UNIVERSIDADE FEDERAL DE SÃO CARLOS

CENTRO DE CIÊNCIAS EXATAS E DE TECNOLOGIA

PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

TRADUÇÃO AUTOMÁTICA ESTATÍSTICA BASEADA EM SINTAXE E LINGUAGENS DE ÁRVORES

DANIEL EMILIO BECK

ORIENTADORA: PROFA. DRA. HELENA DE MEDEIROS CASELI

São Carlos - SP

Junho/2012

UNIVERSIDADE FEDERAL DE SÃO CARLOS

CENTRO DE CIÊNCIAS EXATAS E DE TECNOLOGIA

PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

TRADUÇÃO AUTOMÁTICA ESTATÍSTICA BASEADA EM SINTAXE E LINGUAGENS DE ÁRVORES

DANIEL EMILIO BECK

Dissertação apresentada ao Programa de Pós-Graduação em Ciência da Computação da Universidade Federal de São Carlos como parte dos requisitos para a obtenção de título de Mestre em Ciência da Computação, área de concentração: Inteligência Artificial

Orientadora: Profa. Dra. Helena de Medeiros Caseli

São Carlos - SP

Junho/2012

Ficha catalográfica elaborada pelo DePT da Biblioteca Comunitária da UFSCar

B393ta

Beck, Daniel Emilio.
Tradução automática estatística baseada em sintaxe e linguagens de árvores / Daniel Emilio Beck. -- São Carlos : UFSCar, 2012.
93 f.

Dissertação (Mestrado) -- Universidade Federal de São Carlos, 2012.

1. Processamento da linguagem natural (Computação). 2. Linguística - processamento de dados. 3. Linguagem - tradução automática. I. Título.

CDD: 006.35 (20ª)

Universidade Federal de São Carlos

Centro de Ciências Exatas e de Tecnologia

Programa de Pós-Graduação em Ciência da Computação

"Tradução Automática Estatística baseada em Sintaxe e Linguagens de Árvores"

Daniel Emílio Beck

Dissertação de Mestrado apresentada ao Programa de Pós-Graduação em Ciência da Computação da Universidade Federal de São Carlos, como parte dos requisitos para a obtenção do título de Mestre em Ciência da Computação

Membros da Banca:

Helena de M. Caseli

Profa. Dra. Helena de Medeiros Caseli
(Orientadora - DC/UFSCar)

Prof. Dr. Estevam Rafael Hruschka Júnior
(DC/UFSCar)

Profa. Dra. Lucia Specia
(University Of Sheffield - Inglaterra)

•

São Carlos
Junho/2012

AGRADECIMENTOS

Várias pessoas tiveram sua importância no decorrer do trabalho aqui apresentado. No entanto, essa dissertação jamais teria sido escrita sem o apoio da Helena e essa importância vai muito além de uma simples questão burocrática. Seus conselhos técnicos e científicos, sua compreensão com a minha forma de trabalhar e, principalmente, seu fácil acesso foram fundamentais. E ter me proporcionado tudo isso no meio de todo o trabalho que uma professora tem me faz admirar ainda mais o seu profissionalismo. Se hoje eu posso dizer que tenho uma visão muito melhor das nuances da vida acadêmica e uma maior maturidade para fazer pesquisa é porque eu tive a sorte de ser orientado por ela.

Minha estadia em São Carlos trouxe um apanhado enorme de novas experiências. Na área acadêmica, a troca de ideias feita dentro dos laboratórios do NILC foi de grande valia para minha formação. Fora da universidade, novas amizades fizeram com que eu superasse a distância da família e dos velhos amigos que deixei pra trás. Em especial, talvez não tivesse sobrevivido em São Carlos sem a ajuda da minha "sister" Cairo, que me acolheu, me apresentou a cidade e foi alguém que eu podia contar sempre.

Durante esses dois anos, minhas viagens a São Paulo se tornaram mais frequentes. De lá, tive o apoio do meu irmão Dary e da minha cunhada Adriane, que sempre me acolheram de braços abertos. Muito importante também foram os amigos, em especial Osama e Luan, que nunca se cansam de me aconselhar e ouvir meus choros, e o Adriano, que além de fazer tudo isso também teve tempo pra ser um namorado *extremamente* compreensivo.

Ainda que a saudade de Porto Alegre fosse uma constante durante os estudos, aprendi que a distância não é tão importante quanto as pessoas. Os velhos amigos, em especial Fábio, Kao, Letícia e Ricardo, sempre estiveram presentes, ainda que virtualmente. Além disso, minha família, em especial meu irmão Diego e minha mãe Teresa, nunca deixaram de me incentivar a buscar meus sonhos, mesmo com toda a saudade mútua. Como já disse no momento em que me graduei: não consigo imaginar uma família melhor pra uma pessoa como eu.

Finalmente, esse trabalho não teria sido realizado sem o apoio fundamental da CAPES e da FAPESP (projetos 2010/03807-4 e 2010/07517-0).

A todos vocês, muito obrigado.

“Nada na vida deve ser temido, apenas compreendido.”

Marie Curie

RESUMO

A Tradução Automática (*Machine Translation* - MT) é uma das aplicações clássicas dentro do Processamento da Língua Natural (*Natural Language Processing* - NLP). O estado-da-arte em MT é representado por métodos estatísticos, que buscam aprender o conhecimento linguístico necessário de forma automática por meio de grandes coleções de textos (os *corpora*). Entretanto, ainda que se tenha avançado bastante em relação à qualidade de sistemas estatísticos de MT, hoje em dia esses avanços não estão sendo significativos. Por conta disso, as pesquisas na área têm buscado formas de envolver mais conhecimento linguístico explícito nesses sistemas. Um dos problemas que não é bem resolvido por sistemas de MT puramente estatísticos é o correto tratamento de fenômenos sintáticos. Assim, uma das direções que as pesquisas tomam na hora de incorporar conhecimento linguístico a esses sistemas é através da adição de regras sintáticas. Para isso, uma série de métodos e formalismos foram e são estudados até hoje. Esse texto apresenta a investigação de métodos que se utilizam de informação sintática na tentativa de avançar no estado-da-arte da MT estatística. Foram utilizados métodos e formalismos que lidam com linguagens de árvores, em especial as Gramáticas de Substituição de Árvores (*Tree Substitution Grammars* - TSGs) e os Transdutores Árvore-para-String (*Tree-to-String* - TTS). Desta investigação, obteve-se maior entendimento sobre os formalismos estudados e seu comportamento em aplicações de NLP.

Palavras-chave: Processamento da Língua Natural, Linguística Computacional, Tradução Automática Estatística, Gramáticas de Substituição de Árvores, Transdutores Árvore-para-String

ABSTRACT

Machine Translation (MT) is one of the classic Natural Language Processing (NLP) applications. The state-of-the-art in MT is represented by statistical methods that aim to learn all necessary linguistic knowledge automatically through large collections of texts (*corpora*). However, while the quality of statistical MT systems had improved, nowadays these advances are not significant. For this reason, research in the area have sought to involve more explicit linguistic knowledge in these systems. One issue that purely statistical MT systems have is the lack of correct treatment of syntactic phenomena. Thus, one of the research directions when trying to incorporate linguistic knowledge in those systems is through the addition of syntactic rules. To accomplish this, many methods and formalisms with this goal in mind are studied. This text presents the investigation of methods which aim to advance the state-of-the-art in statistical MT through models that consider syntactic information. The methods and formalisms studied are those used to deal with tree languages, mainly Tree Substitution Grammars (TSGs) and Tree-to-String (TTS) Transducers. From this work, a greater understanding was obtained about the studied formalisms and their behavior when used in NLP applications.

Keywords: Natural Language Processing, Computational Linguistics, Statistical Machine Translation, Tree Substitution Grammars, Tree-to-String transducers

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

BLEU – *Bilingual Evaluation Understudy*

BP – *Brevity Penalty*

CFG – *Context-Free Grammar*

CKY – *Cocke-Kasami-Younger*

DP – *Dirichlet Process*

EM – *Expectation-Maximization*

ITG – *Inversion Transduction Grammar*

MERT – *Minimum Error Rate Training*

MLE – *Maximum Likelihood Estimate*

MT – *Machine Translation*

NLP – *Natural Language Processing*

PB-SMT – *Phrase-Based Statistical Machine Translation*

PCFG – *Probabilistic Context-Free Grammar*

POS – *Part of Speech*

RTL – *Regular Tree Language*

SCFG – *Synchronous Context-Free Grammar*

STSG – *Synchronous Tree Substitution Grammar*

SMT – *Statistical Machine Translation*

STT – *String-to-Tree*

TSG – *Tree Substitution Grammar*

TTS – *Tree-to-String*

TTT – *Tree-to-Tree*

SUMÁRIO

1 INTRODUÇÃO

O objetivo da Tradução Automática (*Machine Translation* - MT) é usar computadores para auxiliar todo ou uma parte do processo de tradução entre duas linguagens naturais (normalmente chamadas de língua fonte e língua alvo) (JURAFSKY; MARTIN, 2000). A MT é uma das aplicações clássicas dentro da área de Processamento de Linguagens Naturais (*Natural Language Processing* - NLP), tendo inclusive originado softwares comerciais como o Apertium[1], o Google Translator[2] e o Systran[3].

Até meados da década de 80, os sistemas de MT existentes eram baseados em recursos que modelavam o conhecimento linguístico de forma explícita, como dicionários e regras de tradução (HUTCHINS, 2007). Na maior parte das vezes, esses recursos eram construídos de forma manual, um trabalho extremamente custoso. Por conta desse custo, surgiram várias tentativas de se criar modelos de MT que pudessem ser construídos automaticamente. Segundo Lopez (2008), as primeiras ideias nesse sentido datam de 1949, quando Warren Weaver sugeriu atacar o problema utilizando um modelo estatístico originário da Teoria da Informação: o modelo *noisy-channel*. No entanto, esse e outros modelos estatísticos tinham duas barreiras: eles exigiam um poder computacional altíssimo para as máquinas da época e também uma quantidade de dados que na época era inexistente.

Com o passar dos anos, essas barreiras foram quebradas:

- O advento da Internet proporcionou acesso a uma fonte gigantesca de textos bilíngues. A partir dessa maior disponibilidade de recursos, esforços linguísticos foram dispendidos no sentido de estruturar esses textos nos chamados *corpora* paralelos[4]. Como consequência desses esforços, obteve-se a quantidade de dados necessária para que se justificasse a utilização de métodos estatísticos.

[1] `www.apertium.org`
[2] `http://translate.google.com/`
[3] `www.systransoft.com`
[4] Um *corpus* (*corpora* no plural) paralelo é um conjunto de textos que são a tradução uns dos outros.

- O avanço da tecnologia na construção de máquinas permitiu um aumento enorme no seu poder de processamento, incluindo o surgimento das arquiteturas *multi-core*. Soma-se a esse fato o desenvolvimento de métodos eficientes na reunião dessas novas máquinas em *clusters*. Desse modo, a barreira do desempenho foi praticamente quebrada: tornou-se possível a realização de cálculos avançados (necessários para alguns dos métodos estatísticos) em questão de minutos ou até segundos.

Como consequência, a partir do final da década de 80 esses modelos voltaram ser pesquisados, agora reunidos sob o nome Tradução Automática Estatística (*Statistical Machine Translation* - SMT). As primeiras bases da SMT foram estabelecidas com o modelo de Brown et al. (1990). Desde então, SMT passou a ser o paradigma mais estudado na área de MT e tornou-se o estado-da-arte, inclusive fazendo parte de sistemas comerciais.

1.1 Tradução Automática Estatística

A SMT busca resolver o problema da tradução entre duas linguagens naturais através de técnicas de aprendizado de máquina (LOPEZ, 2008). A ideia é gerar automaticamente um sistema de tradução a partir de um *corpus* paralelo e algoritmos de aprendizagem. Esse sistema seria então capaz de traduzir novos textos.

Teoricamente, é possível construir um sistema que realiza a tradução no nível do *corpus*. No entanto, tradicionalmente em SMT, busca-se subdividir o problema da tradução no nível da *sentença* e até mesmo ao nível da *palavra* ou conjunto de palavras. Para que seja possível essa subdivisão, o *corpus* de entrada passa por uma etapa de *alinhamento*, que consiste na descoberta de relações de correspondência entre trechos de ambos os lados do *corpus*.

Em geral, os *corpora* paralelos disponíveis já passaram por várias etapas de alinhamento, chegando até o nível sentencial. Entretanto, a maioria dos modelos de SMT existentes lidam com informações no nível lexical e, portanto, passam por uma etapa de *alinhamento lexical*. A figura 1.1 mostra um exemplo onde duas sentenças foram alinhadas dessa forma. Os traços representam correspondências entre os itens lexicais.

Formalmente, o alinhamento lexical é uma etapa independente do treinamento de um modelo de SMT, sendo que alguns trabalhos inclusive não fazem uso dessa etapa (MARCU; WONG, 2002). No entanto, como a maior parte dos modelos existentes na literatura utilizam o alinhamento lexical, essa ideia tornou-se bastante atrelada à SMT. Por consequência, os *toolkits* existentes costumam englobar ferramentas específicas para essa tarefa.

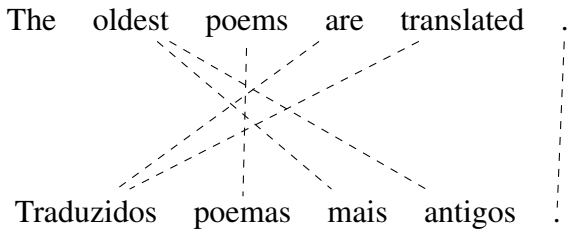


Figura 1.1: Exemplo de um alinhamento lexical de uma sentença em inglês (acima) com uma sentença em português (abaixo)

Após ter sido alinhado, o *corpus* paralelo pode então ser utilizado para treinamento de um sistema de SMT. De acordo com Lopez (2008), a definição de um sistema de SMT passa, necessariamente, por quatro etapas:

Definição do modelo: tem como objetivo simular o processo de tradução de uma sentença fonte em uma sentença alvo. Essa etapa passa pela escolha de um formalismo adequado para representação desse processo (os principais formalismos aplicados para a representação da sintaxe são o tema do capítulo 2) e resulta em um ou mais submodelos. É importante ressaltar que esses submodelos são independentes, ou seja, o formalismo usado em um não necessariamente precisa ser usado no outro.

Parametrização do modelo: nessa etapa é feita a criação de uma função (ou modelo propriamente dito) que determine um valor real para qualquer par de sentenças fonte e alvo, visando à desambiguação dos resultados. É nesse momento que entra a estatística, através dos modelos gerativos e discriminativos. A ideia é definir quais parâmetros serão utilizados na determinação desse valor real. Os submodelos de tradução e língua definidos anteriormente são exemplos desses parâmetros. Essas duas primeiras etapas (definição e parametrização) são frequentemente reunidas em uma só, denominada *modelagem* (tema do capítulo 3).

Estimativa dos parâmetros: após definidos os parâmetros do modelo, é necessário obter os seus valores ou pesos. Na prática, isso inclui a determinação das regras de tradução e suas respectivas probabilidades. É nesse momento que entram os algoritmos de aprendizagem de máquina (descritos no capítulo 4).

Decodificação: finalmente, obtido o modelo com seus respectivos valores, é necessário fazer a tradução propriamente dita, em outras palavras: dada uma sentença de entrada, definir qual a sentença de saída mais adequada de acordo com o modelo (o que é detalhado no capítulo 5).

A formalização completa de um sistema de SMT deve passar pelas definições e métodos utilizados em cada uma dessas etapas.

1.2 Motivação

Através da melhoria constante dos modelos e algoritmos utilizados, foram obtidos vários avanços na SMT durante a década de 90 e início dos anos 2000. No entanto, nos últimos anos esses avanços têm diminuído: mudanças puramente estatísticas não têm mais trazido melhorias significativas de performance. O fato é que tradicionalmente, os modelos de SMT não possuem nenhum conhecimento linguístico explícito. Por conta disso, a maior parte dos trabalhos mais recentes na área têm buscado incorporar esse tipo de conhecimento aos sistemas estatísticos na tentativa de sair do patamar estagnado que a SMT atingiu.

Dentre o conhecimento linguístico ignorado por sistemas de SMT padrão está a sintaxe. Alguns dos problemas obtidos ao se utilizar esses sistemas é que a tradução resultante não leva em conta as diferenças sintáticas entre as duas línguas em questão. Um exemplo para o par de línguas inglês-português é mostrado abaixo, onde a entrada e a referência são as mesmas sentenças da figura 1.1 e a saída é a tradução feita por um sistema de SMT considerado estado-da-arte (no caso, o Moses (KOEHN et al., 2007)):

Entrada: The oldest poems are translated.

Referência: Traduzidos poemas mais antigos.

Saída: O mais antigo poemas são traduzidos.

O exemplo mostra três aspectos sintáticos que o tradutor não conseguiu resolver: a concordância entre as palavras “antigo” e “poemas”, o reordenamento entre o substantivo “poemas” e a locução “mais antigo” e a troca de voz passiva para voz ativa. Esses erros são típicos em sistemas de SMT padrão, especialmente no caso de línguas morfologicamente ricas como o português. Além disso, esses problemas tornam-se mais graves quando a tradução é feita entre línguas com ordem diferente de palavras, como o par inglês-árabe.[5]

A adição explícita de regras sintáticas que definam concordâncias, reordenamentos e outros fenômenos poderia resolver esses tipos de erros. Sendo assim, um dos focos atuais de pesquisa na área é justamente a busca de métodos e formalismos que possibilitem a adição dessas regras em sistemas de SMT. É nesse âmbito que este trabalho está inserido.

[5]Enquanto o inglês possui ordem Sujeito-Verbo-Objeto (SVO), o árabe possui ordem variável de palavras.

1.3 Objetivos

Considerando-se a necessidade de avançar nas pesquisas com MT, esse trabalho visa a investigação de técnicas estatísticas para a MT que incluam informação sintática, com foco no par de línguas inglês e português do Brasil (en-ptBR). Embora as técnicas investigadas sejam independentes de língua, os experimentos são realizados com textos escritos em português do Brasil e traduzidos, por um especialista humano, para o inglês americano. Desse modo, pretende-se avançar nas pesquisas com MT e, ao mesmo tempo, produzir recursos e resultados para uma língua ainda pouco explorada: o português do Brasil.

1.4 Estrutura do texto

O restante deste documento está organizado como segue:

- O capítulo 2 apresenta alguns modelos de representação de sintaxe, em especial, o formalismo utilizado neste trabalho, as Gramáticas de Substituição de Árvores (*Tree Substitution Grammars* - TSGs) e métodos para aprendizado automático dessas gramáticas a partir de *corpora*.
- A modelagem de sistema de SMT baseada em sintaxe é mostrada no capítulo 3. Essa modelagem inclui as etapas de definição e parametrização citadas anteriormente.
- Os algoritmos de aprendizado utilizados em SMT baseada em sintaxe são explicados no capítulo 4.
- O capítulo 5 trata do problema da decodificação e também sobre quais os métodos utilizados para avaliação de um sistema de tradução automática.
- Experimentos realizados com o par de línguas en-ptBR são o foco do capítulo 6, onde são descritos os recursos e ferramentas utilizados e também é feita uma avaliação dos resultados obtidos.
- Por fim, o capítulo 7 traz as considerações finais, mostrando as principais contribuições deste trabalho e possibilidades de trabalhos futuros.

2 FORMALISMOS PARA REPRESENTAÇÃO DE SINTAXE

Em PLN, a representação da informação sintática das sentenças é tradicionalmente feita através de *árvores sintáticas*. Essas árvores são normalmente de dois tipos: baseadas em *constituintes sintáticos*, onde palavras são agrupadas em *sintagmas*, que por sua vez são agrupados formando uma sentença; ou baseadas em *relações de dependência*, onde as palavras são diretamente conectadas umas às outras baseando-se nesse tipo de relação.

As árvores sintáticas são geradas através de gramáticas: formalismos que visam a modelar corretamente os aspectos sintáticos da língua em questão. O formalismo mais clássico para representação sintática são as Gramáticas Livres de Contexto (*Context-Free Grammars* - CFGs), sendo utilizadas em diversas áreas da computação além do NLP. No entanto, a linguagem natural possui diversas ambiguidades díficeis de serem capturadas por CFGs padrão. Por conta disso, diversas extensões foram propostas com o objetivo de tratar melhor essas ambiguidades.

A seção 2.1 define as CFGs e mostra mais detalhadamente quais as suas limitações para a representação da sintaxe em uma linguagem natural. Na seção 2.2 são apresentadas as gramáticas de árvores, um formalismo mais poderoso que consegue capturar uma maior gama de fenômenos sintáticos. Por fim, a seção 2.3 trata do problema de como criar uma gramática a partir de *corpora*.

O foco deste trabalho está na representação da sintaxe através de árvores de constituintes. No entanto, vale ressaltar que os formalismos e métodos apresentados também podem ser aplicados para árvores de dependência.

2.1 Gramáticas Livres de Contexto

Uma CFG é definida formalmente como uma 4-upla (N, W, S, R), onde:

- N é um conjunto de símbolos não-terminais,
- W é um conjunto de símbolos terminais,
- $S \in N$ é o símbolo inicial e
- R é um conjunto de regras de produção

Em uma representação sintática baseada em árvores de constituintes, símbolos terminais podem ser vistos como itens lexicais (palavras, sinais de pontuação, etc.) e símbolos não-terminais como classes morfológicas e constituintes sintáticos (sintagmas). Já as regras dizem como esses elementos devem se combinar de acordo com a língua em questão. Por exemplo, é possível definir um modelo (simples) do português através da seguinte CFG $P = (N_p, W_p, R_p, S_p)$, onde:

- $S_p = S$
- $N_p = \{S, NP, VP, PP, Det, N, V, P\}$
- $W_p = \{o, homem, viu, a, estrela, com, telescópio\}$

E o conjunto de regras R_p é dado por:

$$
\begin{aligned}
S &\rightarrow NP\ VP \\
S &\rightarrow NP\ VP\ PP \\
NP &\rightarrow Det\ N \\
NP &\rightarrow NP\ PP \\
VP &\rightarrow V\ NP \\
PP &\rightarrow P\ NP \\
Det &\rightarrow o \\
Det &\rightarrow a \\
N &\rightarrow homem \\
N &\rightarrow estrela \\
N &\rightarrow telescópio \\
V &\rightarrow viu \\
P &\rightarrow com
\end{aligned}
$$

O modelo descrito acima consegue reconhecer a sentença *o homem viu a estrela com o telescópio*. Note que essa sentença é ambígua (não sabemos se quem possui o telescópio é o homem ou a estrela) sendo possível gerar duas árvores sintáticas para ela a partir da gramática dada, como mostra a figura 2.1. Intuitivamente, podemos concluir que muito provavelmente a árvore correta é a primeira (a que atribui o sintagma *com o telescópio* ao substantivo *homem*) mas a gramática acima não possui formas de assinalar essa preferência. Por outro lado, também não é possível afirmar que a segunda árvore está totalmente incorreta (por exemplo, *estrela* pode ter o significado de "pessoa famosa") e, portanto, não podemos simplesmente modificar a gramática para que ela não gere a segunda árvore.

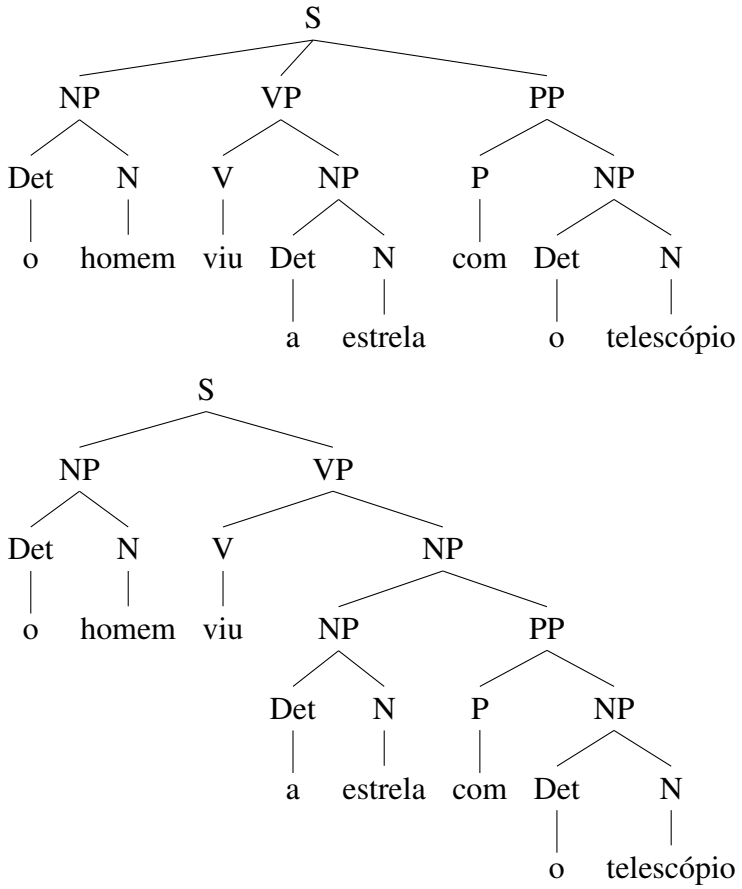


Figura 2.1: Duas árvores sintáticas diferentes para a mesma sentença

As limitações descritas acima são decorrentes do fato das CFGs serem um formalismo *determinístico*. Por conta disso, na maior parte das aplicações de NLP elas são estendidas para um formalismo *probabilístico*, as Gramáticas Livres de Contexto Probabilísticas (*Probabilistic Context-Free Grammars* - PCFGs). Em uma PCFG, cada regra possui uma probabilidade, com a condição de que o somatório das probabilidades de um conjunto de regras que contenha o mesmo lado esquerdo deve ser igual a 1. Por exemplo, é possível estender o conjunto de regras

da gramática anterior da seguinte forma:

$S \rightarrow NP\ VP$	0.6
$S \rightarrow NP\ VP\ PP$	0.4
$NP \rightarrow Det\ N$	0.7
$NP \rightarrow NP\ PP$	0.3
$VP \rightarrow V\ NP$	1.0
$PP \rightarrow P\ NP$	1.0
$Det \rightarrow o$	0.5
$Det \rightarrow a$	0.5
$N \rightarrow homem$	0.5
$N \rightarrow estrela$	0.3
$N \rightarrow telescópio$	0.2
$V \rightarrow viu$	1.0
$P \rightarrow com$	1.0

Nesse modelo, cada árvore sintática possui uma probabilidade correspondente, calculada através do produto das probabilidades das regras utilizadas em sua composição. Dessa forma, é possível apontar entre as várias possíveis árvores sintáticas de uma sentença qual a mais provável. No caso das duas árvores da figura 2.1, por exemplo, a primeira teria probabilidade igual a 0.0005145 e a segunda, 0.0002315. Com esses valores, é possível afirmar então que a primeira árvore é a mais provável pois possui probabilidade maior.

Gramáticas probabilísticas em geral se tornaram extremamente comuns nas aplicações de NLP. Por conta disso, muitas vezes na literatura o termo "probabilísticas" é omitido e elas são referidas simplesmente como "gramáticas" (assim como omite-se o "P" nas siglas correspondentes: "PCFGs" são referidas apenas como "CFGs"). Essa convenção será adotada durante o restante deste texto. Assim, por CFG entenda-se PCFG, a menos que se explicite o contrário.

2.1.1 Limitações das CFGs

Ainda que a adição de probabilidades permita a desambiguação de árvores sintáticas, ela não resolve outros problemas que são inerentes ao formalismo em si. Por exemplo, o seguinte

conjunto de regras:

$S \rightarrow NPVP$	1.0
$NP \rightarrow Det\,N$	1.0
$VP \rightarrow V$	1.0
$Det \rightarrow o$	0.7
$Det \rightarrow a$	0.3
$N \rightarrow menino$	0.6
$N \rightarrow menina$	0.4
$V \rightarrow caiu$	1.0

permite reconhecer as seguintes sentenças com as suas respectivas probabilidades:

- *o menino caiu* p=0.42
- *a menina caiu* p=0.12
- **o menina caiu* p=0.28
- **a menino caiu* p=0.18

Note que a terceira sentença possui um erro gramatical (indicado pelo símbolo '*') mas possui uma probabilidade maior que a segunda sentença, que está correta. Esse tipo de fenômeno ocorre porque em uma CFG, a decisão de qual regra deve ser utilizada é baseada apenas no símbolo do lado esquerdo, quando isso claramente não corresponde à realidade. Por exemplo, no terceiro caso acima, a expansão do símbolo *Det* no artigo *o* se deu apenas considerando o fato de ser um determinante, quando deveria levar em conta também o substantivo seguinte (*menina*). Espera-se, portanto, que um formalismo adequado de representação sintática seja capaz de detectar esse tipo de fenômeno.

Uma alternativa para resolver problemas desse tipo é artificialmente expandir o conjunto de não-terminais da gramática de forma a modelar esses fenômenos. Por exemplo, o conjunto de regras abaixo:

$Det \rightarrow o$	0.7
$Det \rightarrow a$	0.3

poderia ser expandido para:

$$Det\hat{}NP\{menino\} \rightarrow o \qquad 0.9$$
$$Det\hat{}NP\{menino\} \rightarrow a \qquad 0.1$$
$$Det\hat{}NP\{menina\} \rightarrow o \qquad 0.1$$
$$Det\hat{}NP\{menina\} \rightarrow a \qquad 0.9$$

As regras acima são lidas da seguinte forma: um *Det* cujo pai seja um *NP* e o *head*[1] desse pai seja *menino* expande no símbolo *o* com probabilidade 0.9. Esse é o procedimento feito por alguns dos analisadores sintáticos clássicos em NLP: enquanto Charniak (1997) adiciona informações dos nós ancestrais, Collins (1999) utiliza informações de *head*, entre outras características.

Petrov et al. (2006) e Petrov e Klein (2007) também dividem os não-terminais mas ao invés de utilizar essas informações explicitamente eles tentam dividi-los automaticamente buscando minimizar alguma métrica de erro. No exemplo dado, uma divisão possível é mostrada abaixo:

$$Det_0 \rightarrow o \qquad 0.9$$
$$Det_0 \rightarrow a \qquad 0.1$$
$$Det_1 \rightarrow o \qquad 0.1$$
$$Det_1 \rightarrow a \qquad 0.9$$

Uma outra forma de atacar esses problemas é alterando o formalismo utilizado. Formalmente, CFGs reconhecem linguagens que definem *strings* ou sequências de símbolos (no caso, os símbolos são os itens lexicais). Por conta disso, ainda que a representação de uma derivação seja em formato de árvore, fundamentalmente uma CFG tem o poder apenas de reconhecer ou não a sentença propriamente dita. Considerando esse aspecto, trabalhos como Bod (2003), Post e Gildea (2009a) e Cohn, Blunsom e Goldwater (2010) modelam o processo de análise sintática através de *gramáticas de árvores*.

[1] O *head* é a palavra principal de um sintagma.

2.2 Gramáticas de árvores

Como o nome diz, gramáticas de árvores reconhecem as denominadas *linguagens de árvores*. Assim como as linguages de *strings* são definidas como um conjunto de *strings*, essas linguagens são definidas como um conjunto de árvores (que podem ser árvores sintáticas, por exemplo). Além disso, elas também possuem uma hierarquia semelhante à hierarquia de Chomsky (JURAFSKY; MARTIN, 2000, p.474), definindo por exemplo as *Linguagens de Árvores Regulares* (*Regular Tree Languages* - RTLs), as *Linguagens de Árvores Livres de Contexto* e assim sucessivamente.

Teoricamente, *strings* e árvores formam universos diferentes. No entanto, é possível comparar o poder representacional entre eles considerando-se os seguintes fatos:

- Um conjunto de derivações de uma CFG forma uma linguagem de árvores (mais especificamente, uma RTL).
- A sequência de folhas de uma árvore da esquerda para a direita forma uma *string*. Essa sequência é denominada *produto* ou *yield* de uma árvore. O conjunto dos produtos de uma RTL forma uma linguagem livre de contexto.

O maior poder representacional das linguagens de árvores se revela no fato de que a recíproca do primeiro item acima não é verdadeira: ainda que um conjunto de derivações de uma CFG forme uma RTL, existem RTLs que não podem ser modeladas como CFGs. A figura 2.2 mostra um exemplo: uma CFG que reconheça as duas árvores da figura deveria ter a regra recursiva $S \rightarrow SS$. Mas essa regra acarretaria em um conjunto infinito de árvores, diferente do conjunto mostrado na figura.

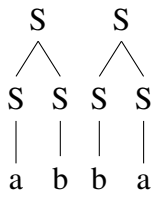


Figura 2.2: Exemplo de uma RTL que não pode ser reconhecida por uma CFG

Existem vários formalismos diferentes que possuem o poder representacional de reconhecer RTLs. Em NLP, um dos mais utilizados são as Gramáticas de Substituição de Árvores (*Tree Substitution Grammars* - TSGs), que são o foco deste trabalho.

2.2.1 Gramáticas de Substituição de Árvores

Diferente das CFGs, existem diversas definições na literatura para as TSGs. Nesse trabalho será utilizada a de Joshi e Schabes (1997), que define uma TSG como uma 4-upla (N, W, S, E), onde:

- N é um conjunto de símbolos não-terminais,
- W é um conjunto de símbolos terminais,
- $S \in N$ é o símbolo inicial e
- E é um conjunto de regras ou *árvores elementares*. As folhas dessas árvores podem ser símbolos terminais ou não-terminais. Os não-terminais são denominados *pontos de substituição* e são marcados com o símbolo $\downarrow$.

A ideia de uma TSG é, a partir de uma árvore elementar, gerar novas árvores através da operação de *substituição*. Essa operação consiste em, dada uma árvore, substituir um de seus não-terminais marcados por outra árvore elementar cuja raiz seja rotulada pelo mesmo não-terminal, gerando uma nova árvore. Assim, uma derivação em uma TSG consiste em uma sequência de substituições a partir de uma árvore inicial cuja raiz seja o símbolo inicial S até que as folhas da árvore resultante sejam somente símbolos terminais, como mostra a figura 2.3.

Nessa definição, uma CFG pode ser vista como uma TSG onde todas as árvores elementares possuem altura igual a 1. Por conta disso, alguns trabalhos classificam as TSGs como uma generalização das CFGs.

A figura 2.3 também mostra um exemplo de como as TSGs conseguem modelar as dependências mostradas na seção 2.1.1: a árvore elementar α_2 expande o sintagma nominal até chegar às palavras *a* e *menina*, fazendo com que a concordância seja respeitada. Diferente das soluções apresentadas na seção 2.1.1, isso é feito mantendo-se o mesmo conjunto de não-terminais, eliminando a necessidade de expansão desse conjunto através das anotações feitas pelos analisadores citados anteriormente (POST, 2010, p.54).

Uma diferença fundamental entre as CFGs e as TSGs é que estas permitem que uma mesma árvore sintática possua mais de uma derivação possível. Por exemplo, a figura 2.4 mostra uma outra derivação possível que resulta na mesma árvore da figura 2.3. Em suma, qualquer conjunto de árvores elementares que forme uma árvore sintática é caracterizado como uma derivação. Por conta disso, a probabilidade de uma árvore é calculada somando-se as probabilidades de cada derivação possível. Por sua vez, a probablidade de uma derivação é o produto das probabilidades

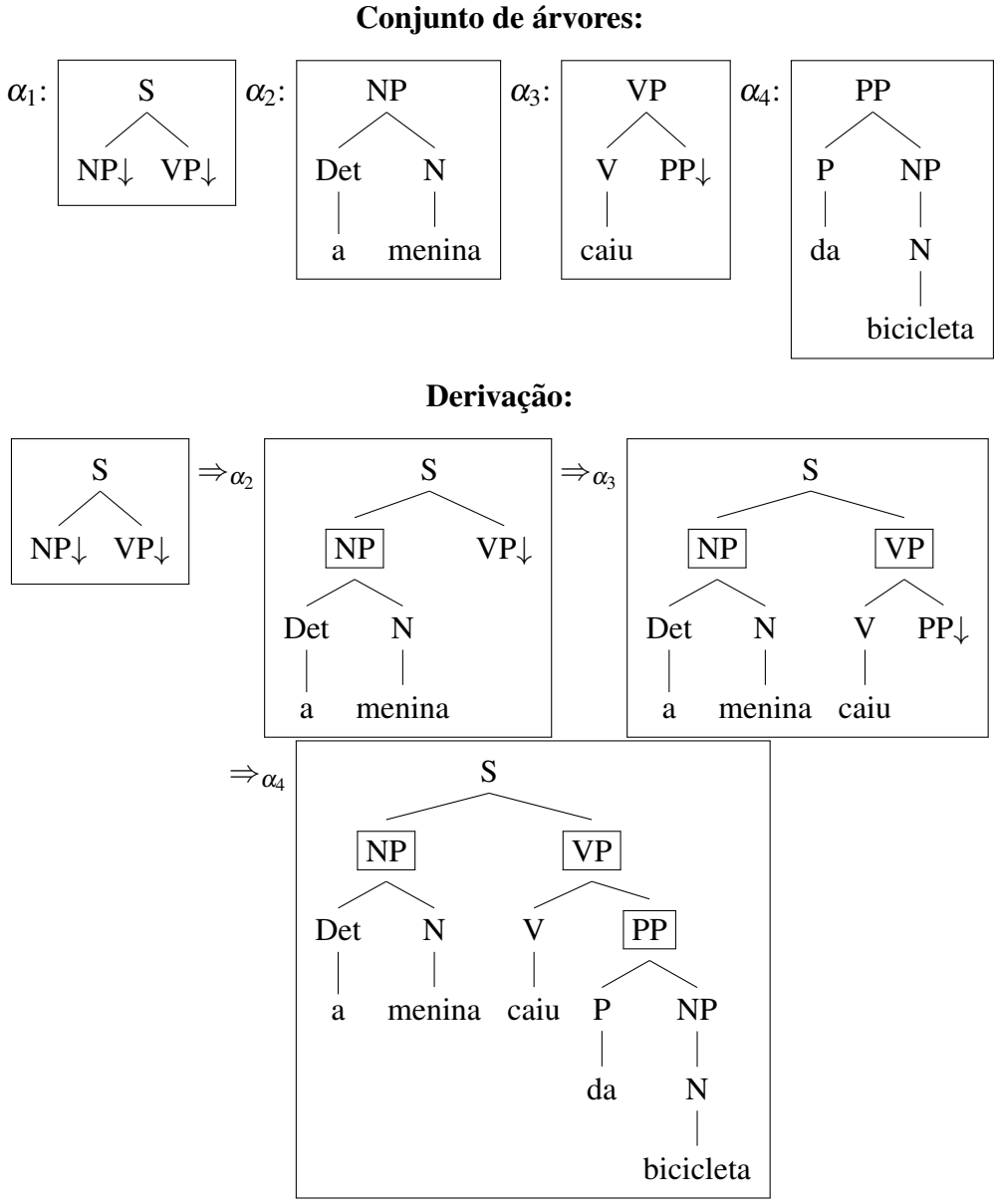


Figura 2.3: Exemplo de derivação em uma TSG. Nodos destacados com retângulos mostram pontos onde houve uma substituição.

de cada regra contida nela, resultando na fórmula abaixo:

$$P(\textit{árvore}) = \sum_{\textit{deriv} \in \textit{derivs}(\textit{árvore})} \prod_{\textit{regra} \in \textit{regras}(\textit{deriv})} P(\textit{regra}) \tag{2.1}$$

Determinar o conjunto $\textit{derivs}(\textit{árvore})$ (ou seja, todas as derivações possíveis para uma árvore sintática) é NP-difícil (SIMA'AN, 1996). Como consequência, a equação acima normalmente é aproximada utilizando somente a melhor derivação (com maior probabilidade) ou as n melhores derivações. Na prática, ambas as aproximações produzem resultados muito similares (COHN; GOLDWATER; BLUNSOM, 2009) mas o primeiro método possui uma vantagem: como apenas uma derivação é considerada, é possível transformar a TSG em uma CFG e utilizar

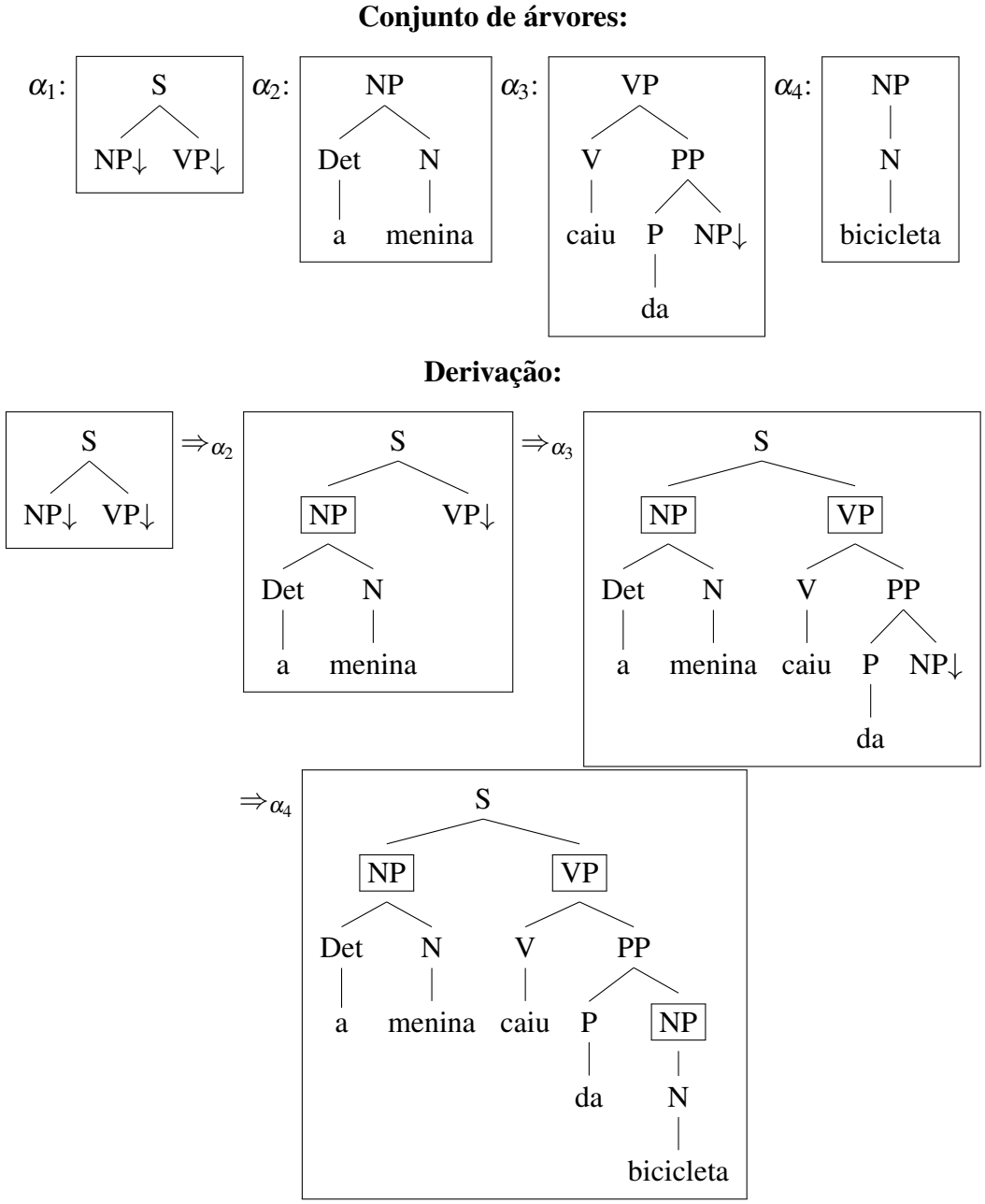


Figura 2.4: Exemplo de derivação alternativa que resulta na mesma árvore da figura 2.3

os algoritmos de análise sintática para esse tipo de gramática. Para isso, as regras são *linearizadas* e novos não-terminais são adicionados com o intuito de recuperar a estrutura sintática perdida na linearização, como mostra a figura 2.5.

2.3 Aprendizado de gramáticas

Uma questão que surge naturalmente ao discutir a utilização de gramáticas em NLP é como construí-las. Teoricamente, é possível elaborar as regras de uma gramática manualmente, assim como atribuir as probabilidades correspondentes a cada regra. Entretanto, construir a gramática

TSG:	**CFG:**	**Novos não-terminais:**
$S(NP\,VP)$	$S \rightarrow NP\,VP$	$ELEMTREE_0$:
$NP(Det(a)\,N(menina))$	$NP \rightarrow ELEMTREE_0$	"$Det(a)\,N(menina)$"
$VP(V(caiu)\,PP(P(da)\,NP))$	$VP \rightarrow ELEMTREE_1$	$ELEMTREE_1$:
$NP(N(bicicleta))$	$NP \rightarrow ELEMTREE_2$	"$V(caiu)\,PP(P(da)\,NP)$"
	$ELEMTREE_0 \rightarrow a\ menina$	$ELEMTREE_2$:
	$ELEMTREE_1 \rightarrow caiu\ da\ NP$	"$N(bicicleta)$"
	$ELEMTREE_2 \rightarrow bicicleta$	

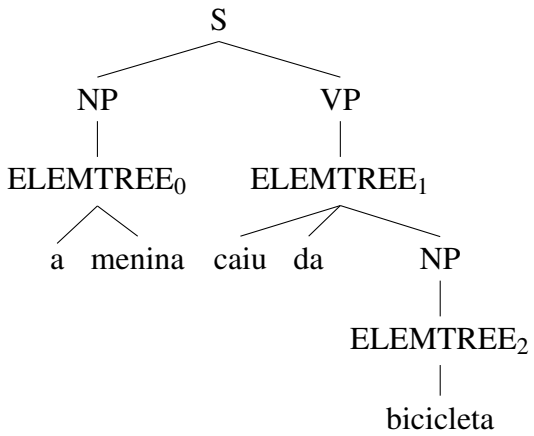


Figura 2.5: Exemplo de transformação de uma TSG em uma CFG e uma análise. As árvores elementares estão representadas em formato de parênteses. A derivação original (feita pela TSG) pode ser recuperada substituindo os não-terminais $ELEMTREE$ pela sua árvore elementar correspondente.

dessa forma é extremamente custoso. Para tentar amenizar esse custo, assim como em várias outras aplicações de NLP, busca-se obter métodos que automatizem essa tarefa.

Uma das formas de realizar essa construção automática é através de aprendizado de máquina a partir de *corpora*. A ideia é utilizar um tipo especial de *corpus*, os *treebanks*, que são conjuntos de sentenças com suas respectivas árvores sintáticas. A partir de um *treebank* (T), busca-se então a gramática (G) que possua a maior probabilidade, ou matematicamente, o objetivo é inferir uma gramática $\hat{G}$ que respeite a equação:

$$\hat{G} = \underset{G}{argmax}\ P(G|T) \tag{2.2}$$

A equação acima é expandida utilizando o teorema de Bayes:

$$\hat{G} = \underset{G}{argmax}\ \frac{P(T|G) \times P(G)}{P(T)} \tag{2.3}$$

O denominador $P(T)$ é constante (pois o *treebank* não muda) e serve apenas para normalizar o termo $P(G|T)$. Como o objetivo é apenas obter a gramática G e não sua probabilidade, esse

termo pode ser descartado:

$$\hat{G} = \underset{G}{argmax}\ P(T|G) \times P(G) \tag{2.4}$$

Intuitivamente, o que se tem é um modelo *gerativo*, onde supomos que a gramática é dada (com probabilidade $P(G)$) e através dessa gramática que o *treebank* é *gerado*. Os termos $P(G|T)$, $P(T|G)$ e $P(G)$ são denominados *posteriori*, *verossimilhança* e *priori*, respectivamente. A priori, como o nome diz, tem como objetivo modelar alguma informação a priori que possamos ter sobre a gramática G. Entretanto, geralmente não possuímos nenhuma informação desse tipo. Isso resulta em uma priori uniforme, que consequentemente pode também ser descartada:

$$\hat{G} = \underset{G}{argmax}\ P(T|G) \tag{2.5}$$

A equação acima corresponde a um método estatístico baseado na *Estimativa de Máxima Verossimilhança* (*Maximum Likelihood Estimate* - MLE). No caso de uma CFG, é possível provar que contando as frequências das regras relativas ao seu símbolo do lado esquerdo chega-se ao MLE. Como em um *treebank* as regras estão implícitas nas árvores sintáticas (cada fragmento de árvore com altura igual a 1 é uma regra), esse é o procedimento adotado para aprender uma CFG.

No caso das TSGs, se as árvores de um *treebank* estiverem anotadas com suas respectivas derivações (ou seja, com as árvores elementares de cada derivação explicitamente anotadas), é possível realizar o mesmo procedimento para se obter o MLE. No entanto, normalmente os *treebanks* não possuem esse tipo de anotação. Ou seja, não é possível saber qual foi o conjunto de árvores elementares utilizado para derivar cada árvore.

O problema de obtenção do MLE em uma TSG pode ser definido da seguinte forma: se soubéssemos as derivações, bastaria ler a gramática e contar as frequências. Por outro lado, se soubéssemos qual é a gramática, poderíamos gerar as derivações das árvores do *treebank*. Um algoritmo clássico para resolver problemas desse tipo é o *Expectation-Maximization* (EM) (MANNING; SCHüTZE, 2000, p.518). Aplicando o EM ao problema em questão, é possível obter o seguinte algoritmo:

1) Gere a gramática com todas as regras existentes no *treebank* (considerando todas as derivações possíveis) e defina suas probabilidades com algum valor (por exemplo, seguindo uma distribuição uniforme);

2) Realize a análise sintática das sentenças do *treebank*, obtendo valores de probabilidades para cada derivação;

3) Atualize as probabilidades das regras de acordo com as probabilidades obtidas no item 2;

4) Volte ao passo 2 e repita o procedimento até as probabilidades convergirem.

O método descrito acima possui um problema no passo 1: o número de árvores elementares é exponencial em relação ao tamanho da sentença de entrada. Isso exige a adoção de heurísticas que limitem o tamanho da gramática, como por exemplo, número máximo de símbolos ou altura máxima da árvore. Definir esse tipo de heurística não é trivial pois elas podem excluir regras interessantes à gramática, assim como podem incluir regras pouco informativas.

Um outro problema com essa abordagem é a questão do *overfitting*. O aprendizado de uma TSG a partir do MLE resulta na otimização da seguinte função:

$$P(G|T) = \prod_{\text{árvore} \in T} \sum_{\text{deriv} \in \text{derivs}(\text{árvore})} \prod_{\text{regra} \in \text{regras}(\text{deriv})} P(\text{regra}) \tag{2.6}$$

No entanto, não é possível otimizá-la pois, como visto anteriormente, obter o termo $\text{derivs}(\text{árvore})$ não é tratável computacionalmente. A equação é então aproximada utilizando o método da melhor derivação:

$$P(G|T) = \prod_{\text{deriv} \in T} \prod_{\text{regra} \in \text{regras}(\text{deriv})} P(\text{regra}) \tag{2.7}$$

O problema da equação acima é que é possível provar que ela é otimizada quando essa derivação é composta por uma única regra que gera a árvore inteira. Intuitivamente, o que ocorre é que o segundo produtório da equação tende a ter um resultado menor à medida que mais regras são utilizadas na derivação pois os termos $P(\text{regra})$ são valores entre 0 e 1. Por conta disso, o EM, ao convergir, terá aprendido uma gramática degenerada, composta apenas por regras que expandam árvores completas. Mesmo que sejam consideradas as melhores n derivações ao invés de simplesmente a melhor, a tendência é que a gramática seja composta apenas por árvores grandes, que simplesmente memorizam o *treebank* e não possuem poder de generalização. Novamente, é possível mitigar esse problema através das heurísticas de restrição de tamanho descritas anteriormente, mas, além dos problemas já citados, a tendência é que a gramática passe a ter regras do maior tamanho possível dentro daquela heurística.

Por conta dessas limitações, trabalhos recentes (POST; GILDEA, 2009b; COHN; GOLDWATER; BLUNSOM, 2009) buscam aprender TSGs através de modelos e métodos Bayesianos. Em especial, o modelo representado na equação 2.3 é modificado. O objetivo deixa de ser inferir uma gramática $\hat{G}$ e passa a ser inferir uma *sequência de derivações* $\hat{S}$, resul-

tando na seguinte equação:

$$\hat{S} = \underset{S}{argmax}\ P(T|S) \times P(S) \tag{2.8}$$

Uma sequência de derivações S representa uma sequência de árvores elementares que, aplicadas na ordem dada, resultam no treebank T. Isso é diferente do modelo anterior pois uma mesma gramática G pode gerar várias sequências S diferentes. Ao mesmo tempo, cada sequência S define um único treebank. O resultado disso é que a verossimilhança $P(T|S)$ pode assumir apenas os valores 1 (quando a S gera T) ou 0. Dessa forma, o trabalho de inferência é feito exclusivamente pela priori $P(S)$. Diferente do modelo anterior, onde a priori é considerada uniforme, nesse modelo ela é definida como um *Processo Dirichlet* (*Dirichlet Process* - DP).

2.3.1 Processo Dirichlet

O Processo Dirichlet é definido como uma distruibuição sobre o espaço infinito de possíveis árvores elementares[2]. Essa distribuição tem a característica de levar em conta todas as árvores elementares utilizadas no processo gerativo de construção do *treebank*. Note que isso é uma diferença fundamental em relação ao modelo anterior, que leva em conta apenas o símbolo raiz.

Formalmente, a distribuição de uma árvore elementar e de acordo com sua raiz r é definida como:

$$\begin{aligned} e|r &\sim G_r \\ G_r|\alpha_r, P_0 &\sim DP(\alpha_r, P_0(\cdot|r)) \end{aligned} \tag{2.9}$$

onde $P_0(\cdot|r)$ (a *distribuição base*) é uma distribuição sobre o conjunto infinito de árvores elementares cuja raiz seja o não-terminal r e α_r é denominado o *parâmetro de concentração* para o símbolo r. Intuitivamente, a distribuição base define quais árvores elementares queremos criar no processo gerativo enquanto o parâmetro de concentração controla a tendência de criar novas árvores ou reutilizar árvores já criadas.

Não é possível amostrar diretamente de uma distribuição DP mas integrando sobre todas as sequências possíveis obtém-se uma distribuição condicional que define a probabilidade da próxima árvore elementar ser gerada (POST, 2010, p. 59). Assim, a probabilidade de uma árvore e_i (sendo i a posição dessa árvore na sequência de derivações) dado a sua raiz r_i é definida como:

$$P(e_i|\vec{e}_{<i}, r_i, \alpha_{r_i}, P_0) = \frac{count(e_i) + \alpha_{r_i} P_0(e_i|r_i)}{count(r_i) + \alpha_{r_i}} \tag{2.10}$$

[2]Para uma explicação generalizada sobre o DP, sugere-se a leitura do Apêndice A em Goldwater, Griffiths e Johnson (2009).

onde $\vec{e}_{<i}$ é o conjunto de todas as outras árvores elementares utilizadas no processo gerativo até o momento, $count(e_i)$ é o total de vezes que e_i aparece em $\vec{e}_{<i}$ e $count(r_i)$ é o total de vezes que uma árvore elementar com raiz r_i aparece em $\vec{e}_{<i}$. No caso em que α_{r_i} é igual a 0, a fórmula acima torna-se a frequência relativa da árvore elementar em questão. Por conta disso, essa fórmula pode ser vista intuitivamente como um cálculo da frequência relativa mas que também leva em conta a distribuição base.

Uma outra forma de entender o DP é considerá-lo como um modelo de *cache*, onde cada vez que uma nova árvore é gerada, o modelo opta por escolher uma a partir dessa *cache* de árvores já existentes ou criar uma nova a partir da distribuição base. À medida que o *treebank* é gerado, a *cache* torna-se cada vez maior e, portanto, o modelo tende a escolher as árvores a partir dela. Esse fenômeno é algo que está presente no uso da linguagem natural: ainda que seja sempre possível criar novas estruturas sintáticas, nós tendemos a utilizar estruturas já existentes mais frequentemente.

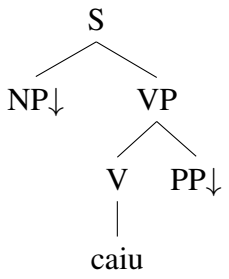
A distribuição base define quais as árvores que irão aparecer na gramática resultante. Conforme visto anteriormente, árvores grandes demais degeneram a gramática e causam *overfitting*. Para atacar esse problema, a distribuição base é definida de forma a dar preferência a árvores menores, que generalizem melhor os fenômenos sintáticos do *treebank*.

Uma forma de fazer isso é definí-la como um processo gerativo próprio onde se considera que cada árvore elementar é criada através de um conjunto de regras de uma CFG. Para cada não-terminal existente na árvore, é feita uma escolha entre expandí-lo ou não e, caso positivo, qual regra é utilizada para expandí-lo. As probabilidades das regras dessa CFG são obtidas usando o mesmo *treebank* de entrada e o método de MLE, enquanto a escolha de expandir o não-terminal é feita utilizando uma distribuição de Bernoulli com parâmetro β_r. A figura 2.6 mostra um exemplo de como esse cálculo é feito. Como a probabilidade de cada árvore é definida em termos de uma multiplicação de probabilidades, árvores menores terão preferência pois possuirão valores maiores.

2.3.2 Amostragem de Gibbs

Utilizando uma priori baseada no DP para prevenir o *overfitting*, poderíamos aplicar o EM e maximizar a equação 2.8. No entanto, mesmo nesse novo modelo ainda persiste o problema da intratabilidade de enumerar todas as derivações do *treebank*. Assim, ao invés de enumerar todas elas, iremos *amostrar* algumas dessas derivações através de um método denominado *Amostragem de Gibbs*.

Árvore elementar:



Parâmetros:
$\beta_{NP} = 0.4$, $\beta_{VP} = 0.3$, $\beta_{V} = 0.8$, $\beta_{PP} = 0.5$
$r_1 : S \rightarrow NP\ VP\ |prob = 0.4$
$r_2 : VP \rightarrow V\ PP\ |prob = 0.2$
$r_3 : V \rightarrow caiu\ |prob = 0.05$

Cálculo:
$P(r_1) \times (1 - \beta_{NP}) \times \beta_{VP} \times P(r_2) \times \beta_{V} \times P(r_3) \times (1 - \beta_{PP}) = 0.000288$

Figura 2.6: Exemplo de cálculo de probabilidade de uma árvore elementar segundo uma distribuição base

A ideia da amostragem de Gibbs é alterar o valor de uma variável do modelo mantendo fixo o valor de todas as outras variáveis. No problema em questão, essas variáveis são todos os símbolos não-terminais do *treebank* (com exceção das raízes das árvores). Considerando que uma derivação é composta por uma sequência de árvores elementares, cada um desses símbolos pode ser a raiz de uma dessas árvores ou um nó interno de outra árvore. Em outras palavras, cada símbolo possui uma *flag* de substituição, que possui o valor *Verdadeiro* quando ele é raiz (um ponto de substituição) e *Falso* caso seja um nó interno. Nesse esquema, cada nodo do *treebank* com seu respectivo valor da *flag* define uma derivação (amostra). Por exemplo, na árvore sintática das figuras 2.3 e 2.4, os nodos marcados com retângulos indicam que suas respectivas *flags* possuem o valor *Verdadeiro*.

O processo realizado pela amostragem de Gibbs consiste em visitar cada nodo existente no *treebank* e decidir entre manter ou trocar o valor da sua respectiva *flag*. Cada um dos dois valores possíveis para uma *flag* define uma derivação possível para o *treebank* e, consequentemente, dois valores possíveis para as respectivas posterioris. O amostrador realiza a decisão baseado na proporção entre essas posterioris. Por exemplo, se o valor da posteriori para $flag = Verdadeiro$ é 0.03 e o valor para $flag = Falso$ for 0.02, o amostrador irá setar a flag para o valor *Verdadeiro* com 60% de chance e para *Falso* com 40% de chance.

Para calcular o valor dessas posterioris, seria necessário levar em conta todas as árvores elementares em cada uma das derivações. No entanto, o amostrador não precisa desses valores exatos, apenas a *proporção* entre eles. Por conta disso, ao fazer o cálculo dessa proporção, o amostrador considera apenas as árvores elementares diretamente associadas com o nodo em questão. Conforme mostra a figura 2.7, um nodo define três árvores elementares:

- Uma árvore *mesclada* (*merged*), onde a *flag* possui o valor *Falso* (significando que esse nodo é interno à essa árvore elementar);

- Uma árvore *interna* e uma árvore *externa*, onde a *flag* possui o valor *Verdadeiro* (significando que o nodo é um ponto de substituição).

As probabilidades das árvores acima são calculadas usando a fórmula 2.10. Esse cálculo é possível de ser feito dessa forma porque a distribuição gerada pelo DP é *intercambiável*, o que significa que podemos considerar cada árvore elementar como a última a ser gerada no processo gerativo de criação do *treebank*. Os valores resultantes para definir o valor de uma *flag* são então calculados da seguinte forma:

$$\begin{aligned} P(flag = False) &= \frac{P(mesclada)}{P(mesclada) + (P(interna) \times P(externa))} \\ P(flag = True) &= \frac{P(interna) \times P(externa)}{P(mesclada) + (P(interna) \times P(externa))} \end{aligned} \tag{2.11}$$

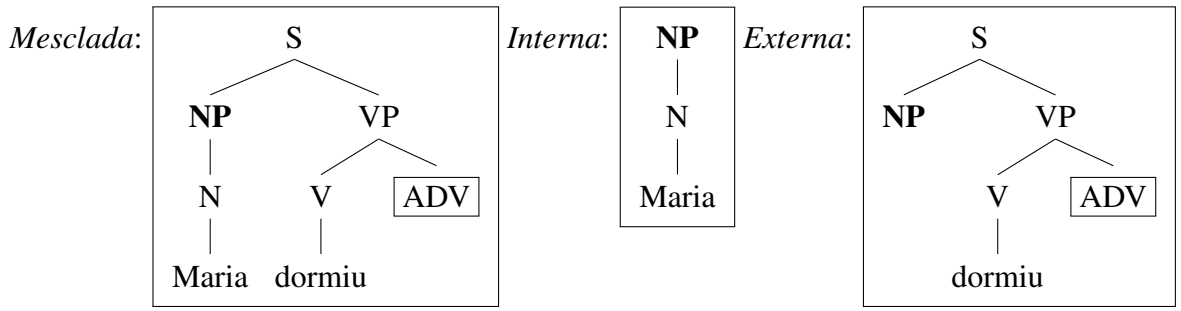


Figura 2.7: Árvores elementares definidas para um nodo visitado pelo amostrador de Gibbs (nesse caso, o nodo destacado NP)

O algoritmo de amostragem começa definindo uma valor para cada *flag* no *treebank* (podendo ser predefinido ou aleatório) e então repete esse processo para cada não-terminal não-raiz. Caso decida trocar o valor de uma *flag*, as contagens na *cache* são atualizadas de acordo com as árvores elementares referentes àquele nodo. Esse processo corresponde a uma iteração do amostrador. Após um número razoável de iterações, a derivação resultante tende a maximizar a posteriori $P(S|T)$, a partir da qual podemos extrair a TSG resultante.[3]

A combinação de modelos Bayesianos baseados no Processo Dirichlet junto com procedimentos de amostragem como o de Gibbs têm sido uma tendência em várias aplicações de NLP além da inferência de TSGs. Goldwater, Griffiths e Johnson (2009) utilizam o mesmo processo para o problema da segmentação em reconhecimento de fala. Em especial, esse trabalho mostra que, em problemas de segmentação em geral, o método de estimar o MLE através do EM leva ao *overfitting* devido às mesmas razões citadas para a inferência de TSGs. Uma forma de enxergar a inferência de uma TSG a partir de um *treebank* é como um problema de

[3]Teoricamente, a amostragem de Gibbs tende a atingir o valor máximo da posteriori à medida que o número de iterações tende a infinito.

segmentação: são dadas árvores sintáticas e o objetivo é segmentá-las em árvores elementares. Outros problemas que também já foram atacados pela combinação do DP com amostragem são a modelagem de documentos (TEH; JORDAN; BEAL, 2006) e a criação de tabelas de frases em SMT (DENERO; BOUCHARD-CôTé; KLEIN, 2008).

3 MODELAGEM DE UM SISTEMA DE SMT BASEADA EM SINTAXE

O modelo de SMT se baseia na ideia de que cada sentença na língua alvo é uma tradução possível da sentença fonte (BROWN et al., 1990). A tarefa desse modelo é fazer a desambiguação dessas possíveis traduções ou *candidatos*, buscando aquela que seja a mais adequada. Em SMT essa desambiguação é feita adicionando-se pesos ou probabilidades a cada candidato. Formalmente, dada uma sentença fonte f, é definida uma probabilidade $P(e|f)$ para cada candidato e. A tarefa de um sistema de SMT é então definida como a busca de uma sentença $\hat{e}$ que satisfaça a seguinte propriedade:

$$\hat{e} = \underset{e}{argmax}\ P(e|f) \tag{3.1}$$

Em outras palavras, busca-se o candidato e que possua a maior probabilidade de ser a tradução da sentença fonte f.

Seguindo o processo explicado por Lopez (2008), o primeiro passo é *definir* um formalismo para o processo de tradução, que irá então gerar um modelo para a função $P(e|f)$. A seção 3.1 mostra alguns desses formalismos, com foco naqueles que utilizam informação sintática.

Independente do formalismo utilizado, calcular o valor da função $P(e|f)$ diretamente traz problemas teóricos e práticos. Por um lado, o conjunto de candidatos tende a ser muito grande ou até mesmo infinito. Por outro lado, mesmo que esse conjunto pudesse ser enumerado, seria necessário uma quantidade de dados muito grande para obter valores confiáveis para essa função. Assim, o segundo passo é *parametrizar* essa função através de modelos *gerativos* (seção 3.2) ou *discriminativos* (seção 3.3).

3.1 Modelagem da tradução

O modelo de tradução é definido como o conjunto de regras utilizadas por um tradutor automático para transformar a sentença fonte na sentença alvo (LOPEZ, 2008). Os primeiros trabalhos em SMT (BROWN et al., 1990, 1993) utilizam regras baseadas na correspondência de palavras, aprendidas através de sentenças alinhadas lexicalmente. Por exemplo, um dos modelos clássicos, o IBM 4, possui os seguintes parâmetros:

Fertilidade: uma palavra em uma língua pode gerar um número variável de palavras em outra língua. O adjetivo "wooden" no inglês normalmente é traduzido para a locução "de madeira" no português. Nesse caso, a fertilidade da palavra "wooden" seria 2. O modelo define um parâmetro para cada par [palavra, fertilidade] (por exemplo $P_{fert}(2|wooden)$), até um certo valor limite de fertilidade. Esse parâmetro permite realizar a tradução entre sentenças de tamanhos diferentes.

Tradução: no exemplo acima, a fertilidade simplesmente diz que "wooden" gera 2 palavras em português, mas não diz quais. Para isso, o modelo define um parâmetro para cada par [palavra-fonte, palavra-alvo]. No exemplo dado, seriam usados os parâmetros $P_{trad}(de|wooden)$ e $P_{trad}(madeira|wooden)$.

Distorção: finalmente, as palavras traduzidas são permutadas ("distorcidas") entre si. Por exemplo, ao traduzir a expressão "wooden spoon", os parâmetros descritos anteriormente gerariam "de madeira colher". Os parâmetros de distorção são então responsáveis pelo reordenamento das palavras, gerando a expressão "colher de madeira".

Os modelos IBM em geral trouxeram ideias que fazem parte dos sistemas de SMT até hoje. Em especial, o modelo considerado estado-da-arte em SMT é o denominado *baseado em frases*[1] (*Phrase-Based Statistical Machine Translation* - PB-SMT) (KOEHN; OCH; MARCU, 2003; OCH; NEY, 2004). Esse modelo é uma extensão do apresentado acima, onde os parâmetros são definidos em termos de frases ao invés de palavras. Assim, a expressão "wooden spoon" poderia ser considerada uma frase que traduz diretamente em "colher de madeira". Não existem parâmetros de fertilidade pois ela está embutida nas correspondências entre frases.

A maior deficiência dos modelos apresentados está na modelagem da distorção. Dada uma sequência de palavras (ou de frases), calcular a melhor permutação possível é um problema NP-completo (KNIGHT, 1999). Por conta disso, os modelos restringem esses valores de distorção

[1] O termo "frase" nesse caso é definido como uma sequência contígua de palavras, sem necessariamente nenhuma conotação linguística.

até um certo limite de distância a partir da palavra. A questão é que, ainda que valores pequenos consigam realizar reordenamentos *locais*, eles não permitem reordenamentos *de longa distância*, que são comuns especialmente em pares de línguas com ordem de palavras diferentes. Essa limitação motivou o surgimento de modelos de tradução sintáticos, baseados em *gramáticas síncronas*.

3.1.1 Gramáticas Livres de Contexto Síncronas

O processo de tradução automática envolve o reconhecimento de textos em uma língua e a geração de textos em outra língua. No entanto, as gramáticas vistas no capítulo 2 lidam com apenas uma língua. Para possibilitar sua utilização como modelos de tradução, elas são expandidas, gerando as denominadas *gramáticas síncronas*.

Gramáticas Livres de Contexto Síncronas (*Synchronous Context-Free Grammars* - SCFGs) são definidas formalmente como uma 5-upla (N, W_1, W_2, S, R), onde:

- N é um conjunto de símbolos não-terminais,
- W_1 e W_2 são um conjunto de símbolos terminais das linguagens 1 e 2, respectivamente,
- $S \in N$ é o símbolo inicial e
- R é um conjunto de regras de produção

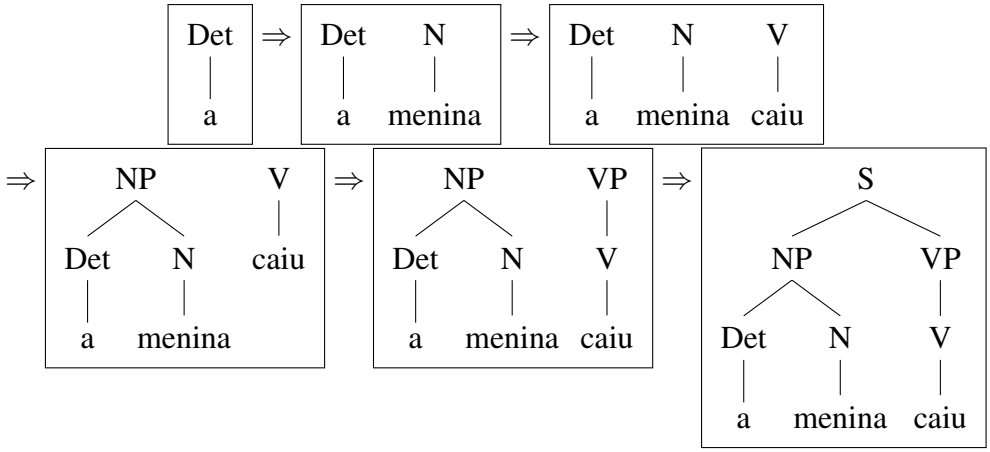
Cada regra de produção em R expande em duas *strings* diferentes, uma para cada língua. Além disso, os símbolos não-terminais de cada *string* são co-indexados. A ideia é realizar a análise sintática da sentença fonte (como se fosse uma CFG) e guardar a sequência de regras utilizada. De posse dessa sequência, ela é aplicada na ordem inversa mas dessa vez expandindo o lado direito correspondente à lingua alvo, obtendo assim a tradução da sentença fonte. A figura 3.1 mostra um exemplo.

Modelos de tradução baseados em SCFGs existem em vários formatos. Os que são baseados em anotação sintática de constituintes como o da figura 3.1 recebem vários nomes como *Syntax-based SMT* (YAMADA; KNIGHT, 2001) ou *Syntax-augmented SMT* (ZOLLMANN; VENUGOPAL, 2006). Outros modelos existentes são os baseados em *Inversion Transduction Grammars* (ITGs) (WU, 1997), que são um subconjunto das SCFGs com propriedades particulares, e a SMT *Hierárquica* (CHIANG, 2007), onde as gramáticas possuem apenas um símbolo não-terminal genérico ao invés de símbolos sintaticamente motivados.

Conjunto de regras:

$r_1 : S \rightarrow \langle NP_1\ VP_2\ ,\ NP_1\ VP_2 \rangle$ | $r_4 : Det \rightarrow \langle a\ ,\ the \rangle$
$r_2 : NP \rightarrow \langle Det_1\ N_2\ ,\ Det_1\ N_2 \rangle$ | $r_5 : N \rightarrow \langle menina\ ,\ girl \rangle$
$r_3 : VP \rightarrow \langle V_1\ ,\ V_1 \rangle$ | $r_6 : V \rightarrow \langle caiu\ ,\ fell \rangle$

Análise da sentença em português:



Sequência de regras utilizadas:

$$r_4 \Rightarrow r_5 \Rightarrow r_6 \Rightarrow r_2 \Rightarrow r_3 \Rightarrow r_1$$

Geração da sentença em inglês:

$$r_1 \Rightarrow r_3 \Rightarrow r_2 \Rightarrow r_6 \Rightarrow r_5 \Rightarrow r_4$$

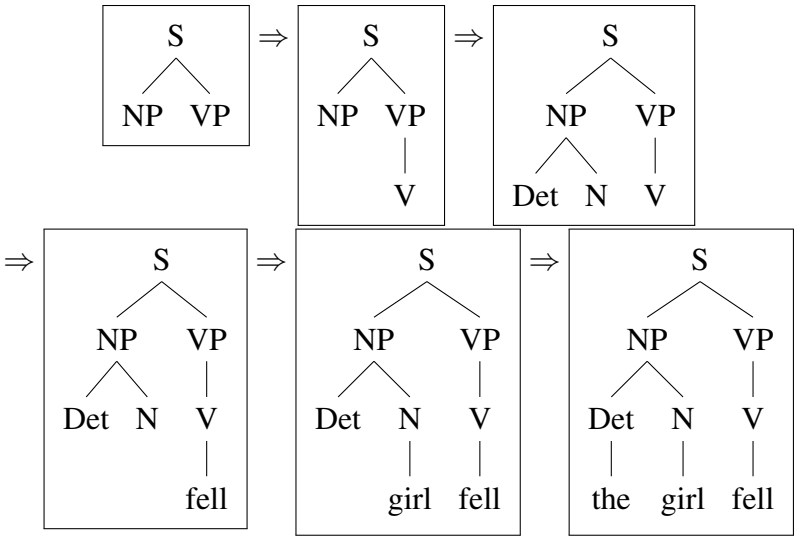


Figura 3.1: Processo de tradução de um sentença em português para o inglês usando uma SCFG

As limitações das CFGs citadas no capítulo 2 ocorrem de forma semelhante para as SCFGs, assim como as soluções: Zollmann e Venugopal (2006), por exemplo, também buscam mitigar o problema da generalização expandindo o conjunto de não-terminais da gramática. Outra possibilidade é expandir o formalismo para uma *gramática de árvores síncrona*.

3.1.2 Gramáticas de Substituição de Árvores Síncronas

Assim como as TSGs possuem um poder representacional maior do que as CFGs, as Gramáticas de Substituição de Árvores Síncronas (*Synchronous Tree Substitution Grammars* - STSGs) também conseguem modelar fenômenos que as SCFGs não conseguem. Uma STSG é definida como uma 5-upla (N, W_1, W_2, S, E), onde:

- N é um conjunto de símbolos não-terminais,
- W_1 e W_2 são conjuntos de símbolos terminais das linguagens 1 e 2, respectivamente,
- $S \in N$ é o símbolo inicial e
- E é um conjunto de regras ou pares de árvores elementares. De forma análoga às SCFGs, os pontos de substituição são co-indexados.

A figura 3.2 mostra um exemplo de STSG para a mesma árvore da figura 3.1. O processo é o mesmo: a sentença em português é analisada e depois a sequência de regras da derivação é aplicada em ordem invertida para gerar a sequência em inglês.

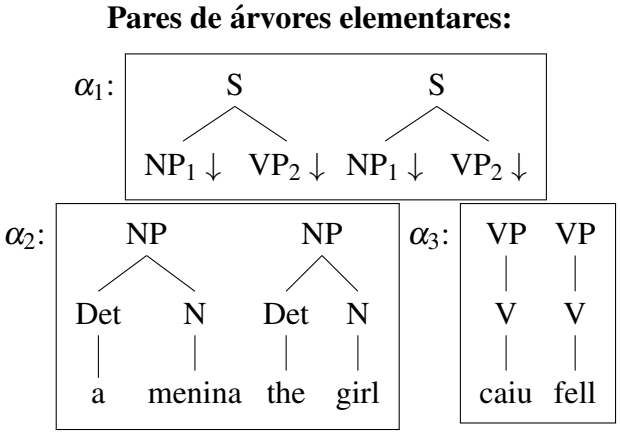


Figura 3.2: Exemplo de uma STSG para o português-inglês

Modelos de tradução baseados em STSGs foram explorados por trabalhos que utilizam informação sintática em ambas as línguas (EISNER, 2003; ZHANG et al., 2007). Essa abordagem é denominada *Árvore-para-Árvore* (*Tree-to-Tree* - TTT). Existem uma série de trabalhos que utilizam informação sintática apenas na língua fonte (LIU; LIU; LIN, 2006) ou na língua alvo (GALLEY et al., 2004, 2006; COHN; BLUNSOM, 2009), abordagens denominadas *Árvore-para-String* (*Tree-to-String* - TTS) e *String-para-Árvore* (*String-to-Tree* - STT), respectivamente. Esses trabalhos utilizam um formalismo semelhante às STSGs chamado *Transdutores Árvore-para-String* (*Tree-to-String* - TTS).

3.1.3 Transdutores Árvore-para-String

Transdutores de árvores[2] são, em geral, equivalentes às gramáticas de árvores síncronas. Entretanto, um tipo especial deles, os transdutores TTS, possuem uma diferença: as regras são definidas como pares [árvore, *string*] (ao invés de [árvore, árvore] como nas STSGs) e existe a noção de *variáveis*. A figura 3.3 mostra um exemplo de transdutor TTS para o mesmo par de sentenças da figura 3.1.

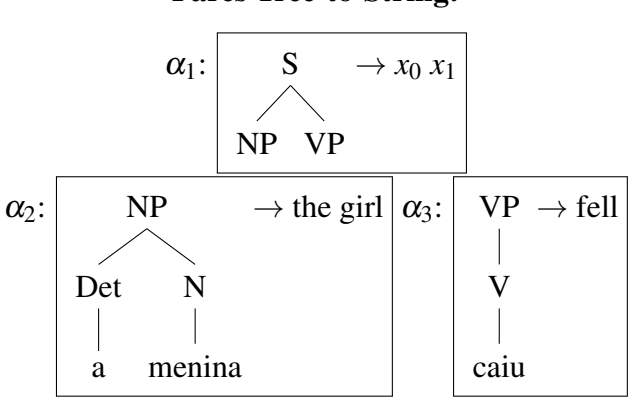


Figura 3.3: Exemplo de um transdutor TTS para o português-inglês. x_0 e x_1 são variáveis relativas aos não-terminais NP e VP, nesse caso. No processo de derivação, eles são substituídos pelas respectivas expansões desses terminais.

Tanto as STSGs quando os transdutores TTS podem ser convertidos para SCFGs se aproximarmos a fórmula de análise pela melhor derivação, através do processo de linearização explicado no capítulo 2. Na figura 3.4 é mostrada a linearização de um transdutor TTS, enquanto a figura 3.5 mostra uma possível sequência de aplicações de regras da SCFG resultante.[3] Por conta dessa linearização, um transdutor TTS pode ser utilizado tanto em um sistema que utilize a abordagem TTS propriamente dita (quando o *corpus* paralelo vem associado às árvores sintáticas das sentenças fonte) quanto em um sistema que use a abordagem STT (quando o *corpus* paralelo vem associado às árvores sintáticas das sentenças alvo).

No processo mostrado na figura 3.5 é possível perceber como o transdutor consegue realizar duas operações sintaticamente motivadas: a troca de ordem entre um substantivo e um sintagma adjetival (regra α_3) e a troca de voz ativa para voz passiva (regra α_1). Enquanto essas operações são modeladas naturalmente por um transdutor TTS, um modelo PB-SMT teria maior dificuldade em realizá-las pois ele depende de parâmetros de distorção que, como visto anteriormente,

[2]Uma explicação generalizada sobre a teoria de transdutores de árvores pode ser encontrada em Comon et al. (1997), enquanto Graehl, Knight e May (2008) discutem suas aplicações em NLP.

[3]O exemplo apresentado nestas figuras é o mesmo introduzido no capítulo 1, no qual há duas inversões: uma localmente dentro do NP (*poemas mais antigos* ⇒ *oldest poems*) e outra globalmente na sentença (entre VP e NP).

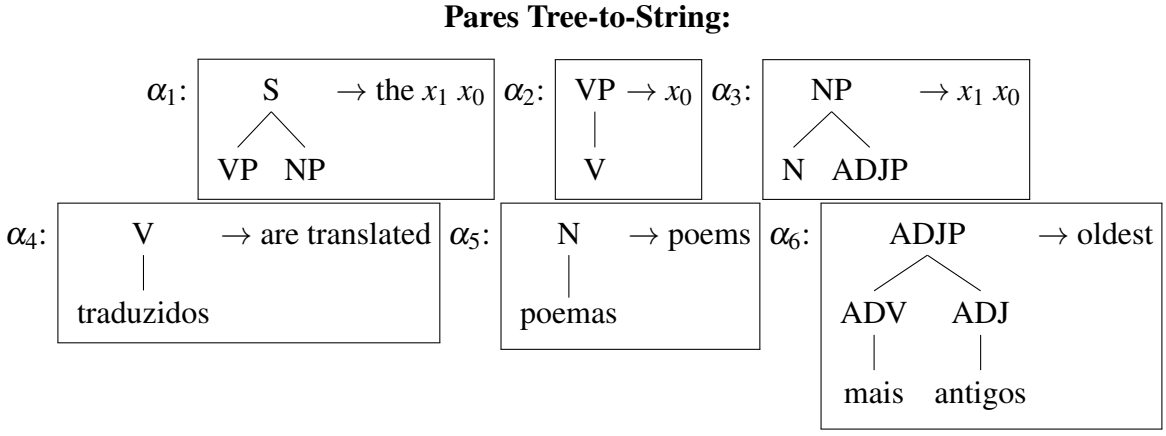


Figura 3.4: Exemplo de linearização de um transdutor TTS

são bastante restritos devido a questões de eficiência computacional[4].

3.2 Modelos gerativos

Definido o formalismo de tradução, o próximo passo é a parametrização do modelo. As primeiras parametrizações utilizadas em SMT usam modelos gerativos (BROWN et al., 1990, 1993). Esses modelos decompõem o termo $P(e|f)$ utilizando o teorema de Bayes, expandindo a equação 3.1 da seguinte forma:

$$\hat{e} = \underset{e}{argmax}\ \frac{P(f|e) \times P(e)}{P(f)} \tag{3.2}$$

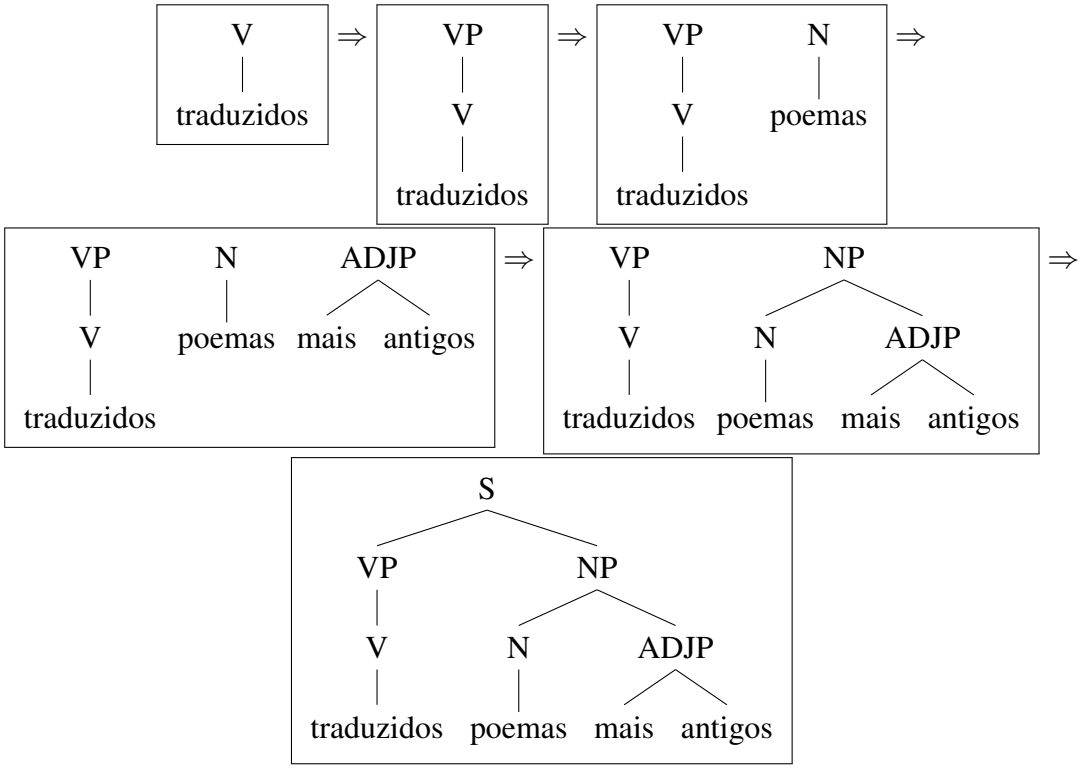
O termo $P(f)$ pode ser excluído pois a sentença fonte é sempre a mesma e estamos interessados apenas na sentença alvo que maximize a equação, sem a necessidade do seu respectivo valor ser normalizado:

$$\hat{e} = \underset{e}{argmax}\ P(f|e) \times P(e) \tag{3.3}$$

A equação acima é semelhante à equação 2.3. No entanto, no contexto da SMT, a verossimilhança $P(f|e)$ corresponde ao *modelo de tradução* e a priori $P(e)$, ao *modelo de língua*. Esse equação é equivalente ao modelo *noisy channel* proposto por Warren Weaver, com origens na teoria da informação (LOPEZ, 2008). Dentro desse modelo, é possível ver

[4]Na prática, se o *corpus* for grande o suficiente, um modelo PB-SMT consegue realizar reordenamentos locais como o mostrado pela regra α_3. Os problemas de eficiência surgem quando se tenta modelar os reordenamentos de longa distância.

Análise da sentença em português:



Geração da sentença em inglês:

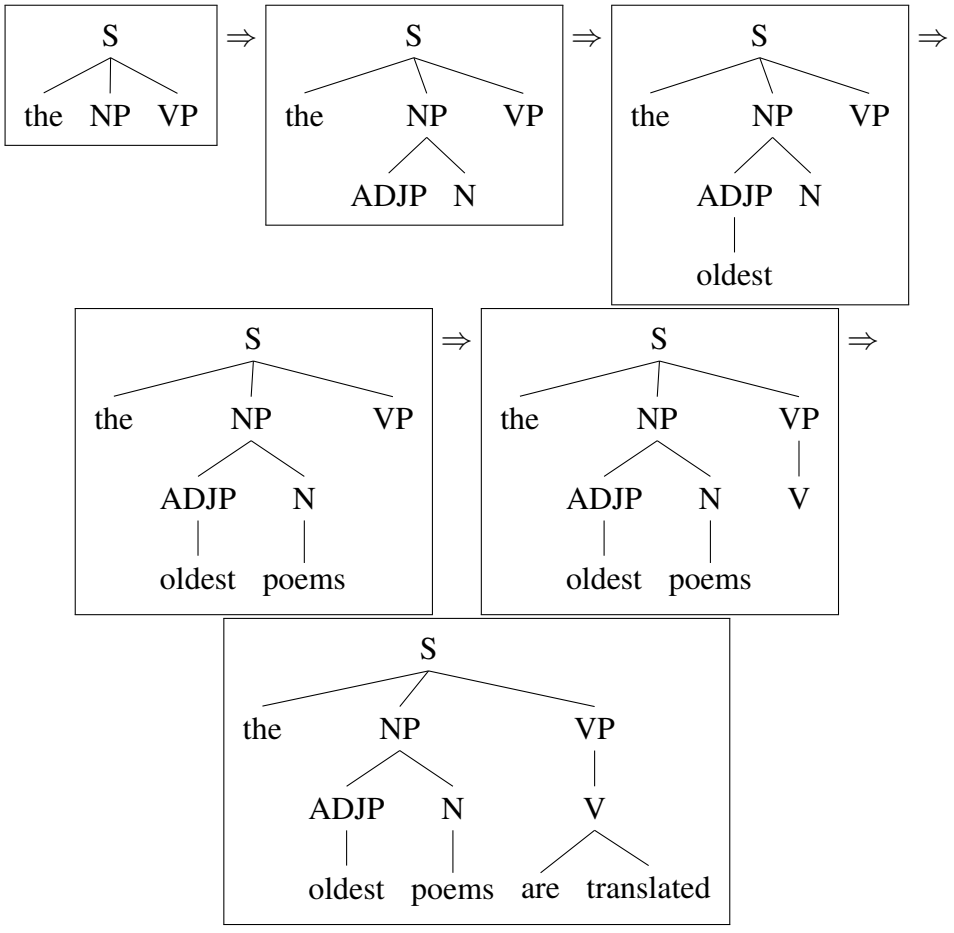


Figura 3.5: Sequência de aplicações de regras para o transdutor TTS da figura 3.4

o processo de tradução como uma história gerativa, onde o modelo de língua $P(e)$ gera estocasticamente novas sentenças e o modelo de tradução $P(f|e)$ "corrompe" essas sentenças, transformando-as em uma outra língua, como mostra a figura 3.6.

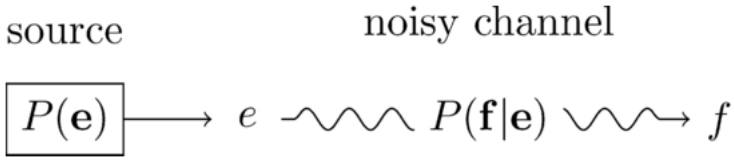


Figura 3.6: Modelo *noisy-channel* de Weaver (adaptado de Lopez (2008))

A grande vantagem de se separar o termo $P(e|f)$ em dois é que se torna possível aplicar dois submodelos independentes para a desambiguação da sentença alvo e. Além disso, o treinamento de cada um desses submodelos pode ser feito separadamente, inclusive utilizando *corpora* diferentes. O submodelo de tradução corresponde ao formalismo definido na primeira etapa enquanto o submodelo de língua é definido separadamente. A modelagem de língua é um tópico por si só em PLN, tendo diversas aplicações, mas em SMT ela se tornou uma parte essencial de qualquer sistema.

3.2.1 Modelagem de língua

Um modelo de língua pode ser definido como um processo estocástico que gera novas palavras em uma sentença de acordo com as palavras anteriores (MANNING; SCHüTZE, 2000, p.191). Sua aplicação vai muito além da SMT: os primeiros modelos de língua foram elaborados com o intuito de desambiguar *emissões* (*utterances*) em reconhecimento de fala. De fato, os modelos utilizados em SMT são normalmente "emprestados" dessa área.

Formalmente, o termo $P(e)$ é definido como:

$$P(e) = \prod_{i=1}^{I} P(e_i|e_1^{i-1}) \tag{3.4}$$

onde I é o tamanho da sentença e em número de *tokens*, e_i é o *token* localizado na posição i e e_1^{i-1} são todos os *tokens* anteriores a ele.

Realizar o cálculo da equação acima é computacionalmente intratável pois o seu lado esquerdo expande em um número exponencial de termos. Por conta disso, os modelos de língua assumem determinadas *suposições de independência* que permitam o cálculo da equação em um tempo tratável.

Os modelos mais utilizados em SMT e considerados estado-da-arte são os que assumem que o *token* e_i é independente de todos os outros com exceção dos $n-1$ anteriores: são os

modelos baseados em *n-gramas*. Com essa suposição, a fórmula 3.4 pode ser reescrita como:

$$P(e) = \prod_{i=1}^{I} P(e_i | e_{i-n}^{i-1}) \tag{3.5}$$

Por exemplo, em um modelo de trigramas, a sentença "os meninos da escola municipal foram ao jogo" teria sua probabilidade calculada da seguinte forma:

$$\begin{aligned} P(e) = {} & P(os|START_2, START_1) \times \\ & P(meninos|START_1, os) \times \\ & P(da|os, meninos) \times \\ & P(escola|meninos, da) \times \\ & P(municipal|da, escola) \times \\ & P(foram|escola, municipal) \times \\ & P(ao|municipal, foram) \times \\ & P(jogo|foram, ao) \times \\ & P(END|ao, jogo) \end{aligned}$$

onde $START_2$, $START_1$ e END são *tokens* artificiais inseridos para marcar início e fim de sentença.

Note que o sexto termo da fórmula acima ($P(foram|escola, municipal)$) possui, localmente, um erro de concordância. