Universidade de São Paulo - USP
Universidade Federal de São Carlos - UFSCar
Universidade Estadual Paulista - UNESP

Aprendizado Bayesiano Aplicado ao Processamento de Línguas Naturais

Thiago Alexandre Salgueiro Pardo
Maria das Graças Volpe Nunes

NILC-TR-02-25

Dezembro, 2002

Série de Relatórios do Núcleo Interinstitucional de Lingüística Computacional
NILC - ICMC-USP, Caixa Postal 668, 13560-970 São Carlos, SP, Brasil

RESUMO

Técnicas de aprendizado de máquina têm sido amplamente usadas para resolver todo tipo de problema de computação, devido, principalmente, a sua flexibilidade, adaptabilidade e bons resultados que têm conseguido. Em face disso, este relatório faz uma revisão de aplicações recentes de processamento de línguas naturais que fazem uso de técnicas de aprendizado de máquina, mais especificamente, o aprendizado bayesiano, dada sua simplicidade, robustez e bons resultados gerados quando aplicado ao processamento de textos, área esta que também tem tido bastante destaque ultimamente.

Este trabalho conta com o apoio financeiro da FAPESP e do CNPq.

ÍNDICE

Aprendizado Bayesiano Aplicado ao Processamento de Línguas Naturais[1]

1. Introdução

Técnicas de Aprendizado de Máquina (AM) têm sido muito usadas em todos os ramos da computação, por exemplo, reconhecimento de imagens, sistemas baseados em conhecimento, roteamento de redes e processamento de textos, conseguindo resultados satisfatórios e, às vezes, até melhor do que se esperava.

As técnicas de AM são classicamente divididas em técnicas de aprendizado supervisionado e não supervisionado. No aprendizado supervisionado, o conjunto de dados do qual se pretende extrair conhecimento já vem todo rotulado, isto é, a cada instância está associada sua classificação, a que o algoritmo de AM deve aprender a realizar. No aprendizado não supervisionado, o conjunto de dados não vem rotulado, sendo o algoritmo de AM incumbido de tentar agrupar os dados de acordo com suas características da melhor maneira possível (isso é o que se chama de *clustering*). As técnicas de AM podem ainda ser classificadas de acordo com o paradigma que seguem, que pode ser simbólico, estatístico, neural ou genético. O aprendizado simbólico se caracteriza por extrair conhecimento que seja acessível e interpretável por seres humanos; o aprendizado estatístico trabalha com fórmulas estatísticas e probabilidades; o aprendizado neural consiste, principalmente, no uso de redes neurais para classificação; o aprendizado genético, por fim, engloba os algoritmos genéticos e suas aplicações.

Em vista da crescente utilização e do sucesso obtido com as técnicas de AM, este relatório dedica-se a mostrar o estado da arte de uma técnica específica de AM, o aprendizado bayesiano, aplicada ao Processamento de Línguas Naturais (PLN), uma área que também tem crescido bastante ultimamente, responsável por desenvolver técnicas, ferramentas e recursos para o tratamento automatizado das línguas naturais. Na Seção 2, uma breve revisão do aprendizado bayesiano é feita. A Seção 3 apresenta diversas aplicações de aprendizado bayesiano ao PLN, tentando mostrar o estado da arte, enquanto a Seção 4 apresenta conclusões e comentários finais sobre o assunto tratado.

2. Aprendizado Bayesiano

O aprendizado bayesiano é do tipo supervisionado, já que são fornecidas ao algoritmo de AM as instâncias juntamente com seus rótulos (ou seja, as classes). Seguindo o paradigma estatístico, o algoritmo faz uso de fórmulas estatísticas e cálculo de probabilidades para realizar a classificação (Mitchell, 1997).

As vantagens do AM estatístico, especialmente o aprendizado bayesiano, são, principalmente: (a) o fato de se poder embutir nas probabilidades calculadas o conhecimento de domínio que se tem (caso se tenha) e (b) a capacidade das classificações feitas pelo algoritmo de AM se basearem em evidências fornecidas, que podem aumentar ou diminuir as probabilidades das classes a serem observadas em uma nova instância que se quer classificar. Por outro lado, as desvantagens do AM estatístico residem justamente no seu caráter estatístico, ou seja, (a) muitas

[1] Esse trabalho teve o apoio da Fapesp e do CNPq.

probabilidades devem ser calculadas e (b) isto pode ocasionar um alto custo computacional. Uma das soluções para o custo do cálculo das probabilidades necessárias para o aprendizado do algoritmo é o uso do classificador *naive-Bayes*[2], o qual será enfocado neste relatório, dada sua vasta utilização e bons resultados obtidos principalmente na área de PLN.

O classificador *naive-Bayes* se baseia na aplicação do Teorema de Bayes para o cálculo das probabilidades necessárias para a classificação. O Teorema de Bayes é mostrado abaixo já no contexto de AM, isto é, dada uma nova instância $A=a_1, a_2 ... a_n$, deseja-se predizer sua classe:

$$P(classe \mid A) = \frac{P(A \mid classe) \times P(classe)}{P(A)}$$

Como $A=a_1, a_2 ... a_n$, tem-se:

$$P(classe \mid a_1 ... a_n) = \frac{P(a_1 ... a_n \mid classe) \times P(classe)}{P(a_1 ... a_n)}$$

Para calcular a classe mais provável da nova instância, calcula-se a probabilidade de todas as possíveis classes e, no fim, escolhe-se a classe com a maior probabilidade como rótulo da nova instância. Em termos estatísticos, isso é equivalente a maximizar a $P(classe|a_1...a_n)$. Para tanto, deve-se maximizar o valor do numerador $P(a_1...a_n|classe)\times P(classe)$ e minimizar o valor do denominador $P(a_1...a_n)$. Como o denominador $P(a_1...a_n)$ é uma constante, pois não depende da variável *classe* que se está procurando, pode-se anulá-lo no Teorema de Bayes, resultando na fórmula abaixo, na qual se procura a classe que maximize o valor do termo $P(classe| a_1...a_n)=P(a_1...a_n|classe)\times P(classe)$:

$$\arg\max P(classe \mid a_1 ... a_n) = \arg\max P(a_1 ... a_n \mid classe) \times P(classe)$$

A suposição "ingênua" que o classificador *naive-Bayes* faz é que todos os atributos $a_1...a_n$ da instância que se quer classificar são independentes. Sendo assim, o complexo cálculo do valor do termo $P(a_1...a_n|classe)$ reduz-se ao simples cálculo $P(a_1|classe)\times... \times P(a_n|classe)$. Assim, a fórmula final utilizada pelo classificador é:

$$\arg\max P(classe \mid a_1 ... a_n) = \arg\max \prod_i P(a_i \mid classe) \times P(classe)$$

Sabe-se, entretanto, que, na maioria dos casos, a suposição de independência dos atributos de uma instância é falsa. Mesmo assim, o classificador *naive-Bayes* produz resultados bastante satisfatórios. Quando os atributos são realmente independentes, o classificador fornece a solução ótima.

Como dito anteriormente, o cálculo da classe de uma nova instância consiste no cálculo da probabilidade de todas as possíveis classes, escolhendo-se, a seguir, a classe com maior probabilidade. De acordo com a fórmula acima, devem-se calcular os termos $P(a_i|classe)$ e $P(classe)$. $P(classe)$ é simplesmente o número de casos[3]

[2] *Naive*, ou *naïve*, significa "ingênuo" em português.
[3] Chama-se de caso uma instância com seu rótulo durante o treinamento do classificador.

pertencente à classe em questão sobre o número total de casos. P(a_i|classe), por sua vez, é o número de casos pertencente à classe em questão com o atributo i com valor a_i sobre o número total de casos.

Como exemplo do uso do classificador *naive-Bayes*, considere o conjunto de dados para treinamento da Tabela 1, no qual se deve decidir se se vai jogar tênis ou não, com base na previsão do tempo (Mitchell, 1997).

Tabela 1 – Exemplo de conjunto de treinamento

Caso	Outlook	Temperature	Humidity	Wind	PlayTennis
1	sunny	hot	high	weak	No
2	sunny	hot	high	strong	No
3	overcast	hot	high	weak	Yes
4	rain	mild	high	weak	Yes
5	rain	cool	normal	weak	Yes
6	rain	cool	normal	strong	No
7	overcast	cool	normal	strong	Yes
8	sunny	mild	high	weak	No
9	sunny	cool	normal	weak	Yes
10	rain	mild	normal	weak	Yes
11	sunny	mild	normal	strong	Yes
12	overcast	mild	high	strong	Yes
13	overcast	hot	normal	weak	Yes
14	rain	mild	high	strong	No

Supondo que se quisesse descobrir a classe mais provável (jogar ou não tênis) para a instância [Outlook=sunny, Temperature=cool, Humidity=high, Wind=strong], com base nos dados de treinamento, dever-se-ia proceder da seguinte forma:

1) calcula-se a probabilidade de cada uma das classes ocorrer:

P(classe) = (número de casos da classe)/(número total de casos)

P(PlayTennis=yes) = 9/14 = 0.64
P(PlayTennis=no) = 5/14 = 0.36

2) calcula-se a probabilidade de cada um dos atributos da instância em questão em relação a cada possível classe. Por exemplo, para o atributo Wind=strong:

P(a_i|classe) = (número de casos da classe com atributo a_i)/(número total de casos da classe)

P(Wind=strong|PlayTennis=yes) = 3/9 = 0.33
P(Wind=strong|PlayTennis=no) = 3/5 = 0.60

3) tendo todas as probabilidades necessárias calculadas, basta calcular a probabilidade de cada classe ocorrer agora:

$$P(classe \mid a_1 ... a_n) = \prod_i P(a_i \mid classe) \times P(classe)$$

P(PlayTennis=yes|Outlook=sunny,Temperature=cool,Humidity=high, Wind=strong) = P(Outlook=sunny|PlayTennis=yes) × P(Temperature=cool|PlayTennis=yes) × P(Humidity=high|PlayTennis=yes) × P(Wind=strong|PlayTennis=yes) × P(PlayTennis=yes) = 0.0053

P(PlayTennis=no|Outlook=sunny,Temperature=cool,Humidity=high, Wind=strong) = P(Outlook=sunny|PlayTennis=no) × P(Temperature=cool|PlayTennis=no) × P(Humidity=high|PlayTennis=no) × P(Wind=strong|PlayTennis=no) × P(PlayTennis=no) = 0.0206

Protanto, a probabilidade de não jogar tênis é maior que a probabilidade de jogar, dados os atributos da instância que se quer classificar. Portanto, a classificação mais provável da nova instância é PlayTennis=no.

A próxima seção apresenta uma revisão literária de aplicações recentes do aprendizado bayesiano ao PLN.

3. Aprendizado Bayesiano Aplicado ao PLN

3.1. Categorização de Documentos

Categorização de documentos é a tarefa de se associar aos documentos suas categorias. Por exemplo, em uma página da Web de notícias jornalísticas, as possíveis categorias são "esporte", "política", "entretenimento", "ciência", "educação", etc. Com a crescente quantidade de informação disponível na Web, principalmente, a tarefa de categorização se torna cada vez mais importante para tarefas de recuperação de informação e sumarização automática, por exemplo.

A categorização de documentos pode ser classificada da seguinte forma:

- binária ou graduada: na categorização binária, um documento pode pertencer somente a uma das categorias possíveis; na categorização graduada, um documento pode pertencer a várias categorias com um certo grau de relevância;
- simples ou múltipla: na categorização simples, não há documentos comuns entre categorias; na categorização múltipla, um documento pode pertencer a mais de uma categoria.

Corrêa e Ludermir (2002) fizeram um estudo comparativo do desempenho de várias técnicas de AM para a categorização automática de documentos. As técnicas utilizadas foram redes neurais *Multilayer Perceptron*, *Self-organizing Maps* (SOM), árvores de decisão, regras de decisão e aprendizado bayesiano pelo uso do classificador *naive-Bayes*.

Os autores desenvolveram um sistema de categorização que segue a arquitetura da Figura 1, onde:

- na etapa de representação, os documentos que se quer categorizar são representados como vetores (Salton, 1988), em que cada posição do vetor representa uma palavra do documento;
- na etapa de pré-processamento, os vetores são analisados, realizando-se os processos de *stemming* (redução das palavras aos seus radicais), remoção de *stop words* (palavras irrelevantes para o processamento) e seleção de termos (ou seja, palavras ou conjuntos de palavras) mais importantes dos documentos para indexação destes;

- na etapa de classificação, novos vetores são montados para os documentos, contendo, agora, somente os termos selecionados na etapa de pré-processamento. Por fim, estes vetores com suas categorias (dentre as possíveis) são utilizados para o treinamento e teste dos algoritmos de AM utilizados, ou seja, na linguagem de AM, as posições de um vetor são os atributos da instância e a categoria associada ao vetor é seu rótulo.

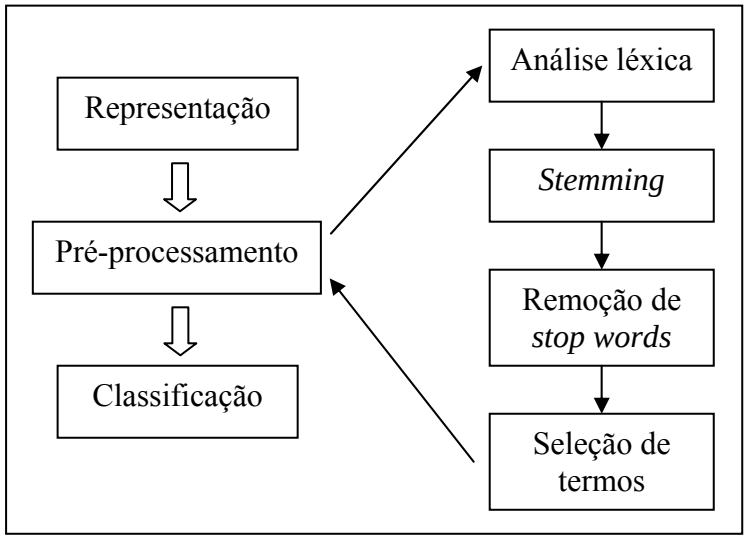

Figura 1 – Arquitetura do sistema de categorização automática de documentos

Três coleções de dados foram codificadas conforme a arquitetura acima e serviram para o estudo comparativo realizado: a primeira coleção consistiu de 2.340 páginas da Web em inglês classificadas em 6 categorias possíveis, com uma dessas categorias possuindo 15 subcategorias; a segunda coleção consistiu de 1.300 páginas da Web em inglês e em português do Brasil classificadas em 2 categorias; a terceira coleção consistiu de 319 artigos classificados em 6 categorias. Durante a avaliação do sistema de categorização, as duas primeiras coleções foram utilizadas para categorização binária e simples, enquanto a última coleção foi utilizada para categorização múltipla. As Tabelas 2, 3 e 4 mostram o desempenho dos classificadores para cada uma das coleções de dados descritas acima:

- para a primeira coleção, o classificador *naive-Bayes* se saiu melhor, ou seja, obteve a menor taxa de erro e a maior *F-Measure*, sendo capaz, portanto, de identificar bem as subclasses da coleção de dados;
- para a segunda e terceira coleções, o classificador *naive-Bayes* obteve a segunda e terceira posições, respectivamente.

Tabela 2 – Desempenho do sistema de categorização para a primeira coleção de dados

Técnica	**Erro**	***F-Measure***
Naive-Bayes	23,25	76,75
Redes neurais	25,90	74,10
SOM	33,68	66,90
Árvores de decisão	37,95	62,05
Regras de decisão	38,38	61,62

Tabela 3 – Desempenho do sistema de categorização para a segunda coleção de dados

Técnica	**Erro**	***F-Measure***
Redes neurais	5,00	95,00
Naive-Bayes	5,33	94,67
SOM	7,00	93,00
Regras de decisão	14,00	86,00
Árvores de decisão	14,33	85,67

Tabela 4 – Desempenho do sistema de categorização para a terceira coleção de dados

Técnica	**Erro**	***F-Measure***
Redes neurais	15,09	88,24
Árvores de decisão	17,92	82,08
Naive-Bayes	20,75	79,25
SOM	27,36	76,62
Regras de decisão	27,36	72,64

Ao fim das avaliações, ao estabelecer um ranque dos classificadores de acordo com o erro que obtiveram, os autores chegaram à Tabela 5, na qual o classificador *naive-Bayes* só teve desempenho pior que as redes neurais, mostrando-se, portanto, bastante promissor para a resolução dos problemas de categorização de documentos em geral.

Tabela 5 – Ranque dos classificadores para categorização de documentos

Técnicas de AM
Redes neurais
Naive-Bayes
SOM
Árvores de decisão
Regras de decisão

Outros trabalhos interessantes sobre categorização de documentos por meio de técnicas de AM são os trabalhos de Busemann et al. (2000) e Yang e Liu (1999).

3.2. Sumarização de Textos

A sumarização de um texto consiste um produzir uma versão mais curta do mesmo para determinado usuário e/ou tarefa. Com a crescente quantidade de informação disponível atualmente e tempo cada vez mais reduzido que as pessoas têm para apreender o máximo dessa informação, a sumarização automática tornou-se um amplo campo de pesquisa.

O conteúdo do sumário, ou resumo, pode variar dependendo da audiência a que se destina e para qual função está sendo produzido:

- um leitor leigo no assunto do texto, pode, por exemplo, querer que o sumário contenha definições e informação contextual suficiente para que ele entenda o restante do conteúdo apresentado, enquanto um leitor especialista no assunto, pode querer somente a idéia principal do texto, ignorando todo o resto;

- se a função para a qual se quer o sumário é que ele sirva de índice para um texto, este sumário pode ser, por exemplo, apenas uma lista das palavras-chave do texto.

Um sumário pode ser classificado de várias formas. Com relação ao conteúdo de um sumário, este pode ser classificado como indicativo, informativo ou crítico: sumários indicativos servem como índice para o texto, listando seu conteúdo, podendo mesmo não estar em forma textual (por exemplo, uma lista de palavras); sumários informativos contêm toda a informação relevante do texto, podendo substituí-lo, inclusive; sumários críticos, por sua vez, avaliam o conteúdo do texto (exemplos típicos desse caso são as resenhas). Sumários também podem ser classificados de acordo com a forma como são construídos, podendo ser chamados de sumários (da palavra *abstracts*, do inglês), propriamente ditos, e extratos: sumários transformam as estruturas superficiais de textos, podendo, por exemplo, agregar informações, agrupar sentenças, separar uma sentença em várias e fazer generalizações e especificações com as informações apresentadas; extratos, por sua vez, são constituídos por sentenças completas extraídas de textos, não passando, portanto, por processos de re-escrita. Quanto a abordagem computacional utilizada para se produzir os sumários, elas podem ser classificadas como profunda, superficial e híbrida: a abordagem profunda faz uso de modelos formais e conhecimento lingüístico para produzir, normalmente, sumários (referentes aos *abstracts*); a abordagem superficial faz uso de conhecimento empírico e técnicas estatísticas para produzir, normalmente, extratos; a abordagem híbrida, por fim, pode combinar técnicas profundas e superficiais para produzir sumários.

Larocca Neto et al. desenvolveram um sumarizador extrativo híbrido (ou seja, produz extratos pela abordagem computacional híbrida) que utiliza técnicas de AM, mais especificamente, árvores de decisão e o classificador *naive-Bayes*. Os dados de treinamento e teste para os algoritmos de AM foram obtidos da seguinte forma:

1. Os textos e os sumários autênticos (produzidos pelo próprio autor do texto) eram pré-processados, ou seja, realizaram-se os processos de *stemming*, remoção de *stop words* e *case folding*, sendo este último responsável por deixar todo o texto em caixa baixa;
2. Os textos e seus sumários autênticos foram alinhados, isto é, para cada texto, identificaram-se as sentenças do texto que correspondiam às sentenças do respectivo sumário, seguindo a proposta de Mani e Bloedorn (1998). Nesta proposta, para cada sentença do sumário, procura-se a sentença do correspondente texto que tenha a maior medida do co-seno, calculada da seguinte forma:
    - Constroem-se os vetores de todas as sentenças do texto e do sumário (Salton, 1988);
    - Calculam-se as medidas do co-seno entre todo par de sentenças x e y do texto e do sumário, respectivamente:

$$co - seno = \frac{< x, y >}{|x| \times |y|}$$

onde <x,y> é o produto escalar de x e y e |x| é o módulo de x.

- Para cada sentença do sumário, seleciona-se a sentença do correspondente texto que tenha a maior medida de co-seno com aquela sentença para formar o novo sumário, chamado ideal.

3. Extraíram-se de cada sentença dos textos atributos representativos de seu conteúdo, a saber:
    - TF-ISF: *term frequency-inverse documento frequency*, que é uma medida da representatividade da sentença no texto (Larocca Neto et al., 2000);
    - tamanho da sentença;
    - posição da sentença no texto;
    - similaridade da sentença com o título do texto: novamente, calculada pela medida de co-seno;
    - similaridade da sentença com as palavras-chave do texto: pela medida de co-seno, novamente;
    - coesão entre sentenças: pelo uso de uma variação da medida de co-seno;
    - coesão entre a sentença e o centróide do texto: pela variação da medida de co-seno, novamente;
    - ocorrência de nomes próprios na sentença;
    - presença de anáforas na sentença;
    - ocorrência de termos que indiquem informação supérflua na sentença;
    - etc.

4. Construíram-se os conjuntos de treinamento e teste para os algoritmos de AM da seguinte forma:
    - cada instância do conjunto de dados representa uma sentença do texto e é formada pelos atributos acima (item 2);
    - a classe, ou rótulo, de cada instância indica se a sentença representada por aquela instância é adequada ou não para estar no sumário, sendo que as sentenças do texto adequadas para estar no sumário são aquelas que foram selecionadas para formar o sumário ideal (item 1).

Os dados utilizados para treinar e testar o sumarizador foram os textos da base de dados do TIPSTER (Harman, 1994). Para comparação com o sumarizador produzido, os autores também geraram sumários com o Microsoft Word Summarizer e com um algoritmo que seleciona as primeiras sentenças de um texto para formar seu sumário. Os resultados obtidos são mostrados na Tabela 6 em termos das medidas de *precision* e *recall*. De acordo com a tabela, o classificador *naive-Bayes* se mostrou bem melhor que todos os outros métodos, embora todos tenham tido um desempenho ruim.

Tabela 6 – Desempenho dos sumarizadores de texto

Sumarizador	***Precision***	***Recall***
Naive-Bayes	53,43	53,43
Microsoft Word Summarizer	38,80	43,67
Árvores de decisão	34,68	34,68
Primeiras sentenças do texto	32,03	32,03

Outros trabalhos interessantes sobre sumarização de textos que utilizam técnicas de AM são Kupiec et al. (1995) e Larocca Neto (2002).

3.3. Desambiguação do Sentido das Palavras

A tarefa de desambiguação do sentido das palavras (DSP) de um texto/sentença consiste em determinar a acepção correta das palavras dentro de seus contextos. Por exemplo, a palavra "banco" pode ter vários significados: "instituição financeira", "artefato humano para se sentar", "dar apoio financeiro", etc. O processo de DSP é muito importante para o desenvolvimento de tradutores automáticos.

Várias abordagens podem ser utilizadas para a detecção automática do sentido das palavras. Pode-se fazer uso de ferramentas como *taggers* (ou seja, etiquetadores morfossintáticos) e *parsers* (analisadores sintáticos), podendo-se variar ainda, por exemplo, o tamanho do contexto ao redor da palavra que se quer desambiguar no texto/sentença, os tipos de conhecimento utilizados para a desambiguação (morfológico, morfossintático, sintático, sintático e semântico, etc), as abordagens utilizadas (profunda *vs* superficial), entre outros critérios.

Cucchiarelli e Velardi (2002) fizeram experimentos de DSP com algoritmos de AM, mais especificamente, árvores de decisão e classificadores *naive-Bayes*, combinando diversos tipos de atributos. Os atributos considerados pelos autores para a desambiguação de uma palavra foram:

- a etiqueta morfológica da palavra;
- o lema da palavra;
- o tipo da palavra: se representa uma entidade ou uma lexia complexa (por exemplo, termos como "avenida principal" e "*world wide web*");
- a categoria semântica da palavra: por exemplo, "instituição financeira", "organização", "local", etc;
- o hiperônimo da palavra: extraído da *WordNet*[4];
- o synset da palavra na *WordNet*;
- etc.

Os experimentos foram conduzidos considerando vários tamanhos de contexto também, isto é, quantas palavras são consideradas à esquerda e à direita da palavra que se quer desambiguar. Os autores utilizaram contextos de tamanhos 2 e 4.

O conjunto de dados utilizados para o treinamento e teste foram os mesmos utilizados pelo programa de avaliação de sistemas de DSP chamado *Senseval*[5]. O primeiro experimento conduzido considerou somente os quatro primeiros atributos relatados acima. O segundo experimento considerou a combinação de todos os atributos. A Tabela 7 mostra o desempenho dos algoritmos de AM para o primeiro caso, enquanto a Tabela 8 mostra o erro médio dos algoritmos de AM para o segundo caso.

Tabela 7 – Desempenho dos sistemas de DSP para o primeiro caso

Técnica de AM	**Desempenho**
Árvores de decisão	70,40%
Naive-Bayes	63,42%

[4] http://www.cogsci.princeton.edu/~wn/
[5] http://www.itri.bton.ac.uk/events/senseval/

Tabela 8 – Erro médio dos sistemas de DSP para o segundo caso

Técnica de AM	**Erro médio**
Naive-Bayes	39,32%
Árvores de decisão	43,54%

Como se pode notar, no primeiro caso, as árvores de decisão se saíram melhor que o classificador *naive-Bayes*. Já no segundo caso, utilizando todos os atributos, o classificador *naive-Bayes* alcançou um erro médio menor do que o erro das árvores de decisão, ou seja, desempenhou-se melhor. Os autores verificaram que o desempenho dos sistemas de DSP melhora quando (a) se consideram mais atributos de natureza semântica (por exemplo, aqueles que utilizavam a *WordNet*), (b) utilizam-se contextos maiores ao redor da palavra que se quer desambiguar e (c) há um grande conjunto de treinamento. Ainda, de acordo com os autores, os resultados das Tabelas 7 e 8 estão muito próximos dos limites de desempenho a serem atingidos por sistemas de DSP que utilizam técnicas de AM, caso não se consiga um modo mais eficiente e robusto de se extrair conhecimento de natureza semântica de textos.

Outros trabalhos interessantes sobre DSP usando técnicas de AM são os trabalhos de Pedersen (2000) e Bruce e Wiebe (1999).

3.4. Análise de Discurso

Dentro do PLN, textos podem ser analisados em vários níveis, desde o mais baixo, o fonético-fonológico, até o mais alto, o nível da pragmática e do discurso. A Figura 2 mostra os níveis de processamento em PLN e sua relação com a complexidade e abstração do tratamento computacional. Por essa figura, pode-se perceber que, quanto mais se sobe nos níveis, mais complexo e abstrato fica o tratamento computacional. A análise do discurso encontra-se nos dois últimos níveis, isto é, entre a semântica e a pragmático e o discurso.

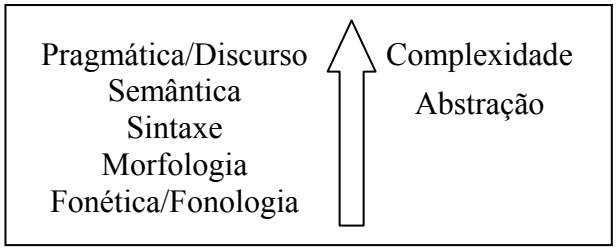

Figura 2 – Níveis de processamento em relação à complexidade e abstração do tratamento computacional

Sabe-se que, em um texto, as sentenças/cláusulas[6] não são, normalmente, entendidas de forma isolada, pois elas se relacionam às outras sentenças/cláusulas do texto, formando uma estrutura altamente elaborada que permite ao leitor do texto ler e entender seu significado. A essa estrutura, dá-se o nome de estrutura discursiva. Uma análise de discurso deve, portanto, produzir uma estrutura dessa natureza. Por exemplo, considere o trecho de texto abaixo já segmentado em 3 cláusulas:

[6] Cláusula, nesse contexto, é uma unidade mínima de significado, podendo ter qualquer estruturação interna.

"[1] Ele queria jogar tênis com Janete, [2] mas também queria jantar com Suzana. [3] Sua indecisão o deixou louco."

Nesse trecho, pode-se perceber, mesmo intuitivamente, que:
- existe uma relação de contraste entre as cláusulas 1 e 2;
- a relação de contraste entre 1 e 2 causou o fato expresso na cláusula 3.

Essas relações, como contraste e causa, são normalmente chamadas de relações discursivas. Esse tipo de conhecimento pode ser muito útil em aplicações práticas de PLN, auxiliando em tarefas de correção gramatical e estilística de textos, sumarização de textos e tradução automática, por exemplo.

Marcu e Echihabi (2002) utilizaram aprendizado bayesiano para tentar detectar automaticamente as relações discursivas, treinando e testando vários classificadores *naive-Bayes* para isso.

Em face das várias teorias discursivas existentes e seus diferentes pressupostos e relações discursivas que assumem, já que esse nível de análise é inerentemente ambíguo e difícil de se formalizar, os autores optaram por selecionar algumas relações discursivas mais genéricas para tratarem. Por exemplo, algumas teorias estabelecem várias relações de oposição possíveis entre dois trechos de texto, como "contraste", "antítese" e "concessão". Neste caso, os autores selecionaram uma relação mais genérica que cobrisse todos esses casos: a relação "contraste" somente. No geral, as relações usadas pelos autores foram:
- contraste: indica oposição entre duas cláusulas;
- explicação-evidência: uma cláusula fornece a explicação ou a evidência do conteúdo de uma outra cláusula;
- condição: o conteúdo de uma cláusula é condicionado à ocorrência do conteúdo de outra cláusula;
- elaboração: uma cláusula apresenta mais informação sobre o conteúdo de outra cláusula;
- nenhuma: quando nenhuma das relações acima se aplica.

Os autores utilizaram dois conjuntos de textos para seus experimentos, a saber: um corpus de 41.147.805 sentenças em inglês e um corpus de 1.796.386 sentenças em inglês com anotações sintáticas feitas por Charniak (2000). Para formar os conjuntos de treinamento e de teste, os autores selecionaram pares de cláusulas dos corpora e associaram suas relações discursivas.

O aprendizado se deu da seguinte forma:
- para cada par de cláusulas C1 e C2, suas palavras foram consideradas como atributos da instância para o aprendizado, enquanto que a relação discursiva R entre as cláusulas C1 e C2 foi considerada a classe, ou rótulo, da instância;
- treinou-se o classificador *naive-Bayes* com a probabilidade de todo par de palavras p1 e p2 das cláusulas C1 e C2, com $p1 \in C1$ e $p2 \in C2$, indicarem a relação R entre as cláusulas em questão, ou seja:

$$\arg\max P(R \mid P1, P2) = \arg\max \prod_{pi \in P1,\ pj \in P2} P(pi, pj \mid R) \times P(R)$$

onde P1 e P2 representam, respectivamente, o conjunto de palavras das cláusulas C1 e C2.

Os autores utilizaram tal modelagem de probabilidade por observarem que a co-ocorrência de palavras pode indicar a relação retórica. Por exemplo, no trecho "João é bom em matemática e ciências. Paulo reprovou em todas as matérias que fez.", a relação de contraste pode ser reconhecida pelas cargas semânticas opostas das palavras "bom" e "reprovou".

- com o método acima, foram treinados classificadores para (a) distinguirem entre todo possível par de relações discursivas (por exemplo, contraste *vs* explicação-evidência, contraste *vs* condição, explicação-evidência *vs* condição, etc) e (b) distinguirem todas as relações em conjunto. No primeiro caso, os classificadores foram treinados com exemplos somente dos pares de relação em questão; no segundo caso, os classificadores foram treinados com todo o conjunto. A Tabela 9 mostra o desempenho dos classificadores *naive-Bayes* para os dois casos.

Tabela 9 – Desempenho dos classificadores para identificação de relações discursivas

Relações	**Desempenho**
Pares de relações	93,00%
Todas as relações	49,70%

Os autores verificaram que ao aumentarem o conjunto de treinamento, a melhoria no desempenho dos classificadores para cada par de relações era muito pequena. Após análises, observaram que alguns pares de palavras não eram representativos para determinar as relações discursivas, o que causava a pequena melhoria de desempenho citada acima. Com isso, resolveram realizar novamente todo o processo de treinamento e teste dos classificadores somente com as palavras das cláusulas que fossem das classes substantivo ou verbo ou que fossem marcadores discursivos, já que estes podem indicar a relação discursiva entre as cláusulas que relaciona (por exemplo, o marcador discursivo "mas" é altamente indicativo da presença de uma relação discursiva de contraste). Como resultado, concluíram que o desempenho dos classificadores pode melhorar com tal técnica.

Marcu e Echihabi concluíram que sua técnica, baseada em pares de palavras, é bastante eficiente para a análise discursiva automática, dada a simplicidade e obviedade dos atributos escolhidos para o treinamento dos classificadores *naive-Bayes*. Conforme os autores, atributos mais representativos e com maior poder de predição das relações discursivas poderiam, ainda, melhorar muito o desempenho dos classificadores.

Mais detalhes sobre como utilizar técnicas de AM para análise de discurso podem ser obtidos em Marcu (2000).

3.5. Desambiguação de Conjuntos de Confusão

Conjuntos de confusão consistem em grupos de palavras muito parecidas que são comumente confundidas ou usadas no contexto errado pelas pessoas. Por exemplo, para a língua inglesa, há os conjuntos {*principle*, *principal*}, {*then*, *than*}, {*to*, *two*, *too*} e {*weather*, *whether*}. Em português, podemos citar *{seção, sessão, cessão}.* A tarefa de desambiguação de conjuntos de confusão (DCC) consiste em determinar a palavra correta a ser usada em seus contextos de ocorrências. Esse tipo

de resolução em PLN pode melhorar o desempenho de aplicações como correção gramatical e reconhecimento de fala.

Banko e Brill (2001) investigaram o uso de classificadores *naive-Bayes* para resolver esse problema, analisando, também, a influência do tamanho do conjunto de dados de treinamento no desempenho do sistema.

A forma de se aplicar AM foi removendo as palavras dos conjuntos de confusão das sentenças em que ocorriam, inserindo, em seu lugar, marcadores para indicar suas posições. A tarefa do sistema de DCC foi, assim, decidir pelo uso da palavra correta dos conjuntos de confusão em foco toda vez em que um dos marcadores era encontrado.

Os autores treinaram e testaram seus classificadores com corpora de tamanho crescente, de forma a verificar a influência do tamanho dos corpora na classificação, partindo de 500.000 palavras até 1.000.000.000 de palavras. Os atributos de cada instância do aprendizado foram:

- palavras do contexto de ocorrência do marcador (que será substituído por uma das palavras dos conjuntos de confusão em foco);
- etiquetas morfológicas das palavras do contexto.

A classe, ou rótulo, dos atributos foram as próprias palavras dos conjuntos de confusão. Como resultado, os classificadores atingiram um desempenho acima de 95%, aumentando seu desempenho conforme o tamanho do corpus de treinamento também aumentava. Com isso, os autores levantaram questões importantes sobre o que é mais eficiente, tanto em termos de tempo gasto quanto de custo financeiro e computacional: desenvolver novos algoritmos de AM ou investir na preparação dos dados para formar conjuntos de treinamento maiores.

Outros trabalhos interessantes sobre DCC usando técnicas de AM são os trabalhos de Golding (1995) e Golding e Schabes (1996).

3.6. Extração de Informação

A tarefa de extração de informação consiste em identificar e extrair de um documento determinadas informações e/ou detalhes. Normalmente, ferramentas de extração de informação são desenvolvidas para domínios específicos e muito limitados, dos quais se conhece muito bem a estrutura dos documentos, que é um requisito essencial para que muitos dos sistemas existentes tenham um bom desempenho.

Em PLN, a extração de informação é muito importante para sistemas de pergunta e resposta, onde se deve reconhecer em uma base de dados/textos a informação solicitada pelo usuário, sistemas de sumarização automática, nos casos em que o sumário a ser produzido deve ser focado em eventos e/ou conceitos descritos no texto (Mani, 2001), entre outros.

Freitag (1998a) investigou como tratar o problema de extração de informação de forma que a ferramenta desenvolvida fosse mais genérica e independente de domínios e estruturas textuais pré-determinadas. Ele concluiu que técnicas de AM seriam boas para essa tarefa, imediatamente observando as seguintes vantagens alcançadas pelo seu uso:

- menos esforço humano para desenvolver ferramentas de extração de informação, já que, com as técnicas de AM, o sistema aprende o que deve extrair de um

documento em vez de um ser humano tentar reconhecer regras (às vezes, até intuitivamente) de como reconhecer e extrair informação de um documento;
- mudar para um novo domínio de documentos não se tornaria mais uma questão de implementar novos métodos para reconhecer outras estruturas textuais, mas uma simples adaptação dos atributos que servirão de treinamento para a técnica de AM;
- fácil extensão da ferramenta de extração de informação desenvolvida.

As técnicas de AM utilizadas por Freitag (1998a, 1998b) foram SRV, Rote e o classificador naive-Bayes. Essas técnicas foram treinadas com atributos genéricos o suficiente para que não invalidasse a aplicação dos classificadores gerados em outros domínios e estruturas textuais. Isso foi feito por meio do treinamento dos classificadores com atributos que não levassem em consideração, por exemplo, informação relativa à estrutura textual em foco, suposições de que o texto fosse gramatical e que toda palavra encontrada devesse estar no léxico utilizado.

Freitag utilizou, para treinar e testar seus classificadores, um conjunto de 600 textos que tratavam de assuntos sobre "aquisição de bens materiais". Metade desse conjunto de textos foi usada para treinar os classificadores, enquanto a outra metade foi usada para testá-los. Sendo textos sobre "aquisições", foi estabelecido que as informações que deveriam ser extraídas do texto, ou seja, reconhecidas pelos classificadores, deveriam ser relativas às seguintes informações:
- entidades/nomes que participaram da aquisição: quem vendeu (referenciado, daqui em diante, pelo termo *seller*), quem comprou (*purchaser*) e o que foi adquirido (*acquired*);
- siglas ou abreviaturas das entidades/nomes que participaram da aquisição: *sellerabr*, *purchabr* e *acqabr*;
- local do bem adquirido: *acqloc*;
- o preço pago: *dlramt*;
- o estado da negociação: *status*.

Para cada instância de treinamento e teste, as classes consistiram nos termos relativos à aquisição dos bens materiais (*acquired*, *seller*, *purchaser*, etc.), com os seguintes atributos:
- se a primeira letra da palavra em foco é maiúscula;
- se a palavra em foco é um numeral;
- se a palavra em foco é um substantivo;
- a relação sintática da palavra em foco com as palavras que a cercam em seu contexto de ocorrência;
- identificadores dos *synsets* da *WordNet* aos quais a palavra em foco pertence.

Para preparação dos dados de treinamento e teste, o quarto atributo acima é obtido pelo uso de um *parser*, enquanto o último atributo é importado diretamente da *WordNet* para cada palavra consultada.

O desempenho dos classificadores é mostrado na Tabela 10 em termos das medidas de *precision* e *recall* calculadas sobre as informações de "aquisição" extraídas dos documentos. O método SRV foi treinado e testado de duas formas: (a) considerando todos os atributos listados anteriormente (representado na Tabela 10 por "SRV") e (b) considerando somente os atributos menos sofisticados, ou seja,

excluindo-se os atributos obtidos por uma análise lingüística das palavras pelo uso do *parser* e da *WordNet* (representado na Tabela 10 por "SRV+ling").

Tabela 10 – Desempenho dos classificadores para extração de informação (Freitag 1998a)

	Precision	***Recall***	***Precision***	***Recall***	***Precision***	***Recall***
Classificador	***acquired***		***purchaser***		***seller***	
Rote	59,6	18,5	43,2	23,2	38,5	15,2
Naive-Bayes	19,8	100	36,9	100	15,6	100
SRV	38,4	96,6	42,9	97,9	16,3	86,4
SRV+ling	38,0	95,6	42,4	96,3	16,4	82,7
	acqabr		***purchabr***		***sellerabr***	
Rote	16,1	42,5	3,6	41,9	2,7	27,3
Naive-Bayes	23,2	100	39,6	100	16,0	100
SRV	31,8	99,8	41,4	99,6	14,3	95,1
SRV+ling	35,5	99,2	43,2	99,3	14,7	91,8
	acqloc		***status***		***dlramt***	
Rote	6,4	63,1	42,0	94,5	63,2	48,5
Naive-Bayes	7,0	100	33,3	100	24,1	100
SRV	12,7	83,7	39,1	89,8	50,5	91,0
SRV+ling	15,4	80,2	41,5	87,9	52,1	89,4

Freitag conclui que, no geral, todos os classificadores tiveram desempenho parecido. Com isso, constatou que o uso de atributos resultantes de análise lingüística (provenientes do *parser* e da *WordNet* utilizados) não aumentou suficientemente o desempenho do classificador SRV para justificar a inclusão desses atributos nas instâncias de aprendizado, já que os atributos mais simples forneceram informação suficiente aos classificadores.

Outros trabalhos interessantes sobre extração de informação pelo uso de técnicas de AM são os trabalhos de Soderland (1996) e Califf e Mooney (1997).

3.7. Análise Sintática

Análise sintática, ou *parsing*, consiste em identificar a função sintática que os termos exercem numa sentença. Sistemas que realizam a análise sintática automática de sentenças, chamados *parsers*, são essenciais para muitas aplicações de PLN, como sumarização automática, interpretação, tradução automática, correção gramatical, entre outras.

Há várias abordagens possíveis para se desenvolver um *parser* (Allen, 1987). Uma delas diz respeito à forma como se analisa uma sentença: (a) podem-se achar regras gramaticais que gerem a sentença em questão (método chamado *top-down*, ou seja, parte-se de um símbolo inicial de uma gramática até se produzir as palavras da sentença); (b) partindo-se das palavras da sentença, procuram-se estruturas sintáticas que relacionem as palavras, de forma que, no final, as regras gramaticais não sejam violadas (método chamado *bottom-up*, ou seja, parte-se das palavras da sentença até se produzir um símbolo inicial da gramática); (c) levam-se em consideração tanto as palavras da sentença quanto as regras gramaticais para se construir a estrutura sintática da sentença (método chamado *mixed*). Um *parser* também pode ser classificado como determinístico ou probabilístico: um *parser* determinístico utiliza

as primeiras regras gramaticais possíveis de serem aplicadas à sentença para produzir sua estrutura sintática, produzindo, assim, somente uma estrutura sintática para a sentença; um *parser* probabilístico, por outro lado, constrói todas as estruturas sintáticas possíveis para uma sentença pela utilização de todas as regras gramaticais possíveis, associando a cada estrutura sua probabilidade de ser a estrutura sintática adequada para a sentença em questão.

As regras gramaticais de um *parser* podem ser desenvolvidas das seguintes formas: (a) pela intuição de um lingüista computacional; (b) com base em estudos de fenômenos lingüísticos ocorrentes em corpora; (c) de forma híbrida, isto é, as regras são desenvolvidas com base em estudos de corpora e depois avaliadas por um lingüista computacional. De qualquer forma, o desenvolvimento de regras gramaticais para um *parser* é complexo, dado que vários fatores devem ser observados, por exemplo:

- as regras gramaticais não devem ser muito restritivas a ponto de não serem aplicadas a sentenças que seguem seus padrões sintáticos;
- as regras gramaticais não devem ser muito genéricas a ponto de serem aplicadas a sentenças que não seguem seus padrões sintáticos;
- as regras gramaticais devem abranger tantos fenômenos lingüísticos quanto possível.

Delisle et al. (1998) investigaram o uso de técnicas de AM para auxiliar o processo de *parsing*, em particular, a escolha de regras gramaticais adequadas para serem aplicadas às sentenças, já que escolhas ruins podem levar o *parser* a produzir análises sintáticas erradas e/ou aumentar o tempo de processamento necessário. O *parser* utilizado pelos autores, chamado DIPPET, possui as características de ser *top-down* e determinístico, com suas regras gramaticais desenvolvidas de forma híbrida. Com o uso de técnicas de AM, os autores buscavam uma adaptação mais fácil do *parser* para outros domínios e o menor esforço humano para o desenvolvimento das regras gramaticais, além de um melhor desempenho do *parser*.

O problema enfocado pelos autores foi a questão da escolha da regra gramatical inicial para analisar sintaticamente uma sentença, que, no DIPPET, são de dois tipos, a saber: (a) uma regra para analisar sentenças simples, sem orações subordinadas (referenciada, daqui em diante, por NON_COMPOUND) e (b) uma regra para analisar sentenças complexas, com coordenações (referenciada por COMPOUND). Desta forma, dada uma nova sentença para ser analisada, os autores desejavam que os algoritmos de AM informassem se tal sentença deveria ser analisada por uma regra NON_COMPOUND ou COMPOUND. Sob esse enfoque, as seguintes questões tiveram que ser analisadas:

- que atributos seriam representativos o suficiente de uma sentença para permitirem a determinação da regra gramatical adequada a ser aplicada;
- como incorporar o que foi aprendido ao DIPPET;
- como avaliar o "novo" DIPPET.

Os autores decidiram por utilizar os seguintes atributos para representar uma instância de aprendizado referente a uma sentença[7]:

- número de palavras da sentença;

[7] Os autores também aplicaram a técnica PCA (*Principal Component Analysis*) (Johnson and Wichern, 1992) aos atributos, mas não conseguiram resultados bons. Segundo os autores, isso ocorreu devido ao fato dos atributos já serem altamente não redundantes.

- número de palavras que podem ser verbos na sentença;
- número de palavras que podem ser verbos auxiliares na sentença;
- a distância entre os verbos de uma sentença, caso mais de um verbo ocorra na sentença;
- número de palavras que podem ser palavras coordenativas na sentença;
- número de fragmentos sintáticos encontrados na sentença;
- número de fragmentos sintáticos verbais encontrados na sentença;
- número de palavras não consideradas durante a detecção dos fragmentos sintáticos da sentença;
- porcentagem da sentença reconhecida durante a detecção dos fragmentos sintáticos;

onde os quatro últimos atributos são obtidos por uma versão do DIPPET que realiza uma análise sintática parcial, identificando os chamados "fragmentos sintáticos" em uma sentença, por exemplo, sujeito, verbo e adjuntos adverbiais. As possíveis classes são as identificações das regras gramaticais a serem aplicadas, ou seja, NON_COMPOUND ou COMPOUND.

Foram várias as técnicas de AM utilizadas pelos autores, a saber: regras de decisão, aprendizado baseado em instâncias, redes neurais e o classificador *naive-Bayes*. O conjunto de dados para treinamento e teste consistiu de 111 sentenças (55 da classe COMPOUND e 56 da classe NON_COMPOUND) extraídas de manuais de usuários para softwares, de um guia de impostos, de um livro sobre fenômenos climáticos e do Brown Corpus[8]. Dois terços do conjunto de sentenças foram reservados para o treinamento; o restante foi reservado para o teste dos classificadores gerados. As taxas de erro obtidas são mostradas na Tabela 11.

Tabela 11 – Desempenho dos classificadores para o reconhecimento da regra gramatical adequada (Delisle et al., 1998)

Classificador	**Taxa de Erro**
Com base em instâncias	10,8%
Rede neural	13,5%
Naive-Bayes	16,2%
Regras de decisão	18,9%

Como o DIPPET faz uso de regras gramaticais para realizar a análise sintática, as regras de decisão aprendidas durante a aplicação do algoritmo de AM foram inseridas no DIPPET no lugar das antigas regras. Antes, entretanto, os autores selecionaram as regras de decisão que seriam inseridas no DIPPET por meio dos seguintes critérios:

- taxa de erro estimada das regras;
- o quanto cada regra parecia fazer sentido para os lingüistas computacionais;
- a legibilidade das regras, de forma que regras mais simples fossem preferidas;
- a novidade das regras, isto é, se apresentavam conhecimento inédito.

As regras de decisão aprendidas foram implementadas no DIPPET de três formas:

1) implementaram-se somente as regras aprendidas para a classe COMPOUND. Caso essas regras não se aplicassem, a classe NON_COMPOUND seria então

[8] http://www.hit.uib.no/icame/brown/bcm.html

assumida como correta. Essa implementação será referenciada por C-imp daqui em diante.

2) implementaram-se somente as regras aprendidas para a classe NON_COMPOUND. Caso essas regras não se aplicassem, a classe COMPOUND seria então assumida como correta. Essa implementação será referenciada por NC-imp daqui em diante.
3) as implementações acima foram combinadas. Essa implementação será referenciada por NC_C-imp daqui em diante. Para uma nova sentença, a classe resultante dessa implementação é dada pelas combinações da Tabela 12.
   Para a primeira linha da tabela, por exemplo, caso C-imp indique a classe COMPOUND e NC_imp também a indique, NC_C-imp indicará a classe COMPOUND. Pode-se perceber que a classe NON_COMPOUND é escolhida quando as duas primeiras implementações discordam, já que a segunda classe ocorre com mais freqüência nos corpora utilizados.

Tabela 12 – Combinação das implementações das regras de decisão (Delisle et al. 1998)

C-imp	**NC-imp**	**NC_C-imp**
COMPOUND	COMPOUND	COMPOUND
NON_COMPOUND	NON_COMPOUND	NON_COMPOUND
NON_COMPOUND	COMPOUND	NON_COMPOUND
COMPOUND	NON_COMPOUND	NON_COMPOUND

As taxas de erro do DIPPET com as antigas e as novas regras são mostradas na Tabela 13. Como se pode notar, todas as implementações das regras de decisão tiveram desempenho melhor que as antigas regras.

Tabela 13 – Desempenho do DIPPET (Delisle et al. 1998)

Versão	**Taxa de Erro**
Regras originais	25,26%
C-imp	20,52%
NC-imp	22,10%
NC_C-imp	16,31%

Os autores concluíram que técnicas de AM podem melhorar muito o desempenho de *parsers*, auxiliando na escolha de regras gramaticais a serem aplicadas a uma nova sentença. Dessa forma, é possível minimizar o tempo e o processamento necessário para a realização da análise sintática, já que se indicam de antemão que regras gramaticais utilizar, evitando, assim, a busca exaustiva no espaço de regras possíveis.

Outros trabalhos interessantes que utilizam técnicas de AM no processo de *parsing* são os trabalhos de Samuelsson (1994), Soderland (1996), Collins (1999) e Bonfante & Nunes (2001), este último envolvendo o português do Brasil.

3.8. Etiquetação Morfológica

A tarefa de etiquetação (ou anotação, ou marcação) morfológica, também chamada *tagging*, consiste, basicamente, em associar às palavras de uma sentença suas classes gramaticais (etiquetas ou, no inglês, *tags*). Sistemas que realizam

automaticamente esse tipo de etiquetação são chamados de *taggers* e são essenciais para muitas aplicações de PLN, como *parsing*, e também para a área de lingüística de corpus.

Atualmente, principalmente para o inglês, os *taggers* atingiram um desempenho quase tão bom quanto o humano. Entre os erros que um *tagger* ainda comete, destacam-se os erros derivados da presença de dados esparsos para o desenvolvimento do *tagger*, isto é, como um *tagger* é desenvolvido a partir de corpora etiquetados (utilizando medidas estatísticas e empíricas com base na ocorrência das palavras e suas classes gramaticais), ele consegue etiquetar "bem" as palavras com as quais ele foi treinado. Quando palavras desconhecidas são apresentadas ao *tagger*, este pode cometer erros ao etiquetá-las. Portanto, grandes esforços têm sido despendidos na tentativa de etiquetar corpora enormes que sejam representativos da língua para treinamento dos *taggers*, de forma que estes consigam atingir desempenho máximo. A etiquetação de corpora para treinamento dos *taggers*, entretanto, é uma tarefa extremamente custosa para humanos, os quais sempre cometem erros, tornando necessário, ainda, um processo de revisão da etiquetação feita.

Eskin (2000), visando auxiliar os processos de etiquetação e revisão comentados acima, desenvolveu um método de detecção de erros na etiquetação morfológica utilizando técnicas de AM, dentre elas, o classificador *naive-Bayes*.

O método de detecção de erro utilizado é derivado de um método de detecção de anomalias em dados estatísticos (Barnett and Lewis, 1994). Nesse contexto, o classificador *naive-Bayes* é usado para estimar os parâmetros do método de detecção de anomalias. Com esse método, consegue-se estimar as possíveis palavras etiquetadas erroneamente em um corpus já etiquetado (por humanos ou mesmo por um *tagger*) e sugerir correções quando necessário. Além da motivação de auxiliar o processo de etiquetação e revisão, como dito anteriormente, Eskin ainda ressalta que as anomalias encontradas pelo método de detecção podem consistir em fenômenos interessantes da língua que merecem ser estudados com maior profundidade.

Com relação ao aprendizado bayesiano, a coleção de dados utilizada para treinar e testar o classificador *naive-Bayes* foi o corpus anotado do Penn Treebank (Marcus et al., 1993) com, aproximadamente, 1,25 bilhões de palavras etiquetadas manualmente. Para cada instância de aprendizado, que consiste em uma palavra etiquetada e seu contexto, os atributos usados foram:

- a própria palavra;
- a etiqueta da palavra anterior à palavra em foco;
- a etiqueta da palavra posterior à palavra em foco.

As possíveis classes, neste caso, foram todas as possíveis etiquetas do conjunto de etiquetas utilizado (*tagset*).

Ao final do processo de classificação, as probabilidades de uma palavra pertencer às classes gramaticais do *tagset* são utilizadas para o cálculo de detecção de anomalias. Caso esse processo indique que a classe gramatical associada à palavra em foco seja uma anomalia, utiliza-se esse mesmo processo, com as probabilidades da palavra pertencer às outras classes gramaticais, para determinar a possível classe gramatical correta da palavra.

Eskin, em sua avaliação, pediu que lingüistas computacionais julgassem os resultados da aplicação do método ao corpus anotado do Penn Treebank, associando às anomalias identificadas pelo método uma das seguintes categorias:

- erro no corpus: a etiquetação da palavra no corpus realmente estava errada, ou seja, o método fez uma indicação correta;
- erro do método: a etiquetação da palavra no corpus estava correta, ou seja, o método a identificou erradamente como anomalia;
- não há certeza: quando o lingüista computacional não sabia se a etiquetação da palavra no corpus estava errada ou não.

Além disso, para os casos em que havia erro no corpus, foi julgado se a sugestão de etiqueta feita pelo método estava correta. Como resultado, foi verificado que (a) 44% dos casos indicados pelo método como anômalos eram realmente anômalos (sem computar as palavras em que os lingüistas computacionais avaliaram como não havendo certeza sobre a etiquetação) e (b) caso fossem aceitas as sugestões de etiquetação feitas pelo método, 91% dos erros do corpus seriam corrigidos.

Eskin concluiu que seu método pode ser muito eficaz para o propósito a que foi desenvolvido, mas também destacou limitações, principalmente o fato de o método não ser capaz de reconhecer etiquetações ambíguas, podendo indicar como anomalia uma etiquetação correta.

Outros trabalhos interessantes sobre etiquetação morfológica com uso de técnicas de AM são os trabalhos de Hajič (2000), Roth e Zelenko (1998) e Aires et al. (2000), este último para o português do Brasil.

A próxima seção conclui este relatório, fazendo algumas considerações finais sobre aplicações de PLN que utilizam técnicas de PLN.

