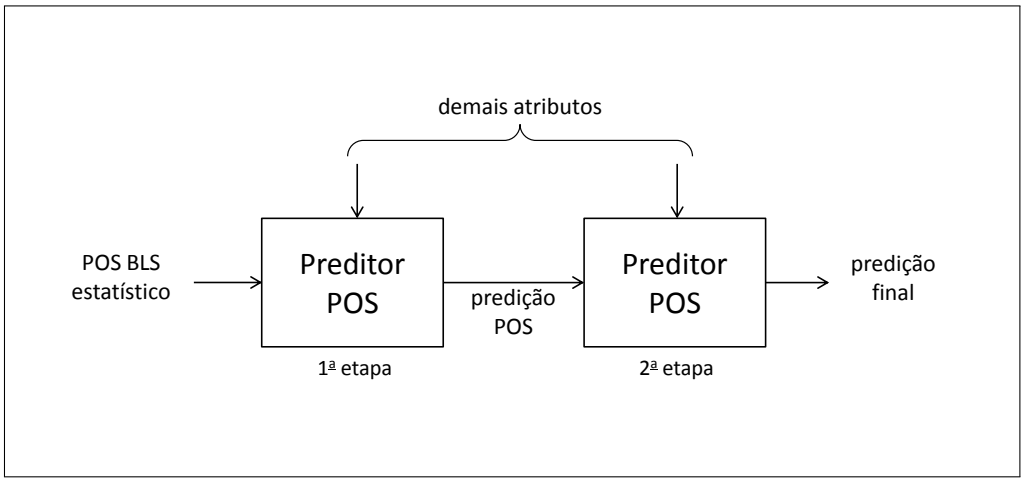

Figura 5.2: Etapas do Modelo de POS.

Tabela 5.8: Atributos usados para o corpus Mac-Morpho.

Atributo	Descrição
sucedePrep	Indica se a palavra corrente ocorre até duas posições depois das preposições do, de, da, dos ou das.

Na primeira etapa, usamos o atributo *POS BLS* como estimativa do POS no contexto da palavra, além dos outros atributos descritos. Na segunda etapa, as predições obtidas com o modelo da primeira etapa são utilizadas no lugar do atributo *POS BLS*.

5.3.3 Palavras Desconhecidas

No português, para as palavras do teste não vistas no treino, usamos a regra de predição

```
se Palavra começa com maiúscula
          POS = NPROP (nome próprio)
senão
          POS = N (nome comum)
```

No inglês, para as palavras do teste não vistas no treino, usamos a regra de predição

```
se Palavra começa com maiúscula
  se Palavra termina em 's'
        POS = nps (nome próprio no singular)
  senão
        POS = nnp (nome próprio no plural)
senão
  se Palavra termina em 's'
        POS = nns (nome comum no plural)
  senão
        POS = nn (nome comum no singular)
```

5.4 Resultados Experimentais

Nesta seção, apresentamos os resultados experimentais da aplicação do IFIS à tarefa de anotação morfossintática. A medida de qualidade dos modelos é a acurácia por palavra, ou seja, o percentual de etiquetas morfossintáticas atribuídas corretamente.

A tabela 5.9 mostra os resultados obtidos com o modelo da primeira etapa.

Tabela 5.9: Acurácias da primeira etapa de modelos para os corpora Mac-Morpho e WSJ.

Corpus	**Acurácia (%)**
Mac-Morpho	97,08
WSJ	97,06

Na tabela 5.10, mostramos os resultados da regularização dos modelos da segunda etapa, para os dois corpora.

Tabela 5.10: Regularização dos Ciclos 0 e 1 na tarefa de anotação morfossintática.

Corpus	**Ciclo**	**Limiar**	**Número de Atributos**		
			Original	**Regularizado**	**Regularização (%)**
Mac-Morpho	0	2	299.726	78.941	73,7
	1	5	665.364	120.235	81,9
WSJ	0	1	205.848	64.588	68,6
	1	50	675.125	65.488	90,3

O número de atributos reportado na tabela é o número de atributos não nulos do modelo SVM, isto é, somente aqueles atributos que têm peso diferente de zero para pelo menos uma das classes. A acurácia após a regularização foi mantida dentro de uma oscilação não superior a 0,04%, como listado na tabela 5.11.

Tabela 5.11: Regularização dos Ciclos 0 e 1 na tarefa de anotação morfossintática do WSJ.

Corpus	**Ciclo**	**Acurácia (%)**	
		Original	**Regularizada**
Mac-Morpho	0	97,07	97,04
	1	97,11	97,13
WSJ	0	97,14	97,12
	1	97,10	97,14

Os gabaritos utilizados para a indução de atributos no ciclo 1 têm tamanhos 2 e 3 para o Mac-Morpho e tamanhos de 2 a 5 para o WSJ.

Este comprimento de gabaritos bem como os limiares do perceptron foram selecionados através de validação no conjunto de desenvolvimento.

O percentual de regularização indica que é possível obter uma redução significativa do número de atributos, sem que haja perda expressiva na qualidade dos modelos. Eventualmente, a regularização aumenta a qualidade, como resultado da redução da dimensão do modelo.

A redução da dimensão do modelo é obtida por meio da diminuição da cardinalidade de cada atributo básico do modelo. Nas tabelas 5.12 e 5.13, mostramos a regularização alcançada para os atributos com cardinalidade maior que dois, no ciclo 0. Os atributos binários não sofrem regularização.

Tabela 5.12: Regularização dos atributos no ciclo 0 no corpus Mac-Morpho.

Atributo Básico	**Cardinalidade**		**Regularização (%)**
	Original	**Regularizado**	
Palavra	37.488	8.452	77,5
Sufixo	398	294	26,1
Prefixo1	106	99	6,6
Prefixo2	1.536	1.034	32,7
Prefixo3	6.636	3.302	50,2
Prefixo4	14.163	5.571	60,7
Prefixo5	20.368	6.838	66,4
Sufixo1	100	95	5,0
Sufixo2	1.363	813	40,4
Sufixo3	5.229	2.131	59,2
Sufixo4	11.430	4.272	62,6
Sufixo5	19.432	6.544	66,3

5.5 Regularizacão do Léxico

No cálculo do BLS estatístico, observamos que a maior parte das palavras apresentam somente uma etiqueta morfossintática. Estas palavras, que são 86,7% do léxico do corpus WSJ e 91,2% do corpus Mac-Morpho, não apresentam ambiguidade na determinação da etiqueta POS. Entretanto, há palavras que têm mais de uma etiqueta POS possível.

As tabelas 5.14 e 5.15 mostram a distribuição do léxico, de acordo com o número de etiquetas POS que cada palavra pode assumir. Palavras com muitas etiquetas POS possíveis são mais difíceis de classificar.

Tabela 5.13: Regularização dos atributos no ciclo 0 no corpus WSJ.

Atributo Básico	**Cardinalidade**		**Regularização (%)**
	Original	**Regularizado (%)**	
Palavra	27.815	8.145	70,5
Sufixo	193	168	13,0
Prefixo 1	78	76	2,6
Prefixo 2	1.103	749	32,1
Prefixo 3	5.155	2.744	46,8
Prefixo 4	11.353	4.909	43,2
Prefixo 5	16.631	6.344	61,9
Sufixo 1	77	77	0
Sufixo 2	993	618	62,2
Sufixo 3	3.781	1.803	52,3
Sufixo 4	8.915	3.924	56,0
Sufixo 5	14.859	5.970	40,2

Tabela 5.14: Distribuição do léxico do corpus WSJ por Quantidade de Etiquetas POS.

Número de Etiquetas	**Quantidade de palavras**
1	37.448
2	4.631
3	985
4	105
5	29
6	10
7	2
Total	43.210

Há casos menos frequentes, em que uma mesma palavra pode assumir mais de 10 etiquetas distintas. Por exemplo, no corpus em inglês do WSJ, há duas palavras que podem ter sete etiquetas distintas cada. São elas *a* e *down*.

Em português, a palavra *a*, no corpus Mac-Morpho, pode ter 16 diferentes etiquetas dependendo do contexto onde aparece.

Ainda assim, há uma concentração em poucas etiquetas para cada cada palavra, como podemos ver na tabela 5.16 para o corpus Mac-Morpho e nas tabelas 5.17 e 5.18, para o corpus WSJ.

O apêndice A contém a lista completa de etiquetas do corpus Mac-Morpho.

Tabela 5.15: Distribuição do léxico do corpus Mac-Morpho por Quantidade de Etiquetas POS.

Número de Etiquetas	**Quantidade de palavras**
1	55.638
2	4.657
3	511
4	106
5	46
6	16
7	10
8	8
9	2
10	3
11	2
12	1
14	1
16	1
Total	61.002

O apêndice B contém a lista completa de etiquetas do corpus WSJ.

A regularização de palavras é variável de acordo com a posição na janela de contexto. A posição central da janela, ou seja, a palavra corrente, tem um léxico maior preservado que a vizinhança.

Observamos que as palavras com menor quantidade de possíveis etiquetas POS sofrem regularização mais intensa do que as palavras com muitas etiquetas POS possíveis.

Na tabela 5.19 temos a regularização do léxico tabulada em duas dimensões. Nas colunas estão representadas as posições na janela de contexto. Nas linhas, o léxico está segmentado por número de etiquetas que cada palavra pode assumir. O número de etiquetas diversas está relacionada à dificuldade de classificação da palavra, pois quanto maior este número, maior é a ambiguidade na determinação da etiqueta POS. Observe, que ao longo das linhas da tabela, quanto maior a quantidade de etiquetas, menor é a regularizacão. A regularização também é variável ao longo da janela de contexto. Na posição central, da palavra corrente, as palavras com somente uma etiqueta tem regularização aproximadamente igual a de toda a janela. Por outro lado, palavras com mais de uma etiqueta sofrem regularização consideravelmente menor na posição central quando comparado com o restante da janela. Este fenômeno

Tabela 5.16: Distribuição das Etiquetas para a palavra *que* no corpus Mac-Morpho.

Etiqueta	Significado	Quantidade
PRO-KS-REL	pronome relativo conectivo subordinativo	8.472
KS	conjunção subordinativa	7.321
PRO-KS	pronome conectivo subordinativo	699
PROSUB	pronome substantivo	228
ADV-KS-REL	advérbio relativo subordinativo	77
PDEN	palavra denotativa	52
ADV	advérbio	40
PROADJ	pronome adjetivo	30
NPROP	nome próprio	26
PREP	preposição	11
ADV-KS	advérbio conectivo subordinativo	7
IN	interjeição	2
ADJ	adjetivo	2
ART	artigo	1
VAUX	verbo auxiliar	1
KC	conjunção coordenativa	1
Total		**16.970**

Tabela 5.17: Distribuição das Etiquetas para a palavra *a* no corpus WSJ.

Etiqueta	Significado	Quantidade
DT	artigo, demonstrativo ou possessivo	18.446
SYM	símbolo	10
FW	palavra estrangeira	7
LS	marcador de item de lista	2
JJ	adjectivo	2
NNP	nome próprio singular	2
IN	conjunção subordinativa ou preposição	1
Total		**18.470**

pode ser interpretado como a maior necessidade do modelo preservar palavras difíceis na posição central do que nas vizinhanças.

Um fenômeno similar é observado no corpus Mac-Morpho, em português, como exibido na tabela 5.20.

Tabela 5.18: Distribuição das Etiquetas para a palavra *down* no corpus WSJ.

Etiqueta	Significado	Quantidade
RB	advérbio	319
RP	partícula	207
IN	preposição/conjunção subordinativa	124
JJ	adjetivo	10
VBP	verbo, singular, presente, não 3ª pessoa	1
NN	substantivo, singular ou grupo	1
RBR	advérbio, comparativo	1
Total		**663**

5.6 Benefícios da Compactação dos Modelos

A regularização de domínios e dos modelos traz o benefício de controlar o *overfitting* e reduzir o número de atributos, produzindo modelos mais compactos. Esta compactação dos modelos se reflete na redução do tamanho físico dos arquivos de modelos e na diminuição dos tempos de predição dos modelos.

A tabela 5.21 mostra a redução no tamanho físico do modelo e no tempo de execução de predição, comparando o modelo original com o modelo compacto, para os corpora Mac-Morpho e WSJ.

Podemos observar reduções de 66% a 82% no tamanho dos modelos e reduções de 50% a 67% nos tempos de predição dos modelos. As medidas de tempo de predição foram tomadas numa CPU Intel i7 950@3.07Ghz.

5.7 Experimentos com *Dropout*

Para a tarefa de anotação morfossintática experimentamos com o perceptron esparso com *dropout*, no corpus MacMorpho.

Conforme descrito na seção 4.6, o *dropout* quando combinado ao perceptron estruturado permite obter maior regularização através da eliminação de atributos correlacionados.

No gráfico da figura 5.3 mostramos o efeito do *dropout* no número de atributos regularizados no ciclo 0, no corpus de desenvolvimento.

Na tabela 5.22 mostramos a média e o desvio padrão do número de atributos selecionados, e as medidas de acurácia em desenvolvimento e treino.

Para obter estes números, executamos 100 vezes a geração do modelo do ciclo 0 regularizado, com *dropout* com $P = 0.5$

Tabela 5.19: Regularização do léxico ao longo da janela de contexto por número de etiquetas POS no corpus WSJ.

Núm. de Etiquetas		**Posição na Janela**						
		-3	**-2**	**-1**	**0**	**1**	**2**	**3**
1	Orig.	14.416	14.664	13.579	22.183	13.966	14.915	15.030
	Reg.	2.703	2.705	2.387	3.954	2.857	2.585	2.822
	%	81,3	81,6	82,4	82,2	79,5	82,7	81,2
2	Orig.	3.525	3.533	3.524	4.511	3.306	3.445	3.464
	Reg.	1.397	1.390	1.374	3.137	1.215	1.337	1.333
	%	60,4	60,7	61,0	30,5	63,2	61,2	61,5
3	Orig.	888	906	895	978	836	869	867
	Reg.	539	565	563	911	506	519	526
	%	39,3	37,6	37,1	6,9	39,5	40,3	39,3
4	Orig.	102	103	102	103	103	102	102
	Reg.	82	80	88	103	75	82	84
	%	19,6	22,3	13,7	0,0	27,2	19,6	17,6
5	Orig.	27	27	28	28	27	27	27
	Reg.	24	25	27	28	23	24	24
	%	11,1	7,4	3,6	0,0	14,8	11,1	11,1
6	Orig.	10	10	10	10	10	10	10
	Reg.	9	10	10	10	9	9	9
	%	10,0	0,0	0,0	0,0	10,0	10,0	10,0
7	Orig.	2	2	2	2	2	2	2
	Reg.	2	2	2	2	2	2	2
	%	0,0	0,0	0,0	0,0	0,0	0,0	0,0
Total	Orig.	18.970	19.245	18.140	27.815	18.250	19.370	19.502
	Reg.	4.756	4.777	4.451	8.145	4.687	4.558	4.800
	%	74,9	75,2	75,5	70,7	74,3	76,5	75,4

A cada execução do perceptron esparso com *dropout* temos um conjunto de atributos diferente. Isto pode ser útil para implementar métodos de comitê [60], como, por exemplo, o *bagging* [61].

Tabela 5.20: Regularização do léxico ao longo da janela de contexto por número de etiquetas POS no corpus Mac-Morpho.

Núm. de Etiquetas		Posição na Janela						
		-3	**-2**	**-1**	**0**	**1**	**2**	**3**
1	Orig.	24.625	25.454	23.238	32.270	27.727	26.145	25.543
	Reg.	3.825	4.012	3.806	4.611	4.674	4.161	4.254
	%	84,5	84,2	83,6	85,7	83,1	84,1	83,3
2	Orig.	3.702	3.811	3.646	4.522	3.824	3.677	3.704
	Reg.	1.605	1.641	1.623	3.110	1.812	1.584	1.602
	%	56,6	56,9	55,5	31,2	52,6	56,9	56,7
3	Orig.	484	478	477	501	469	465	467
	Reg.	345	350	359	489	342	331	320
	%	28,7	26,8	24,7	2,4	27,1	28,8	31,5
4	Orig.	106	106	104	106	96	101	102
	Reg.	93	89	96	106	88	85	83
	%	12,3	16,0	7,7	0,0	8,3	15,8	18,6
5	Orig.	45	45	45	45	43	42	44
	Reg.	42	42	44	45	37	38	39
	%	6,7	6,7	2,2	0,0	14,0	9,5	11,4
6	Orig.	16	16	16	16	16	16	16
	Reg.	15	16	16	16	14	15	15
	%	6,3	0,0	0,0	0,0	12,5	6,3	6,3
7	Orig.	10	10	10	10	10	10	10
	Reg.	10	10	10	10	10	10	10
	%	0,0	0,0	0,0	0,0	0,0	0,0	0,0
8 ou mais	Orig.	18	18	18	18	18	18	18
	Reg.	18	18	18	18	18	18	18
	%	0,0	0,0	0,0	0,0	0,0	0,0	0,0
Total	Orig.	29.006	29.938	27.554	37.448	32.203	30.474	29.904
	Reg.	5.953	6.178	5.972	8.405	6.995	6.242	6.341
	%	79,5	79,4	78,3	77,6	78,3	79,5	78,8

5.8 Considerações Finais

O IFIS permite a redução da cardinalidade de atributos, regularizando seus domínios e gerando modelos regularizados mais compactos. A aplicação

Tabela 5.21: Tamanho do modelo e tempo de predição para os modelos originais e compactos na tarefa de anotação morfossintática.

Corpus	Tamanho do Modelo (Mb)		Tempo Predição (s)	
	Original	Regularizado	Original	Regularizado
Mac-Morpho	80	27	9,7	4,8
WSJ	90	16	9,1	3,0

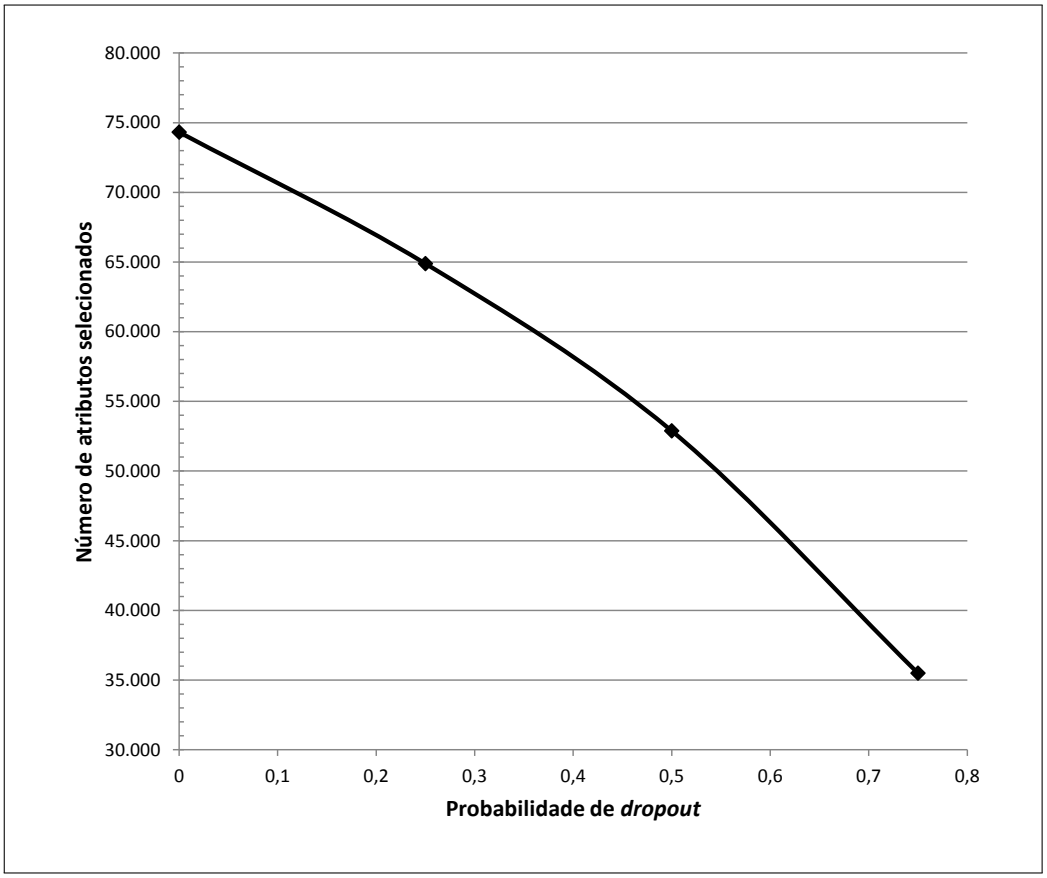

Figura 5.3: Redução do número de atributos com a variação da probabilidade de *dropout*.

Tabela 5.22: Média e Desvio Padrão de 100 Execuções do *Dropout*.

Medida	Média	σ
Número de atributos	52.810	104
Acurácia em treino	98,89%	0,0031%
Acurácia em desenvolvimento	97,16%	0,014%

do IFIS à tarefa de anotação morfossintática mostra que este método permite obter resultados competitivos com o estado da arte, utilizando um número reduzido de atributos, mesmo sem usar dados externos aos corpora.

6
Análise de Dependência

Neste capítulo apresentamos a aplicação do IFIS à tarefa de análise de dependência em língua portuguesa. A seguir descrevemos a tarefa, o corpus e a modelagem utilizados, os resultados experimentais alcançados, a regularização dos léxicos e, por fim, as considerações finais.

6.1
Introdução

A tarefa de análise de dependência[1] consiste em identificar, em uma sentença, as dependências entre as palavras. Mais especificamente, determinar que palavra governa cada palavra da sentença, baseado em uma gramática de dependências [62]. Por exemplo, na sentença "Alfredo fala...", *fala* governa *Alfredo* [63]. Podemos dizer também que *Alfredo* é dependente de *fala* e que *fala* é o pai de *Alfredo*.

Na tabela 6.1, mostramos um exemplo de sentença anotada com as etiquetas que apontam o pai, ID Pai.

Tabela 6.1: Exemplo de Sentença Anotada com as Relações de Dependência.

ID	***Token***	POS	ID Pai
1	É	adv	9
2	por	prp	9
3	isso	pron	2
4	que	adv	9
5	,	punc	6
6	diz	v	0
7	,	punc	6
8	não	adv	9
9	tem	v	6
10	pena	n	9
11	de	prp	10
12	Bill	prop	11
13	.	punc	6

[1] *dependency parsing*, em inglês.

Como cada palavra tem um e somente um pai, a tabela 6.1 representa a árvore de dependências, onde $IDPai = 0$ indica que o *token* é a raiz da árvore.

A representação gráfica da árvore de dependências desta sentença é apresentada na figura 6.1.

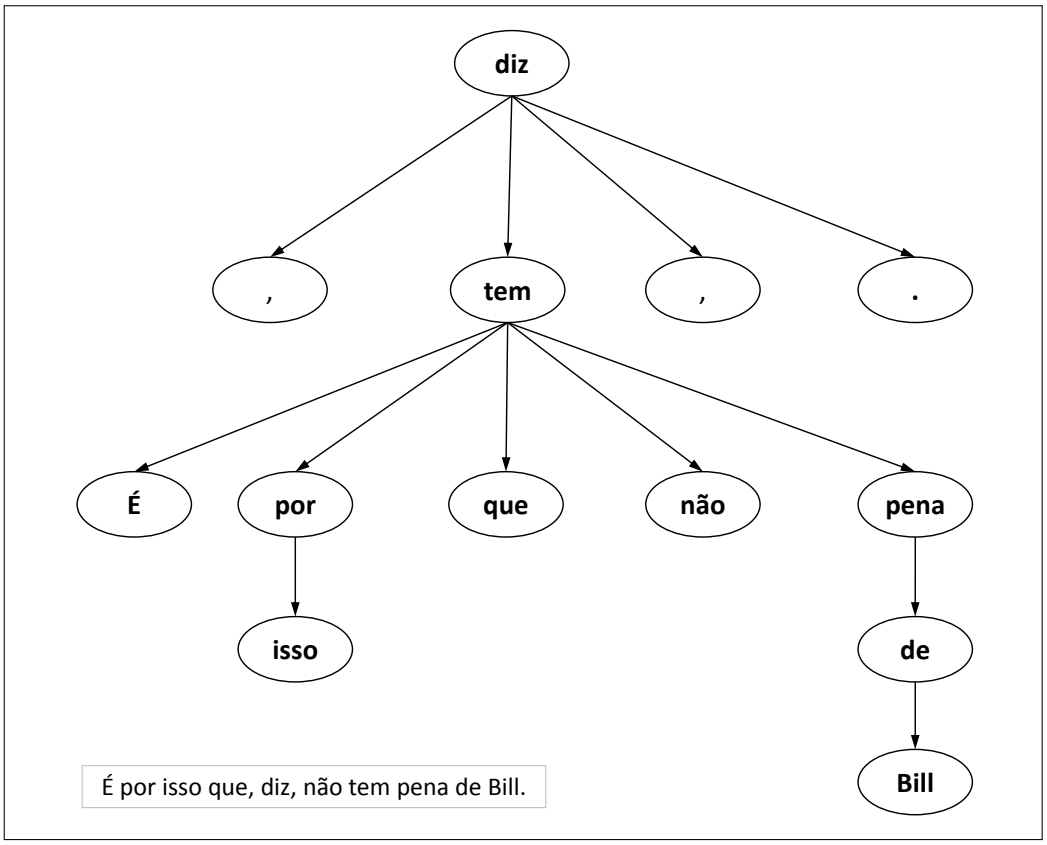

Figura 6.1: Exemplo de Árvore de Dependência.

Aplicando o IFIS a esta tarefa, obtemos o resultado de 92,01%. Este resultado é superior ao sistema *MSTParser*, melhor sistema na *Conference on Natural Language Learning* – CoNLL 2006 [64], proposto por McDonald *et al* [65], que alcançou 91,36%.

Em 2010, este resultado foi superado por Koo *et. al* [66], com o sistema *Dual Decomposition* atingindo 93,03%, que é o estado-da-arte para esta tarefa, em português.

Na tabela 6.2, comparamos nosso resultado, com o estado-da-arte, que usa um algoritmo *online* de aprendizado estruturado com atributos de terceira ordem, e também com o melhor resultado alcançado durante a CoNLL 2006, que usa uma algoritmo de árvore geradora máxima. Comparamos ainda com o sistema EFG, de Fernandes [19], que aplica um *perceptron* estruturado, e com o sistema de Crestana [67], que, como neste trabalho, adota uma estratégia de classificação de *tokens*, aplicando o algoritmo ETL.

Tabela 6.2: Desempenho de diversos sistemas de análise de dependência usando o corpus do CoNLL-2006.

Ano	Sistema	UAS (%)
2010	Dual Decomposition	93,03
2012	EFG	92,66
2014	Este trabalho	92,01
2006	MSTParser	91,36
2010	ETL	89,74

É importante destacar que nosso modelo usa somente 16.100 atributos, um número bastante reduzido, considerando a complexidade desta tarefa.

6.2 Corpus

O corpus para análise de dependência fornecido pelo CoNLL-2006 é derivado do Bosque [68], um corpus de português que inclui notícias de jornais de Portugal e do Brasil. Na tabela 6.3, mostramos a partição padrão utilizada neste corpus.

Tabela 6.3: Partições do corpus.

Dataset	Sentenças	*Tokens*
Treino	9.071	206.678
Teste	288	5.867

Para calibração de parâmetros criamos um conjunto de desenvolvimento com 10% da partição de treino, selecionados aleatoriamente.

6.3 Modelagem

A análise de dependência é tratada como uma tarefa de classificação de *tokens*. Para tal, construímos um preditor para o pai de cada palavra da sentença. A seguir, descrevemos a Engenharia de Atributos, destacando a representação relativa do atributo de classe e os atributos básicos. Além disso, apresentamos as subtarefas utilizadas para a resolução da tarefa principal.

6.3.1 Representação Relativa

A árvore de dependência é codificada no corpus original de modo absoluto, indicando o ordinal do pai, ID pai, dentro da sentença. Esta representação

é muito susceptível a pequenas modificações nas sentenças. Por exemplo, se uma palavra for adicionada à sentença, isto causará o deslocamento de todos os identificadores de pai. Como resultado, esta representação tem um baixo poder de generalização.

Para contornar este ponto, utilizamos as etiquetas relativas propostas por Crestana [67], que permitem maior generalização. A etiqueta absoluta do pai é mapeada em uma etiqueta relativa, composta de três partes: *lado*, *POS* e *distância*. O *lado* indica se o pai está à esquerda ou à direita do filho, isto é, se antecede ou sucede o filho na sentença. O lado pode assumir os valores L para esquerda, R para direita. O *POS* é a etiqueta morfossintática do pai e pode assumir qualquer um dos 12 valores de POS. A *distância* é a contagem dos *tokens* que têm a mesma etiqueta do pai, entre o filho e o pai. Nas três partes, usamos a etiqueta *root* no caso que o *token* é a raiz da sentença.

Esta representação, por ser relativa à posição do filho e considerar somente *tokens* com mesmo POS do pai, é mais robusta a alterações na sentença.

Tabela 6.4: Exemplo da Codificação da Etiquetas Relativas

ID	***Token***	**POS**	**ID Pai**	**Etiqueta Relativa**
1	É	adv	9	2_v_R
2	por	prp	9	2_v_R
3	isso	pron	2	1_prp_L
4	que	adv	9	2_v_R
5	,	punc	6	1_v_R
6	diz	v	0	root
7	,	punc	6	1_v_L
8	não	adv	9	1_v_R
9	tem	v	6	1_v_L
10	pena	n	9	1_v_L
11	de	prp	10	1_n_L
12	Bill	prop	11	1_prp_L
13	.	punc	6	2_v_L

Utilizando esta representação relativa, o número total de etiquetas relativas é 135. A distância varia de 1 até 25, com baixa frequência de distâncias maiores que 3. Para controlar o número de classes necessárias no problema de predição, adotamos uma notação compacta, atribuindo a etiqueta especial *maior que 2* a qualquer distância maior que 2. Isto reduz o número de etiquetas necessárias de 135 para 59, cobrindo ainda 97,53% dos exemplos de treino. A tabela 6.4 mostra um exemplo da codificação relativa.

6.3.2 Atributos Básicos

Os atributos básicos são de três tipos: contexto do filho, atributos da relação e contexto do pai. A seguir, detalhamos cada atributo básico utilizado na modelagem.

Contexto do filho

A tabela 6.5 mostra os atributos que dependem somente do contexto do filho.

Tabela 6.5: Atributos de Contexto do Filho.

Atributo	**Descrição**
token	Palavra
POS	Etiqueta POS da palavra
fts	Atributos morfossintáticos adicionais
Núm. Sub Esq	Número de substantivos à esquerda
Núm. Sub Dir	Número de substantivos à direita
Núm. Verbos Esq	Número de verbos à esquerda
Núm. Verbos Dir	Número de verbos à direita
Lema Verbo Esq	Lema do verbo à esquerda
Etiqueta de *chunk*	Etiqueta de *chunk*
Início de oração	Etiqueta de *clause start*
Término de oração	Etiqueta de *clause end*
Etiqueta de oração	Etiqueta de *clause*
Deprel	Natureza da relação de dependência com o pai

A segmentação de orações [69] é uma tarefa de processamento de linguagem natural, que consiste em separar a sentença em orações. A presente tarefa de análise de dependência se beneficia desta informação, utilizando as marcações de oração para determinar se filho e pai estão na mesma oração, ou a quantas orações de distância se encontram. A informação de segmentação de orações é obtida usando o sistema de Fernandes *et al* [70]. A representação da segmentação das orações adota as etiquetas descritas no apêndice C.

Chunking [71] é uma tarefa de processamento de linguagem natural, que consiste em segmentar a sentença em partes que não se superpõem, de modo que palavras que estão relacionadas sintaticamente fiquem na mesma

parte. Esta informação fornece pistas sintáticas para a tarefa de análise de dependências. Os atributos de *chunk* são obtidos com o sistema de Ferreira [72]. As etiquetas usadas para delimitar os *chunks* são descritas no apêndice C.

Estes atributos são calculados em uma janela de contexto de tamanho 7, com três palavras à esquerda, a palavra corrente e três palavras à direita.

Atributos da relação

Os atributos que dependem do filho e do pai são descritos na tabela 6.6. Estes atributos são originais deste trabalho. A intuição para os atributos de

Tabela 6.6: Atributos de Relação Entre Palavra com os Candidatos a Pai.

Atributo	**Descrição**
Núm. Inícios Oração Pai n	Número de inícios de oração até o n-ésimo candidato a pai
Núm. Términos Oração Pai n	Número de términos de oração até o n-ésimo candidato a pai
Núm. Ck Nome Pai n	Número de inícios de *chunk* nominais até o n-ésimo candidato a pai
Núm. Ck Verbo Pai n	Número de inícios de *chunk* verbais até o n-ésimo candidato a pai
Núm. Ck Prep Pai n	Número de inícios de *chunk* preposicionais até o n-ésimo candidato a pai
Núm. POS Art Pai n	Número de artigos até o n-ésimo candidato a pai
Núm. POS Conj Pai n	Número de conjunções até o n-ésimo candidato a pai
Núm. POS Sub Pai n	Número de substantivos até o n-ésimo candidato a pai
Núm. POS Nprop Pai n	Número de nomes próprios até o n-ésimo candidato a pai
Núm. POS Prep Pai n	Número de preposições até n-ésimo candidato a pai
Núm. POS Verbo Pai n	Número de verbos até n-ésimo candidato a pai
Núm. Tok Pai n	Número de *tokens* até o n-ésimo candidato a pai

relação entre a palavra com os candidatos a pai vem da estrutura sintática

da sentença e de como a segmentação de orações pode capturar informações relevantes à determinação da relação de dependência. Todos os atributos são contadores que usam diferentes métricas para representar a distância entre filho e candidato a pai. Os atributos *Núm. Inícios Oração Pai* e *Núm. Términos Oração Pai* indicam quantas orações iniciam e terminam no segmento entre filho e candidato a pai.

Os atributos *Núm. Ck Nome Pai*, *Núm. Ck Verbo Pai* e *Núm. Ck Prep Pai* indicam quantos *chunks* dos tipos nominais, verbais e preposicionais iniciam no segmento entre filho e candidato a pai.

Os atributos *Núm. POS Art Pai*, *Núm. POS Sub Pai*, *Núm. POS Conj Pai*, *Núm. POS Nprop Pai*, *Núm. POS Prep Pai* e Núm. POS Verbo Pai indicam a quantidade de *tokens* entre o filho e o candidato a pai.

O atributo *Núm. Tok Pai* é a contagem do número de *tokens* entre o filho e o candidato a pai. Estes atributos são calculados somente para a palavra corrente. O cálculo é feito para o primeiro e segundo candidatos a pai, isto é, $n = [1, 2]$. Para tal, são consideradas a primeira e segunda palavras com a etiqueta POS predita, no lado previsto.

Contexto do Pai

A tabela 6.7 mostra os atributos relativos ao contexto do pai. Todos estes atributos são originais deste trabalho.

Estes atributos são calculados somente para a palavra corrente, para os dois candidatos a pai mais próximos do filho. Os candidatos a pai são determinados com o auxílio da subtarefa *POS + lado*, descrita a seguir. A saída desta subtarefa é uma predição para o POS do pai e para o lado do pai.

Tabela 6.7: Atributos de Contexto dos Candidatos a Pai.

Atributo	**Descrição**
Lema Pai n	Lema do n-ésimo candidato a pai
POS Esq Pai n	Etiqueta POS da palavra à esquerda do n-ésimo candidato a pai
POS Dir Pai n	Etiqueta POS da palavra à direita do n-esimo candidato a pai
POS Esq Pai 1 + POS Esq Pai 2	Concatenação das etiquetas POS da palavra à esquerda dos primeiro e segundo candidatos a pai
POS Dir Pai 1 + POS Dir Pai 2	Concatenação das etiquetas POS da palavra à direita dos primeiro e segundo candidatos a pai

Estes atributos são calculados somente para a palavra corrente. O cálculo é feito para o primeiro e segundo candidatos a pai, isto é, $n = [1, 2]$. Para tal, são consideradas a primeira e segunda palavras com a etiqueta POS predita, no lado previsto.

6.3.3 Subtarefas

Para resolver a tarefa de análise de dependências utilizamos três subtarefas, que permitem obter atributos para melhorar seu desempenho. As predições destas subtarefas são usadas como entrada para o preditor da tarefa principal de análise de dependências. O diagrama da figura 6.2 ilustra as subtarefas.

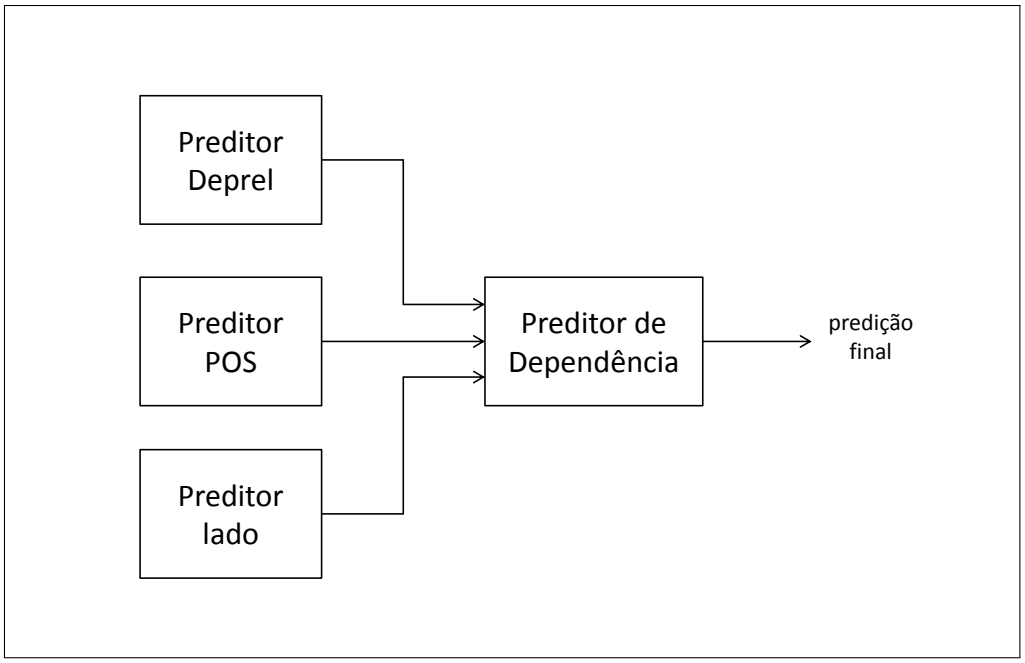

Figura 6.2: Subtarefas da Análise de Dependências.

A seguir, descrevemos estas três subtarefas, a saber: *Deprel*–Relação de dependência, *POS* e *lado*.

Deprel

Como auxiliar para a tarefa principal, desenvolvemos um classificador para *Deprel*, que representa o tipo de dependência entre a palavra e seu pai. Esta etiqueta tem origem no corpus Bosque e carrega informação sobre função gramatical. A tabela 6.8 mostra as etiquetas *Deprel* da sentença exemplo.

A predição de *Deprel* é usada como atributo de entrada para a tarefa principal. A tabela 6.9 mostra a acurácia do BLS e do preditor para *Deprel*. O sistema BLS é baseado na estatística do corpus, considerando uma janela de tamanho 3 da etiqueta POS.

Tabela 6.8: Etiquetas *Deprel*.

ID	*Token*	POS	ID Pai	Deprel
1	É	adv	9	FOC
2	por	prp	9	ADVL
3	isso	pron	2	P¡
4	que	adv	9	FOC
5	,	punc	6	PUNC
6	diz	v	0	STA
7	,	punc	6	PUNC
8	não	adv	9	ADVL
9	tem	v	6	ACC
10	pena	n	9	ACC
11	de	prp	10	N¡
12	Bill	prop	11	P¡
13	.	punc	6	PUNC

Tabela 6.9: Acurácias para a subtarefa *Deprel*.

Sistema	Acurácia (%)
Preditor	93,54
BLS	82,04

POS

Para calcular os atributos de relação entre filho e pai e os atributos de contexto do pai, precisamos de um preditor para a etiqueta *POS* do pai. Desenvolvemos um preditor de *POS* para calcular os candidatos a pai, cuja acurácia é 95,67%.

A função deste preditor é determinar a etiqueta POS do pai, como ilustrado na tabela 6.10.

Lado

Para calcular os atributos de relação entre filho e pai e os atributos de contexto do pai, precisamos ter um preditor para *lado* do pai. Desenvolvemos um preditor de *lado* para calcular os candidatos a pai, cuja acurácia é 98,90%.

A função deste preditor é determinar a etiqueta do lado do pai, como ilustrado na tabela 6.11.

6.4 Resultados Experimentais

Nesta seção, apresentamos os resultados experimentais da aplicação do IFIS à tarefa de análise de dependência. A medida de qualidade dos modelos é

Tabela 6.10: Exemplo da Codificação da Etiquetas POS

ID	*Token*	POS	ID Pai	Etiqueta POS
1	É	adv	9	v
2	por	prp	9	v
3	isso	pron	2	prp
4	que	adv	9	v
5	,	punc	6	v
6	diz	v	0	root
7	,	punc	6	v
8	não	adv	9	v
9	tem	v	6	v
10	pena	n	9	v
11	de	prp	10	n
12	Bill	prop	11	prp
13	.	punc	6	v

Tabela 6.11: Exemplo da Codificação da Etiquetas de Lado

ID	*Token*	POS	ID Pai	Etiqueta Lado
1	É	adv	9	R
2	por	prp	9	R
3	isso	pron	2	L
4	que	adv	9	R
5	,	punc	6	R
6	diz	v	0	root
7	,	punc	6	L
8	não	adv	9	R
9	tem	v	6	L
10	pena	n	9	L
11	de	prp	10	L
12	Bill	prop	11	L
13	.	punc	6	L

o UAS [64], *unlabeled attachment score* que é o percentual de pais identificados corretamente. De acordo com a definição da CoNLL 2006, são desconsiderados os sinais de pontuação no cálculo do UAS.

A tabela 6.12 mostra a evolução do aprendizado ao longo dos ciclos.

Tabela 6.12: Regularização dos Ciclo 0 e 1 na tarefa de análise de dependência.

Ciclo	Limiar	Original		Regularizado		
		Núm. de Atributos	Acurácia (%)	Núm. de Atributos	Acurácia (%)	Regularização (%)
0	5	169.268	91,36	8.142	91,28	95,2
1	20	351.126	91,89	16.100	92,01	95,4

No ciclo 1, após a indução de novos atributos, o modelo tem 351.126 atributos. Após a regularização do ciclo 1, obtemos 16.100 atributos. Neste caso, a acurácia aumenta, como resultado da regularização.

O número de atributos reportado nas tabelas é o número de atributos não nulos do modelo SVM, isto é, somente aqueles atributos que têm peso diferente de zero para pelo menos uma das classes. O percentual de regularização indica que é possível obter uma redução significativa do número de atributos, sem que haja perda expressiva na qualidade dos modelos.

Eventualmente, a regularização aumenta a qualidade, como resultado da redução da dimensão do modelo.

A redução da dimensão do modelo é obtida por meio da diminuição da cardinalidade de cada atributo básico do modelo.

A tabela 6.13 mostra a regularização dos atributos de contexto do filho. Observe que a redução do léxico é drástica, indicando que o modelo usa pouco a informação da palavra, privilegiando as informações de estrutura.

Tabela 6.13: Regularização dos atributos no ciclo 0 na tarefa de análise de dependência.

Atributo Básico	**Cardinalidade**		**Regularização (%)**
	Original	**Regularizado**	
token	13.594	134	99,0
POS	15	13	13,3
fts	459	84	81,7
Núm. Sub Esq	37	21	43,2
Núm. Sub Dir	37	18	51,4
Núm. Verbos Esq	17	13	23,5
Núm. Verbos Dir	17	12	29,4
Lema Verbo Esq	1.817	373	79,4
Etiqueta de *chunk*	8	7	12,5
Início de oração	2	2	0
Término de oração	2	2	0
Etiqueta de oração	15	7	53,3
Deprel	55	35	36,3

6.5 Regularização do Léxico

Um dos principais produtos da aplicação do IFIS é a redução automática do léxico. A redução do léxico é obtida através da regularização do domínio dos atributos de palavra e lema. A redução considerável do tamanho do léxico introduz a possibilidade de análise por linguistas das palavras e lemas relevantes para o modelo do problema. Esta análise pode trazer a compreensão de fenômenos linguísticos e sugerir novos atributos para a modelagem da tarefa.

A tabela 6.14 mostra o léxico reduzido após a regularização do ciclo 0, contando com 144 palavras.

O apêndice D mostra o conjunto de lemas dos verbos à esquerda, reduzido após a regularização do ciclo 0 a um conjunto de 383 lemas. O símbolo {inexistente} indica que não há verbo à esquerda e, portanto, não há lema correspondente.

Tabela 6.14: Léxico regularizado do ciclo 0.

! % ’ ( ) - – ; , ? $. :] ≪ ≫ A Apesar_de Até Com Como De Depois E Em Há Mas O Os Para Por Quando Segundo US$ a_partir_de ano anos antes apenas após as assim através até cerca_de chegou com como conseguem contos contra de depois desde deu deve deverá disse diz do_que durante É é e em em_relação_a entre era estava estavam este está estão faz feita filho fizeram foi for foram fosse havia há já mais mais_de mas menos mesmo mil milhões muito nada nem não o o_que onde os ou para parece pode poderá por por_parte_de porque presidente preço quando que quer se segundo seis seja sem sempre ser seria será sobre são só também tem tendo tenha ter terá tão têm um vai

6.6 Benefícios da Compactação dos Modelos

A regularização de domínios e dos modelos traz o benefício de controlar o *overfitting* e reduzir o número de atributos, produzindo modelos mais compactos. Esta compactação dos modelos se reflete na redução do tamanho físico dos arquivos de modelos e na diminuição dos tempos de predição dos modelos.

A tabela 6.15 mostra a redução no tamanho físico do modelo e no tempo de execução de predição, comparando o modelo original com o modelo compacto.

Tabela 6.15: Tamanho do modelo e tempo de predição para os modelos originais e compactos na tarefa de análise de dependência.

Tamanho do Modelo (Mb)		**Tempo Predição (s)**	
Original	**Regularizado**	**Original**	**Regularizado**
60	12	3,2	0,5

Podemos observar reduções de 80% no tamanho do modelo e redução de 84% no tempo de predição dos modelos. As medidas de tempo de predição foram tomadas numa CPU Intel i7 950@3.07Ghz.

6.7 Considerações Finais

O IFIS permite a redução da cardinalidade de atributos, em alguns casos, a menos de 1% do número original. Isto permite gerar modelos regularizados bastante compactos. A aplicação do IFIS à análise de dependência mostra que este método obtém bons resultados, com UAS de 92,01%, cerca de 1% inferior ao estado-da-arte, que é de 93,03%. Esta qualidade é alcançada com um modelo bastante compacto, utilizando somente 16.100 atributos. A principal vantagem de se ter um modelo compacto é ter requisitos limitados de memória e processamento em tempo de predição.

7
Conclusões

Neste trabalho, investigamos o problema de regularização de domínios, combinado com a indução de atributos, com o objetivo de controlar o fenômeno de *overfitting*. A regularização permite selecionar atributos mais informativos para a criação de modelos com maior poder de generalização.

Para fazer a regularização de domínios, introduzimos o IFIS, um *framework* que permite criar classificadores com modelos mais compactos, com desempenho similar ou melhor que os modelos originais.

A seguir, na seção 7.1, apresentamos o resumo dos resultados alcançados pelo IFIS nas tarefas linguísticas de análise de dependência e anotação morfossintática. Na seção 7.2, indicamos as principais contribuições deste trabalho. Na seção 7.3, elencamos algumas propostas de extensão do trabalho.

7.1
Resumo dos Resultados

A aplicação do IFIS a duas tarefas de processamento de linguagem natural indica que o método é bastante promissor, alcançando resultados competitivos com o estado-da-arte, com um número reduzido de atributos. Modelos compactos, além de generalizarem mais, têm a vantagem de executarem mais rapidamente em tempo de predição. Isto é especialmente importante para ambientes de produção, onde o tempo de resposta é crucial para respostas online e considerando que em geral, na cadeia de processamento de linguagem natural, são executadas várias etapas para resolver progressivamente uma tarefa mais complexa. A anotação morfossintática está entre as tarefas mais básicas, portanto, várias outras etapas dependem de sua execução. No caso da análise de dependências, a estrutura de saída pode ser usada em aplicações como análises semânticas.

7.2
Contribuições

As principais contribuições desta tese são cinco, a saber:

– o *framework* IFIS para a construção de preditores compactos, apresentando desempenhos iguais ou melhores que suas versões completas, porém utilizando um número consideravelmente menor de atributos;

– um método para compactar domínios de atributos, permitindo aprimorar a etapa de engenharia de atributos;

– uma versão incremental do algoritmo EFI, permitindo combinar indução e seleção de atributos;

– resultados competitivos para as tarefas de anotação morfossintática e análise de dependência;

– uma variação do algoritmo do perceptron esparso, com a combinação deste com a técnica de *dropout*.

7.3 Trabalhos Futuros

A seguir elencamos algumas extensões propostas ao *framework* IFIS.

A implementação apresentada neste trabalho usa como classificador central o algoritmo SVM, um separador linear de margem máxima. Entretanto, a ideia da regularização de domínios pode ser aplicada em conjunto com outros algoritmos como, por exemplo, perceptron estruturado ou SSVM.

Aplicar a ideia do *dropout* na indução de atributos, selecionando aleatoriamente atributos dos gabaritos para serem eliminados.

Implementar a seleção de atributos com comitê de perceptrons esparsos com *dropout*.

Explorar a regularização de domínios como ferramenta para facilitar da engenharia de atributos. Como o IFIS permite reduzir drasticamente o léxico em alguns casos, este léxico reduzido pode ser analisado por um linguista para trazer novas intuições sobre a tarefa linguística em estudo.

O IFIS pode ser aplicado a problemas de outros domínios, como o mercado financeiro. Uma aplicação adicional do IFIS que já está em andamento é o trabalho de Borges Filho [73]. O trabalho é um preditor do comportamento do mercado financeiro utilizando notícias em português, relacionadas à Petrobras. A modelagem utilizada inclui vários atributos linguísticos, como palavra, POS, *chunk* e segmentação de orações.

A utilização do IFIS permitiu uma regularização de cerca de 55%, resultando em um aumento na acurácia de classificação de 66,60% para 68,57%.

O IFIS pode utilizar outro algoritmo central em lugar do SVM. O trabalho de Barroso [74] utiliza um perceptron esparso com estrutura latente como classificador central para resolver as tarefas de análise de dependência

e de resolução de correferências, que consiste na identificação e agrupamento das menções textuais que se referem à mesma entidade ou evento do mundo real.

Glossário

C5.0	Implementação do algoritmo de árvores de decisão
CoNLL	*Conference on Natural Language Learning*, Conferência sobre Aprendizado de Linguagem Natural
EFI	*Entropy-Guided Feature Induction*, Indução de Atributos Guiada por Entropia
ESL	*Entropy-Guided Structure Learning*, Aprendizado Estruturado Guiado por Entropia
ETL	*Entropy-Guided Transformation Learning*, Aprendizado Baseado em Transformação Guiado por Entropia
IFIS	*Incremental Feature Induction and Selection*, Indução e Seleção Incrementais de Atributos
L1	Regularização baseada na soma da norma dos pesos do modelo
L2	Regularização baseada na soma dos quadrados dos pesos do modelo
LIBLINEAR	Implementação do algoritmo SVM
POS	*Part-of-Speech*, Etiqueta morfossintática
SVD	*Singular Value Decomposition*, Decomposição em Valores Únicos
SVM	*Support Vector Machines*, Máquinas de Vetores de Suporte

Referências Bibliográficas

[1] MURPHY, K. P. **Machine Learning: A Probabilistic Perspective (Adaptive Computation and Machine Learning series)**. [S.l.]: The MIT Press, 2012. Hardcover. ISBN 0262018020.

[2] SANTOS, C. N. dos; MILIDIÚ, R. L. **Entropy Guided Transformation Learning - Algorithms and Applications.** [S.l.]: Springer, 2012. I-XIII, 1-78 p. (Springer Briefs in Computer Science). ISBN 978-1-4471-2977-6.

[3] FERNANDES, E. R.; SANTOS, C. N. D.; MILIDIÚ, R. L. Latent structure perceptron with feature induction for unrestricted coreference resolution. In: ASSOCIATION FOR COMPUTATIONAL LINGUISTICS. **Joint Conference on EMNLP and CoNLL-Shared Task**. [S.l.], 2012. p. 41–48.

[4] SCHMIDHUBER, J. Deep learning in neural networks: An overview. **CoRR**, abs/1404.7828, 2014.

[5] ABU-MOSTAFA, Y. S.; MAGDON-ISMAIL, M.; LIN, H.-T. **Learning From Data**. [S.l.]: AMLBook, 2012. ISBN 1600490069, 9781600490064.

[6] VAPNIK, V. N. **The Nature of Statistical Learning Theory**. New York, NY, USA: Springer-Verlag New York, Inc., 1995. ISBN 0-387-94559-8.

[7] FAN, R.-E. et al. Liblinear: A library for large linear classification. **J. Mach. Learn. Res.**, JMLR.org, v. 9, p. 1871–1874, jun. 2008. ISSN 1532-4435. Disponível em: <http://dl.acm.org/citation.cfm?id=1390681.1442794>.

[8] SANTOS, C. N. dos; MILIDIÚ, R. L. **Entropy Guided Transformation Learning - Algorithms and Applications.** [S.l.]: Springer, 2012. I-XIII, 1-78 p. (Springer Briefs in Computer Science). ISBN 978-1-4471-2977-6.

[9] QUINLAN, J. R. Simplifying decision trees. **Int. J. Man-Mach. Stud.**, Academic Press Ltd., London, UK, UK, v. 27, n. 3, p. 221–234, set. 1987. ISSN 0020-7373. Disponível em: <http://dx.doi.org/10.1016/S0020-7373(87)80053-6>.

[10] MARCUS, M. et al. **Information on See5/C5.0 - RuleQuest Research Data Mining Tools**. 2013. Disponível em: <http://www.rulequest.com/see5-info.html>.

[11] GUYON, I.; BOSER, B.; VAPNIK, V. Automatic capacity tuning of very large VC-dimension classifiers. In: HANSON, S. J.; COWAN, J. D.; GILES, C. L. (Ed.). **Advances in Neural Information Processing Systems**. [S.l.]: Morgan Kaufmann, San Mateo, CA, 1993. v. 5, p. 147–155.

[12] NADARAYA, E. A. On estimating regression. **Theory of Probability and its Applications**, v. 9, p. 141–142, 1964.

[13] WATSON, G. S. Smooth regression analysis. **Sankhyā Ser.**, v. 26, p. 359–372, 1964.

[14] AIZERMAN, M. A.; BRAVERMAN, E. A.; ROZONOER, L. Theoretical foundations of the potential function method in pattern recognition learning. In: **Automation and Remote Control,**. [S.l.: s.n.], 1964. (Automation and Remote Control,, 25), p. 821–837.

[15] SHAWE-TAYLOR, J.; CRISTIANINI, N. **Kernel Methods for Pattern Analysis**. New York, NY, USA: Cambridge University Press, 2004. ISBN 0521813972.

[16] HOFMANN, T.; SCHöLKOPF, B.; SMOLA, A. J. **KERNEL METHODS IN MACHINE LEARNING**. 2008.

[17] JIN, R.; LIU, H. Robust feature induction for support vector machines. In: **Proceedings of the Twenty-first International Conference on Machine Learning**. New York, NY, USA: ACM, 2004. (ICML '04), p. 57–. ISBN 1-58113-838-5. Disponível em: <http://doi.acm.org/10.1145/1015330.1015370>.

[18] SCHAPIRE, R. E. **The Boosting Approach to Machine Learning: An Overview**. 2001. In MSRI Workshop on Nonlinear Estimation and Classification, Berkeley, CA, USA.

[19] FERNANDES, E. L. R. **Entropy Guided Feature Generation for Structure Learning**. Tese (Doutorado) — Pontifícia Universidade Católica do Rio de Janeiro, 2012.

[20] SCHMIDHUBER, J. **Deep Learning in Neural Networks: An Overview**. [S.l.], 2014.

[21] COLLOBERT, R. et al. Natural language processing (almost) from scratch. **J. Mach. Learn. Res.**, JMLR.org, v. 12, p. 2493–2537, nov. 2011. ISSN 1532-4435. Disponível em: <http://dl.acm.org/citation.cfm?id=1953048.2078186>.

[22] COLLOBERT, R.; WESTON, J. A unified architecture for natural language processing: Deep neural networks with multitask learning. In: **Proceedings of the 25th International Conference on Machine Learning**. New York, NY, USA: ACM, 2008. (ICML '08), p. 160–167. ISBN 978-1-60558-205-4. Disponível em: <http://doi.acm.org/10.1145/1390156.1390177>.

[23] MIKOLOV, T. et al. Efficient estimation of word representations in vector space. **CoRR**, abs/1301.3781, 2013. Disponível em: <http://arxiv.org/abs/1301.3781>.

[24] MIKOLOV, T. et al. Distributed Representations of Words and Phrases and their Compositionality. In: BURGES, C. J. C. et al. (Ed.). **Advances in Neural Information Processing Systems 26**. Curran Associates, Inc., 2013. p. 3111–3119. Disponível em: <http://papers.nips.cc/paper/5021-distributed-representations-of-words-and-phrases-and-their-compositionality.pdf>.

[25] In: . [S.l.: s.n.].

[26] TANG, Y. Deep learning using support vector machines. **CoRR**, abs/1306.0239, 2013. Disponível em: <http://arxiv.org/abs/1306.0239>.

[27] BISHOP, C. M. **Pattern Recognition and Machine Learning (Information Science and Statistics)**. Secaucus, NJ, USA: Springer-Verlag New York, Inc., 2006. ISBN 0387310738.

[28] RASMUSSEN, C. E.; GHAHRAMANI, Z. Occam's razor. In: **In Advances in Neural Information Processing Systems 13**. [S.l.]: MIT Press, 2001. p. 294–300.

[29] PEARSON, K. On lines and planes of closest fit to systems of points in space. **Philosophical Magazine**, v. 2, n. 6, p. 559–572, 1901.

[30] SORZANO, C. O. S.; VARGAS, J.; PASCUAL-MONTANO, A. D. A survey of dimensionality reduction techniques. **CoRR**, abs/1403.2877, 2014. Disponível em: <http://arxiv.org/abs/1403.2877>.

[31] GUYON, I.; ELISSEEFF, A. An introduction to variable and feature selection. **J. Mach. Learn. Res.**, JMLR.org, v. 3,

p. 1157–1182, mar. 2003. ISSN 1532-4435. Disponível em: <http://dl.acm.org/citation.cfm?id=944919.944968>.

[32] R.O. Duda, P.E. Hart, and D.G. Stork, Pattern Classification, New York: John Wiley & Sons, 2001, Pp. Xx + 654, ISBN: 0-471-05669-3. **J. Classif.**, Springer-Verlag New York, Inc., Secaucus, NJ, USA, v. 24, n. 2, p. 305–307, set. 2007. ISSN 0176-4268. Disponível em: <http://dx.doi.org/10.1007/s00357-007-0015-9>.

[33] HINTON, G. E. et al. Improving neural networks by preventing co-adaptation of feature detectors. **CoRR**, abs/1207.0580, 2012. Disponível em: <http://dblp.uni-trier.de/db/journals/corr/corr1207.htmlabs-1207-0580>.

[34] KOHAVI, R.; JOHN, G. H. Wrappers for feature subset selection. **Artif. Intell.**, Elsevier Science Publishers Ltd., Essex, UK, v. 97, n. 1-2, p. 273–324, dez. 1997. ISSN 0004-3702. Disponível em: <http://dx.doi.org/10.1016/S0004-3702(97)00043-X>.

[35] AMALDI, E.; KANN, V. On the Approximation of Minimizing Non-Zero Variables or Unsatisfied Relations in Linear Systems. **Theoretical Computer Science**, v. 209, p. 237–260, 1998.

[36] NG, A. Y. Feature selection, l1 vs. l2 regularization, and rotational invariance. In: **In ICML**. [S.l.: s.n.], 2004.

[37] MOORE, R. C.; DENERO, J. L1 and l2 regularization for multiclass hinge loss models. In: **Symposium on Machine Learning in Speech and Natural Language Processing**. [s.n.], 2011. Disponível em: <http://www.ttic.edu/sigml/symposium2011/papers/Moore+DeNero$_{R}egularization.pdf$¿.

[38] FLACH, P. **Machine Learning: The Art and Science of Algorithms That Make Sense of Data**. New York, NY, USA: Cambridge University Press, 2012. ISBN 1107422221, 9781107422223.

[39] MARSLAND, S. **Machine Learning: An Algorithmic Perspective**. 1st. ed. [S.l.]: Chapman & Hall/CRC, 2009. ISBN 1420067184, 9781420067187.

[40] MITCHELL, T. M. **Machine learning**. New York, NY [u.a.: McGraw-Hill, 1997. ISBN 0071154671 9780071154673. Disponível em: <http://www.worldcat.org/search?qt=worldcat$_{o}rg_{a}llq$ = 9780071154673¿.

[41] DENG, H.; RUNGER, G. C.; TUV, E. Bias of importance measures for multi-valued attributes and solutions. In: **ICANN (2)'11**. [S.l.: s.n.], 2011. p. 293–300.

[42] CHANG, C.-C.; LIN, C.-J. LIBSVM: A library for support vector machines. **ACM Transactions on Intelligent Systems and Technology**, v. 2, p. 27:1–27:27, 2011. Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`.

[43] QUINLAN, J. R. **C4.5: Programs for Machine Learning**. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1993. ISBN 1-55860-238-0.

[44] GOLDBERG, Y.; ELHADAD, M. **Learning Sparser Perceptron Models**. [S.l.], 2011.

[45] ROSENBLATT, F. The perceptron: a probabilistic model for information storage and organization in the brain. **Psychological Review**, v. 65, n. 6, p. 386–408, nov. 1958.

[46] DAGUM, L.; MENON, R. Openmp: An industry-standard api for shared-memory programming. **IEEE Comput. Sci. Eng.**, IEEE Computer Society Press, Los Alamitos, CA, USA, v. 5, n. 1, p. 46–55, jan. 1998. ISSN 1070-9924. Disponível em: <http://dx.doi.org/10.1109/99.660313>.

[47] SANTOS, C. N. dos. **Entropy Guided Transformation Learning**. Tese (Doutorado) — Pontifícia Universidade Católica do Rio de Janeiro, 2009.

[48] TSOCHANTARIDIS, I. et al. Large margin methods for structured and interdependent output variables. **JOURNAL OF MACHINE LEARNING RESEARCH**, v. 6, p. 1453–1484, 2005.

[49] GÜNGÖR, T. Part-of-speech tagging. In: INDURKHYA, N.; DAMERAU, F. J. (Ed.). **Handbook of Natural Language Processing, Second Edition**. Boca Raton, FL: CRC Press, Taylor and Francis Group, 2010. ISBN 978-1420085921.

[50] SANTOS, C. D.; ZADROZNY, B. Learning character-level representations for part-of-speech tagging. In: JEBARA, T.; XING, E. P. (Ed.). **Proceedings of the 31st International Conference on Machine Learning (ICML-14)**. JMLR Workshop and Conference Proceedings, 2014. p. 1818–1826. Disponível em: <http://jmlr.org/proceedings/papers/v32/santos14.pdf>.

[51] SANTOS, C. N. dos; MILIDIÚ, R. L.; RENTERÍA, R. P. Portuguese part-of-speech tagging using entropy guided transformation learning. In: **PROPOR**. [S.l.: s.n.], 2008. p. 143–152.

[52] COLLINS, M. Discriminative training methods for hidden markov models: theory and experiments with perceptron algorithms. In: **Proceedings of the ACL-02 conference on Empirical methods in natural language processing - Volume 10**. Stroudsburg, PA, USA: Association for Computational Linguistics, 2002. (EMNLP '02), p. 1–8. Disponível em: <http://dx.doi.org/10.3115/1118693.1118694>.

[53] SØGAARD, A. Semi-supervised condensed nearest neighbor for part-of-speech tagging. In: _____. **Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies (ACL-HLT**. [S.l.]: Association for Computational Linguistics, 2011.

[54] SHEN, L.; SATTA, G.; JOSHI, A. K. Guided learning for bidirectional sequence classification. In: CARROLL, J. A.; BOSCH, A. van den; ZAENEN, A. (Ed.). **ACL**. The Association for Computational Linguistics, 2007. Disponível em: <http://dblp.uni-trier.de/db/conf/acl/acl2007.html#ShenSJ07>.

[55] TOUTANOVA, K. et al. Feature-rich part-of-speech tagging with a cyclic dependency network. In: **Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology - Volume 1**. Stroudsburg, PA, USA: Association for Computational Linguistics, 2003. (NAACL '03), p. 173–180. Disponível em: <http://dx.doi.org/10.3115/1073445.1073478>.

[56] ALUÍSIO, S. et al. 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**. Berlin, Heidelberg: Springer-Verlag, 2003. (PROPOR'03), p. 110–117. ISBN 3-540-40436-8. Disponível em: <http://dl.acm.org/citation.cfm?id=1758748.1758769>.

[57] MARCUS, M. et al. **Treebank-3**. 1999. Disponível em: <https://catalog.ldc.upenn.edu/LDC99T42>.

[58] PORTER, M. An algorithm for suffix stripping. **Program: electronic library and information systems**, MCB UP Ltd, v. 14, n. 3, p. 130–137, 1980.

[59] ORENGO, V. M.; HUYCK, C. A Stemming Algorithm for Portuguese Language. In: **Proc. of Eigth Symposium on String Processing and Information Retrieval (SPIRE 2001) - Chile**. [S.l.: s.n.], 2001. p. 186–193.

[60] ZHOU, Z.-H. **Ensemble Methods: Foundations and Algorithms**. 1st. ed. [S.l.]: Chapman & Hall/CRC, 2012. ISBN 1439830037, 9781439830031.

[61] BREIMAN, L. Bagging predictors. **Mach. Learn.**, Kluwer Academic Publishers, Hingham, MA, USA, v. 24, n. 2, p. 123–140, ago. 1996. ISSN 0885-6125. Disponível em: <http://dx.doi.org/10.1023/A:1018054314350>.

[62] NIVRE, J. **Dependency Grammar and Dependency Parsing**. [S.l.], 2005. Disponível em: <http://www.vxu.se/msi/~nivre/papers/05133.pdf>.

[63] TESNIÈRE, L. **Eléments de Syntaxe Structurale**. Paris: Klincksieck, 1959.

[64] BUCHHOLZ, S.; MARSI, E. Conll-x shared task on multilingual dependency parsing. In: **Proceedings of the Tenth Conference on Computational Natural Language Learning**. Stroudsburg, PA, USA: Association for Computational Linguistics, 2006. (CoNLL-X '06), p. 149–164. Disponível em: <http://dl.acm.org/citation.cfm?id=1596276.1596305>.

[65] MCDONALD, R.; LERMAN, K.; PEREIRA, F. Multilingual dependency analysis with a two-stage discriminative parser. In: **Proceedings of the Conference on Computational Natural Language Learning (CoNLL)**. [S.l.: s.n.], 2006. p. 216–220.

[66] KOO, T. et al. Dual decomposition for parsing with non-projective head automata. In: **Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing**. Stroudsburg, PA, USA: Association for Computational Linguistics, 2010. (EMNLP '10), p. 1288–1298. Disponível em: <http://dl.acm.org/citation.cfm?id=1870658.1870783>.

[67] CRESTANA, C. E. M. **A Token Classification Approach to Dependency Parsing**. Dissertação (Mestrado) — Pontifícia Universidade Católica do Rio de Janeiro, 2010.

