UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
INSTITUTO DE INFORMÁTICA
MESTRADO EM COMPUTAÇÃO

PAULO SCHREINER

Alinhamento Léxico Utilizando Técnicas Híbridas Discriminativas e de Pós-Processamento

Dissertação apresentada como requisito parcial
para a obtenção do grau de
Mestre em Ciência da Computação

Profª. Drª. Aline Villavicencio
Orientador

Profª. Drª. Helena de Medeiros Caseli
Co-orientador

Porto Alegre, abril de 2010

CIP – CATALOGAÇÃO NA PUBLICAÇÃO

Schreiner, Paulo

Alinhamento Léxico Utilizando Técnicas Híbridas Discriminativas e de Pós-Processamento / Paulo Schreiner. – Porto Alegre: PPGC da UFRGS, 2010.

66 f.: il.

Dissertação (mestrado) – Universidade Federal do Rio Grande do Sul. Mestrado em Computação, Porto Alegre, BR–RS, 2010. Orientador: Aline Villavicencio; Co-orientador: Helena de Medeiros Caseli.

1. Processamento de linguagem natural, aprendizado de máquina, alinhamento léxico, corpora paralelo, expressões multipalavra. 2. UFRGS. I. Villavicencio, Aline. II. Caseli, Helena de Medeiros. III. Título.

SUMÁRIO

LISTA DE ABREVIATURAS E SIGLAS

AER	*Alignment Error Rate* (taxa de erros de alinhamento)
CALL	*Computer-Assisted Language Learning* (aprendizado de línguas auxiliado por computador)
CVP	Construção verbo-partícula
EBMT	*Example Based Machine Translation* (tradução de máquina baseada em exemplos)
EMP	Expressão multi-palavra
EMS	Etiquetador morfossintático
FF	*Feature Function*
LLR	Log-likelihood Ratio
MAE	Método de associação estatística
OCR	*Optical Character Recognition* (reconhecimento ótico de caracteres)
PLN	Processamento de Linguagem Natural
POS	*Part-of-speech* (etiqueta morfossintática)
SN	Sintagma Nominal
SVM	*Support Vector Machine* (Máquina de Vetores de Suporte)

LISTA DE SÍMBOLOS

α	Fator de *trade-off* entre precisão e revocação no escore-f
ϵ	Constante pequena
δ	Delta de Kronecker
θ	Parâmetros de uma *feature function*
κ	Coeficiente kappa de concordância entre anotadores
λ_i	Peso da *feature* i

LISTA DE FIGURAS

LISTA DE TABELAS

RESUMO

O alinhamento léxico automático é uma tarefa essencial para as técnicas de tradução de máquina empíricas modernas. A abordagem gerativa não-supervisionado têm sido substituída recentemente por uma abordagem discriminativa supervisionada que facilite inclusão de conhecimento linguístico de uma diversidade de fontes. Dentro deste contexto, este trabalho descreve uma série alinhadores léxicos discriminativos que incorporam heurísticas de pós-processamento com o objetivo de melhorar o desempenho dos mesmos para expressões multi-palavra, que constituem um dos desafios da área de processamento de linguagens naturais atualmente. A avaliação é realizada utilizando um *gold-standard* obtido a partir da anotação de um corpus paralelo de legendas de filmes. Os alinhadores propostos apresentam um desempenho superior tanto ao obtido por uma *baseline* quanto ao obtido por um alinhador gerativo do estado-da-arte (Giza++), tanto no caso geral quanto para as expressões foco do trabalho.

Palavras-chave: Processamento de linguagem natural, aprendizado de máquina, alinhamento léxico, corpora paralelo, expressões multi-palavra, UFRGS.

Text alignment

ABSTRACT

Lexical alignment is an essential task for modern empirical machine translation techniques. The unsupervised generative approach is being replaced by a supervised, discriminative one that considerably facilitates the inclusion of linguistic knowledge from several sources. Given this context, the present work describes a series of discriminative lexical aligners that incorporate post-processing heuristics with the goal of improving the quality of the alignments of multiword expressions, which is one of the major challanges in natural language processing today. The evaluation is conducted using a gold-standard obtained from a movie subtitle parallel corpus. The aligners proposed show an alignment quality that is superior both to our baseline and to a state-of-the-art generative aligner (Giza++), for the general case as well as for the expressions that are the focus of this work.

Keywords: natural language processing, lexical alignment, machine learning, parallel corpora, multiword expressions, UFRGS.

1 INTRODUÇÃO

O Processamento de Linguagens Naturais (PLN) é uma área de computação que estuda problemas de geração e compreensão automática de línguas humanas naturais. Ao contrário das linguagens desenvolvidas especificamente para o computador, em um idioma humano uma frase normalmente contém ambiguidades, nuances e interpretações que dependem de contexto, conhecimento de mundo, cultura, entre outros (JURAFSKY; MARTIN, 2000).

A tarefa de tradução de máquina é a mais antiga da área, e continua sendo de grande relevância atualmente. Sua importância é cada vez mais evidente quando consideramos comunidades multilíngues (tanto países, como Suíça e Canadá, como entidades multinacionais como a Comunidade Europeia), ou a disponibilização de conteúdo nas mais diversas línguas na Internet. Mesmo que ainda estejamos longe de substituir completamente os tradutores humanos, alguns métodos empíricos modernos de tradução de máquina têm obtido resultados promissores (CHIANG, 2007).

A ideia dos métodos empíricos é utilizar-se somente de grandes corpora[1] de textos traduzidos para aprender automaticamente as regras de tradução, em oposição aos métodos mais tradicionais que consistem de regras criadas por especialistas. Um exemplo de método empírico é a **tradução baseado em exemplos** (SOMERS, 1999), cuja ideia básica é reutilizar traduções existentes como base para a nova tradução. Um sistema desse tipo trabalha normalmente com três etapas: identificar exemplos parecidos, estabelecer as correspondências que serão reusadas, e recombinar os fragmentos gerados na língua alvo. Podemos também citar a **tradução de máquina estatística** (BROWN et al., 1990). Similarmente, a ideia é aprender modelos de tradução a partir de exemplos. No entanto, nos métodos estatísticos, a tarefa de tradução é modelada probabilisticamente, ou seja, para traduzir uma sentença a é preciso encontrar, dentre todas as traduções possíveis b, aquela tradução com a maior probabilidade de acordo com o modelo.

Os métodos empíricos têm em comum o fato de que necessitam, para o seu treinamento, grandes quantidades de *exemplos*, que são sentenças traduzidas. Ou seja, precisam do que se chama de um **corpus paralelo** (MIHALCEA; SIMARD, 2006), de preferência de larga escala. Além disso, esse corpus precisa estar **alinhado lexicalmente**, ou seja, ele deve possuir marcadas quais palavras ou expressões em uma língua correspondem a quais palavras ou expressões na outra língua. Um exemplo de alinhamento léxico entre duas sentenças pode ser visto na figura 1.1, e uma definição mais detalhada está contida na seção 2.1. Quanto maior for esse corpus alinhado lexicalmente, e quanto melhor for este alinhamento, melhores condições o método de tradução de máquina terá de inferir suas regras ou modelos de tradução. Esses alinhamentos não pode ser realizado manual-

[1] Um corpus é um conjunto de textos que serve como base de análise linguística

mente, devido ao seu custo proibitivo para corpora de grandes dimensões, dando origem aos **alinhadores léxicos**.

		1	2	3	4	5	6	7	8
		Life	's	not	fair	,	is	it	?
1	A		•						
2	vida			•					
3	é				•				
4	injusta				•				
5	,					•			
6	não						•	•	
7	é						•	•	
8	?								•

Figura 1.1: Exemplo de alinhamento

Um alinhador léxico é um programa que tem como entrada um corpus paralelo, normalmente já alinhado quanto as suas frases. A saída do programa é o mesmo corpus paralelo, mas com as correspondências **léxicas** (palavras, ou expressões multi-palavra) marcadas. Esse problema está longe de ser trivial pois, dados um corpus paralelo alinhado por sentença, há várias divergências que podem ocorrer: correspondências entre palavras de expressões idiomáticas, traduções livres, *function words* faltantes, palavras com mais de uma tradução possível, diferenças na estrutura morfológica das línguas, tornam a tarefa bastante complexa.

Uma das primeiras tentativas de lidar com essas complexidades do alinhamento léxico foi a abordagem proposta por Brown et al. (1993), que introduzem 5 modelos do processo de tradução, conhecidos como modelos IBM. O modelo IBM 1 é o mais simples e ignora fatores como a ordem das palavras e fertilidade[2]. Basicamente, a probabilidade de uma palavra t na tradução ser a correspondente de uma palavra o na sentença original depende somente da correlação destas duas palavras no corpus. Os modelos mais avançados (IBM 3, 4, etc) incorporam na sua modelagem a posição das palavras e a **fertilidade** delas, o que permite, em princípio, modelar traduções $m : n$. Tendo um modelo do processo de tradução e um corpus de sentenças traduzidas, resta estimar os parâmetros do modelo. Isso pode (mas não precisa) ser feito utilizando o algoritmo *Expectation-Maximization*, para obter a estimativa de máxima verossimilhança. Essa é a chamada **abordagem gerativa** do problema. Entretanto, elaborar modelos mais complexos, que capturem mais informações linguísticas, pode ser difícil, e estimar os parâmetros destes modelos muitas vezes acaba tendo custo computacional proibitivo.

Por esse motivo, recentemente cresceu o foco das pesquisas em uma **abordagem discriminativa** (MOORE, 2005). Alinhadores discriminativos se utilizam de técnicas de aprendizado de máquina e heurísticas linguisticamente informadas para gerar os alinhamentos. Por exemplo, Liu et al. (2005) utilizam as probabilidades de alinhamento do modelo IBM 3, correspondência de etiqueta morfossintática e ocorrência em dicionários bilíngues como *features*, combinadas de maneira log-linear.

Apesar da pesquisa abundante na área, há muitas oportunidades para melhora nos métodos de alinhamento léxico automático. O surgimento de parsers de alta qualidade para o

[2]Fertilidade é a propriedade de uma palavra se ligar a 0, 1, ou mais palavras na tradução

inglês (como o RASP (BRISCOE; CARROLL; WATSON, 2006)) e em menor escala para o Português (Palavras (BICK, 2000)) oferecem uma abundância de conhecimento linguístico para se experimentar. Ademais, poucos experimentos foram realizados utilizando o Português como uma das línguas. Um dos poucos trabalhos que abordam a língua portuguesa é o alinhador LIHLA (CASELI; NUNES; FORCADA, 2005), que foi parte do projeto Retratos (CASELI, 2007). O objetivo deste projeto foi a obtenção de recursos necessários à criação de tradutores de máquina (léxicos bilíngues e regras de tradução). Para português-espanhol o LIHLA conseguiu desempenho superior ao do Giza++[3], no entanto para português-inglês o alinhador obteve um desempenho próximo, porém inferior (por isso o Giza foi usado no projeto para criação do corpus português-inglês alinhado lexicalmente). Além disso, até recentemente, muitos estudos utilizaram como métrica de avaliação o *Alignment Error Rate* (AER), que não está correlacionado com a qualidade da tradução gerada e, no geral, não é muito útil, como apontado por Fraser e Marcu (FRASER; MARCU, 2007a), que também sugerem que a revocação é mais importante do que a precisão para a qualidade de tradução gerada a partir de um alinhamento.

Dentre as complexidades que devem ser tratadas na tarefa de alinhamento léxico estão as **expressões multi-palavra** (EMPs) (SAG et al., 2002) que constituem um dos principais desafios para o PLN atualmente, tais como substantivos compostos (*homem-bomba*), verbos frasais (no inglês *make up*, com o sentido de inventar) e expressões idiomáticas (*bater as botas*). É estimado que o número de expressões multi-palavra conhecidas por um falante é da mesma ordem de grandeza do número de expressões simples (JACKENDOFF, 1997), portanto, para modelar com precisão linguagens naturais, aplicações de PLN precisam tratar dessas expressões de forma apropriada.

Dada a sua natureza heterogênea e flexível essas expressões multi-palavra causam problemas significativos a sistemas de PLN (SAG et al., 2002), de modo que métodos específicos para tratá-las têm sido desenvolvidos.

As expressões multi-palavra também causam problemas na área de tradução de máquina, gerando traduções incorretas. Por exemplo (Babelfish [4]):

> *The oil tanker sank in the Pacific.*
> O *tanker de óleo afundou-se no Pacífico.
>
> *The employees disliked the new policy set up by the company.*
> Os empregados não gostaram da política nova *ajustada acima pela companhia.

As expressões multi-palavra também constituem um desafio para os métodos de alinhamento léxico, com a maioria estabelecendo restrições a alinhamentos $1:1$ ou $1:n$ ou, quando permitem alinhamentos $m:n$, não realizando nenhum tratamento especial para encontrar EMPs (FRASER; MARCU, 2006). Desta forma, este trabalho tem como objetivo desenvolver técnicas para melhorar o desempenho de um alinhador léxico com respeito a um tipo específico de EMP, as *verb-particle constructs*, ou construções verbo-partícula (CVPs), que são combinações entre um verbo e uma partícula em inglês, tais como *turn up* (aparecer) e *made up* (inventou).

Este trabalho propõe uma nova abordagem para o alinhamento léxico para a inclusão de conhecimento linguístico de uma maneira direta, através de heurísticas de pós-processamento executadas após um alinhamento preliminar realizado por um alinhador

[3] O Giza++ é um programa que implementa os modelos IBM, descrito em mais detalhes na seção 3.3.1
[4] http://babelfish.altavista.com/tr, acessado em 13/2/2010

discriminativo. Como resultado uma melhoria significativa no alinhamento destas expressões foi obtida como apresentado neste documento.

Dado esse contexto, o presente trabalho está organizado da seguinte forma: no capítulo 2 descrevemos os trabalhos relacionados, começando com a definição do problema na seção 2.1 seguindo com uma descrição de alinhadores gerativos na seção 2.2, técnicas de simetrização de alinhamentos na seção 2.3, métodos discriminativos de alinhamento (seção 2.4), formas de avaliação (seções 2.5 e 2.6), um levantamento geral sobre expressões multi-palavra (seção 2.7) e formas de detectá-las (seção 2.8) e trabalhos na interseção entre alinhamento léxico e tratamento de EMPs (seção 2.9). Após, descrevemos os recursos utilizados e criados no decorrer do projeto, como corpora, gold standard e ferramentas, no capítulo 3. Então, no capítulo 4 descrevemos o alinhador desenvolvido e apresentamos os experimentos realizados e no capítulo 5 resultados obtidos. Por fim, concluímos o trabalho com as considerações finais no capítulo 6.

2 TRABALHOS RELACIONADOS

2.1 Definição do problema

Alinhamento léxico é definido como um objeto que indica quais palavras (expressões) em um língua correspondem a quais palavras (expressões) em outra língua, num texto paralelo (BROWN et al., 1993). Um exemplo pode ser visto na figura 2.1.

The$_1$ *FIFA*$_2$ *world*$_3$ *cup*$_4$ *starts*$_5$ *today*$_6$
Começa$_1$ hoje$_2$ a$_3$ copa$_4$ do$_5$ mundo$_6$

$$\{(1,4)(3,6)(4,4)(5,1)(6,2)\}$$

		1	2	3	4	5	6
		The	FIFA	world	cup	starts	today
1	Começa					•	
2	hoje						•
3	a						
4	copa	•			•		
5	do						
6	mundo			•			

Figura 2.1: Exemplo de alinhamento

Frequentemente, é difícil até para humanos encontrar as correspondências corretas num texto. Isso ocorre especialmente quando existem expressões idiomáticas, traduções livres, *function words* faltando, etc. O problema é que a noção de correspondência é subjetiva, por exemplo *world cup* deve ser alinhado com *copa do mundo* como no exemplo (*world* com *mundo* e *cup* com *copa*), ou devemos alinhar as expressão inteiras (*world cup* com *copa do mundo*)? Podemos definir recomendações para casos deste tipo, mas sempre acabam ocorrendo casos em que o alinhador tem que fazer um julgamento.

Muitas vezes um alinhamento inclui efeitos que dificultam a tarefa. Podemos ter reordenações, quando palavras em uma língua seguem uma ordem diferente da outro, como no exemplo acima com *hoje* e *today*. Esse efeito é mais presente em línguas com estrutura gramatical distantes. Além disso, podemos ter omissões, quando uma palavra não possui correspondência, como no alinhamento acima com *FIFA* e *de* ou alinhamento de palavras com expressões (como por exemplo *take a walk* com *passear*). Por isso,

é necessário uma representação poderosa, capaz de captar todos estes efeitos, mesmo que muitos métodos depois restrinjam os alinhamentos aceitos. O alinhamento pode ser definido formalmente considerando-se uma string na língua de destino (português) $pt_1^J = pt_1, \ldots, pt_j, \ldots, pt_J$ e uma string na língua de origem (inglês) $e_1^I = e_1, \ldots, e_i, \ldots, e_I$ que devem ser alinhadas. Definimos, então, o alinhamento entre 2 strings de palavras como um subconjunto do produto cartesiano das posições das palavras, ou seja, um alinhamento $\mathcal{A}$ é definido como

$$\mathcal{A} \subseteq \{(j, i) : j = 1, \ldots, J; i = 1, \ldots, I\} \tag{2.1}$$

Para tornar mais claro, vejamos mais um exemplo, obtido do corpus utilizado no trabalho:

The$_1$ *Minutes*$_2$ *of*$_3$ *yesterday*$_4$ *'s*$_5$ *sitting*$_6$ *have*$_7$ *been*$_8$ *distributed*$_9$.

A$_1$ acta$_2$ da$_3$ sessão$_4$ de$_5$ ontem$_6$ já$_7$ foi$_8$ distribuída$_9$.

Temos uma sentença em inglês (origem) e uma em português (destino). Um alinhamento pode ser representado de forma algébrica (fig. 2.2), matricial (fig. 2.3) ou visual (fig. 2.4).

$$\{(1,1), (2,2), (3,3), (4,6), (5,5), (6,4), (7,8), (8,8), (9,9)\}$$

Figura 2.2: Representação algébrica

Veja que nesse alinhamento foi considerado que o sétimo token em português, *já*, não possui correspondência com o texto em inglês (nenhuma relação com o elemento 7) e *foi* está alinhado com a expressão *have been*. Neste exemplo extremamente simples e com tradução literal, já podemos vislumbrar algumas das dificuldades da tarefa.

		1	2	3	4	5	6	7	8	9
		The	Minutes	of	yesterday	's	sitting	have	been	distributed
1	A	•								
2	acta		•							
3	da			•						
4	sessão						•			
5	de					•				
6	ontem				•					
7	já									
8	foi							•	•	
9	distribuida									•

Figura 2.3: Representação por matriz de alinhamento

Na notação deste trabalho um **alinhamento** entre duas sentenças é formado pelo conjunto de todos os **alinhamentos individuais**, que são as correspondências, ou ligações,

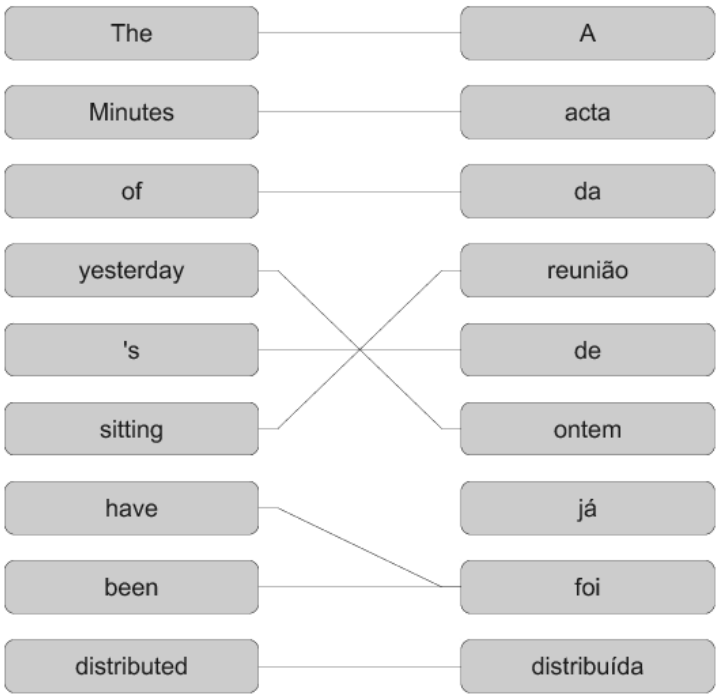

Figura 2.4: Representação visual

entre tokens. Isto se torna bastante explícito na representação algébrica da figura 2.2 que é uma representação direta da definição de alinhamento da equação 2.1. Neste trabalho utilizaremos a representação algébrica, por ser bastante compacta, e e alguns casos a representação por matriz que permite uma melhor visualização do alinhamento, além de ser a representação interna adotada pelo alinhador desenvolvido.

A definição da equação 2.1 é poderosa, mas o desenvolvimento de alinhadores que utilizem essa representação é bastante difícil. Usualmente, os métodos encontrados na literatura impõe restrições adicionais. Uma estratégia comum é fazer com que cada palavra da string de origem esteja conectada a *exatamente* uma palavra na string de destino, ou seja, permite-se alinhamentos do tipo $n : 1$ somente. Normalmente, inclui-se um token 0 (vazio) na string de destino, para representar o caso de um token não possuir ligação alguma. Melamed (2000) vai ainda mais longe, restringindo as ligações ao tipo 1 para 1, apenas a palavra vazia (0) aceita mais de uma ligação. Em princípio, essas restrições tornam impossível modelar todas as correspondências.

Colocado isto, a tarefa de alinhamento léxico é:

Dado um corpus paralelo alinhado por sentença, encontrar, para todas as

sentenças do corpus, o alinhamento léxico que melhor descreve as correspondências léxicas entre elas.

Existem muitas aplicações para alinhamentos léxicos na área de processamento de linguagens naturais. A mais óbvia é a extração automática de léxicos bilíngues e terminologia a partir de corpora (MELAMED, 2000). Além disso, muitas vezes são utilizados por sistemas de tradução de máquina estatísticos (BROWN et al., 1990); e são usados como ponto de partida de sistemas estatísticos *phrase-based* (CHIANG, 2007). Em sistemas desse tipo, a qualidade da tradução está diretamente relacionada com a qualidade dos alinhamentos utilizados. Outra aplicação é a utilização de sentenças alinhadas no nível léxico para o ensino de línguas auxiliado por computador (CALL) (WU et al., 2003).

2.2 Métodos gerativos

A abordagem gerativa ao problema de alinhamento léxico automático faz uso do fato de que as traduções seguem determinados padrões, por exemplo, palavras que co-ocorrem com frequencia em uma sentença e sua tradução tem maior probabilidade de serem elas mesmo traduções entre si. Esses padrões são modelados como um processo estatístico de tradução com alguns parâmetros que são estimados a partir de corpora paralelos de forma não-supervisionada e, por fim, o modelo é utilizado para encontrar o melhor alinhamento possível. Esta seção está fortemente baseada no artigo de Brown et al. (1993), onde foram introduzidos os modelos IBM do 1 ao 5.

Uma string de palavras em inglês ($\mathbf{en}$) pode ser traduzida em uma string de palavras em português ($\mathbf{pt}$) de diferentes maneiras. Na tradução estatística, adota-se a visão de que todas as strings $\mathbf{pt}$ são uma tradução possível de $\mathbf{en}$. A cada par de sentenças $(\mathbf{en}, \mathbf{pt})$ é associado um número, $\Pr(\mathbf{pt}|\mathbf{en})$, que é intepretado como a probabilidade que um tradutor, ao se deparar com a sentença $\mathbf{en}$, produza $\mathbf{pt}$ como sua tradução. Dada uma sentença $\mathbf{pt}$, a tarefa do sistema de tradução de máquina estatístico é encontrar a string $\hat{\mathbf{en}}$ que maximiza $\Pr(\mathbf{pt}|\mathbf{en})$.

Usando o teorema de Bayes, obtemos

$$\Pr(\mathbf{en}|\mathbf{pt}) = \frac{\Pr(\mathbf{en})\Pr(\mathbf{pt}|\mathbf{en})}{\Pr(\mathbf{pt})} \tag{2.2}$$

como o denominador é independente de $\mathbf{en}$, encontrar $\hat{\mathbf{en}}$ é equivalente a maximizar o produto $\Pr(\mathbf{en})\Pr(\mathbf{pt}|\mathbf{e})$ o que nos leva a equação fundamental do tradução de máquina:

$$\hat{\mathbf{en}} = \arg\max_{\mathbf{en}} \Pr(\mathbf{en})\Pr(\mathbf{pt}|\mathbf{en}) \tag{2.3}$$

Esta equação torna evidente os 3 desafios da tradução de máquina estatística: (1) estimar $\Pr(\mathbf{en})$ é um problema de modelar a própria língua destino da tradução; (2) estimar $\Pr(\mathbf{pt}|\mathbf{en})$ é o problema com o qual essa seção trabalho lida, a modelagem do processo de tradução; e por fim (3) precisamos realizar uma busca efetiva por $\hat{\mathbf{en}}$ uma vez que não temos como enumerar todas as sentenças de uma língua.

O objeto matemático básico com o qual o artigo trabalha é a probabilidade conjunta da distribuição $\Pr(\mathbf{pt}, \mathbf{a}, \mathbf{en})$, onde $\mathbf{a}$ é um alinhamento. Por exemplo, podemos escrever $\Pr(\mathbf{pt}|\mathbf{en})$ em termos da probabilidade condicional $\Pr(\mathbf{pt}, \mathbf{a}|\mathbf{en})$ como:

$$\Pr(\mathbf{pt}|\mathbf{en}) = \sum_{\mathbf{a}} \Pr(\mathbf{pt}, \mathbf{a}|\mathbf{en}) \tag{2.4}$$

Inicialmente, restringe-se os alinhamentos àqueles em que cada palavra em $\mathbf{pt}$ está relacionada a exatamente uma ou nenhuma palavra em $\mathbf{en}$. Se a string em inglês, $\mathbf{en} = e_1^l \equiv e_1 e_2 \dots e_l$, tem l palavras, e a string em português $\mathbf{pt} = pt_1^m \equiv pt_1 pt_2 \dots pt_m$, tem m palavras, então o alinhamento $\mathbf{a}$ pode ser representado por uma série, $a_1^m \equiv a_1 a_2 \dots a_m$, de m valores, cada um entre 0 e l, de forma que se uma palavra na posição j da string em português está conectada a palavra i da string em inglês $a_j = i$, e caso não esteja conectada a nenhuma palavra, $a_j = 0$.

Sem perda de generalidade, podemos escrever

$$\Pr(\mathbf{pt}, \mathbf{a}|\mathbf{en}) = \Pr(m|\mathbf{en}) \prod_{j=1}^{m} \Pr(a_j|a_1^{j-1}, pt_1^{j-1}, m, \mathbf{en}) \Pr(pt_j|a_1^j, pt_1^{j-1}, m, \mathbf{en}) \quad (2.5)$$

Esta é apenas uma das muitas maneiras que $\Pr(\mathbf{pt}, \mathbf{a}|\mathbf{en})$ pode ser escrita como o produto de uma série de probabilidades condicionais. É importante observar que a equação 2.5 não é uma aproximação. O que essa equação está dizendo é que quando nós geramos uma string em português junto com um alinhamento a partir de uma string em inglês, nós podemos primeiro escolher o tamanho da string em português dada a string em inglês. A seguir, podemos escolher onde conectar a primeira posição da string em português a partir do nosso conhecimento da string em inglês e do comprimento da string em português. Então nós podemos escolher a identidade da primeira palavra na string em português, dado o nosso conhecimento da string em inglês, o comprimento da string em português e a posição na string em inglês a qual a primeira posição da string em português está conectada, e assim por diante. Na medida em que passamos pela string em português, a cada passo fazemos nossa próxima escolha com o conhecimento da string em inglês e de todas as nossa escolhas anteriores.

2.2.1 Modelo IBM 1

As probabilidades condicionais do lado direito da equação 2.5 não podem todas ser tomadas como parâmetros independentes, pois há muitas delas. No modelo IBM 1 (BROWN et al., 1993), assumimos que $\Pr(m|\mathbf{en})$ não depende de $\mathbf{en}$ e de m; que $\Pr(a_j|a_1^{j-1}, pt_1^{j-1}, m, e)$ depende somente de l, o comprimento da string em inglês, e portante deve assumir o valor $(l+1)^{-1}$; e que $\Pr(pt_j|a_1^j, pt_1^{j-1}, m, \mathbf{en})$ dependem somente de pt_j e de e_{a_j}. Os parâmetros são, então, $\epsilon \equiv \Pr(m|\mathbf{en})$ e $t(pt_i|e_{a_j}) \equiv \Pr(pt_j|a_1^j, pt_1^{j-1}, m, \mathbf{en})$ que chamamos de *probabilidade de tradução* de pt_j dado e_{a_j}. Consideramos ϵ um número fixo, pequeno.

Nos focamos agora em como estimar os parâmetros para o modelo 1 (estimar $t(pt_i|e_{a_j})$). A probabilidade conjunta de uma string em português e um alinhamento, dada uma string em inglês, é

$$\Pr(\mathbf{pt}, \mathbf{a}|\mathbf{en}) = \frac{\epsilon}{(l+1)^m} \prod_{j=1}^{m} t(pt_j|e_{a_j}) \quad (2.6)$$

O alinhamento é determinado especificando os valores a_j para j de 1 até m, cada qual pode tomar um valor de 0 até l. Portanto

$$\Pr(\mathbf{pt}|\mathbf{en}) = \frac{\epsilon}{(l+1)^m} \sum_{a_1=0}^{l} \dots \sum_{a_m=0}^{l} \prod_{j=1}^{m} t(pt_j|e_{a_j}) \quad (2.7)$$

Queremos ajustar as probabilidades de tradução ($t(pt|e)$) de forma a maximizar $\Pr(\mathbf{pt}|\mathbf{en})$ sujeito a condição de que para cada e

$$\sum_{pt} t(pt|e) = 1 \tag{2.8}$$

Seguindo prática padrão de otimização condicionada, introduz-se multiplicadores de Lagrange, λ_e, e procura-se o extremo da função auxiliar:

$$h(t,\lambda) \equiv \frac{\epsilon}{(l+1)^m} \sum_{a_1=0}^{l} \cdots \sum_{a_m=0}^{l} \prod_{j=1}^{m} t(pt_j|e_{a_j}) - \sum_{e} \lambda_e \left(\sum_{pt} t(pt|e) - 1\right) \tag{2.9}$$

Um extremo ocorre quando todas as derivadas parciais de h em relação aos componentes de t e λ são 0. Que as derivadas em relação a λ sejam 0 é simplesmente uma outra forma de expressar a condição de que a soma das probabilidades de tradução deve ser 1. A derivada parcial de h em relação a $t(pt|e)$ é

$$\frac{\partial h}{\partial t(pt|e)} = \frac{\epsilon}{(l+1)^m} \sum_{a_1=0}^{l} \cdots \sum_{a_m=0}^{l} \sum_{j=1}^{m} \delta(pt, pt_j)\delta(e, e_{a_j}) t(pt|e)^{-1} \prod_{k=1}^{m} t(pt_k|e_{a_k}) - \lambda_e \tag{2.10}$$

onde δ é a função delta de Kronecker, igual a 1 quando seus argumentos são iguais, e igual a 0 caso contrário. Essa derivada parcial será 0 desde que

$$t(pt|e) = \lambda_e^{-1} \frac{\epsilon}{(l+1)^m} \sum_{a_1=0}^{l} \cdots \sum_{a_m=0}^{l} \sum_{j=1}^{m} \delta(pt, pt_j)\delta(e, e_{a_j}) \prod_{k=1}^{m} t(pt_k|e_{a_k}) \tag{2.11}$$

Superficialmente, a equação 2.11 parece a solução do problema de encontrar o extremo, porém isto não é o caso, pois as probabilidades de tradução ($t(pt|e)$) aparecem em ambos os lados da igualdade. Entretanto, essa equação sugere o uso de um método iterativo para a solução do problema: dada uma estimativa inicial para os valores da probabilidade de tradução, podemos estimar um novo valor para $t(pt|e)$ utilizando o lado direito da equação 2.11. Esse processo, quando aplicado repetidamente, chama-se algoritmo *Expectation-Maximization* (EM) (DEMPSTER; LAIRD; RUBIN, 1977).

Com o auxílio da equação 2.6, podemos reescrever a equação 2.11 como

$$t(pt|e) = \lambda^{-1} \sum_{\mathbf{a}} \Pr(\mathbf{pt}, \mathbf{a}|\mathbf{en}) \underbrace{\sum_{j=1}^{m} \delta(pt, pt_j)\delta(e, e_{a_j})}_{\text{numero de vezes que } e \text{ conecta a } pt \text{ em } \mathbf{a}} \tag{2.12}$$

Chamamos o número esperado de vezes que e conecta em pt na tradução $(\mathbf{pt}|\mathbf{en})$ de *contagem* de pt dado e para $(\mathbf{pt}|\mathbf{en})$ e denotamos por $c(pt|e; \mathbf{pt}, \mathbf{en})$. Por definição,

$$c(pt|e; \mathbf{pt}, \mathbf{en}) = \sum_{\mathbf{a}} \Pr(\mathbf{a}|\mathbf{en}, \mathbf{pt}) \sum_{j=1}^{m} \delta(pt, pt_j)\delta(e, e_{a_j}) \tag{2.13}$$

onde $\Pr(\mathbf{a}|\mathbf{en}, \mathbf{pt}) = \Pr(\mathbf{pt}, \mathbf{a}|\mathbf{en}) / \Pr(\mathbf{pt}|\mathbf{en})$. Se substituirmos λ_e por $\lambda_e \Pr(\mathbf{pt}|\mathbf{en})$, a equação 2.12 pode ser expressa de forma compacta por

$$t(pt|e) = \lambda_e^{-1} c(pt|e; \mathbf{pt}, \mathbf{en}) \tag{2.14}$$

Na prática, os dados vão consistir de um conjunto de traduções, $(\mathbf{pt}^{(1)}|\mathbf{en}^{(1)}), (\mathbf{pt}^{(2)}|\mathbf{en}^{(2)}), \ldots, (\mathbf{pt}^{(S)}|\mathbf{en}^{(S)})$, então a equação fica

$$t(pt|e) = \lambda_e^{-1} \sum_{s=1}^{S} c(pt|e; \mathbf{pt}^{(s)}, \mathbf{en}^{(s)}) \tag{2.15}$$

Aqui, λ_e serve apenas como um lembrete que as probabilidades de tradução devem ser normalizadas.

Normalmente, não é viável computar as expectativas na equação 2.13 de forma exata. Mesmo sem alinhamentos $n : n$, temos ainda $(l+1)^m$ alinhamentos possíveis para $(\mathbf{pt}|\mathbf{en})$. No caso do modelo 1, entretanto, temos que

$$\sum_{a_1=0}^{l} \cdots \sum_{a_m=0}^{l} \prod_{j=1}^{m} t(pt_j|e_{a_j}) = \prod_{j=1}^{m} \sum_{i=0}^{l} t(pt_j|e_i) \tag{2.16}$$

Desta forma, podemos manipular a equação 2.7 de forma a obter

$$\Pr(\mathbf{pt}|\mathbf{en}) = \frac{\epsilon}{(l+1)^m} \prod_{j=1}^{m} \sum_{i=0}^{l} t(pt_j|e_i) \tag{2.17}$$

Se usarmos essa expressão no lugar da equação 2.7 quando escrevemos a função auxiliar na equação 2.9, encontramos que

$$c(pt|e; \mathbf{pr}, \mathbf{en}) = \frac{t(pt|e)}{t(pt|e_0) + \ldots + t(pt|e_l)} \underbrace{\sum_{j=1}^{m} \delta(pt, pt_j)}_{\text{número de } pt \text{ em } \mathbf{pt}} \overbrace{\sum_{i=0}^{l} \delta(e, e_i)}^{\text{número de } e \text{ em } \mathbf{en}} \tag{2.18}$$

Assim, o número de operações necessário para calcular uma contagem é proporcional a $l + m$, e não a $(l+1)^m$ como a equação 2.13 sugere.

Então, usando as equações 2.15 e 2.18, podemos estimar os parâmetros $t(pt|e)$ da seguinte maneira:

1. Escolher valores iniciais para $t(pt|e)$.

2. Para cada par de sentenças $(\mathbf{pt}^{(s)}, \mathbf{en}^{(s)})$, $1 \leq s \leq S$, usar a equação 2.18 para computar as contagems $c(f|e; \mathbf{pt}^{(s)}, \mathbf{en}^{(s)})$. Essas contagem vão ser diferentes de zero somente quando pt é uma das palavras em $\mathbf{pt}^{(s)}$ e e é uma das palavras em $\mathbf{en}^{(s)}$, e não dependem da ordem das palavras nas sentenças, mas somente do número de vezes que as palavras ocorrem nas respectivas sentenças.

3. Para cada e que ocorrer ao menos uma vez em um dos $\mathbf{en}^{(s)}$,

   - Computar λ_e de acordo com a equação

$$\lambda_e = \sum_{pt} \sum_{s=1}^{S} c(pt|e; \mathbf{pt}^{(s)}, \mathbf{en}^{(s)}) \tag{2.19}$$

- Para cada pt que aparecer em ao menos um $\mathbf{pt}^{(s)}$, usar a equação 2.15 para obter um novo valor de $t(pt|e)$.

4. Repetir passos 2 e 3 até que os valores de $t(pt|e)$ tenham atingido algum critério de convergência.

Os valores iniciais de $t(pt|e)$ não são importantes (desde que não sejam 0), pois o modelo 1 possui um único máximo (global).

2.2.2 Modelo IBM 2

O modelo 1 não leva em consideração a posição das palavras na string ao calcular as probabilidades. Especialmente para línguas parecidas, e com ordem das palavras não livre, essa informação pode ser importante.

No modelo 2 (BROWN et al., 1993), são feitas as mesmas suposições que no modelo 1, com a diferença que se assume que $\Pr(a_j|a_1^{j-1}, pt_1^{j-1}, m, \mathbf{en})$ depende não somente de l (o comprimento da string $\mathbf{en}$), mas também de m (o comprimento da string em $\mathbf{pt}$), j (posição do token em $\mathbf{pt}$) e a_j (posição do token correspondente em $\mathbf{en}$. É introduzida uma *probabilidade de alinhamento*:

$$a(a_j|j, m, l) \equiv \Pr(a_j|a_1^{j-1}, pt_1^{j-1}, m, \mathbf{en}) \tag{2.20}$$

O resto das fórmulas é deduzido de maneira similar ao modelo 1 e ao modelo POS.

2.2.3 Modelos IBM 3, 4 e 5

Nos modelos subsequentes, utiliza-se um processo gerativo diferente do descrito na equação 2.5. Como havíamos mencionado, aquele era apenas uma de muitas maneiras de descrever o processo na forma de várias probabilidades condicionais.

No modelo 3, são introduzidos parâmetros para modelar a **fertilidade** das palavras, isto é, quantas palavras em português são geradas a partir de uma palavra em inglês, e a **distorção**, que busca capturar diferenças na ordem natural de palavras nas sentenças de cada língua.

O modelo 4 adicionalmente introduz um parâmetro que busca modelar o efeito de *orações*, que aparecem em uma posição diferente na tradução. No modelo 3, isto teria que ser capturado pela distorção das palavras individuais.

Os modelos 3 e 4 são **deficientes**. Isso quer dizer que eles atribuem probabilidades a strings mal formadas. Na tarefa de alinhamento isto normalmente não é um problema desde que as probabilidades sejam corretas (divididas por um fator constante, da probabilidade "perdida" por causa da deficiência), pois sempre temos strings bem formadas. O modelo 5 é igual ao modelo 4, porém não-deficiente.

É importante ressaltar que nestes modelos mais complexos, o algoritmo EM não pode ser aplicado de forma exata, pois temos muitas combinações de parâmetros possíveis. Isso é resolvido utilizando heurísticas de busca que se concentram apenas naqueles alinhamentos mais prováveis, e ignoram a grande maioria de alinhamentos que teriam uma probabilidade desprezível.

Os modelos mais avançados são sempre inicializados com os parâmetros do modelo anterior.

Mais detalhes sobre esses métodos podem ser encontrados no artigo original (BROWN et al., 1993)

2.2.4 Outros modelos

Na verdade, com algumas exceções tal como Ma (2009), poucos trabalhos foram realizados nesta área (alinhamento com processos gerativos) desde a publicação do artigo de Brown et al. em 1993. Ainda hoje utiliza-se como *baseline* de comparação os modelos IBM.

A principal contribuição desde lá é provavelmente Och e Ney (2003), que realizam extensivos testes com os modelos propostos, e descrevem também um modelo de alinhamento baseado em cadeias escondidas de Markov. Além deste, podemos citar Ganchev et al. (2008), que apresentam uma extensão do algoritmo EM que busca permitir a inclusão de informações adicionais sobre os alinhamentos desejados no processo de treinamento não supervisionado e Fraser e Marcu (2007b) que buscam uma estrutura para o alinhamento gerativo que permita alinhamentos $m : n$.

2.3 Simetrização

Os métodos de alinhamento vistos até agora são todos unidirecionais, ou seja, eles realizam treinamento em apenas uma direção. No caso dos modelos IBM, é realizado um treinamento para um modelos de tradução origem $\rightarrow$ destino e outro para destino $\rightarrow$ origem, o que gera alinhamentos diferentes para cada caso (como pode ser vistos nas figuras 2.5 e 2.6. Além disso, permite apenas alinhamentos $1 : n$ ou $n : 1$, dependendo da direção utilizada.

Por isso, introduz-se as técnicas de simetrização (OCH; NEY, 2003), como um passo de pós-processamento que combina os alinhamentos nas duas direções (origem $\rightarrow$ destino, destino $\rightarrow$ origem).

Realizando o alinhamento nas 2 direções, ficamos com 2 alinhamos, a_1^J e b_1^I para cada par de sentenças do corpus. Seja $A_1 = \{(a_j, j|a_j > 0\}$ e $A_2 = \{(i, b_i|b_i > 0)\}$ os conjuntos dos alinhamentos. Para aumentar a qualidade dos alinhamentos, combinamos A_1 e A_2 em um alinhamento A, das seguintes formas:

- Interseção: $A = A_1 \cap A_2$
- União: $A = A_1 \cup A_2$
- Refinado: Primeiramente, a interseção $A = A_1 \cap A_2$ é calculada, fornecendo alinhamentos bastante prováveis. Então, A vai sendo extendido iterativamente adicionando-se alinhamentos ocorrendo somente em A_1 ou A_2 onde nem pt_j nem e_i possuem um alinhamento em A, ou se as duas condições a seguir forem verdadeiras
  - O alinhamento (i, j) possui um vizinho horizontal $(i - 1, j), (i + 1, j)$ ou um vizinho vertical $(i, j - 1), (i, j + 1)$ que já esteja em A.
  - O conjunto $A \cup \{(i, j)\}$ não contém alinhamentos com ambos os vizinhos horizontais e verticais.

Por exemplo, o par de sentenças

life$_1$ *'*$_2$ *s*$_3$ *not*$_4$ *fair*$_5$ *,*$_6$ *is*$_7$ *it*$_8$ *?*$_9$
a$_1$ vida$_2$ é$_3$ injusta$_4$,$_5$ não$_6$ é$_7$?$_8$

foi alinhado pelo Giza++ de acordo com os modelos IBM e as heurísticas de simetrização apresentadas, e os resultados podem ser vistos nas figuras 2.7, 2.8 e 2.9

$$\{(1,2)(3,3)(5,4)(6,5)(7,7)(8,6)(9,8)\}$$

		1	2	3	4	5	6	7	8	9
		life	,	s	not	fair	,	is	it	?
1	a									
2	vida	•								
3	é			•						
4	injusta					•				
5	,						•			
6	não								•	
7	é							•		
8	?									•

Figura 2.5: Alinhamento en → pt

$$\{(1,2)(3,3)(4,4)(5,4)(6,5)(7,7)(8,7)(9,8)\}$$

		1	2	3	4	5	6	7	8	9
		life	,	s	not	fair	,	is	it	?
1	a									
2	vida	•								
3	é			•						
4	injusta				•	•				
5	,						•			
6	não									
7	é							•	•	
8	?									•

Figura 2.6: Alinhamento en ← pt

2.4 Métodos discriminativos

Nos métodos discriminativos de alinhamento léxico não é necessário gerar uma "história" de como os dados foram gerados para depois encontrar os parâmetros deste modelo. Nos métodos discriminativos usualmente utiliza-se *features* e então métodos de aprendizado de máquina, normalmente supervisionados, ou seja, que possuem exemplos de alinhamentos corretos para treinar um alinhador.

Os primeiros métodos discriminativos surgiram em 2005, e o objetivo é justamente simplificar a adição de mais parâmetros ao sistema, o que é bastante difícil de fazer com os métodos gerativos.

Moore (2005) descreve um alinhador léxico baseado no treinamento discriminativo de uma combinação linear ponderada de um pequeno número de *features*. Para uma dada sentença, para cada alinhamento considerado, multiplica-se o valor de cada uma das *features* pelo peso correspondente, e depois soma-se esses valores de forma a obter um escore final para o alinhamento. Assim, o problema se resume em encontrar o alinhamento com maior escore total. Para um par de sentença $(\mathbf{org}, \mathbf{dst})$ busca-se o alinhamento $\hat{\mathbf{a}}$ tal que

$$\{(1,2)(3,3)(4,4)(5,4)(6,5)(7,7)(8,6)(8,7)(9,8)\}$$

		1	2	3	4	5	6	7	8	9
		life	,	s	not	fair	,	is	it	?
1	a									
2	vida	•								
3	é			•						
4	injusta				•	•				
5	,						•			
6	não								•	
7	é							•	•	
8	?									•

Figura 2.7: União dos Alinhamentos

$$\{(1,2)(3,3)(5,4)(6,5)(7,7)(9,8)\}$$

		1	2	3	4	5	6	7	8	9
		life	,	s	not	fair	,	is	it	?
1	a									
2	vida	•								
3	é			•						
4	injusta					•				
5	,						•			
6	não									
7	é							•		
8	?									•

Figura 2.8: Interseção dos Alinhamentos

$$\hat{\mathbf{a}} = \arg\max_{a} \sum_{i=1}^{n} \lambda_i f_i(a, org, dst) \tag{2.21}$$

onde org é a sentença na língua de origem, dst a sentença na língua de destino[1], f_i são as *features*, e λ_i os pesos.

Em linhas gerais, podemos descrever os alinhadores discriminativos a partir de 4 componentes: *feature functions*, combinação, busca e otimização, que a princípio são independentes. Por serem independentes, diferentes solução para um mesmo componente podem ser combinadas entre si. Por exemplo, poderíamos pegar um sistema pronto e alterar um de seus componentes sem modificar os outros.

A ideia dos alinhadores discriminativos em geral é utilizar uma série de informações sobre o que faz um bom alinhamento para encontrar o melhor alinhamento possível. Essa

[1]Mantemos a nomenclatura de *origem* e *destino*, mas devemos ressaltar que essa distinção não possui peso algum nos métodos discriminativos pois eles são simétricos, podendo-se trocar org por dst mantendo o mesmo resultado.

$$\{(1,2)(3,3)(4,4)(5,4)(6,5)(7,7)(8,6)(9,8)\}$$

		1	2	3	4	5	6	7	8	9
		life	,	s	not	fair	,	is	it	?
1	a									
2	vida	•								
3	é			•						
4	injusta				•	•				
5	,						•			
6	não								•	
7	é							•		
8	?									•

Figura 2.9: Alinhamento Refinado

série de informações é fornecida pelo componente das ***feature functions***. Cada *feature function* deve fornecer alguma medida que indique quão bom é um alinhamento. Em geral, quanto maior seu valor, melhor o alinhamento de acordo com a *feature function* em questão. Definimos em mais detalhes as *feature functions* na seção 2.4.1, e apresentamos as principas encontradas na literatura. Cada *feature function* captura um aspecto do alinhamento, por isso utiliza-se várias, que devem ser combinadas, através do componente de **combinação**. Na combinação, descrita em mais detalhes na seção 2.4.2 é gerado um escore combinado global a partir dos escores individuais das *feature functions* e parâmetros que atribuem pesos distintos às diferentes *feature functions*. Esses parâmetros precisam ser otimizados de forma que o alinhamento final resultante seja ótimo no componente de **combinação**. Vemos algumas maneiras de fazer isso na seção 2.4.4. O processo de obter um valor de escore global para um dado alinhamento candidato está representado graficamente pela figura **??**. Por fim, temos o componente que de fato realiza o alinhamento, o componente de **busca**. O componente de busca tem a função de encontrar, dentre todos os alinhamentos possíveis entre duas sentenças, aquele cujo valor do escore combinado global seja máximo. Vemos como isso pode ser feito na seção 2.4.3.

2.4.1 Feature Functions

Uma *feature function* (FF) tem a forma $f_i(a, org, dst, \theta_i)$, onde f_i é a *feature function* em questão, a é um alinhamento candidato entre duas sentenças, org e dst são vetores com as palavras das sentenças de origem e destino respectivamente, e θ_i são os parâmetros de f_i. Cada *feature function* tem a função de identificar uma característica de um bom alinhamento. Definimos como FF **local** uma FF cujo valor seja dependente exclusivamente da soma de todos os alinhamentos individuais, ignorando vizinhos, e como FF **global** uma FF onde há interação entre alinhamentos vizinhos na definição do valor.

Todo alinhador discriminativo necessita de pelo menos uma FF que funcione aproximadamente como um dicionário, ou seja, que capture as propriedades de tradução entre palavras. Várias FFs para isto foram propostas na literatura, e todas elas são locais. Além de no mínimo uma FF que capture relações de equivalência entre palavras, normalmente outras FFs também são sugeridas, capturando fenômenos como ordem de tradução, correspondência entre determinadas categorias gramaticais, entre outras. No decorrer desta

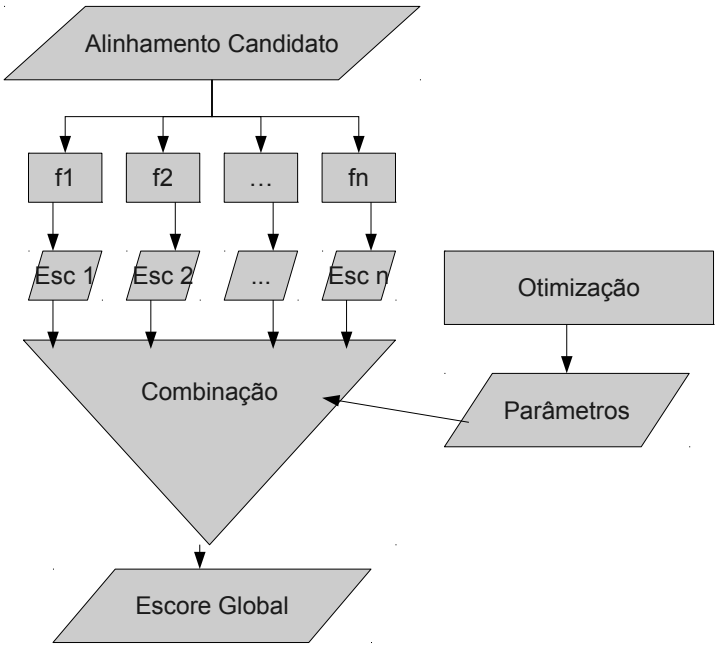

Figura 2.10: Obtenção do valor do escore combinado global

seção, descreveremos as principais FFs encontradas na literatura.

2.4.1.1 *Log-likelihood ratio*

Uma *feature* que captura probabilidades de tradução entre palavras é a *log-likelihood-ratio* (LLR), que é usada para a construção automática de léxicos de tradução (MELAMED, 2000) e foi proposta para o contexto de alinhamento discriminativo por Moore (2005). A fórmula para o cálculo do LLR é:

$$LLR(f,e) = \sum_{f? \in \{f, \neg f\}} \sum_{e? \in \{e, \neg e\}} C(f?, e?) \log \frac{p(f?|e?)}{f(f?)} \tag{2.22}$$

Nesta fórmula, f e e significam que as palavras cujo grau de associação está sendo medido ocorrem nas respectivas sentenças destino e origem de uma sentença alinhada (por sentença, não lexicalmente), $\neg f$ e $\neg e$ significam que as palavras correspondentes não ocorrem nas respectivas sentenças e $f?$ e $e?$ são variáveis sobre estes valores e $C(f?, e?)$ é a contagem conjunta dos valores de $f?$ e $e?$. Todas as probabilidades na fórmula correspondem a estimativas de máxima verossimilhança. O escore LLR para um par de palavras é alto caso estas tenham uma forte associação tanto positiva quanto negativa. Como é esperado que palavras correspondentes por tradução estejam associadas positivamente, Moore (2005) remove quaisquer pares associados negativamente, ou seja, onde $p(f,e) \leq p(f) * p(e)$. O escore da *feature* é formado pela soma de todos os LLRs para cada alinhamento individual.

Colocando isso em termos de nossa definição de *feature function*, $f_i(a, org, dst, \theta_i)$ é a **soma** dos LLRs para cada alinhamento individual contido em $\mathbf{a}$, sendo que estes LLRs são fornecidos como parâmetro da função, θ_i.

2.4.1.2 *Probabilidade de tradução de palavra em Modelos IBM*

Um dos parâmetros treinados pelos modelos IBM é a **probabilidade de tradução de palavra**, que específica a probabilidade $\Pr(destino|origem)$, onde *origem* é uma palavra na língua de origem e *destino* uma palavra na língua de destino. Estas probabilidades podem ser usadas de forma similar ao LLR para formar um *feature*. Por exemplo, Liu et al. (2005) utilizam a probabilidade de tradução de palavra extraída do modelo IBM 3 como uma *feature* do alinhador, Fraser e Marcu (2006) utilizam as do IBM 4, assim como acabamos utilizando neste trabalho. Independente de qual modelo IBM é utilizado, a *feature function* acaba tendo a mesma estrutura, parecida com a LLR: $f_i(a, org, dst, \theta_i)$ é a soma das probabilidades de tradução de todos os alinhamentos individuais em $\mathbf{a}$, e o parâmetro θ_i contém essas probabilidades.

2.4.1.3 *Coeficiente de Dice*

O coeficiente de Dice (DICE, 1945) é mais uma maneira de encontrar associações entre palavras a partir de um corpus paralelo alinhado por sentença. O coeficiente de Dice para duas palavras e e f nas línguas de origem e destino respectivamente pode ser calculado a partir da seguinte fórmula:

$$Dice(e,f) = \frac{2Count(e,f)}{Count_s(e)Count_t(f)} \tag{2.23}$$

onde $Count(e,f)$ é o número de co-ocorrência das palavras, $Count_s(e)$ é o total de ocorrências de e nas sentenças de origem e $Count_t(f)$ é o total de ocorrências de f nas sen-

tenças de destino.

Uma *feature* utilizando coeficientes de Dice é proposta em Taskar et al. (2005), onde $f_i(a, org, dst, \theta_i)$ é a soma dos coeficientes de Dice dos alinhamentos individuais de a, parâmetro θ_i da função.

2.4.1.4 Dicionários

Além de medidas estatísticas obtidas a partir de corpora paralelos, podemos utilizar dicionários manualmente construídos como forma de se obter uma *feature* que capture as propriedades de qual palavra é traduzida para qual palavra em um alinhamento. Liu et al. (2005) sugerem uma *feature* a partir de dicionários manualmente construídos, onde θ_i contém as correspondências de tradução entre palavras e $f_i(a, org, dst, \theta_i)$ é a soma de todos alinhamentos individuais contidos em a que também são uma correspondência de tradução de acordo com o dicionário. O uso de um dicionário pode ser bastante positivo, no entanto muitas vezes não é possível obter um bom dicionário em formato eletrônico que possa ser usado, tanto por motivos de direitos autorais quanto para línguas com recursos mais escassos. Além disso, muito cuidado deve ser tomado com inflexões das expressões, já que o dicionário normalmente contém somente uma forma canônica das palavras.

2.4.1.5 Features de Localidade na Tradução

Pode ser observado que em línguas fortemente relacionadas, os alinhamentos tendem a ser aproximadamente monotônicos (palavras correspondentes tendem a ficar em posições aproximadamente correspondentes nas sentenças). Mesmo para línguas relacionadas de forma mais distante este efeito aparece em menor grau, pois sequências tendem a ser traduzidas por *chunks* contíguos.

Liu et al. (2005) sugerem duas medidas do que são chamadas de não-monotonicidade. Os alinhamentos são ordenados, primeiramente por posição da palavra na sentença de origem, após por posição da palavra na sentença de destino. Em cada uma dessas ordenações, pontos de não-monotinicidade podem ser encontrados observando a sequencia de posições das palavras não-ordenadas. Por exemplo, suponha o alinhamento ordenado por posição de origem $\{(1,1),(2,4),(2,5),(3,2),(6,6)\}$: a sequência das posições não-ordenadas é $(1,4,5,2,6)$, ou seja, um ponto de não-monotonicidade ocorre quando a posição 2 segue a posição 5. O processo é repetido de forma análoga ordenando por ordem de palavra do destino. São sugeridas 2 *features* a partir desse processo: em uma delas soma-se as magnitudes do salto para trás em cada ponto de não-monotonicidade (no exemplo, 5, saltando de 5 para 2), na outra simplesmente conta-se o número de ocorrências de pontos de não-monotonicidade.

Além dessas *features*, outras similares podem ser propostas, como a que sugerimos neste trabalho, ou mesmo pode se usar as probabilidades de distorção geradas pelos modelos IBM mais complexos (FRASER; MARCU, 2006). *Features* de localidade na tradução de modo geral são globais.

2.4.1.6 Fertilidade

A fertilidade de uma palavra é definida pelo número de palavras pelas quais ela é traduzida, por exemplo, se *morrer* for traduzido como *kick the bucket*, a fertilidade de *morrer* neste caso é 3, porém se fosse traduzido por *die* seria 1. Muitas vezes ocorre também de termos fertilidade 0, quando palavras são omitidas na tradução. A probabilidade de fer-

tilidade é a probabilidade de cada um desses casos para cada palavra, por exemplo, para *morrer* teríamos uma probabilidade para a fertilidade 0, uma para a fertilidade 1, uma para 2, e assim por diante até uma fertilidade máxima. Esta é definida como uma constante alta, de forma a permitir que todas as fertilidades encontradas na prática sejam possíveis.

A fertilidade é um do parâmetros estimados pelos modelos IBM, e foi utilizada em Fraser e Marcu (2006) e também neste trabalho como uma *feature*, $f_i(a, org, dst, \theta_i)$ onde θ_i contém as fertilidades de todas as palavras conforme estimadas pelos modelos IBM. f_i é calculada da seguinte forma: para todas as palavras presentes na origem e no destino, é feito o cálculo da fertilidade real no alinhamento a, ou seja, quantas vezes essa palavra está alinhada, e a probabilidade de tal fertilidade é consultada na tabela. O escore final da *feature* é a multiplicação de todas as probabilidades de fertilidade individuais.

2.4.1.7 Outras features

Além dessas *features* descritas, uma série de outras pode ainda ser encontrada na literatura. Por exemplo, em Liu et al. (2005) é apresentada uma *feature* que busca capturar probabilidades de tradução de etiqueta morfossintática. Funções de "bônus" como em Moore (2005) e a proposta nesse trabalho também são encontradas. Há ainda variações sobre as medidas descritas, que também são comuns. A criação de nova *features* é na prática limitada apenas pela imaginação.

2.4.2 Combinação

Na combinação, precisamos obter um escore combinado global a partir dos escores individuais das funções. Esta etapa do processo de alinhamento léxico discriminativo é a que menos tem variação na literatura, na prática, todos os trabalhos acabam utilizando um combinação linear:

$$\sum_{i=1}^{n} \lambda_i f_i(a, org, dst, \theta_i) \tag{2.24}$$

os valores de de cada *feature function* são multiplicados por um peso λ_i e somados para se obter o escore global.

2.4.3 Busca

De posse de uma função de escore combinado global, resta encontrar aquele alinhamento que a maximiza. O espaço de busca de todos os alinhamentos possíveis para a definição de alinhamento descrita na seção 2.1 é de 2^{m*n} onde m e n são o tamanho em tokens das sentenças de origem e destino respectivamente. Isso torna uma busca exaustiva proibitiva. Idealmente, gostaríamos de um algoritmo rápido e exato para encontrar o alinhamento máximo, sem precisar percorrer todos os alinhamentos. Porém, quando *feature functions* globais são utilizadas, não é conhecido nenhum algoritmo para a tarefa.

Por isso, as funções de busca são baseadas em algum tipo de heurística, como o **hill-climbing**. Numa busca pelo melhor alinhamento utilizando *hillclimbing*, inicia-se com um alinhamento nulo, ou aleatório, e a partir faz-se uma busca pela vizinhança do alinhamento corrente, buscando aquele vizinho cujo escore global seja máximo, que torna-se o novo alinhamento corrente. O processo segue até que não seja mais possível melhorar o escore global. O *hillclimbing* está sujeito a mínimos locais, assim como qualquer heurística. Liu et al. (2005) sugerem uma busca utilizando o *hillclimbing* de forma mais simples possível, definindo a vizinhança como o alinhamento corrente somado de um ali-

nhamento individual. Desta forma, o alinhamento inicial deve ser o alinhamento nulo, ou vazio. Fraser e Marcu (2006) expandem o método de *hillclimbing* básico, criando uma fila de prioridade de outros alinhamentos a serem considerados, como forma de fugir de máximos locais. Um forma de *hillclimbing* também foi utilizada neste trabalho, por ser de implementação simples e de boa performance.

Apesar de *hillclimbing* ser o método mais popular para a busca, outros também existem. Por exemplo, Moore (2005) utiliza *beam search* para buscar o melhor alinhamento. No *beam search* proposto, um número n (largura do *beam*) de alinhamentos é considerado em cada passo, todos os vizinhos dos alinhamentos contidos no *beam* são considerados e ordenados de acordo com o escore global, e os n melhores permanecem no *beam* para o próximo passo. Alternativamente, caso os alinhamentos permitidos sejam restritos a 1:1, existem algoritmos melhores, por exemplo o proposto por Melamed (2000).

2.4.4 Otimização

Os pesos λ_i das diferentes *features* unidas através da combinação precisam ser escolhidos de forma a maximizar a performance do alinhador. Para isso, utiliza-se um *gold standard* e algum algoritmo para induzir os valores. Por exemplo, Moore (2005) utiliza um *averaged perceptron* para realizar a otimização dos parâmetros. Iniciando com um conjunto inicial de pesos λ_i, o aprendizado com perceptron itera pelo corpus anotado múltiplas vezes, comparando, para cada par de sentenças, o melhor alinhamento a_{hyp} de acordo com o modelo corrente com o alinhamento de referência a_{ref}. Em cada par de sentenças, o peso de cada *feature* é incrementado pela diferença entre o valor da *feature* para o melhor alinhamento de acordo com o modelo e o valor da *feature* no alinhamento de referência:

$$\lambda_i \leftarrow \lambda_i + (f_i(a_{ref}, org, dst, \theta_i) - f_i(a_{hyp}, org, dst, \theta_i)) \tag{2.25}$$

os valores atualizados são utilizados para computar a_{hyp} do próximo par de sentenças, e o algoritmo prossegue até atingir uma condição de parada.

2.5 Métodos de avaliação

Como sempre é o caso em métodos de processamento de linguagens naturais empíricos, é de vital importância realizarmos uma avaliação adequada dos resultados obtidos. Com base nos resultados encontrados, podemos decidir qual o método mais adequado para cada caso.

Podemos dividir as formas de avaliação em 2 grandes grupos. Na avaliação **intrínseca**, buscamos avaliar a saída dos métodos diretamente, independente de como estes resultados serão utilizados depois; enquanto que na avaliação **extrínseca**, buscamos verificar o efeito que o resultado do alinhamento causa em uma tarefa maior, que o utiliza como uma das entradas. Neste trabalho será usada a avaliação intrínseca.

2.5.1 Avaliação intrínseca

Como já mencionado, na avaliação intrínseca queremos verificar diretamente a qualidade dos alinhamentos gerados. O procedimento geralmente usado em experimentos da literatura é a criação de um ***gold standard***, uma coleção de sentenças paralelas alinhadas manualmente por um especialista. Realiza-se então o teste do método de alinhamento e, em seguida, compara-se a saída do alinhador com as sentenças do *gold standard*. Com-

parando o *gold standard* com o resultado gerado pelo sistema que deseja-se avaliar é possível calcular algumas métricas de performance tradicionalmente usadas em PLN, tais como precisão e revocação.

Na construção do *gold standard* é importante observar que, como visto na seção 2.1, a tarefa de realizar o alinhamento é às vezes subjetiva, possuindo ambiguidades, e difícil de ser realizada mesmo por um humano. Isso não torna essa avaliação menos valiosa, mas requer certos cuidados, como estabelecimento de boas "regras" (*guidelines*) para os anotadores, e também uma análise de concordância entre anotadores (*inter-annotator agreement*). Por causa dessa ambiguidade Och e Ney (2003) permitem que os anotadores marquem correspondências como *Sure* ou *Possible* e propõe a métrica de avaliação do *Alignment Error Rate* (AER).

Para explicar as medidas utilizadas em termos da tarefa de alinhamento, começamos definindo como G os alinhamentos (individuais) presentes no *gold standard* (em teoria, todos os alinhamentos corretos), e como A os alinhamentos (individuais) encontrados pelo algoritmo.

2.5.1.1 Precisão

Precisão é definida como

$$\text{Precisão}(A, G) = \frac{|G \cap A|}{|A|} \tag{2.26}$$

ou seja, dentre os alinhamentos individuais encontrados pelo método, a precisão representa quantos deles estavam corretos. É possível aumentar a precisão selecionado somente aqueles alinhamentos dos quais mais temos certeza, porém ao custo de uma revocação menor.

2.5.1.2 Revocação

A revocação (ou *recall*) é definida como

$$\text{Revocação}(A, G) = \frac{|G \cap A|}{|G|} \tag{2.27}$$

ou seja, dentre todos os alinhamentos individuais corretos existentes no *gold standard*, a revocação representa quantos deles o método conseguiu identificar. Podemos atingir uma revocação de 1 (máximo) meramente gerando *todos* os alinhamentos possíveis.

2.5.1.3 Escore-f

Entre precisão e revocação, sempre é possível aumentar um em detrimento do outro, diminuindo ou aumentando o grau de certeza necessário para a criação de um alinhamento. O que queremos então, é um certo equilíbrio entre os 2, podendo um ter mais peso que o outro, mas em geral não é apropriado maximizar um deles em detrimento total do outro.

Para capturar esse equilíbrio, utiliza se o escore-f, definido como

$$\text{escore-f}(A, G, \alpha) = \frac{1}{\frac{\alpha}{\text{Precisão(A, G)}} + \frac{(1-\alpha)}{\text{Revocação}(A,G)}} \tag{2.28}$$

onde α é um fator que define os pesos que queremos dar à precisão e revocação; quanto maior for α, maior é a importância da precisão em detrimento da revocação, e vice-versa.

2.5.1.4 AER

Uma medida que foi usada extensivamente na literatura até pouco tempo atrás é o *Alignment Error Rate* (AER) (OCH; NEY, 2003). Ele é baseado na anotação que permite criação de links como *Sure* (definitivamente há o alinhamento) e *Possible* (possivelmente há o alinhamento). Para definir AER, é necessário redefinir precisão e revocação para este novo esquema de anotação. Sejam S os alinhamentos definidos como *Sure* no gold-standard, P os alinhamentos definidos como *Possible* e A os alinhamentos encontrados:

$$\text{Precisão}_2(A, P) = \frac{|P \cap A|}{|A|} \tag{2.29}$$

$$\text{Revocação}_2(A, S) = \frac{|S \cap A|}{|S|} \tag{2.30}$$

$$\text{AER}(A, P, S) = \frac{|P \cap A| + |S \cap A|}{|S| + |A|} \tag{2.31}$$

Entretanto, Fraser e Marcu (2007a) mostram que o AER não é uma métrica adequada para a tarefa de alinhamento léxico. Os principais defeitos são que o AER **não** possui uma correlação boa com a qualidade da tradução gerada por sistemas de tradução treinados, e o pior, é possível aumentar o AER dando preferência grande à precisão o que resulta em uma métrica que não se correlaciona fortemente com tarefas associadas, em especial a tradução de máquina que obtém melhor desempenho quando há um equilíbrio entre precisão e revocação no alinhamento.

Eles sugerem o uso de escore-f, que, dependendo do valor de α, possui uma boa correlação com a qualidade de tradução ou ainda a realização uma avaliação extrínseca.

2.5.2 Avaliação extrínseca

Na avaliação extrínseca, nós buscamos avaliar o quanto uma melhora no alinhamento se reflete em melhora em uma tarefa que utilize alinhamentos léxicos. Desta forma avaliamos esta tarefa variando apenas o algoritmo de alinhamento léxico. Assim, quanto melhor for a avaliação desta tarefa, melhor é o alinhamento (pelo menos para essa tarefa em específico...). Normalmente isso é feito treinando um sistema de tradução de máquina estatístico com a saída do alinhador, e avaliando este sistema de tradução utilizando uma métrica da área, como por exemplo o BLEU (PAPINENI et al., 2002). De acordo com Fraser e Marcu (2007a), o escore-f possui uma forte correlação com avaliação extrínseca realizada utilizando o BLEU.

2.6 Concordância entre anotadores

Como apresentado na seção 2.5.1, a avaliação intrínseca adotada neste trabalho envolve a comparação de alinhamentos gerados automaticamente com alinhamentos criados manualmente (*gold standard*). Para validar as anotações manuais criadas pode-se realizar a anotação com mais de um anotador humano, seguindo regras estabelecidas, e avaliar o grau de concordância destes. Caso haja uma boa concordância, podemos concluir que a anotação é coerente e pode ser utilizada.

Para medir a concordância entre anotadores, Carletta (1996) sugere o uso da estatística κ (kappa), que mede o grau de concordância entre dois anotadores levando em conta a probabilidade deles concordarem ao acaso. O κ é utilizado em tarefas onde os anotadores

podem classificar uma série de observações em um número finito de classes. Para o caso do alinhamento léxico, as observações são todos os alinhamentos individuais possíveis para uma dada sentença, e as classes são *alinhado* e *não-alinhado*. A equação para o cálculo de κ é:

$$\kappa = \frac{\Pr(a) - \Pr(e)}{1 - \Pr(e)} \tag{2.32}$$

onde $\Pr(a)$ é a concordância observada entre os anotadores, e $\Pr(e)$ é a probabilidade hipotética de concordância ao acaso, utilizando os dados observados para calcular as probabilidades de cada observador escolher uma categoria. Quando a concordância entre os anotadores é total, temos $\kappa = 1$, quando a concordância é meramente aquela que podemos esperar de forma aleatória, temos $\kappa = 0$, e entre esses valores, quanto maior o κ, maior a concordância.

Suponha a matriz de confusão entre dois anotadores representada pela tabela 2.1. O primeiro alinhador (linhas) classificou as instâncias como *alinhado* 130 vezes ou numa proporção de 0.15, e *não-alinhado* como 720 vezes (proporção de 0.85). Já o segundo alinhador (colunas) considerou alinhado 120 vezes (0.14) e não alinhado 730 (0.86). Assim, $\Pr(a) = 800/850 = 0.94$ ou o número de vezes em que os anotadores concordam dividido pelo total de alinhamentos considerados. Vamos agora calcular $\Pr(e)$, a proporção de que eles concordem ao acaso nas marcações de *alinhado*:

$$0.15 * 0.14 = 0.021 \tag{2.33}$$

e de que eles concordem ao acaso nas marcações de *não-alinhado* é

$$0.85 * 0.86 = 0.731 \tag{2.34}$$

e a proporção que eles concordam total, $\Pr(e)$ é

$$0.021 + 0.731 = 0.752 \tag{2.35}$$

	alinhado	**não-alinhado**
alinhado	100	30
não-alinhado	20	700

Tabela 2.1: Matriz de confusão

Voltando à equação 2.36 e substituindo os valores obtidos, temos

$$\kappa = \frac{\Pr(a) - \Pr(e)}{1 - \Pr(e)} = \frac{0.94 - 0.752}{1 - 0.752} = 0.758 \tag{2.36}$$

Foi desta forma que calculamos a estatística κ do *gold standard* deste trabalho, descrito na seção 3.2.

De acordo com Carletta (1996), entre outros autores, um valor de κ entre 0.67 e 0.80 indica uma boa concordância, possibilitando conclusões preliminares, enquanto que um κ acima de 0.80 indicia grande confiabilidade e abaixo de 0.67 geralmente não permite que se tire conclusões.

2.7 Expressões Multi-palavra

O foco deste trabalho é em tradução automática (mais especificamente alinhamento léxico) de expressões multi-palavra. Em Sag et al. (2002), as expressões multi-palavras são definidas como "interpretações idiossincráticas que cruzam limite de palavras (ou espaços)", tais como expressões idiossincráticas (*against the grain*), verbos frasais (*take a walk*) ou colocações (*salt and pepper*). A importância do tratamento correto destas expressões fica evidente se levarmos em consideração que o número de EMPs conhecidas por um falante é da mesma ordem de grandeza que o de palavras simples. Além disso, vocabulários de domínios específicos consistem muitas vezes principalmente de EMPs, como por exemplo, termos jurídicos (*habeas corpus*), esportivos (*running back*, *team manager*), turísticos (*book up*). Assim, ao aumentar a cobertura de um sistema da PLN para novos domínios, estaremos aumentando a proporção de EMPs necessárias ao processamento.

As EMPs aparecem em todo tipo de textos e causam problemas significativos para os tipos de PLN que necessitam de algum tipo de interpretação semântica, como sistemas de geração automática ou tradução automática.

Há duas principais abordagens para o tratamento de EMPs: usando **métodos composicionais** e listando EMPs como **palavras-com-espaço**. Por exemplo, tratando as EMPs com métodos composicionais de análise linguística, um sistema geraria expressões corretas, como *telephone booth*, mas não teria como diferenciar esta de outras perfeitamente composicionais, mas incorretas, como *telephone closet*. Além disso, um sistema desse tipo não seria capaz de lidar com a idiomaticidade, ou seja, nos casos em que o significado da EMP não pode ser inferido de suas partes (não é composicional), como por exemplo *kick the bucket* que (normalmente) significa algo como *morrer*.

Seguindo um caminho diametralmente oposto, poderíamos tratar as EMPs simplesmente como palavras com espaços. Porém, EMPs muitas vezes possuem variação interna distinta. Por exemplo, *kicked the bucket* não permite uma forma na voz passiva **the bucket was kicked*, mas *light their spirits* permite *their spirits were lightened*. A opção seria listar todas as formas permitidas, porém isso geraria uma proliferação léxica muito grande, com várias entradas que na verdade se referem à mesma coisa (*kick/kicked/kicking the bucket*).

Na próxima seção, discutimos uma classificação dos tipos de EMP (SAG et al., 2002) encontrada na literatura.

2.7.1 Tipos de EMPs

As expressões multi-palavra podem ser divididas em **frases lexicalizadas** e **frases institucionalizadas**. Frases lexicalizadas são aquelas que possuem algum tipo de irregularidade semântica ou sintática, e ainda podem ser classificadas em **expressões fixas**, **expressões semi-fixas** e **expressões sintaticamente flexíveis**. As frases institucionalizadas são sintaticamente e semanticamente composicionais, porém ocorrem com frequência muito alta em um determinado contexto.

2.7.1.1 Expressões fixas

Expressões fixas são expressões como *by and large*, *kingdom come*, *ad hoc*, que não seguem convenções gramaticais e composicionais de interpretação.

Estas expressões não sofrem variação morfossintática (**by and larger* a partir de *by and large*), nem modificação interna (*by and very large* também a partir de *by and large*). Por isso, uma representação do estilo palavra-com-espaços é adequada ao pro-

blema. Se tentássemos representá-las de maneira composicionais, teríamos que incluir entradas como *hoc*, e impedir que elas se combinassem em outros contextos, pois neste caso, *hoc* só aparece na EMP *ad hoc*.

2.7.1.2 Expressões semi-fixas

Essas expressões seguem restrições rígidas na ordem das palavras e composição, mas sofrem algum tipo de variação léxica, como por exemplo variação na forma reflexiva ou de inflexão. Com isto, elas podem ser tratadas como um complexo de palavras com uma única classe sintática. Alguns tipos de expressões semi-fixas são:

Expressão idiomática não-composicional Algumas expressões idiomáticas podem ser decompostas, mesmo quando seu significado não é imediatamente previsível pelas suas partes. Por exemplo *spill the beans* pode ser analisado, considerando-se um sentido de *revelar* para *spill* e um sentido de *segredo(s)* para *beans*, resultando em *revelar os segredos*.

Outras, porém, não permitem tal decomposição, como *kick the bucket* e *shoot the breeze*. Este tipo de EMP não é suscetível a variações sintáticas como modificação interna (**kick the great bucket* a partir de *kick the bucket*) ou transformação na voz passiva. Os únicos tipos de variação léxica que elas permitem são inflexão (*kicked the bucket*) e variação morfológica de pronome (*wet oneself/himeself/herself/...*). Usando uma abordagem de palavra-com-espaços, não conseguimos descrever a variação interna das expressões sem incorrer em proliferação léxica. Por outro lado, usando uma abordagem totalmente composicional, não teríamos problema para reconhecer as variações corretas, porém acabaríamos aceitando também as formas incorretas como *kick the buckets*. Além disso, o método composicional tem problemas com a idiomaticidade, ou seja, não teria como derivar a semântica correta para a EMP (por exemplo, *morrer* a partir de *kick*, *the*, *bucket*)

Substantivos compostos como *flight attendant*, *congressman at large* também não sofrem alteração sintática, somente inflexão de número. Portanto, precisaríamos saber qual o núcleo da expressão para pode aplicar o plural ou não, por exemplo, *flight attendants* em oposição à *congressmen at large*. Em português estas expressões são de fácil detecção, pois a gramática requer o uso de hífen, porém ainda se faz necessário o conhecimento do núcleo da expressão, por exemplo, *homens-bomba*.

2.7.1.3 Expressões sintaticamente flexíveis

Enquanto que as expressões semi-fixas mantém uma mesma ordem básica das palavras as expressões sintaticamente flexíveis apresentam uma variabilidade sintática muito maior.

As **construções verbo-partícula** (CVP), formadas por um verbo e uma ou mais partículas, como em *make up*, são o foco do trabalho e serão tratadas em mais detalhes na seção 2.7.2.

Expressões idiomáticas composicionais como *let the cat out of the bag* tendem a ter uma variabilidade sintática maior, porém quais exatamente são possíveis para cada expressão é altamente imprevisível. Por sua alta variabilidade sintática, eles não são compatíveis com uma estratégia do tipo palavra-com-espaços, e devido à imprevisibilidade das variações possíveis, uma técnica totalmente composicional aceitaria construções inválidas.

Light Verbs são expressões do tipo *make a mistake*, *give a demo*, em que um verbo se combina com um substantivo. Esse tipo de expressão está sujeito à variabilidade sintática completa (por exemplo, *a mistake was made*, *How many more mistakes are you going to*

make?), e portanto parecem ser candidatas a serem tratadas de maneira composicional. Mas, como as combinações verbo-substantivo são limitadas, é preciso evitar que sejam aceitas expressões inválidas como **do a mistake*.

2.7.1.4 Frases institucionalizadas

Frases institucionalizadas são semanticamente composicionais, porém estatisticamente idiossincráticas. Por exemplo, em *traffic light*, tanto *traffic* como *light* mantém o seu sentido original e se combinam composicionalmente. Assim, poderíamos supor que outras combinações com significado análogo também seriam possíveis, como *intersection regulator*, porém isso não ocorre, pois o uso consagrou uma forma como uso correto. Como as frases institucionalizadas são inteiramente composicionais, têm variabilidade sintática completa. Por isso, uma técnica baseada em palavras-com-espaço teria o problema de proliferação léxica, e uma técnica inteiramente composicional aceitaria combinações inválidas.

2.7.2 Construções verbo-partícula

Construções verbo-partícula (CVPs) consistem de um verbo e uma ou mais partículas, como *set up*, *run away* e *look up*.

Estas construções são sintaticamente flexíveis no caso do verbo, e podem ser intransitivas (*I give up*) ou transitivas, necessitando uma oração nominal como complemento (*eat the cereal up*).

Uma outra característica dos CVPs é que a partícula não precisa necessariamente seguir o verbo, como por exemplo em *He looked the word up*. Porém quando o complemento de uma CVP transitiva é um pronome pessoal como *him*, *it* isto é obrigatório (*she made it up* mas não **she made up it*). Algumas combinações aceitam ambas as variações, enquanto outras aceitam apenas um ou outra configuração (por exemplo, *come up with an idea* mas não *come with an idea up*).

Como nos outros tipos de EMPs, uma descrição seguindo uma técnica de palavra-com-espaço geraria proliferação léxica, e uma técnica totalmente composicional teria problemas em aceitar combinações inválidas, e não funcionaria para as CVPs idiomáticas descritas adiante. Para a tradução automática, isso pode causar problemas, exemplificados a seguir[2]:

> *The story was made up.*
> A história foi *feita.
>
> *He looked the word up.*
> Ele *olhou a palavra *acima.
>
> *He looked up the word.*
> Ele *olhou para a palavra.

Dehe (2002) e Jackendoff (2002) classificam as CVPs em **composicionais**, **idiomáticas** ou **aspectuais**, de acordo com seu sentido.

[2]Google Translator: http://translate.google.com.br/ - Acesso em 04/2010.

2.7.2.1 CVPs composicionais

O significado da construção pode ser inferido completamente a partir de interpretações literais do verbo a da partícula que o compõe. Normalmente, a partícula tem um sentido direcional ou espacial como em *carry in* e *take down* em *She carried in the bags* e *He took the picture down* respectivamente.

2.7.2.2 CVPs aspectuais

A partícula provê um ponto final ao verbo, indicando que a ação ocorreu completamente, ou continuamente. Por exemplo *eat up*, *clear up* e *tear up*. Quando traduzidas para o português, essas construções muitas vezes mantém a mesma tradução do verbo (*eat up the cookie* = *comer a bolacha*) mas diferentemente do verbo sozinho, com a partícula ele denota completar a ação de comer, onde a bolacha é toda ingerida.

2.7.2.3 CVPs idiomáticas

Esse tipo de CVP tem um sentido que não é relacionado diretamente com aquele de seu verbo e partícula, por exemplo, *look up* como *consultar*, *root out* como *erradicar* e *play down* como *minimizar*. Esse tipo de CVP é muitas vezes traduzido para um verbo simples em português (*root out* = *erradicar*), e cria problemas para os métodos estatísticos de tradução automática, pois não se pode inferir seu significado a partir dos significados de seus componentes.

Na próxima seção serão discutidos alguns métodos propostos na literatura para a detecção e extração de EMPs a partir de corpora.

2.8 Extração de EMPs

Além do tratamento correto de EMPs existentes, muitas vezes se faz necessária a detecção e extração automática deste tipo de expressão de um corpus de texto. Como vimos, as EMPs podem ser produtivas, e devido ao grande número delas, a construção manual de recursos lingüísticos se torna cara, difícil e trabalhosa. Além disso, muitos métodos de NLP são intrinsecamente estatísticos, por exemplo a tradução automática estatística (GEER, 2005), onde todas as regras de tradução são descobertas automaticamente. Neste caso, torna-se vital detectarmos EMPs para que as regras geradas não causem traduções literais incorretas.

A seguir citaremos alguns dos experimentos recentes em extração de EMPs.

2.8.1 Construções verbo-partícula

2.8.1.1 Extração de CVPs

Extração de EMPs do tipo verbo-partícula, com valência (transitivo ou intransitivo) é o tema de Baldwin (2005). O objetivo do método é extrair entradas de CVPs completamente especificados (com informações sobre transitividade, por exemplo) a partir de corpora para inclusão automática em uma gramática ou dicionário. Isto poderia ser feito, por exemplo, para aumentar a cobertura da gramática e prepará-la para um novo domínio.

Primeiramente é testada uma técnica de extração baseada no uso de um etiquetador morfossintático. Depois de etiquetado o texto, é feita uma procura por etiquetas de partícula, e a partir daí, procura-se o verbo correspondente àquela partícula, que deve estar no máximo 5 posições à esquerda. Somente são aceitos nomes, pronomes, adjetivos e

Método	Tagger (CLAWS2)	Chunker	Chunk grammar	Parser	Combinado
Precisão	0.971	0.991	0.984	0.975	0.976
Revocação	0.740	0.740	0.897	0.721	0.963
escore-f	0.840	0.847	0.939	0.829	0.969

Tabela 2.2: Resultados da extração de CVPs (BALDWIN, 2005)

determinantes entre o verbo e a partícula. A valência (transitivo/intransitivo) é testada verificando se um sintagma nominal está entre o verbo e a partícula, ou para partícula imediatamente após o verbo, se um sintagma nominal sucede a partícula. Para cada verbo e partícula, a frequência de ocorrências classificadas como transitivas e a frequência de ocorrências classificadas como intransitivas são as duas *features* extraídas. Para a classificação propriamente dita, o sistema é treinado em um *gold standard* usando um *Memory Based Learner* (DAELEMANS et al., 2002). Este método obtém uma boa precisão para a simples extração de CVPs, ignorando a transitividade. A *revocação* no entanto é bastante baixa, devido à dificuldade dos etiquetadores morfossintáticos utilizados em distinguir entre partículas e preposições, quando o comportamento padrão em casos ambíguos é considerar como uma preposição.

Um segundo método testado é utilizando, em vez do etiquetador um ***chunker***, de forma a evitar as limitações do etiquetador. Um *chunker* particiona o texto em segmentos, sem tentar unir os segmentos em constituintes como faz o parser. O processo para identificar os candidatos a CVP é análogo ao do etiquetador, para cada *chunk* considerado como partícula, procura-se o verbo correspondente à esquerda. São geradas 7 *features* para realizar a classificação, de novo realizada com o *Memory Based Learner*. Este método possui uma precisão ainda melhor do que o anterior, mas devido às restrições sobre quais *tags* podem ocorrem entre verbos e partículas ou após a partícula, possui uma revocação um pouco menor.

A seguir, o *chunker* é utilizado novamente, porém sem procurar por um *chunk* classificado como partícula. Desta vez, a busca é por um verbo, e algo que possa ser uma partícula até 5 palavras a sua direita. Alguns testes linguísticos são realizados para eliminar combinações inválidas e gerar *features* que serão usadas na detecção. Esse método é denominado *chunk grammar* na tabela 2.2.

O último método básico utiliza a saída de um parser completo para extrair CVPs, que são caracterizados pela relação MOD entre um verbo e uma partícula, sendo a valência considerada a mesma do verbo. São geradas *features* para a frequência que a CVP é classificada como transitiva, e a frequência com que a CVP é classificada como intransitiva.

Por fim, como cada um dos métodos se destacou em algum aspecto, fez-se a classificação utilizando todas as *features* do métodos básicos concatenadas, o que proporcionou melhora nos resultados para quase todos os casos.

Os resultados estão resumidos na tabela 2.2

2.8.1.2 CVPs na web

Villavicencio (2005) usa a classificação de verbos em classes propostas por Levin (1993) para investigar como essas classes se combinam com partículas para formar CVPs, e quais classes e partículas são mais produtivas no que se refere à CVPs. Levin define 190 classes que abrangem 3100 verbos diferentes. Enquanto que neste trabalho não iremos investigar a produtividade das classes, a lista de CVPs atestadas pelo método é de grande

valia para verificarmos se os candidatos à CVP encontrados pelo nosso método são realmente válidos.

Todos os verbos incluídos nas classes de Levin são combinados com todas as partículas, e então estas combinações são atestadas (ou não) através de uma busca no Google.

Para manter as buscas simples e auto-contidas, nenhuma fonte de informação extra foi utilizada. Assim, cada busca por CVP é realizada utilizando somente à forma base do verbo como ela é listada nas classes de Levin (*eat up* mas não *ate up* ou *eats up*). Desta forma as estimativas obtidas são bastante conservadoras.

Como o objetivo é buscar somente CVPs e não verbos preposicionais (ex. *rely on*), e para não precisar pós-processar os resultados, a busca realizada usa um delimitador após a partícula, para evitar que esta seja seguida por um sintagma nominal, o que caracterizaria um verbo preposicional (ou CVP transitivo). Desta forma, para atestar CVPs intransitivos realizamos uma busca por:

<VERBO><PARTÍCULA><DELIMITADOR>

Para delimitador, são utilizadas preposições, que são freqüentes o suficiente para permitir a coleta de evidências (VILLAVICENCIO; COPESTAKE, 2003). Neste trabalho, foram utilizadas as preposições *for*, *from* e *with*.

A escolha do delimitador é importante pois delimita o contexto da busca para evitar ambigüidade com verbos preposicionais. No entanto, a adição de um termo extra também restringe consideravelmente o número de páginas retornadas.

Também foi utilizado um padrão de busca para encontrar possíveis CVPs que ocorram predominantemente na forma transitiva.

<VERBO><SN><PARTÍCULA><DELIMITADOR>

onde SN (sintagma nominal) é um pronome como *you*, *it* e *them*. O uso de pronomes simplifica a forma do SN, mas também abstrai do problema da ordem dos termos numa CVP, uma vez que os pronomes tendem a ocorrer adjacentes ao verbo em CVPs transitivas.

Foram consideradas atestadas as combinações verbo-partícula com mais de 5 ocorrências.

2.8.1.3 Sentido de CVP

Cook e Stevenson (2006) propõe um método de identificar qual sentido uma partícula está contribuindo em uma construção verbo-partícula. O artigo se foca na partícula *up* e em 4 sentidos que ela usualmente contribui; gera uma série de *features* como co-ocorrência de palavras e outras linguisticamente motivadas para cada ocorrência e realiza um treinamento com elas.

As *features* utilizadas englobam como os ***slots*** deste verbo são utilizados, ou seja, com que frequência este verbo tem um sujeito, objeto direto, objeto indireto, etc, associados, tanto ao verbo simples quanto à CVP em questão. Outra *feature* utilizada é a frequência com que cada partícula se combina com o verbo em questão, e também a distância média da partícula em relação ao verbo. Estas *features* são então comparadas com vetores de co-ocorrência de palavras, na sua capacidade de prever corretamente o sentido contribuído pela partícula numa CVP composicional.

Para a classificação das partículas em seu sentido, é usada uma *support vector machine*. São usados 389 CVPs, e as *features* para elas são geradas usando o BNC. As partículas são classificadas em 2 ou 3 classes distintas de acordo com a contribuição de

sentido. Os resultados encontrados são de que o uso das *features* linguísticas (slots do verbo e frequência de partículas) proporciona resultados melhores que de uma *baseline* baseada na distribuição uniforme dos sentidos e que do uso da co-ocorrência de palavras. Porém a precisão do método fica abaixo de 65 e 55 por cento para a classificação em 2 e 3 classes respectivamente.

2.8.1.4 Composicionalidade de CVP

Bannard (2005) propõe um método de identificar a composicionalidade de CVPs através de seu contexto léxico. A intuição é que se o contexto léxico da partícula for similar ao do CVP, a partícula estará contribuindo com sua semântica para o CVP, o mesmo valendo para o verbo.

Usando o *British National Corpus*, Bannard cria um contexto léxico para um conjunto de CVPs selecionado para testes, para os verbos destes CVPs e para as partículas destes CVPs. O contexto léxico de uma palavra p é um vetor contendo as palavras que ocorrem até 5 tokens antes ou depois de p e a sua freqüência.

Foi criado também um *gold standard*, onde para as mesmas CVPs especialistas e não especialistas (mas falantes nativos) julgaram se o verbo e/ou a partícula contribuem com sua semântica para a CVP. A lista de CVPs foi então ordenada de acordo com a contribuição semântica do verbo de acordo com os especialistas e separadamente de acordo com a valor co-seno entre o contexto léxico do verbo e da CVP, medida que deveria indicar também a contribuição semântica do verbo. O coeficiente de correlação de Spearman destas duas listas foi altamente significativo.

O mesmo procedimento foi adotado para as partículas, onde o coeficiente de correlação atingido não é significativo (por pouco) com um nível de confiança aceitável. Em ambos os caso o coeficiente de correlação foi maior do que o da *baseline* proposta.

2.9 Alinhamento Léxico e Expressões Multi-palavra

Em geral, na evolução dos métodos de alinhamento léxico, expressões do tipo n:m tem sido ignoradas, mantendo-se restrições a alinhamento dos tipos 1:1 ou 1:n. Em alguns alinhadores mais recentes, estas restrições não existem (FRASER; MARCU, 2006), ou seja, eles permitem alinhamentos arbitrários, no entanto, nenhuma tentativa especial é realizada para encontrar expressões multi-palavra.

Uma exceção a essa regra é o trabalho de Venkatapathy e Joshi (2006), que propõe um algoritmo parecido com o nosso. Eles propõe um técnica para identificar verbos frasais em Hindi a então alinhá-los a um verbo correspondente em Inglês. Para fazer isto, eles utilizam informações obtidas a partir de uma medida de associação estatística (*mutual information*) em um alinhador discriminativo, na forma de uma nova *feature function*. Os resultados são similares aos apresentados neste trabalho; a adição de tratamento especial para EMPs ocasiona uma pequena melhoria no desempenho geral. Não são apresentados resultados levando-se em conta apenas as expressões de interesse. Enquanto que o trabalho de Venkatapathy e Joshi (2006) adiciona uma *feature estatística* para reconhecimento de EMPs, neste trabalho apresentamos uma heurística de pós-processamento que utiliza conhecimentos linguísticos e métodos de aprendizado de máquina.

Caseli et al. (2009) buscam identificar EMPs a partir de um alinhamento léxico. Ou seja, o trabalho parte do pressuposto de que alinhamentos n:m em um alinhador léxico tem mais chance de serem EMPs. Um corpus paralelo é alinhado com o Giza++ em ambos os sentidos, e a heurística de simetrização da união é utilizada. Para identificar as EMPs, são

selecionados todos os alinhamentos $m : n$ onde $m > 1 \vee n > 1$ que ocorrem um número mínimo de vezes (o *threshold* básico é 2) e dentre estes é feita uma filtragem por etiqueta morfossintática, removendo alinhamentos com configuração de etiquetas morfossintáticas pouco prováveis de serem EMPs.

Um dos padrões de etiqueta morfossintática analisados é justamento o de verbo seguido de partícula e, de acordo com a avaliação realizada, até 97% dos candidatos extraídos eram de fato EMPs (para um threshold de 10 ocorrências). No entanto, essa é uma avaliação de extração de *tipos* de CVP, enquanto que em um alinhador, queremos uma avaliação de extração de *tokens*. Isso e o *threshold* de número de ocorrências explica em parte a aparente contradição entre a performance do Giza++ apresentada na seção 5 e os resultado do artigo.

3 RECURSOS

Nesse capítulo são descritos os recursos utilizados para o desenvolvimento deste trabalho, em especial os corpora de textos e ferramentas.

3.1 Corpus

Nos nossos experimentos, foi utilizado um corpus paralelo: o corpus de legendas de filmes Opus (TIEDEMANN, 2009). Esse corpus é composto de legendas de filmes livremente disponíveis na Web, que são automaticamente alinhadas por sentença.[1]

O corpus é formado predominantemente por sentenças curtas, o que facilita o trabalho de alinhamento, pois há menos opções de alinhamento possíveis, no entanto, muitas traduções livres ocorrem no corpus, o que dificulta a tarefa. Vejamos alguns exemplos de sentenças encontradas no corpus:

> *Life ' s not fair , is it ?*
> A vida é injusta , não é ?

Esse primeiro exemplo está na forma ideal, uma tradução quase literal, parte a parte, de uma sentença curta. Porém, também temos os casos com tradução livre:

> *I ' m afraid I ' m at the shallow end of the gene pool .*
> Receio não ser um bom representante da espécie .

Estes casos são difíceis para um alinhador e são uma das fontes de erros. É importante notar que a tradução livre é diferente de uma EMP; numa EMP temos uma correspondência de $m : n$ *com o mesmo significado*, por exemplo, *kick the bucket* é equivalente a *bater as botas* ou *morrer*; já numa tradução livre, como acima, as expressões contém uma correspondência mais fraca, elas não são equivalentes, mas cumprem a mesma função no contexto. Por fim, temos omissões e erros de alinhamento lexical:

> *A pouncing lesson . Oh very good .*
> Oh muito bem .

Este último exemplo contém uma omissão causada por erro no alinhamento sentencial. Decidimos não corrigir esse tipo de erro por não ser foco deste trabalho o alinhamento sentencial.

Para os experimentos realizados neste trabalho, utilizamos a parte inglês-português do corpus Opus depois de alguns passos de pré-processamento para remover marcações e tokenizar o texto. A Tabela 3.1 apresenta a quantidade total de sentenças e de tokens em cada língua, onde **en** significa Inglês e **pt** significa Português.

[1] O alinhamento automático por sentença não foi corrigido manualmente.

Língua	# Sentenças	# Tokens
en	351,106	3,077,113
pt	351,106	2,605,376

Tabela 3.1: O corpus de legendas Opus en-pt

3.2 Gold standard

Para o desenvolvimento do trabalho, precisamos de um *gold standard* de sentenças alinhadas lexicalmente de forma manual, tanto para estimar os parâmetros do alinhador discriminativo quanto para avaliá-lo. Nós selecionamos um subconjunto do corpus Opus para uma anotação manual.

Essa anotação foi realizada por dois falantes nativos de português fluentes em inglês, seguindo as orientações estabelecidas em Caseli et al. (2005), e marcando explicitamente as CVPs encontradas. Foi considerado uma CVP qualquer verbo em inglês seguido por uma partícula que tivessem que ser alinhados como uma unidade. A anotação foi realizada utilizando a ferramenta visual de anotação YAWAT (GERMANN, 2008). Para medir o grau de concordância entre os alinhadores, utilizamos e medida κ descrita na seção 2.6, e obtivemos um valor de 0.78. De acordo com (CARLETTA, 1996), entre outros autores, um valor de κ entre 0.67 e 0.80 indica uma boa concordância.

Para melhor avaliar o impacto das técnicas de identificação de CVPs, dois tipos diferentes de anotação foram realizados:

- a anotação **completa** contém todas as correspondências lexicais.
- a anotação **parcial** contém correspondências lexicais *apenas* dos CVPs.

Por exemplo, a sentença

The giraffe falls over.

seria anotada manualmente por completo no caso da anotação completa, ou somente as correspondências envolvendo *falls over* no caso da anotação parcial.

You're from the hospital.

seria anotada manualmente caso encontrada no corpus completo, mas seria ignorada no corpus parcial.

Essa anotação seletiva nos permite obter um bom número de alinhamentos de CVP, sem precisar anotar uma porção proibitivamente extensa do corpus. Um exemplo de anotação completo pode ser visto na figura 3.1.

Separamos cada *gold standard* em dois sub-conjuntos, o conjunto de ***tuning*** e o conjunto de **teste**. O conjunto de alinhamentos de *tuning* foi utilizado durante o desenvolvimento e ajuste de parâmetros do alinhador, enquanto que o conjunto de teste foi utilizado para avaliá-lo. Estes diferentes corpora anotados estão sumarizados na tabela 3.2. Para os corpora com anotação parcial, o número de sentenças não é igual ao número de sentenças de fato anotadas, pois apenas aquelas que continham CVPs o foram. As sentenças anotadas para a formação da cada um dos corpora foram selecionadas de diferentes partes do corpus de legendas Opus, isto é, de filmes diferentes, de forma a evitar um sobre-ajuste do alinhador a alguma porção específica do corpus.

$$\{(2,1)(3,2)(4,3)(4,4)(5,5)(6,6)(6,7)(7,6)(7,7)(8,8)\}$$

		1	2	3	4	5	6	7	8
		Life	's	not	fair	,	is	it	?
1	A		•						
2	vida			•					
3	é				•				
4	injusta				•				
5	,					•			
6	não						•	•	
7	é						•	•	
8	?								•

Figura 3.1: Exemplo de alinhamento do gold standard

Corpus	**# Sentenças**	**# Alinhamentos**	**# CVPs**
Corpus de *tuning*, alinhamento completo	600	4019	78
Corpus de *tuning*, alinhamento parcial	900	206	103
Corpus de teste, alinhamento completo	500	4395	107
Corpus de teste, alinhamento parcial	600	142	71

Tabela 3.2: Corpora utilizados nos experimentos

3.3 Ferramentas

3.3.1 Giza++

Nos métodos desenvolvidos neste trabalho são utilizadas informações induzidas de forma automática com o alinhador Giza++. O Giza++ é um extensão do programa Giza, desenvolvido pela equipe de Tradução de Máquina Estatística da John-Hopkins University. O Giza++[2] inclui uma série de extensões sobre o Giza (AL-ONAIZAN et al., 1999), mas o mais relevante para este é que implementa os modelos IBM (OCH; NEY, 2003). Assim, podemos utilizar o Giza++ para induzir os parâmetros dos modelos IBM, alguns deles utilizados neste trabalho. Como essa indução é realizada de forma não-supervisionada, ela foi feita utilizando todo o corpus Opus, incluindo as partes não anotadas para o *gold standard*. As informações obtidas e como elas são utilizadas estão descritas na seção 4.1.

3.3.2 Etiquetadores morfosintáticos

Para a realização dos experimentos projetados para este trabalho foi necessário obter a etiqueta morfossintática e o lema de cada palavra, para tanto utiliza-se um etiquetador morfossintático (EMS).

Um etiquetador morfossintático (*Part-of-speech tagger*) atribui uma etiqueta morfossintática a cada palavra presente no texto. Por exemplo, a saída do Tree Tagger (SCHMID, 1994) para sentença em inglês:

[2]Disponível em: http://code.google.com/p/giza-pp/

Does that flight server dinner?

```
Does    VVZ     do
that    DT      that
flight  NN      flight
server  NN      server
dinner  NN      dinner
?       SENT    ?
```

E para um sentença em português:

Este vôo serve jantar?

```
Este    DET     este
vôo     NOM     vôo
serve   V       servir
jantar  NOM     jantar
?       SENT    ?
```

Neste exemplo cada linha da saída corresponde a um token do texto de entrada. O primeiro elemento de cada linha é a palavra na sua forma original, o segundo elemento é a etiqueta morfossintática da palavra, e o terceiro é a palavra na sua forma lematizada.

O conjunto de etiquetas possíveis varia de acordo com a língua, e também o grau de detalhamento desejado. Para o inglês, um conjunto de *tags* bastante usado é o Penn Treebank tagset (MARCUS; MARCINKIEWICZ; SANTORINI, 1993), que é também utilizado pelo Tree Tagger. Grande parte das palavras pode pertencer à somente uma classe morfossintática. Porém, muitas das palavras mais comuns são ambíguas, podendo pertencer a mais de uma classe, por exemplo em:

Book that flight.

Book pode tanto ser um verbo (significando "reservar" como acima), ou um substantivo (significando "livro"). Portanto, torna-se necessário um método para desambiguar estes casos, o que normalmente é feito usando se o contexto, o que dá origem aos etiquetadores morfossintáticos.

Neste trabalho utilizamos o EMS Tree Tagger (SCHMID, 1994). Este etiquetador constrói automaticamente, a partir de um corpus de treinamento, uma árvore de decisão binária, sempre realizando aquela divisão que maximiza o ganho de informação sobre a etiqueta que está sendo prevista. Este processo é repetido recursivamente, até que um critério de parada é atingido, quando se chega à folha da árvore, que contém uma tabela das etiquetas possíveis para aquela sequência de regras e sua respectiva probabilidade.

O Tree Tagger foi escolhido por sua performance no estado da arte (mais de 96% de acertos para etiquetamento em Inglês), e da disponibilidade do próprio e de arquivos de parâmetros para outras línguas[3], inclusive o português. Para o português o Tree Tagger utiliza apenas 11 etiquetas que podem ser vistas no apêndice A. Para ambas as línguas, foi utilizada a configuração default do Tree Tagger.

[3]Disponível em: http://www.ims.uni-stuttgart.de/projekte/corplex/TreeTagger/, acesso em 10/03/2010

3.3.3 Weka

Neste trabalho, utilizamos o Weka para gerar árvores de classificação, utilizadas numa etapa de pós-processamento nos alinhadores Hill Post (seção 4.5), Hill VPC 1 e Hill VPC 2 (seção 4.6). Uma árvore de decisão (QUINLAN, 1986) pode ser definida como uma estrutura em forma de árvore que, dados uma série de valores de parâmetros de um objeto, o classifica em uma determinada classe. Cada folha da árvore representa uma **classe**, cada aresta representa uma **decisão** ou **característica**. Para classificar um objeto em sua classe, inicia-se a partir da raiz da árvore, e segue-se sempre o nodo cuja condição é verdadeira para o objeto, ao final chega-se numa folha que representa a classe deste objeto. Um exemplo de árvore de decisão pode ser visto na figura 4.9.

O Weka[4] (HALL et al., 2009) é uma suíte de softwares para aprendizado de máquina e descoberta de conhecimento. O Weka permite ao pesquisador importar uma base de dados a partir de uma série de formatos padrão, e experimentar com vários algoritmos distintos de aprendizado de máquina, fornecendo informações sobre a performance. Além disso, o Weka provê uma série de recursos de visualização dos dados quanto à distribuição de classes, que pode dar informações valiosas sobre o problema e, caso se utilize algoritmos de aprendizado de máquina que gerem árvores de decisão, permite visualizá-las.

[4]Disponível em: http://www.cs.waikato.ac.nz/ml/weka/

4 ALINHADOR

O novo alinhador léxico automático proposto nesse trabalho é formado por um núcleo discriminativo independente de língua e um pequeno número de heurísticas de pós-processamento dependentes da língua. Em um alinhador discriminativo, uma série de *feature functions* são combinadas formando um escore de alinhamento global, ficando a tarefa de encontrar aquele alinhamento que possui o máximo escore de alinhamento global. A grande vantagem do alinhamento discriminativo é que ele torna mais fácil a inclusão de novas *features*.

O processo de alinhamento proposto é divido nas seguintes partes:

- *feature functions* (descritas na seção 4.1) - as *feature functions* individuais, que informação relevante cada uma busca capturar e como é obtido seu valor
- *combinação* (seção 4.2) - as *feature functions* individuais são combinadas para formar uma função de escore global;
- *busca* (descrita na seção 4.3) - dada uma função de escore global, o método busca pelo alinhamento que maximiza tal função;
- *otimização* (seção 4.4) - os respectivos pesos das *feature functions* são otimizados; e
- *pós-processamento* (seções 4.5, 4.6) - as heurísticas de pós-processamento são aplicadas.

4.1 Features

O alinhador léxico proposto neste trabalho tem 4 *feature functions* descritas a seguir. Primeiramente, trabalhamos com os 3 primeiras *features* e foi com estas 3 *features* que testamos todas as heurísticas de pós-processamento. A quarta *feature* foi adicionada posteriormente, para testar a função de otimização para diferentes valores de α, posto que a mesma é realizada diretamente em relação ao escore-f.

4.1.1 Probabilidade de tradução de palavra

A medida de probabilidade de tradução de palavra é uma medida da probabilidade de que uma palavra em Inglês seja traduzido para uma palavra especifica em Português, e vice-versa. Para estimar essas probabilidades utilizamos os modelos IBM com o Giza++ em ambas as direções (en $\rightarrow$ pt e pt $\rightarrow$ en), e usamos a média das probabilidades de tradução encontrados pelo Giza++. Como os valores estão todos próximos entre 0 e 1,

utilizamos o logaritmo da probabilidade de tradução de palavra de cada alinhamento individual, e o valor final da *feature* é composto da soma destes valores. Mais formalmente

$$f_{trad}(\mathbf{a}, \mathbf{en}, \mathbf{pt}, \theta) = \sum_{(i,j)\in a} \log p_{trad}(en[i], pt[j], \theta) \tag{4.1}$$

onde $ptrad(en[i], pt[j], \theta)$ corresponde à probabilidade de tradução entre as palavras na posição i e na posição j, passadas no parâmetro θ[1].

4.1.2 Fertilidade

A fertilidade é uma medida da probabilidade de uma palavra em Inglês (Português) ser traduzida para 0,1,2,3,... palavras em Português (Inglês). Por exemplo, tomemos o token *to* em Inglês. Ele pode ser omitido na tradução para o Portguês ($fertility = 0$), ou ele pode ser traduzido para uma única palavra em Português ($fertility = 1$), ou pode ser traduzido para uma expressão, com 2 ou mais palavras ($fertility = 2, 3, \ldots$). Cada um destes casos possui uma probabilidade, que é capturada pela *feature* da fertilidade, que também é obtida a partir do Giza++. Utilizamos a soma do logaritmo das fertilidades de cada palavra no alinhamento como o valor da *feature*.

4.1.3 Coerência

A medida de coerência assume que palavras em uma sentença ocorrem na sua tradução aproximadamente na mesma posição, e caso a posição mude, normalmente a distância é pequena ou um bloco inteiro de palavras é movido. O corpus de *tuning* completo foi utilizado para estimar as probabilidades de, dado um alinhamento individual (x, y), qual a probabilidade de existir um alinhamento individual $(x+1, y)$, $(x+1, y+1)$, $(x+1, y-1)$ e assim por diante, até $y-3$ e $y+3$. Essas médias foram calculadas também utilizando a linha $x-1$, apenas invertendo o sinal, ou seja $(x+i, y-j)$ é equivalente a $(x-i, y+j)$, pois em média os alinhamentos seguem a diagonal principal. Por esse mesmo motivo essas probabilidades foram extrapoladas para $(x-2), (x-3)$, etc, fazendo um *shift* nas probabilidades, que podem ser vistas na tabela 4.1.

	$y-3$	$y-2$	$y-1$	y	$y+1$	$y+2$	$y+3$
$x-3$	0.553	0.177	0.107	0.041	0.015		
$x-2$	0.157	0.553	0.177	0.107	0.041	0.015	
$x-1$	0.061	0.157	0.553	0.177	0.107	0.041	0.015
x				•			
$x+1$	0.015	0.041	0.107	0.177	0.553	0.157	0.061
$x+2$		0.015	0.041	0.107	0.177	0.553	0.157
$x+3$			0.015	0.041	0.107	0.177	0.553

Tabela 4.1: Alinhamento inicial

No nosso método, é considerada somente a **maior** probabilidade de cada linha, ou 0.005 caso não exista nenhum alinhamento individual na linha em questão, e elas são multiplicadas formando o escore de coerência para o alinhamento individual (x, y). Esse

[1] O parâmetro θ foi definido na seção 2.4.1 e representa uma forma genérica de passar parâmetros para uma *feature function*

escore é obtido para todos os alinhamentos individuais existentes, a soma dos logaritmos destes escores forma o valor da *feature* coerência.

4.1.4 Bônus

O bônus é uma *feature* que soma um valor constante para cada alinhamento individual, ou seja, quanto mais alinhamentos individuais o alinhamento possui, maior será seu valor, de forma a proporcionar uma possibilidade de privilegiar a precisão ou a revocação. Caso a precisão seja privilegiada o peso dessa *feature* encontrada na otimização terá um valor baixo, caso seja privilegiado a revocação, terá um valor alto.

$$f_{bonus}(a, en, pt, \theta) = \sum_{(i,j)\in a} 2 \tag{4.2}$$

4.2 Combinação

As diferentes *feature functions* do alinhador são combinadas para formar o escore combinado global. Isto é feito com uma simples combinação linear:

$$escore_{global} = \sum_{i} \lambda_i * escore_i \tag{4.3}$$

onde $escore_i$ é o escore da *feature function* i para a sentença sendo processada e λ_i é o peso da *feature function* i.

Por exemplo, suponha que tenhamos 3 *feature functions*, resultando nos escores 1.75, 2.34 e 0.23 respectivamente para um determinado alinhamento, e que os pesos sejam 0.5, 0.9 e 0.2 respectivamente. Substituindo na equação 4.3 temos:

$$escore_{global} = 0.5 * 1.75 + 0.9 * 2.34 + 0.2 * 0.23 \tag{4.4}$$

O que resulta em um escore combinado global de 3.027.

4.3 Busca

No alinhador léxico discriminativo proposto, a busca pelo melhor alinhamento é feita utilizando um algoritmo de *hillclimbing* relativamente simples.[2] Portanto, começando com um alinhamento NULO (onde não existe correspondência entre nenhuma palavra) o algoritmo busca na vizinhança do alinhamento atual um alinhamento cujo escore combinado global seja maior. Caso seja encontrado um ou mais, aquele alinhamento com o escore combinado global máximo é selecionado como o novo alinhamento corrente, e o processo é repetido. Caso não seja encontrado nenhum alinhamento com escore superior, o algoritmo termina.

A vizinhança é definida como qualquer alinhamento que possa ser atingido a partir das operações básicas descritas a seguir. Dado que x corresponde à posição de uma palavra na sentença de origem e y e z correspondem às posições de palavras na sentença de destino, temos quatro possíveis operações:

- **adicionar ligação**, onde um alinhamento individual é adicionado;

[2] Mesmo que algoritmos de *hillclimbing* algumas vezes sofram com máximos locais, eles em geral possuem boa performance em tarefas parecidas com essa (BROWN et al., 1993).

- **remover ligação**, onde um alinhamento individual existente é removido;
- **mover linha**, onde um alinhamento individual existente é movido ao longo da linha da matriz de alinhamento, ou seja, se tivermos um alinhamento (x, y) ele é removido e cria-se um alinhamento (x, z) para todos os zs possíveis onde nenhum alinhamento existe; e
- **mover coluna**, que é como mover linha, mas onde é mantida fixa a posição da palavra no destino em vez de na origem.

Estas operações são executadas na ordem acima, primeiramente todas as adições, seguido por todas as remoções e assim por diante, mas como estamos interessados no máximo, essa ordem poderia ser mudada sem alteração no resultado final. Para ilustrar essas quatro operações, vamos considerar o seguinte exemplo:

Help$_1$ *!*$_2$
Socorro$_1$!$_2$

Considere também que já possuímos um único alinhamento individual, $(1, 1)$ entre os primeiros tokens de cada sentença, isto é , entre *Help* e *Socorro*. Em outras palavras, o alinhamento corrente é $\{(1, 1)\}$[3] , como pode ser visto na figura 4.1.

		1	2
		Help	!.
1	Socorro	•	
2	!		

Figura 4.1: Alinhamento inicial

A operação *adicionar ligação* iria considerar adicionar todos os novos alinhamentos individuais possíveis, primeiro $(1, 2)$ (resultando em $\{(1, 1), (1, 2)\}$), depois $(2, 1)$ e finalmente $(2, 2)$. Estes três alinhamentos estão representados de forma compacta na figura 4.2, onde cada $\star$ indica um alinhamento individual que é adicionado.

		1	2
		Help	!.
1	Socorro	•	$\star$
2	!	$\star$	$\star$

Figura 4.2: Adicionando alinhamentos

		1	2
		Help	!.
1	Socorro	•	
2	!		

Figura 4.3: Removendo alinhamentos

A operação *remover ligação* iria considerar remover todos os alinhamentos existentes, neste caso, somente $(1, 1)$ pode ser removido (resultando em $\{\}$, o alinhamento NULO), exemplificado pela figura 4.3, com o elemento removido em cinza.

[3]Note que o alinhamento completo entre duas sentenças é composto pelo conjunto de todos os alinhamentos individuais, como descrito na seção 2.1.

Mover linha iria tentar mover todos os alinhamentos existentes ao longo da mesma linha, no nosso exemplo tentaria mover $(1, 1)$ para $(1, 2)$, como visto na figura 4.4. Finalmente, *mover coluna* tentaria mover $(1, 1)$ para $(2, 1)$, exemplificado na figura 4.5. Cada um dos vizinhos gerados dessa forma é avaliado com a função de escore combinado global, e aquele com o escore máximo torna-se o novo alinhamento corrente.

		1	2
		Help	! .
1	Socorro	•	⋆
2	!		

Figura 4.4: Movendo alinhamento na linha

		1	2
		Help	! .
1	Socorro	•	
2	!	⋆	

Figura 4.5: Movendo alinhamento na coluna

4.3.1 Exemplo real de alinhamento

Iremos agora mostrar um alinhamento real completo realizado com o algoritmo de busca descrito. Considere a sentença

> *And*$_1$ *you*$_2$ *...*$_3$ *shall*$_4$ *never*$_5$ *see*$_6$ *the*$_7$ *light*$_8$ *of*$_9$ *another*$_{10}$ *day*$_{11}$ *...*$_{12}$.$_{13}$
> E$_1$ tu$_2$ ..$_3$.$_4$ Não$_5$ verás$_6$ a$_7$ luz$_8$ do$_9$ dia$_{10}$ novamente$_{11}$ ..$_{12}$.$_{13}$ Adeus ..$_{14}$.$_{15}$

Conforme descrito, o algoritmo começa com um alinhamento vazio $\{\}$ e segue o processamento.

- Alinhamento corrente: $\{\}$, Escore: -75.376026
  - Tentar adicionar: (1, 1) Escore: -72.024756
  - Tentar adicionar: (1, 2) Escore: -81.257941
  - Tentar adicionar: (1, 3) Escore: -79.400871
  - Tentar adicionar: (1, 4) Escore: -79.308020
  - Tentar adicionar: (1, 5) Escore: -80.576603
  - Tentar adicionar: (1, 6) Escore: -84.744355
  - Tentar adicionar: (1, 7) Escore: -77.066895
  - Tentar adicionar: (1, 8) Escore: -92.081734
  - Tentar adicionar: (1, 9) Escore: -83.109593
  - Tentar adicionar: (1, 10) Escore: -82.784487
  - ...(mais 193 *Tentar adicionar*)
  - Tentar adicionar: (13, 12) Escore: -76.817989
  - Tentar adicionar: (13, 13) Escore: -72.144266
  - Tentar adicionar: (13, 14) Escore: -88.138878
  - Tentar adicionar: (13, 15) Escore: -76.817989
  - Tentar adicionar: (13, 16) Escore: -72.144266

  - **Selecionar melhor alinhamento → adicionar (1, 1)**

- Alinhamento corrente: $\{(1, 1)\}$, Escore: -72.024756
  - Tentar remover: (1, 1) Escore: -75.376026
  - Tentar adicionar: (1, 2) Escore: -80.847866
  - Tentar adicionar: (1, 3) Escore: -78.990796
  - Tentar adicionar: (1, 4) Escore: -78.897945
  - Tentar adicionar: (1, 5) Escore: -80.166528
  - Tentar adicionar: (1, 6) Escore: -84.334281
  - ... (mais 92 *Tentar adicionar*)
  - Tentar adicionar: (13, 4) Escore: -68.792996
  - ... (mais 110 *Tentar adicionar*)
  - Tentar mover coluna: (1, 1) ⇒ (2, 1) Escore: -76.467034
  - Tentar mover coluna: (1, 1) ⇒ (3, 1) Escore: -79.492954
  - Tentar mover coluna: (1, 1) ⇒ (4, 1) Escore: -80.997357
  - Tentar mover coluna: (1, 1) ⇒ (5, 1) Escore: -96.312192
  - Tentar mover coluna: (1, 1) ⇒ (6, 1) Escore: -96.535611
  - Tentar mover coluna: (1, 1) ⇒ (7, 1) Escore: -79.805429
  - Tentar mover coluna: (1, 1) ⇒ (8, 1) Escore: -96.680917
  - Tentar mover coluna: (1, 1) ⇒ (9, 1) Escore: -79.699309
  - Tentar mover coluna: (1, 1) ⇒ (10, 1) Escore: -96.429124
  - Tentar mover coluna: (1, 1) ⇒ (11, 1) Escore: -96.489350
  - Tentar mover coluna: (1, 1) ⇒ (12, 1) Escore: -79.492954
  - Tentar mover coluna: (1, 1) ⇒ (13, 1) Escore: -77.437440
  - Tentar mover linha: (1, 1) ⇒ (1, 2) Escore: -81.257941
  - Tentar mover linha: (1, 1) ⇒ (1, 3) Escore: -79.400871
  - Tentar mover linha: (1, 1) ⇒ (1, 4) Escore: -79.308020
  - Tentar mover linha: (1, 1) ⇒ (1, 5) Escore: -80.576603
  - Tentar mover linha: (1, 1) ⇒ (1, 6) Escore: -84.744355
  - Tentar mover linha: (1, 1) ⇒ (1, 7) Escore: -77.066895
  - Tentar mover linha: (1, 1) ⇒ (1, 8) Escore: -92.081734
  - Tentar mover linha: (1, 1) ⇒ (1, 9) Escore: -83.109593
  - Tentar mover linha: (1, 1) ⇒ (1, 10) Escore: -82.784487
  - Tentar mover linha: (1, 1) ⇒ (1, 11) Escore: -98.979232
  - Tentar mover linha: (1, 1) ⇒ (1, 12) Escore: -79.400871
  - Tentar mover linha: (1, 1) ⇒ (1, 13) Escore: -79.308020
  - Tentar mover linha: (1, 1) ⇒ (1, 14) Escore: -86.328818

  - Tentar mover linha: (1, 1) $\Rightarrow$ (1, 15) Escore: -79.400871
  - Tentar mover linha: (1, 1) $\Rightarrow$ (1, 16) Escore: -79.308020
  - **Selecionar melhor alinhamento $\rightarrow$ adicionar (13, 4)**

- Alinhamento corrente: $\{(1,1)(13,4)\}$, Escore: -68.792996
  - ...(261 vizinhos analisados)
  - **Selecionar melhor alinhamento $\rightarrow$ adicionar (11, 10)**

- Alinhamento corrente: $\{(1,1)(11,10)(13,4)\}$, Escore: -66.245417
  - ...(289 vizinhos analisados)
  - **Selecionar melhor alinhamento $\rightarrow$ adicionar (9, 9)**

- Alinhamento corrente: $\{(1,1)(9,9)(11,10)(13,4)\}$, Escore: -63.713699
  - ...(316 vizinhos analisados)
  - **Selecionar melhor alinhamento $\rightarrow$ adicionar (8, 8)**

- Alinhamento corrente: $\{(1,1)(8,8)(9,9)(11,10)(13,4)\}$, Escore: -60.850689
  - ...(343 vizinhos analisados)
  - **Selecionar melhor alinhamento $\rightarrow$ adicionar (7, 7)**

- Alinhamento corrente: $\{(1,1)(7,7)(8,8)(9,9)(11,10)(13,4)\}$, Escore: -57.203721
  - ...(370 vizinhos analisados)
  - **Selecionar melhor alinhamento $\rightarrow$ adicionar (6, 6)**

- Alinhamento corrente: $\{(1,1)(6,6)(7,7)(8,8)(9,9)(11,10)(13,4)\}$, Escore: -54.700607
  - ...(397 vizinhos analisados)
  - **Selecionar melhor alinhamento $\rightarrow$ adicionar (2, 2)**

- Alinhamento corrente: $\{(1,1)(2,2)(6,6)(7,7)(8,8)(9,9)(11,10)(13,4)\}$, Escore: -52.747953
  - ...(424 vizinhos analisados)
  - **Selecionar melhor alinhamento $\rightarrow$ adicionar (3, 3)**

- Alinhamento corrente: $\{(1,1)(2,2)(3,3)(6,6)(7,7)(8,8)(9,9)(11,10)(13,4)\}$, Escore: -49.680730
  - ...(451 vizinhos analisados)
  - **Selecionar melhor alinhamento $\rightarrow$ adicionar (5, 5)**

- Alinhamento corrente: $\{(1,1)(2,2)(3,3)(5,5)(6,6)(7,7)(8,8)(9,9)(11,10)(13,4)\}$, Escore: -47.916296
  - ...(478 vizinhos analisados)
  - **Selecionar melhor alinhamento $\rightarrow$ adicionar (12, 12)**

$\{(1,1)(2,2)(3,3)(5,5)(6,6)(7,7)(8,8)(9,9)(11,10)(12,12)(13,4)\}$

		1	2	3	4	5	6	7	8	9	10	11	12	13
		And	you	...	shall	never	see	the	light	of	another	day	::	.
1	E	•												
2	tu		•											
3	..			•										
4	.													•
5	Não					•								
6	verás						•							
7	a							•						
8	luz								•					
9	do									•				
10	dia											•		
11	novamente													
12	..												⋆	
13	.													
14	Adeus													
15	..													
16	.													

Figura 4.6: Alinhamento depois de adicionar (12, 12)

- Alinhamento corrente (figura 4.6):
  $\{(1,1)(2,2)(3,3)(5,5)(6,6)(7,7)(8,8)(9,9)(11,10)(12,12)(13,4)\}$, Escore: -46.272797
  - ...(505 vizinhos analisados)
  - **Selecionar melhor alinhamento $\rightarrow$ mover coluna (13, 4) $\Rightarrow$ (13,13)**

- Alinhamento corrente (figura 4.7):
  $\{(1,1)(2,2)(3,3)(5,5)(6,6)(7,7)(8,8)(9,9)(11,10)(12,12)(13,13)\}$, Escore: -44.324037
  - ...(605 vizinhos analisados)
  - **Selecionar melhor alinhamento $\rightarrow$ adicionar (3,15)**

- Alinhamento corrente:
  $\{(1,1)(2,2)(3,3)(3,15)(5,5)(6,6)(7,7)(8,8)(9,9)(11,10)(12,12)(13,13)\}$, Escore: -43.769915
  - ...(530 vizinhos analisados)
  - **Selecionar melhor alinhamento $\rightarrow$ adicionar (10,11)**

- Alinhamento corrente:
  $\{(1,1)(2,2)(3,3)(3,15)(5,5)(6,6)(7,7)(8,8)(9,9)(10,11)(11,10)(12,12)(13,13)\}$,
  Escore: -43.283502

$\{(1,1)(2,2)(3,3)(5,5)(6,6)(7,7)(8,8)(9,9)(11,10)(12,12)(13,13)\}$

		1	2	3	4	5	6	7	8	9	10	11	12	13
		And	you	...	shall	never	see	the	light	of	another	day	...	.
1	E	•												
2	tu		•											
3	..			•										
4	.													●
5	Não					•								
6	verás						•							
7	a							•						
8	luz								•					
9	do									•				
10	dia											•		
11	novamente													
12	..												•	
13	.													⋆
14	Adeus													
15	..													
16	.													

Figura 4.7: Alinhamento depois de mover coluna (13,4) ⇒ (13,13)

  - ...(557 vizinhos analisados)
  - **Selecionar melhor alinhamento → adicionar (3,4)**

- Alinhamento corrente:
  $\{(1,1)(2,2)(3,3)(3,4)(3,15)(5,5)(6,6)(7,7)(8,8)(9,9)(10,11)(11,10)(12,12)(13,13)\}$,
  Escore: -43.216788
  - ...(580 vizinhos analisados)
  - **Selecionar melhor alinhamento → mover coluna (3,15) ⇒ (12,15)**

- Alinhamento corrente:
  $\{(1,1)(2,2)(3,3)(3,4)(5,5)(6,6)(7,7)(8,8)(9,9)(10,11)(11,10)(12,12)(12,15)(13,13)\}$,
  Escore: -42.928464
  - ...(582 vizinhos analisados)
  - **Selecionar melhor alinhamento → adicionar (3, 16)**

- Alinhamento corrente:
  $\{(1,1)(2,2)(3,3)(3,4)(3,16)(5,5)(6,6)(7,7)(8,8)(9,9)(10,11)(11,10)(12,12)(12,15)(13,13)\}$,
  Escore: -42.537656
  - ...(607 vizinhos analisados)
  - **Nenhum alinhamento melhor encontrado.**

- Alinhamento corrente (figura 4.8):
  $\{(1,1)(2,2)(3,3)(3,4)(3,16)(5,5)(6,6)(7,7)(8,8)(9,9)(10,11)(11,10)(12,12)(12,15)(13,13)\}$,
  Escore: -42.537656

$\{(1,1)(2,2)(3,3)(3,4)(3,16)(5,5)(6,6)(7,7)(8,8)(9,9)(10,11)(11,10)(12,12)(12,15)(13,13)\}$

		1	2	3	4	5	6	7	8	9	10	11	12	13
		And	you	...	shall	never	see	the	light	of	another	day	..	.
1	E	•												
2	tu		•											
3	..			•										
4	.			•										
5	Não					•								
6	verás						•							
7	a							•						
8	luz								•					
9	do									•				
10	dia											•		
11	novamente										•			
12	..												•	
13	.													•
14	Adeus													
15	..												•	
16	.			•										

Figura 4.8: Alinhamento final

Este exemplo dá uma ideia de como o algoritmo de busca converge para o seu resultado. Inicialmente, domina a *feature* de probabilidade de tradução de palavra, pois o alinhamento está totalmente vazio, e por isso temos muitas operações de adicionar ligação. Uma após a outra as palavras vão sendo ligadas àquelas com as quais possuem grande probabilidade de tradução, em ordem decrescente. Entretanto, quando muitos alinhamentos já foram criados, as *features* (globais) de fertilidade e coerência (como definidas na seção 4.1) passam a ter uma influência maior sobre qual vizinho será selecionado, causando operações de mover coluna ou linha e eventualmente remover um alinhamento, como pode ser evidenciado nas figuras 4.6 e 4.7.

Este alinhamento é um exemplo real extraído do corpus de teste, podemos ver que ele possui um erro (provavelmente de OCR), reconhecendo "..." como ".. .", além de uma omissão no final, referente aos tokens 14, 15 e 16 da sentença em português.

4.4 Otimização

É necessário escolher pesos λ_i que ocasionem a melhor performance possível. Para isso, desenvolvemos um algoritmo de otimização inspirado em métodos de descida (su-

bida) de gradiente (WEISSTEIN, 2010) que otimiza os pesos diretamente com respeito ao escore-f:

- Inicializar todos $\lambda_i = 0.5$, resultando no vetor de pesos corrente λ
- Inicializar $step\_size = 0.05$
- LOOP:
  - Criar candidatos de vetor de pesos adicionando ou subtraindo $step\_size$ de cada λ_i
  - Calcular escore-f para cada vetor de pesos candidato
  - Selecionar novo vetor w com o maior escore-f entre os candidatos
  - Se w mudou, **repetir**
  - Senão
    - Se $step\_size < 0.01$ então **terminar**
    - Fazer $step\_size = step\_size/2$
    - **Repetir**

Seguindo este processo, o algoritmo de otimização atualiza os vetor de pesos de forma iterativa, de maneira a maximizar o escore-f, e quando nenhuma melhoria é possível, reduz o tamanho do passo para refinar a busca. Assume-se que o escore-f varia suavemente e monotonicamente com variações em cada peso w_i, porém isso é uma simplificação e consequentemente não há garantia de que o algoritmo irá encontrar o melhor vetor de pesos. Não obstante, bons resultados foram obtidos utilizando um *gold standard* de tamanho razoável e valores iniciais definidos empiricamente.

Como descrito até agora o algoritmo é independente de língua, e chamamos essa versão de **Hill Base** no texto.

4.5 Heurísticas

Uma análise empírica da performance do alinhador Hill Base indicou problemas com alguns tipos específicos de alinhamento, em particular com alinhamentos mais complexos ($m : n$) como CVPs. Portanto, para aprimorar o alinhamento, uma série de heurísticas, a maioria dependentes de língua, foi adicionada ao núcleo independente de língua do alinhador. Para evitar adicionar um excesso de *features* ao núcleo, que teriam de ser executadas em cada passo do processo de *hillclimbing*, as heurísticas foram implementadas em um passo de pós processamento, em que os alinhamentos são atualizados com uma função apenas uma vez, no final do processo. Como uma consequência, torna-se fácil treinar classificadores de aprendizado de máquina que adicionam ou removem alinhamentos individuais. Essas heurísticas são:

- **interseção:** adiciona qualquer alinhamento individual encontrado na interseção dos alinhamentos do Giza++ em ambas as direções.
- **Pron + V $\Leftrightarrow$ V:** alinha pronomes pessoais (Pron) seguidos por verbos (V) em inglês para verbos em português, posto que em português esse tipo de pronome precedendo o verbo é opcional e muitas vezes omitido e a informação fornecida por

ele é codificada na inflexão do verbo (p.e. *I needed* ⇔ *precisei*, ou *We realized* ⇔ *percebemos*).

Para criar esta heurística, foi construído um classificador baseado em árvores de decisão utilizando o algoritmo J48 (ZHAO; ZHANG, 2008) implementado no Weka (HALL et al., 2009). Ele foi treinado utilizando todas as sequências Pron + V e várias *features* relevantes coletadas do corpus de *tuning*.

- **pontuação:** alinha sinais de pontuação utilizando um classificador baseado em árvores de decisão, uma vez que alguns dos erros mais frequentes realizados pelo alinhador envolviam pontuação, o que pode ser explicado em parte pelo fato de que as traduções do coprus frequentemente omitem ou adicionam pontuação. As *features* relevantes utilizadas são a coerência descrita na seção 4.1 e uma série de *features* booleanas para os vizinhos imediatos do sinal de pontuação em questão, com a ideia de que se eles estiverem alinhados, a probabilidade de que o sinal de pontuação também deve ser alinhado é maior.

Nos referimos à este alinhamento resultante da fase de pós-processamento como **Hill Post**.

4.6 Processamento de CVPs

4.6.1 Features

Para verificar o impacto de adicionar informações dependentes de língua sobre construções complexas, neste trabalho focamos no alinhamento de construções verbo-partícula, implementando duas heurísticas alternativas para auxiliar a detecção de CVPs, que são executadas **após** as heurísticas do alinhador Hill Post. Para fazer isto, foram coletadas do corpus de *tuning* todos candidatos a CVP e o seguinte conjunto de *features*:

- **verbo alinhado**: caso *verdadeiro* indica que o verbo em inglês foi alinhado ao verbo em português pelo alinhador Hill Post, caso contrário é *falso*.
- **partícula alinhada**: é *verdadeiro* se a partícula em inglês está alinhada ao verbo em português pelo alinhador Hill Post, caso contrário é *falso*.
- **verbo giza org→dst**: onde *org* e *dst* são a origem (inglês) e o destino (português) respectivamente, e indica o mesmo que *verbo alinhado*, porém para os alinhamentos do Giza++ inglês → português.
- **verbo giza dst→org**: como acima, porém usando o sentido de alinhamento inverso do Giza++.
- **partícula giza org→dst**: o mesmo que *partícula alinhada*, mas para os alinhamentos do Giza++ inglês → português.
- **partícula giza dst→org**: como acima, porém usando o sentido de alinhamento inverso do Giza++.
- **escore do verbo**: a *probabilidade de tradução de palavra* do verbo em inglês com o verbo em português, como explicado na seção 4.1.

- **escore da partícula**: a *probabilidade de tradução de palavra* da partícula em inglês com o verbo em português.
- **verbo outro**: a *probabilidade **máxima** de tradução de palavra* do verbo em inglês com qualquer palavra diferente do verbo em Português sendo considerado.
- **partícula outro**: a *probabilidade **máxima** de tradução de palavra* da partícula em inglês com qualquer palavra diferente do verbo em português sendo considerado.
- **partícula POS**: a *part-of-speech* (etiqueta morfossintática) da partícula, de acordo com o Tree Tagger. O Tree Tagger possui uma POS específica para partículas, RP, porém muitas vezes ele classifica partículas incorretamente como IN (preposições) ou RB (advérbios), e vice-versa. Portanto utilizamos a POS de fato atribuída pelo Tree Tagger como uma das *features*.

4.6.2 Heurísticas

A anotação da etiqueta morfossintática dos corpora com o Tree Tagger possui uma etiqueta específica para partículas, RP, no entanto como já foi mencionado, partículas são frequentemente ambíguas com preposições (IN) ou com advérbios (RB). Portanto, nós realizamos duas heurísticas diferentes para encontrar CVPs a partir dos alinhamentos gerados pelo alinhador Hill Post.

Na primeira heurística, **Hill CVP 1**, buscamos por qualquer alinhamento entre verbos em Inglês e Português, e selecionamos aqueles em que uma palavra nas 3 posições seguintes ao verbo em inglês foram etiquetados como RP, RB ou IN. Um classificador de árvore de decisão (o algoritmo J48 implementado no Weka) foi treinado para determinar se um dado candidato a CVP com seu conjunto de *features* correspondente é um CVP genuíno ou não. Esse classificador foi adicionado como uma heurística de pós-processamento ao alinhador proposto. Deve ser ressaltado que, *dado* um alinhamento individual contendo um verbo em Inglês, essa heurística encontra partículas próximas e decide se elas devem ser adicionadas ao verbo formando uma CVP (classificador $\rightarrow$ verdadeiro) ou não (classificador $\rightarrow$ falso). Se o verbo não estiver alinhado para começar, nada é feito.

Na segunda heurística, **Hill CVP 2**, não é mais necessário que os verbos em ambas as línguas estejam alinhados. Diferentemente da primeira heurística, procuramos por verbos seguidos de alguma palavra etiquetadas como RP, RB ou IN em algumas das 3 posições seguintes, e geramos um candidato a CVP com o verbo, a partícula e **cada** um dos verbos encontrados na sentença em português. Esses candidatos são, então, utilizados para treinar um classificador que é adicionado como uma heurística de pós-processamento, e pode ser visto na figura 4.9 onde *d* significa *dst* (destino), *o* significa *org* (origem), *f* significa falso e *v* verdadeiro. Se o classificador tiver saída *verdadeiro*, alinhamos tanto o verbo quanto a partícula em inglês ao verbo em português, caso a saída seja *falso*, removemos o alinhamento da partícula com o verbo em Português.

Consideremos o seguinte par de sentenças, com as etiquetas morfossintáticas marcadas como na saída do Tree Tagger:

My_PP$_1$ *interest_NN*$_2$ *in_IN*$_3$ *Johnny_NP*$_4$ *is_VBZ*$_5$ *only_RB*$_6$ *in_IN*$_7$ *finding_VVG*$_8$ *out_RP*$_9$ *if_IN*$_{10}$ *he_PP*$_{11}$ *'_POS*$_{12}$ *s_JJS*$_{13}$ *alive_JJ*$_{14}$ *or_CC*$_{15}$ *if_IN*$_{16}$ *he_PP*$_{17}$ *'_POS*$_{18}$ *s_NNS*$_{19}$ *dead_JJ*$_{20}$ *._SENT*$_{21}$

Meu_ADJ$_1$ interesse_NOM$_2$ é_V$_3$ descobrir_V$_4$ se_P$_5$ está_V$_6$ vivo_ADJ$_7$ ou_CONJ$_7$ morto_ADJ$_8$._SENT$_9$

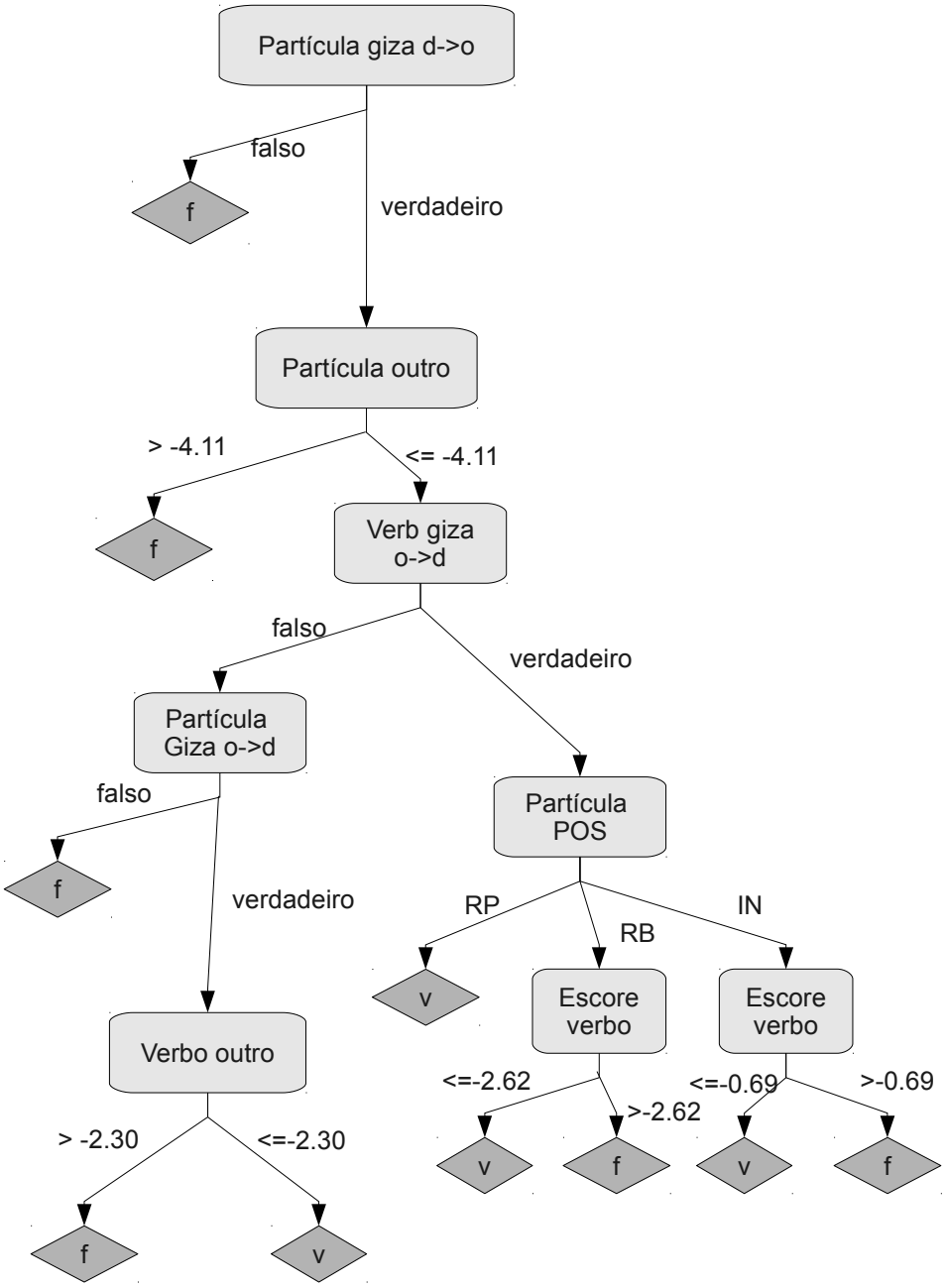

Figura 4.9: Árvore de decisão da heurística CVP 2

e o alinhamento gerado pelo alinhador Hill Post, mostrado na figura 4.10.

Vamos primeiramente descobrir quais candidatos a CVP o método CVP 1 encontra. O método CVP 1 parte de correspondências entre um verbo em inglês e um verbo em português já alinhados, então nesse caso temos 2 ocorrências deste tipo no exemplo, no caso, os alinhamentos individuais $\{(5,3)(8,4)\}$. Então, a partir das posições dos verbos em inglês buscamos nas três próximas posições algo que possar ser uma partícula, encontrando os seguintes candidatos a CVP: $(5+6,3)$ (*is only* $\leftrightarrow$ é), $(5+7,3)$ (*is in* $\leftrightarrow$ é), $(8+9,4)$ (*finding out* $\leftrightarrow$ descobrir) e $(8+10,4)$ (*finding if* $\leftrightarrow$ descobrir). Já no método CVP 2, não há mais o requerimento de um alinhamento pré-existente entre os verbos em

$$\{(1,1)(2,2)(5,3)(8,4)(10,5)(13,6)(14,7)(15,8)(16,5)(20,9)(21,10)\}$$

		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
		PP	NN	IN	NP	VBZ	RB	IN	VVG	RP	IN	PP	POS	JJS	JJ	CC	IN	PP	POS	NNS	JJ	SENT
		my	interest	in	johnny	is	only	in	finding	out	if	he	,	s	alive	or	if	he	,	s	dead	.
1	meu	•																				
2	interesse		•																			
3	é					•																
4	descobrir								•													
5	se										•						•					
6	está													•								
7	vivo														•							
8	ou															•						
9	morto																				•	
10	.																					•

Figura 4.10: Alinhamento antes de heurística de CVP

inglês e português, partimos de todas as combinações possíveis entre um verbo em inglês e um verbo em português $\{(5,3)(5,4)(5,6)(8,3)(8,4)(8,6)\}$ e então prosseguimos como no método CVP 1.

Vamos considerar dois candidatos a CVP de acordo com o segundo método para exemplificar a aplicação da heurística, $(8+9,4)$ (*finding out* $\leftrightarrow$ descobrir) e $(8+10,4)$ (*finding if* $\leftrightarrow$ descobrir). Os valores das *features* para esses 2 exemplos podem ser vistos na tabela 4.2

Candidato	*finding out* $\leftrightarrow$ descobrir	*finding if* $\leftrightarrow$ descobrir
verbo alinhado	verdadeiro	verdadeiro
particula giza org→dst	falso	falso
particula giza dst→org	verdadeiro	falso
verbo giza org→dst	verdadeiro	verdadeiro
verbo giza dst→org	verdadeiro	verdadeiro
escore do verbo	-3.25	-3.25
escore da partícula	-5.10	-27.63
verbo outro	-100	-100
partícula outro	-100	-0.56
partícula POS	RP	IN

Tabela 4.2: Valores das *features* relevantes

A classificação é feita simplesmente seguindo as regras da árvore de decisão da figura 4.9, para *finding out* $\leftrightarrow$ descobrir temos:

- Partícula giza dst→org = verdadeiro

- Partícula outro <= -4.11
- Verbo giza org→dst = verdadeiro
- Partícula POS = RP
- **Verdadeiro!**

ou seja, a heurística considera *finding out* como sendo uma CVP. Já para o segundo exemplo, *finding if* ↔ descobrir, temos:

- Partícula giza org→dst = falso
- **Falso!**

ou seja, não é considerado uma CVP.

O alinhamento final deste par de sentenças está na figura 4.11. A heurística CVP 1 funciona de maneira análoga, no entanto mudam os candidatos considerados e portanto, também muda a árvore de decisão.

$$\{(1,1)(2,2)(5,3)(8,4)(9,4)(10,5)(13,6)(14,7)(15,8)(16,5)(20,9)(21,10)\}$$

		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
		my	interest	in	johnny	is	only	in	finding	out	if	he	,	s	alive	or	if	he	,	s	dead	.
1	meu	•																				
2	interesse		•																			
3	é					•																
4	descobrir								⋆	⋆												
5	se										•						•					
6	está													•								
7	vivo														•							
8	ou															•						
9	morto																				•	
10	.																					•

Figura 4.11: Alinhamento após a heurística CVP 2

5 RESULTADOS

Nesta seção reportamos os experimentos realizados para avaliar os resultados obtidos para os métodos de alinhamento propostos: os alinhadores de *hillclimbing* descritos previamente em comparação com o alinhador Giza++, utilizando-se a heurística de simetrização refinada descrita na seção 2.3. Mostramos os resultados em termos das métricas precisão, revocação e escore-f. Para a análise dos alinhamentos de CVP, consideramos o alinhamento correto somente quando tanto o verbo quanto a partícula estão alinhados corretamente[1]. Em seguida, reportamos testes de significância estatística, discutimos os resultados e por fim reportamos resultados de testes variando o α do escore-f.

5.1 Comparação entre os alinhadores

A tabela 5.1 mostra os resultados do alinhamento realizado no corpus de *tuning* completo, onde podemos ver que os alinhadores Hill são bastante competitivos com Giza++ em termos de performance geral. As heurísticas de CVP mostram uma pequena melhoria, o que não é surpreendente considerando-se que em nossas anotações encontramos um total de 149 alinhamentos de CVP, para um total de 8.414 alinhamentos, o que resulta em um proporção de apenas 1,7%. Assim, mesmo uma melhora significativa no processamento de CVPs causa uma melhor geral pequena.

Alinhador	**Precisão**	**Revocação**	**Escore-f**
Hill Base	0.730	0.624	0.673
Hill Post	0.733	0.679	0.705
Hill CVP 1	**0.734**	0.683	0.707
Hill CVP 2	0.733	0.686	**0.708**
Giza++	0.652	**0.704**	0.677

Tabela 5.1: Resultados para o corpus de *tuning* completo

Porém, para ressaltar as diferenças de performance dos métodos no tratamento de CVPs, é feita uma análise mais detalhada dos alinhamentos de sentenças envolvendo CVPs. Na tabela 5.2 mostramos os resultados considerando somente os candidatos a CVP. Aqui podemos observar mais claramente os efeitos que as heurísticas de CVP têm sobre o nosso alinhador *baseline*, o alinhador Hill Base. Enquanto que os alinhadores Base e Post não conseguem encontrar **nenhum** CVP completo, tanto Hill CVP 1 quanto Hill CVP 2 mostram uma excelente precisão e uma revocação satisfatória, obtendo uma

[1]Desta forma são descartados os casos onde apenas um dos alinhamentos esteja correto.

comparação vantajosa em relação ao alinhador Giza++ refined. CVP 1 possui uma precisão melhor mas revocação pior do que CVP 2, o que é esperado pois o alinhador CVP 1 requer que o verbo em inglês já esteja alinhado ao verbo em português, enquanto que o CVP 2 não. A tabela 5.3 mostra os resultados para o corpus de *tuning* parcial, que apresenta as mesmas tendências.

Alinhador	**Precisão**	**Revocação**	**Escore-f**
Hill Base	0.000	0.000	0.000
Hill Post	0.000	0.000	0.000
Hill CVP 1	**0.952**	0.187	0.316
Hill CVP 2	0.765	**0.364**	**0.494**
Giza++	0.677	0.196	0.304

Tabela 5.2: Resultados para o corpus de *tuning* completo (Somente CVPs)

Alinhador	**Precisão**	**Revocação**	**Escore-f**
Hill Base	0.000	0.000	0.000
Hill Post	**1.000**	0.028	0.005
Hill CVP 1	**1.000**	0.197	0.330
Hill CVP 2	0.795	**0.437**	**0.563**
Giza++	0.656	0.295	0.408

Tabela 5.3: Resultados para o corpus de *tuning* parcial (Somente CVPs)

Entretanto, para evitar *over-tuning*, precisamos ainda avaliar os métodos em dados não vistos, que não foram utilizados durante o processo de desenvolvimento e ajuste de parâmetros. Por isso, repetimos os mesmos testes com o corpus de teste, e a tabela 5.4 mostra os resultados para o alinhamento geral. Comparando esses resultados com os da tabela 5.1 podemos ver que a tendência é mantida, com os alinhadores Hill Post e Hill CVP com melhor desempenho que o Giza++ e as heurísticas de CVP proporcionando uma pequena melhora.

Alinhador	**Precisão**	**Revocação**	**Escore-f**
Hill Base	0.736	0.612	0.668
Hill Post	0.736	0.656	0.694
Hill CVP 1	**0.737**	0.658	**0.695**
Hill CVP 2	0.734	0.657	0.694
Giza++	0.643	**0.678**	0.660

Tabela 5.4: Resultados para o corpus de teste completo

Na tabela 5.5 analisamos somente as CVPs do corpus de teste completo, e na tabela 5.6 temos os resultados para as CVPs no corpus de teste parcial. Os resultados são similares aos das tabelas 5.2 e 5.3, com o alinhador Hill CVP 1 dominando na precisão, e o Hill CVP 2 sendo melhor em revocação e escore-f.

Alinhador	Precisão	Revocação	Escore-f
Hill Base	0.000	0.000	0.000
Hill Post	0.000	0.000	0.000
Hill CVP 1	**0.833**	**0.128**	**0.222**
Hill CVP 2	0.370	**0.128**	0.190
Giza++	0.208	**0.128**	0.158

Tabela 5.5: Resultados para o corpus de teste completo (Somente CVPs)

Alinhador	Precisão	Revocação	escore-f
Hill Base	0.000	0.000	0.000
Hill Post	**1.000**	0.010	0.019
Hill CVP 1	0.946	0.340	0.500
Hill CVP 2	0.676	**0.447**	**0.538**
Giza++	0.548	0.330	0.412

Tabela 5.6: Resultados para o corpus de teste parcial (Somente CVPs)

O fato de o padrão geral dos resultados permanecer o mesmo ao longo dos diferentes corpora, com respeito a performance relativa dos diferentes alinhadores, indica a robustez das técnicas propostas. No entanto, para garantir a significância estatística dos resultados, precisamos ainda realizar um teste de significância.

5.2 Significância Estatística

Observando as tabelas da seção anterior podemos ter uma ideia do comportamento dos alinhadores propostos, porém não podemos afirmar com nenhum grau de certeza se um alinhador x é melhor do que um alinhador y, com respeito a determinada métrica.

Para realizar a comparação entre os métodos, seguimos a metodologia de *bootstraping* proposta por Zhang et al. (2004) para comparar os sistemas com um intervalo de confiança de 95%, realizando 1000 re-amostragens aleatórias. Primeiramente, comparamos os alinhadores com o corpus anotado de forma completa. Na tabela 5.7 $\ll$ significa que o alinhador x é significativamente pior que o alinhador y com respeito ao escore-f. De forma similar, $\gg$ significa que o alinhador x é significativamente melhor com respeito ao escore-f e $\equiv$ significa que não há diferença estatística significante ente x e y com um intervalo de confiança de 95%.

x \ y	Hill Base	Hill Post	Hill CVP 1	Hill CVP 2	Giza++
Hill Base	-	$\ll$	$\ll$	$\ll$	$\equiv$
Hill Post	$\gg$	-	$\ll$	$\equiv$	$\gg$
Hill CVP 1	$\gg$	$\gg$	-	$\gg$	$\gg$
Hill CVP 2	$\gg$	$\equiv$	$\ll$	-	$\gg$
Giza++	$\equiv$	$\ll$	$\ll$	$\ll$	-

Tabela 5.7: Tabela de significância estatística

O mesmo processo foi repetido para o corpus de teste com anotação parcial, para avaliarmos a performance de cada método exclusivamente para os CVPs. Os resultados estão na tabela 5.8.

x \ y	Hill Base	Hill Post	Hill CVP 1	Hill CVP 2	Giza++
Hill Base	-	$\equiv$	$\ll$	$\ll$	$\ll$
Hill Post	$\equiv$	-	$\ll$	$\ll$	$\ll$
Hill CVP 1	$\gg$	$\gg$	-	$\equiv$	$\equiv$
Hill CVP 2	$\gg$	$\gg$	$\equiv$	-	$\gg$
Giza++	$\gg$	$\gg$	$\equiv$	$\ll$	-

Tabela 5.8: Tabela de significância estatística para CVPs

Com estes testes podemos concluir com intervalo de confiança de confiança de 95% que todos os alinhadores Hill X são superiores com respeito ao escore-f (com $\alpha = 0.5$) à nossa *baseline* e ao Giza++ para o caso geral, ou seja, levando em conta todos os tipos de alinhamento. Também podemos afirmar que o método Hill CVP 1 é superior ao Hill CVP 2, apesar destes estarem bastante próximos. Também com 95% de confiança podemos concluir que os métodos Hill CVP 1 e 2 são superiores à nossa *baseline* levando em conta somente alinhamentos de CVP, e também que o Hill CVP 2 é superior ao Giza++ refined nesse aspecto, porém nada podemos concluir entre Hill CVP 1 e Giza++. Esses resultados são verdadeiros para o corpus de legendas Opus, pois é dele que foram selecionadas sentenças para o *gold standard*, esperamos que a tendência seja mantida em outros corpora, porém mais estudos são necessários para confirmar isto.

5.3 Discussão

Observando os alinhamentos gerados pelos diferentes métodos, pudemos observar que tanto Hill Base quanto Giza++ Refined obtém um escore-f bastante similar, mas com Hill Base tendendo um pouco mais em direção à precisão, e o Giza++ tendendo a uma revocação maior. Isso é explicado pelo menos em parte pela grande dificuldade que o nosso alinhador Hill Base tem com alinhamentos individuais complexos n:m, como ficou evidenciado pela avaliação das CVPs acima. Giza++, em contraste, começa com uma performance sensivelmente melhor para alinhamentos complexos, mas ao custo de uma menor precisão. Ao adicionarmos processamento explícito para aqueles alinhamentos complexos que causam problemas ao alinhador Hill Base, resultando nos alinhadors Hill Post, Hill CVP 1 e Hill CVP 2, vemos uma melhoria em revocação, porém mantendo a precisão. Esse incremento na revocação advém daqueles alinhamentos que são explicitamente processado (i.e., Pron + V e CVPs): quanto mais padrões de alinhamento complexos pudermos identificar e processar, melhor será a qualidade que podemos esperar do alinhamento resultante.

Com respeito às CVPs, uma inspeção manual dos alinhamos mostra que tanto Hill CVP 1 quanto Hill CVP 2 conseguem alinhar CVPs corretamente independente destes serem composicionais ou não, desde que a expressão seja suficientemente frequente. CVPs não composicionais como *hung over* e *carry on* foram corretamente identificados e alinhados, assim como também o foram CVPs mais composicionais como *tied up* e *get down*. O problema ocorre quando o CVP é não composicional **e** raro, fazendo com que

as probabilidades de tradução de palavra geradas pelo Giza++ não capturem a correspondência do verbo em Inglês com o verbo em Português. Isso impossibilita que CVPs como *put off* sejam identificados nos nossos testes. A maior parte dos falsos positivos são causados pela dificuldade de identificar o CVP, isto é, de diferenciar corretamente entre as etiquetas morfossintáticas RP, RB e IN, fazendo com que sequencias como *reason with* sejam identificadas como CVP. Outra fonte de erros ocorre com as traduções livres, onde um CVP não é traduzido literalmente, mas uma expressão funcionalmente equivalente é utilizada em seu lugar na língua de destino.

Estes problemas podem ser mitigados utilizando-se corpora maiores e fontes alternativas de informação tal como dicionários. Além disso, utilizar corpora mais literais do que de legendas de filmes, onde existem muitas traduções livres, pode ajudar. Por fim, o uso de melhores ferramentas de etiquetamento, em especial com parâmetros ajustados para o corpus paralelo em questão, também deve melhorar a performance.

5.4 Teste variando α

O alinhador Hill Base proposto nesse trabalho possui uma característica interessante, que é o fato de a otimização ser feita diretamente com respeito ao escore-f. Realizamos então um experimento para verificar quão bem nosso alinhador se adapta a diferentes valores de α no escore-f, comparando com o Giza. Para esse experimento, nós ativamos a *feature* de bônus (seção 4.1) e rodamos o processo de otimização para vários valores de α no corpus de *tuning*. Com os vetores de peso encontrados, realizamos então o alinhamento com o corpus de teste, e os resultados reportados podem ser vistos na tabela 5.9.

alinhador / α	**0.1**	**0.25**	**0.4**	**0.5**	**0.6**	**0.75**	**0.9**	**Média**
Hill Base	0.725	0.683	0.674	0.676	0.693	0.729	0.778	**0.71**
Giza++ Union	0.740	0.698	0.661	0.638	0.617	0.587	0.561	**0.64**
Giza++ Intersection	0.569	0.604	0.643	0.671	0.703	0.757	0.820	**0.68**
Giza++ Refined	0.675	0.669	0.664	0.660	0.657	0.652	0.647	**0.66**

Tabela 5.9: Variando alpha no teste completo

Incluímos na comparação além do Giza Refined também as duas outras heurísticas de simetrização (união e interseção), pois intuitivamente elas podem possuir melhor desempenho para valores de α baixos (priorizando a revocação) e altos (priorizando a precisão) respectivamente. O método Giza Refined gerou resultado de precisão e de revocação bastante equilibrados (como pode ser visto nas tabelas 5.1 e 5.4, por isso seus resultados de escore-f mudam pouco com a variação do valor de α. As heurísticas de união e interseção demonstram os resultados esperados, variando bastante com a variação do α. Por fim, podemos observar que o método de otimização diretamente com respeito ao escore-f atinge seu objetivo, pois o alinhador Hill Base consegue se aproveitar das mudanças de α positivamente, melhorando o desempenho tanto com α grande quanto com α pequeno. Essas variações são apresentadas na tabela 5.9 e ilustradas graficamente na figura 5.1.

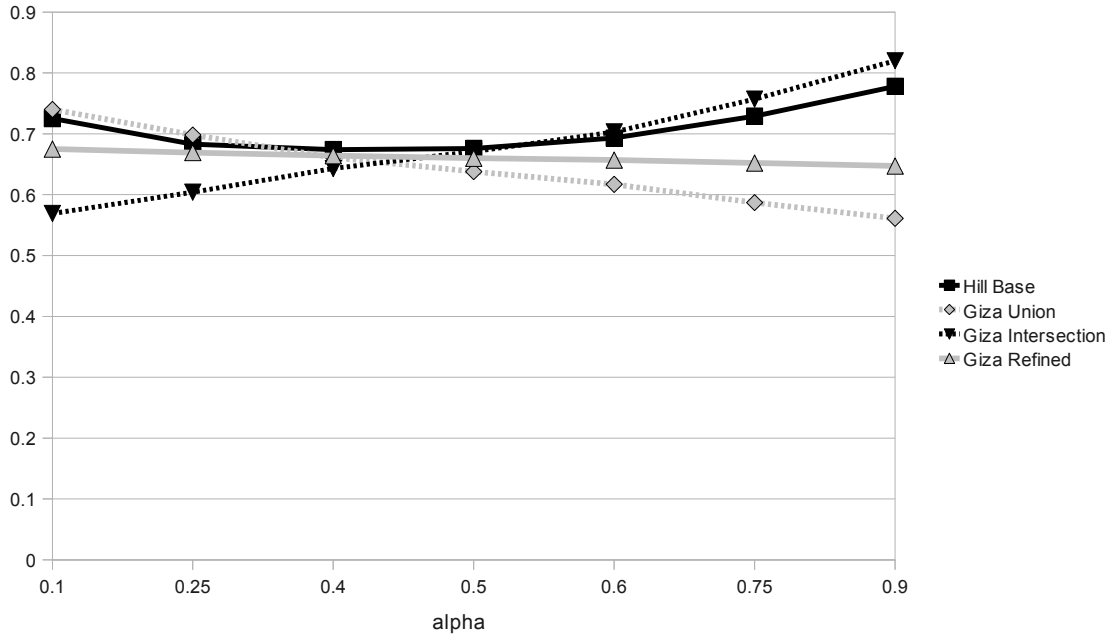

Figura 5.1: Variando alpha no teste completo

6 CONCLUSÃO

Como discutido, EMPs são um dos grandes desafios para aplicações de PLN que envolvam alguma análise semântica. Para a tradução automática elas representam uma fonte de erros que devem ser evitados. Sistemas empíricos modernos de tradução de máquina utilizam um alinhamento léxico gerado automaticamente por um alinhador, de forma que um melhor alinhamento cause um melhor tradutor. Assim, nosso objetivo foi desenvolver um alinhador léxico que incorporasse alguma técnica visando melhor o desempenho com respeito e EMPs.

Neste trabalho apresentamos uma alinhador léxico discriminativo para Português-Inglês, que possui um núcleo baseado em *hillclimbing*. Este núcleo baseado em *hill-climbing* (Hill Base) é seguido por regras de pós-processamento dependentes de língua, para lidar com alinhamentos mais complexos do tipo $n : m$. Avaliamos essa abordagem em CVPs e os resultados mostram uma melhora em todas as métricas de avaliação para o nosso corpus de teste, o que mostra que as técnicas de aprendizado de máquina utilizadas podem ser bastante eficientes para induzir tais regras de pós-processamento. Os alinhadores com pós-processamento propostos (Hill Post, CVP 1 e CVP 2) obtiveram uma melhora de desempenho estatisticamente significativa tanto sobre o nosso alinhador de *baseline* (Hill Base) quanto sobre o alinhador de comparação (Giza++). Com respeito exclusivamente a CVPs, os alinhadores com heurísticas de processamento de CVPs (Hill CVP 1 e 2) obtiveram melhora estatisticamente significativa sobre os alinhadores propostos sem essas heurísticas (Hill Base e Hill Post) e sobre o alinhador de comparação.

Visto que lidar com expressões multi-palavra é um importante problema aberto em processamento de linguagens naturais, a abordagem apresentada neste trabalho mostra que é possível melhorar o alinhamento de pelo menos algumas classes de EMPs usando conhecimento linguístico superficial. Tentativas prévias de lidar com MWEs na tarefa de alinhamento léxico (como em Venkatapathy e Joshi (2006)) utilizam apenas informação estatística, enquanto que aqui combinamos informação estatística gerado pelo Giza++ com conhecimento linguístico (na forma de etiquetas morfossintáticas) utilizando aprendizado de máquina.

Também apresentamos um algoritmo de otimização dos parâmetros que os otimiza diretamente com respeito as métricas de avaliação final (escore-f), o que fez com que este se mostrasse bastante robusto quanto a mudanças na avaliação priorizando revocação ou precisão (variação de α no escore-f).

Além destas contribuições, este trabalho gerou um *gold standard* de alinhamentos português-inglês de alta qualidade que pode ser utilizados em outros experimentos, assim como o código-fonte dos alinhadores aqui propostos. Estes recursos estão disponíveis

gratuitamente[1], e com a disponibilização deles esperamos fomentar mais pesquisa na área, especialmente com português como uma das línguas.

Para trabalhos futuros, pode-se investigar a aplicação da mesma técnica para outros tipos de EMPs, tais como *light verbs* ou *compound nouns*, assim como adicionar algumas medidas estatísticas de associação na forma de *features*, para observar se uma combinação de conhecimento gramatical com mais informação estatística pode melhorar os resultados. Além disso, uma avaliação mais ampla pode ser realizada, utilizando diferentes corpora e também utilizando avaliação extrínseca.

Por fim, podemos concluir que neste trabalho investigamos técnicas de alinhamento léxico discriminativo e implementamos com sucesso medidas para o tratamento de expressões multi-palavra visando melhorar o desempenho, que foi o objetivo proposto.

[1] http://redmine.jorjao81.com/wiki/mestrado-paulo/Resources

REFERÊNCIAS

AL-ONAIZAN, Y.; CURIN, J.; JAHR, M.; KNIGHT, K.; LAFFERTY, J.; MELAMED, D.; OCH, F.-J.; PURDY, D.; SMITH, N. A.; YAROWSKY, D. **Statistical machine translation. Final Report**. [S.l.]: JHU Summer Workshop, 1999.

BALDWIN, T. Deep lexical acquisition of verb-particle constructions. **Computer Speech and Language**, Colchester, v.19, n.4, p.398–414, Oct. 2005.

BANNARD, C. Learning about the meaning of verb-particle constructions from corpora. **Computer Speech and Language**, Colchester, v.19, n.4, p.467–478, Oct. 2005.

BICK, E. **The Parsing System Palavras, Automatic Grammatical Analysis of Portuguese in a Constraint Grammar Framework**. 1st.ed. [S.l.]: Aarhus University Press, 2000.

BRISCOE, T.; CARROLL, J.; WATSON, R. The Second Release of the RASP System. In: COLING/ACL 2006 INTERACTIVE PRESENTATION SESSIONS, 2006, Sydney, Australia. **Proceedings...** Association for Computational Linguistics, 2006. p.77–80.

BROWN, P. F.; COCKE, J.; PIETRA, S. A. D.; PIETRA, V. J. D.; JELINEK, F.; LAFFERTY, J. D.; MERCER, R. L.; ROOSSIN, P. S. A Statistical Approach to Machine Translation. **Computational Linguistics**, [S.l.], v.16, n.2, p.79–85, 1990.

BROWN, P. F.; PIETRA, V. J. D.; PIETRA, S. A. D.; MERCER, R. L. The mathematics of statistical machine translation: parameter estimation. **Computational Linguistics**, Cambridge, MA, USA, v.19, n.2, p.263–311, 1993.

CARLETTA, J. Assessing Agreement on Classification Tasks: the kappa statistic. **Computational Linguistics**, [S.l.], v.22, n.2, p.249–254, 1996.

CASELI, H. M. **Indução de léxicos bilíngües e regras para a tradução automática**. 2007. 158p. Tese (Doutorado em Ciência da Computação) — Universidade de São Paulo, São Paulo.

CASELI, H. M.; NUNES, M. G. V.; FORCADA, M. L. LIHLA: shared task system description. In: ACL WORKSHOP ON BUILDING AND USING PARALLEL TEXTS, 2005, Ann Arbor. **Proceedings...** Association for Computational Linguistics, 2005. p.111–114.

CASELI, H. M.; RAMISCH, C.; NUNES, M. G. V.; VILLAVICENCIO, A. Alignment-based extraction of multiword expressions. **Language Resources and Evaluation**, [S.l.], 2009.

CASELI, H. M.; SCALCO, M. A. G.; NUNES, M. G. V. **Annotation style guide for lexical alignment (NILC-TR-05-09)**. [S.l.: s.n.], 2005. 24 p. (256).

CHIANG, D. Hierarchical Phrase-Based Translation. **Computational Linguistics**, [S.l.], v.33, n.2, p.201–228, 2007.

COOK, P.; STEVENSON, S. Classifying Particle Semantics in English Verb-Particle Constructions. In: COLING/ACL WORKSHOP ON MULTIWORD EXPRESSIONS: IDENTIFYING AND EXPLOITING UNDERLYING PROPERTIES, 2006, Sydney, Australia. **Proceedings...** Association for Computational Linguistics, 2006. p.19–27.

DAELEMANS, W.; ZAVREL, J.; SLOOT, K. van der; BOSCH, A. van den. **TiMBL**: tilburg memory-based learner - version 4.3 - reference guide. Disponível em http://citeseer.ist.psu.edu/article/daelemans02timbl.html.

DEHE, N. **Particle Verbs in English**: syntax, information structure and intonation. 1st.ed. Amsterdam/Philadelphia: John Benjamins, 2002.

DEMPSTER, A. P.; LAIRD, N. M.; RUBIN, D. B. Maximum Likelihood from Incomplete Data via the EM Algorithm. **Journal of the Royal Statistical Society. Series B (Methodological)**, [S.l.], v.39, n.1, p.1–38, 1977.

DICE, L. R. Measures of the Amount of Ecologic Association Between Species. **Ecology**, [S.l.], v.26, n.3, p.297 – 302, 1945.

FRASER, A.; MARCU, D. Semi-Supervised Training for Statistical Word Alignment. In: INTERNATIONAL CONFERENCE ON COMPUTATIONAL LINGUISTICS AND 44TH ANNUAL MEETING OF THE ASSOCIATION FOR COMPUTATIONAL LINGUISTICS, 21., 2006, Sydney, Australia. **Proceedings...** Association for Computational Linguistics, 2006. p.769–776.

FRASER, A.; MARCU, D. Measuring Word Alignment Quality for Statistical Machine Translation. **Computational Linguistics**, Cambridge, MA, USA, v.33, n.3, p.293–303, 2007.

FRASER, A.; MARCU, D. Getting the structure right for word alignment: leaf. In: EMNLP-CONLL, 2007. **Proceedings...** [S.l.: s.n.], 2007. v.4, n.June, p.51–60.

GANCHEV, K.; GRACA, J. V.; TASKAR, B. Better Alignments = Better Translations? In: ACL-08: HLT, 2008, Columbus, Ohio. **Proceedings...** Association for Computational Linguistics, 2008. p.986–993.

GEER, D. Statistical Machine Translation Gains Respect. **Computer**, Los Alamitos, CA, USA, v.38, n.10, p.18–21, 2005.

GERMANN, U. Yawat: Yet Another Word Alignment Tool. In: ACL-08: HLT DEMO SESSION, 2008, Columbus, Ohio. **Proceedings...** Association for Computational Linguistics, 2008. p.20–23.

HALL, M.; FRANK, E.; HOLMES, G.; PFAHRINGER, B.; REUTEMANN, P.; WITTEN, I. H. The WEKA data mining software: an update. **SIGKDD Explor. Newsl.**, New York, NY, USA, v.11, n.1, p.10–18, 2009.

JACKENDOFF, R. **The Architecture of the Language Faculty**. 1st.ed. Cambridge, MA: MIT Press, 1997.

JACKENDOFF, R. **English particle constructions, the lexicon and the autonomy of syntax**. 1st.ed. Berlin/New York: Mouton de Gruyter, 2002. 67–94p.

JURAFSKY, D.; MARTIN, J. **SPEECH and LANGUAGE PROCESSING**: an introduction to natural language processing, computational linguistics and speech recognition. 1st.ed. New Jersey, USA: Prentice Hall PTR, 2000.

LEVIN, B. **English Verb Classes and Alternations**: a preliminary investigation. 1st.ed. [S.l.]: University of Chicago Press, 1993.

LIU, Y.; LIU, Q.; LIN, S. Log-Linear Models for Word Alignment. In: ANNUAL MEETING OF THE ASSOCIATION FOR COMPUTATIONAL LINGUISTICS (ACL'05), 43., 2005, Ann Arbor, Michigan. **Proceedings...** Association for Computational Linguistics, 2005. p.459–466.

MA, Y. **Constrained Word Alignment Models for Statistical Machine Translation**. 2009. Tese (Doutorado em Ciência da Computação) — Dublin City University.

MARCUS, M. P.; MARCINKIEWICZ, M. A.; SANTORINI, B. Building a large annotated corpus of English: the penn treebank. **Computational Linguistics**, Cambridge, MA, USA, v.19, n.2, p.313–330, 1993.

MELAMED, I. D. Models of translational equivalence among words. **Computational Linguistics**, Cambridge, MA, USA, v.26, n.2, p.221–249, 2000.

MIHALCEA, R.; SIMARD, M. Parallel texts. **Natural Language Engineering**, [S.l.], v.11, n.3, p.239–246, 2006.

MOORE, R. C. A Discriminative Framework for Bilingual Word Alignment. In: HUMAN LANGUAGE TECHNOLOGY CONFERENCE AND CONFERENCE ON EMPIRICAL METHODS IN NATURAL LANGUAGE PROCESSING, 2005, Vancouver, British Columbia, Canada. **Proceedings...** Association for Computational Linguistics, 2005. p.81–88.

OCH, F. J.; NEY, H. A systematic comparison of various statistical alignment models. **Computational Linguistics**, Cambridge, MA, USA, v.29, n.1, p.19–51, 2003.

PAPINENI, K.; ROUKOS, S.; WARD, T.; ZHU, W.-J. Bleu: a method for automatic evaluation of machine translation. In: ANNUAL MEETING OF THE ASSOCIATION FOR COMPUTATIONAL LINGUISTICS, 40., 2002, University of Pennsylvania. **Proceedings...** [S.l.: s.n.], 2002. p.00–00.

QUINLAN, J. R. Induction of Decision Trees. **Machine Learning**, Netherlands, v.1, n.1, p.83–106, 1986.

SAG, I. A.; BALDWIN, T.; BOND, F.; COPESTAKE, A. A.; FLICKINGER, D. Multiword Expressions: a pain in the neck for nlp. In: CICLING '02: PROCEEDINGS OF THE THIRD INTERNATIONAL CONFERENCE ON COMPUTATIONAL LINGUISTICS AND INTELLIGENT TEXT PROCESSING, 2002, London, UK. **Anais...** Springer-Verlag, 2002. p.1–15.

SCHMID, H. Probabilistic Part-of-Speech Tagging Using Decision Trees. In: INTERNATIONAL CONFERENCE ON NEW METHODS IN LANGUAGE PROCESSING, 1994, Manchester, UK. **Proceedings...** [S.l.: s.n.], 1994.

SOMERS, H. Review Article: example-based machine translation. **Machine Translation**, Hingham, MA, USA, v.14, n.2, p.113–157, 1999.

TASKAR, B.; SIMON, L.-J.; DAN, K. A Discriminative Matching Approach to Word Alignment. In: HUMAN LANGUAGE TECHNOLOGY CONFERENCE AND CONFERENCE ON EMPIRICAL METHODS IN NATURAL LANGUAGE PROCESSING, 2005, Vancouver, British Columbia, Canada. **Proceedings...** Association for Computational Linguistics, 2005. p.73–80.

TIEDEMANN, J. News from OPUS - A Collection of Multilingual Parallel Corpora with Tools and Interfaces. In: NICOLOV, N.; BONTCHEVA, K.; ANGELOVA, G.; MITKOV, R. (Ed.). **Recent Advances in Natural Language Processing**. Borovets, Bulgaria: John Benjamins, Amsterdam/Philadelphia, 2009. v.V, p.237–248.

VENKATAPATHY, S.; JOSHI, A. K. Using Information about Multi-word Expressions for the Word-Alignment Task. In: WORKSHOP ON MULTIWORD EXPRESSIONS: IDENTIFYING AND EXPLOITING UNDERLYING PROPERTIES, 2006, Sydney, Australia. **Proceedings...** Association for Computational Linguistics, 2006. p.20–27.

VILLAVICENCIO, A. The availability of verb-particle constructions in lexical resources: how much is enough? **Computer Speech and Language**, Colchester, v.19, n.4, p.415–432, Oct. 2005.

VILLAVICENCIO, A.; COPESTAKE, A. Verb-particle constructions in the World Wide Web. In: ACL-SIGSEM WORKSHOP ON THE LINGUISTIC DIMENSIONS OF PREPOSITIONS AND THEIR USE IN COMPUTATIONAL LINGUISTICS FORMALISMS AND APPLICATIONS, 2003, Toulouse, France. **Proceedings...** Association for Computational Linguistics, 2003.

WEISSTEIN, E. W. **Methods of Steepest Descent**. Disponível em: <http://mathworld.wolfram.com/MethodofSteepestDescent.html>. Acesso em: março 2010.

WU, J.-C.; YEH, K. C.; CHUANG, T. C.; SHEI, W.-C.; CHANG, J. S. Totalrecall: a bilingual concordance for computer assisted translation and language learning. In: ACL-2003 INTERACTIVE POSTERS AND DEMONSTRATIONS COMPANION VOLUME TO THE PROCEEDINGS OF 41ST ANNUAL MEETING OF THE ASSOCIATION FOR COMPUTATIONAL LINGUISTICS, 2003, Sapporo, Japan. **Anais...** [S.l.: s.n.], 2003. p.201–204.

ZHANG, Y.; VOGEL, S.; WAIBEL, A. Interpreting BLEU / NIST Scores : how much improvement do we need to have a better system ? In: LANGUAGE RESOURCES AND EVALUATION (LREC-2004), 2004. **Proceedings...** [S.l.: s.n.], 2004. p.2051–2054.

ZHAO, Y.; ZHANG, Y. Comparison of decision tree methods for finding active objects. **Advances in Space Research**, [S.l.], v.41, n.12, p.1955 – 1959, 2008.

APÊNDICE A

Tagset utilizado nos arquivos de parâmetros para o Tree Tagger para o português, disponibilizado por Pablo Gamallo.

```
Adjectivo ADJ
Advérbio ADV
Determinante DET
Número Cardinal / Ordinal CARD
Nome Comum / Próprio   NOM
Pronome P
Preposição PREP
Verbo V
Interjeição   I
Separadores dentro da oração  VIRG
Separadores de orações  SENT

Existem também cominações de tags:

PREP+DET  (por exemplo: do, das, etc.)
V+P (por exemplo: levou-me, disse-lhe, etc.)
```