4. Conclusões e Considerações Finais

As técnicas de AM têm sido cada vez mais usadas para resolver todos os tipos de problemas da computação. São vários os motivos pelo seu uso, destacando-se sua maior flexibilidade, adaptabilidade e bons resultados que têm gerado.

Neste relatório, foi feita uma breve introdução ao AM estatístico, mais especificamente, o classificador *naive-Bayes*, sendo realizada, a seguir, uma revisão de aplicações recentes em PLN que fazem uso de tal técnica, buscando mostrar o estado da arte desse ramo de pesquisa.

Não somente o classificador *naive-Bayes* é bastante utilizado em PLN, podendo-se destacar, também, as árvores de decisão e as redes neurais. Entretanto, este relatório dedicou-se somente ao classificador *naive-Bayes* pelos seguintes fatos: (a) o aprendizado bayesiano é uma abordagem bastante simples e robusta, (b) tem sido amplamente usado e (c) tem conseguido bons resultados quando aplicado a processamento de textos, principalmente.

Referências

Allen, J. (1987). *Natural Language Understanding*. The Benjamim/Cummings Publ. Co., Inc., USA.

Aires, R. V. X.; Aluísio, S. M.; Kuhn, D. C. S.; Andreeta, M. L. B.; Oliveira Jr., O. N. (2000). Combining Multiple Classifiers to Improve Part of Speech Tagging: A Case Study for Brazilian Portuguese. In *Proceedings of the XVII Brazilian Symposium of Artificial Intelligence - SBIA'2000.* Atibaia, SP, November, 20-22.

Banko, M. and Brill, E. (2001). Scaling to Very Very Large Corpora for Natural Language Disambiguation. In *Proceedings of the 39th Annual Meeting of the Association for Computational Linguistics* – ACL'01.

Barnett, V. and Lewis, T. (1994). *Outliers in Statistical Data*. John Wiley and Sons.

Bonfante, A.G. and Nunes, M.G.V. (2001). The Implementation Process of a Statistical Parser for Brazilian Portuguese. In *Proceedings of the Seventh International Workshop on Parsing Technologies* - IWPT'01, pp. 217-221. Beijing, China.

Busemann, S.; Schmeier, S.; Arens, R.G. (2000). Message Classification in the Call Center. In *Proceedings of the 6th Applied Natural Language Processing Conference* – ANLP'00, pp. 158-165.

Califf, M.E. and Mooney, R.J. (1997). Relational Learning of Pattern-Match Rules for Information Extraction. In *Working Papers of the Association for Computational Linguistics Workshop on Natural Language Learning*.

Charniak, E. (2000). A maximum-entropy-inspired parser. In *Proceedings of the First Annual Meeting of the North American Chapter of the Association for Computational Linguistics* – NAACL'00, pp. 132-139. Seattle, Washington.

Collins, M.J. (1999). *Head-Driven Statistical Models for Natural Language Parsing.* PhD Thesis, University of Pennsylvania. Bruce, R.F. and Wiebe. J.M. (1999). Decomposable Modeling in Natural Language Processing. *Computational Linguistics*, Vol. 25, No. 2, pp. 195-207.

Corrêa, R.F. and Ludermir, T.B. (2002). Categorização Automática de Documentos: Estudo de Caso. In *Anais do Simpósio Brasileiro de Redes Neurais* – SBRN'02. Porto de Galinhas/Recife – PE, Brazil.

Cucchiarelli, A. and Velardi, P. (2002). Feature-Based WSD: Why We Are at a Dead-End. In *Proceedings of the Portugal for Natural Language Processing* – PorTAL, pp. 5-14. Faro, Portugal.

Delisle, S.; Létourneau, S; Matwin, S. (1998). Experiments with Learning Parsing Heuristics. In *Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics* – ACL'98, pp. 307-313.

Eskin, E. (2000). Detecting Errors within a Corpus using Anomaly Detection. In *Proceedings of the 6th Applied Natural Language Processing Conference* – ANLP'00, pp. 148-153.

Freitag, D. (1998a). Toward General-Purpose Learning for Information Extraction. In *Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics* – ACL'98, pp. 404-408.

Freitag, D. (1998b). Information Extraction from HTML: Application of a General Machine Learning Approach. In *Proceedings of the 15th National Conference on Artificial Intelligence* – AAAI'98.

Golding, A.R. (1995). A Bayesian Hybrid Method for Context-Sensitive Spelling Correction. In *Proceedings of the 3rd Workshopon Very Large Corpora*.

Golding, A.R. and Schabes, Y. (1996). Combining Trigram-Based and Feature-Based Methods for Context-Sensitive Spelling Correction. In *Proceedings of the 34th Annual Meeting of the Association for Computational Linguistics* – ACL'96, pp. 71-78.

Hajič, J. (2000). Morphological Tagging: Data vs. Dictionaries. In *Proceedings of the 6th Applied Natural Language Processing Conference* – ANLP'00, pp. 94-101.

Harman, D. (1994). Data Preparation. In R. Merchant (ed.), *Proceedings of the TIPSTER Text Program Phase I*. Morgan Kaufmann Publishing Co.

Johnson, R.A. and Wichern, D.W. (1992). *Applied Multivariate Statistical Analysis*. Prentice Hall.

Kupiec, J.; Pedersen, J.; Chen, F. (1995). A trainable document summarizer. *ACM SIGIR*, pp. 68-73.

Larocca Neto, J. (2002). *Contribuição ao Estudo de Técnicas para Sumarização Automática de Textos.* Dissertação de Mestrado. Pontifícia Universidade Católica do Paraná, Brasil.

Larocca Neto, J.; Freitas, A.A.; Kaestner, C.A.A. (2002). Automatic Text Summarization Using a Machine Learning Approach. In *Anais do Simpósio Brasileiro de Inteligência Artificial* – SBIA'02, pp. 205-215. Porto de Galinhas/Recife – PE, Brazil.

Larocca Neto, J.; Santos, A.D.; Kaestner, C.A.A.; Freitas, A.A. (2000). Document Clustering and Text Summarization. In *Proceedings of the 4th International Conference on Practical Applications of Knowledge Discovery and Data Mining* – PADD'00, pp. 41-55.

Mani, I. and Bloedorn E. (1998). Machine Learning of Generic and User-Focused Summarization. In *Proceedings of the Fifteenth National Conference on Artificial Intelligence* – AAAI'98, pp. 821-826.

Marcu, D. (2000). *The Theory and Practice of Discourse Parsing and Summarization*. The MIT Press. Cambridge, Massachusetts.

Marcu, D. and Echihabi, A. (2002). An Unsupervised Approach to Recognizing Discourse Relations. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics* – ACL'02, pp. 368-375. Philadelphia.

Marcus, M; Santorini, B.; Marcinkiewicz, M.A. (1993). Building a Large Annotated Corpus of English: The Penn Treebank. *Computational Linguistics*, Vol. 19, No. 2, pp. 313-330.

Mitchell, T.M. (1997). *Machine Learning*. McGraw Hill, New York.

Pedersen, T. (2000). A Simple Approach to Building Ensembles of Naive Bayesian Classifiers for Word Sense Disambiguation. In *Proceedings of the 1st Meeting of the North American Chapter of the Association for Computational Linguistics* – NAACL'00, pp. 63-69.

Roth, D. and Zelenko, D. (1998). Part of Speech Tagging Using a Network of Linear Separators. In *Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics* – ACL'98, pp.1136-1142.

Salton, G. (1988). *Automatic Text Processing*. Reading, MA: Addison-Wesley.

Samuelsson, C. (1994). Grammar Specialization Through Entropy Thresholds. In *Proceedings of the 32th Annual Meeting of the Association for Computational Linguistics* – ACL'94, pp. 188-195.

Soderland, S.G. (1996). *Learning Texts Analysis Rules for Domain-Specific Natural Language Processing*. PhD Thesis. Department of Computer Science, University of Massachusets.

Yang, Y. and Liu, X. (1999). A Re-examination of Text Categorization Methods. In *Proceedings of the ACM SIGIR Conference on Research and Developmento in Information Retrieval* – SIGIR'99, pp. 42-49.