[68] FREITAS, C.; ROCHA, P.; BICK, E. Floresta Sintá(c)tica: Bigger, thicker and easier. In: TEIXEIRA, A. et al. (Ed.). **Computational Processing of the Portuguese Language**. [S.l.: s.n.], 2008. (Lecture Notes in Computer Science, v. 5190), p. 216–219.

[69] TJONG, E. F.; SANG, K.; DÉJEAN, H. Introduction to the conll-2001 shared task: Clause identification. In: **Proceedings of the 2001 Workshop on Computational Natural Language Learning - Volume 7**. Stroudsburg, PA, USA: Association for Computational Linguistics, 2001. (ConLL '01). Disponível em: <http://dx.doi.org/10.3115/1117822.1455626>.

[70] FERNANDES, E.; SANTOS, C. dos; MILIDIÚ, R. A machine learning approach to portuguese clause identification. In: PARDO, T. et al. (Ed.). **Computational Processing of the Portuguese Language**. Springer Berlin Heidelberg, 2010, (Lecture Notes in Computer Science, v. 6001). p. 55–64. ISBN 978-3-642-12319-1. Disponível em: <http://dx.doi.org/10.1007/978-3-642-12320-7_8>.

[71] SANG, E. F. T. K.; BUCHHOLZ, S. Introduction to the conll-2000 shared task: Chunking. In: **Proceedings of the 2Nd Workshop on Learning Language in Logic and the 4th Conference on Computational Natural Language Learning - Volume 7**. Stroudsburg, PA, USA: Association for Computational Linguistics, 2000. (ConLL '00), p. 127–132. Disponível em: <http://dx.doi.org/10.3115/1117601.1117631>.

[72] FERREIRA, G. C. D. N. **A Machine Learning Approach for Portuguese Text Chunking**. Dissertação (Mestrado) — Pontifícia Universidade Católica do Rio de Janeiro, 2011.

[73] FILHO, H. P. B. **Predição do Comportamento do Mercado Financeiro Utilizando Notícias em Português**. Dissertação (Mestrado) — Pontifícia Universidade Católica do Rio de Janeiro, 2014.

[74] BARROSO, Y. M. **Unrestricted Coreference Resolution with Latent Structured Sparse Perceptron**. Dissertação (Mestrado) — Pontifícia Universidade Católica do Rio de Janeiro, 2014.

A
Conjunto de Etiquetas POS do MacMorpho

Tabela A.1: Anotação Morfossintática do Corpus MacMorpho.

Etiqueta	**Classe Gramatical**
ADJ	adjetivo
ADV	advérbio
ADV-KS	advérbio conectivo subordinativo
ADV-KS-REL	advérbio relativo subordinativo
ART	artigo, definido ou indefinido
KC	conjunção coordenativa
KS	conjunção subordinativa
IN	interjeição
N	nome
NPROP	nome próprio
NUM	numeral
PCP	particípio
PDEN	palavra denotativa
PREP	preposição
PROADJ	pronome adjetivo
PRO-KS	pronome conectivo subordinativo
PROPESS	pronome pessoal
PRO-KS-REL	pronome relativo conectivo subordinativo
PROSUB	pronome substantivo
V	verbo
VAUX	verbo auxiliar
CUR	símbolo de moeda corrente

B
Conjunto de Etiquetas POS do WSJ

Tabela B.1: Conjunto de Etiquetas POS do WSJ.

Etiqueta	**Classe Gramatical**	**Exemplo**
CC	coordinating conjunction	and
CD	cardinal number	1, third
DT	determiner	the
EX	existential there	there is
FW	foreign word	d'hoevre
IN	preposition/subordinating conjunction	in, of, like
JJ	adjective	green
JJR	adjective, comparative	greener
JJS	adjective, superlative	greenest
LS	list marker	1)
MD	modal	could, will
NN	noun, singular or mass	table
NNS	noun plural	tables
NNP	proper noun, singular	John
NNPS	proper noun, plural	Vikings
PDT	predeterminer	both the boys
POS	possessive ending	friend's
PRP	personal pronoun	I, he, it
PRP$	possessive pronoun	my, his
RB	adverb	however, usually, naturally, here, good
RBR	adverb, comparative	better
RBS	adverb, superlative	best
SYM	symbol	%
RP	particle	give up
TO	to	to go, to him
UH	interjection	uhhuhhuhh
VB	verb, base form	take
VBD	verb, past tense	took
VBG	verb, gerund/present participle	taking
VBN	verb, past participle	taken
VBP	verb, sing. present, non-3d	take
VBZ	verb, 3rd person sing. present	takes
WDT	wh-determiner	which
WP	wh-pronoun	who, what
WP$	possessive wh-pronoun	whose
WRB	wh-abverb	where, when

C
Etiquetas de Oração e *chunk*

A seguir descrevemos as etiquetas de limites de orações e de *chunks*, que são usadas como atributos básicos na tarefa de análise de dependência.

C.1
Etiquetas de Oração

Os limites das orações são uma informação sintática que está disponível no corpus Bosque. Um exemplo de sentença do corpus Bosque, quebrada em oração por parênteses, é mostrada na figura C.1:

(Ninguém percebe (que ele quer (impor sua presença)) .)

Figura C.1: Uma sentença anotada com informações dos limites das orações, indicados por parênteses

As etiquetas que delimitam as orações são descritas tabela C.1. A coluna *Start* contém o atributo binário que indica se pelo menos uma oração começa neste *token*. A coluna *End* contém o atributo binário que indica se pelo menos uma oração termina neste *token*. O atributo *Oração* codifica as orações da sentença, utilizando parênteses, como no exemplo da figura C.1.

Estas etiquetas que marcam as orações, são utilizadas como atributos básicos na tarefa de análise de dependência.

C.2
Etiquetas de *Chunk*

Na tabela C.2 descrevemos as etiquetas que marcam os *chunks*, utilizadas na tarefa de análise de dependência.

Tabela C.1: Etiquetas de oração

Token	**POS**	***Start***	***End***	***Oração***
Ninguém	pron-indp	S	X	(S*
percebe	v-fin	X	X	*
que	conj-s	S	X	(S*
ele	pron-pers	X	X	*
quer	v-fin	X	X	*
impor	v-inf	S	X	(S*
sua	pron-det	X	X	*
presença	n	X	E	*S)S)
.	.	X	E	*S)

Tabela C.2: Etiquetas de *chunk*

Informação de *chunk* do *token*	**Etiqueta**
Início de *chunk* nominal	B-NP
Início de *chunk* preposicional	B-PP
Início de *chunk* verbal	B-VP
Contido em um *chunk* nominal	I-NP
Contido em um *chunk* preposicional	I-PP
Contido em um *chunk* verbal	I-VP
Fora de qualquer *chunk*	O

D
Lemas dos verbos na análise de dependência

A tabela D.1 mostra o conjunto de lemas dos verbos à esquerda, na tarefa de análise de dependência, após a regularização do ciclo 0.

Tabela D.1: Conjunto de lemas dos verbos à esquerda, após a regularização do ciclo 0.

{inexistente} abandonar abordar abrir acabar aceitar acentuar acertar achar acompanhar acontecer acreditar acrescentar actuar acusar adaptar adequar adiantar admitir adquirir afastar afirmar aguardar alargar alegar aliar alimentar alinhar analisar andar anular anunciar apanhar aparecer aplicar apoiar apontar apreciar apresentar aprovar aproximar apurar armar articular assegurar assinar assistir assumir atingir atravessar atribuir aumentar autorizar avançar basear bater brincar buscar caber cair casar causar ceder celebrar chamar chegar chumbar citar classificar cobrir colocar comer cometer começar comparar complicar compor comprar compreender comunicar conceder concentrar concluir concretizar condenar conduzir confirmar conhecer conjugar conquistar conseguir considerar consistir constatar constituir construir contar conter continuar contribuir controlar convencer converter correr cotar crescer criar criticar cumprir custar dar datar debater decidir declarar decorrer dedicar defender definir deixar demonstrar deparar derrotar descer descobrir desconhecer descrever desejar desenvolver designar deslocar destacar destinar destruir desvincular detectar deter determinar dever diminuir dirigir discutir dispensar dispor disputar distribuir divertir dividir divulgar dizer dormir dotar efectuar elaborar eleger elevar eliminar emigrar emitir empatar empenhar encerrar encontrar enfrentar entender enterrar entrar entregar enumerar enviar envolver escolher escrever esgotar esperar esquecer estabelecer estar estimar estragar estudar esvoaçar evitar evocar exigir existir explicar expor extinguir falar faltar fazer fechar ferir ficar filmar fixar formar fornecer funcionar fundar ganhar garantir gerir golear governar gritar guardar haver identificar ilustrar imaginar impedir implicar impressionar inaugurar incluir indicar informar iniciar inspirar instalar integrar interpretar investigar investir ir jogar juntar justificar lançar lembrar ler levar liderar ligar limar mandar manifestar manter marcar matar melhorar menosprezar merecer montar morrer mostrar mudar nascer negociar obrigar observar obter ocorrer ocupar odiar oferecer opor optar organizar orientar ouvir pagar parar parecer participar partir passar pedir pegar pensar perder permitir pertencer pisar poder praticar prender preocupar preparar prestar pretender prever priorizar privilegiar proceder procurar proibir promover pronunciar propor prosseguir proteger provar provocar publicar pôr querer radicar realizar receber recolher recomendar reconhecer recordar recuperar recém-formar reduzir referir registar registrar regressar reivindicar relacionar remeter render renovar reparar repartir repetir representar resolver respeitar responder resultar retirar reunir revelar romper roubar saber sair seduzir seguir sentar sentir ser servir significar situar sofisticar sofrer solicitar subir sublinhar substituir suceder sujeitar superar supor surgir surpreender suspender sustentar tentar ter terminar tirar tomar tornar trabalhar traduzir transformar transmitir tratar trazer ultrapassar usar utilizar variar vasculhar vencer vender ver verificar vigiar vir virar visitar viver voltar votar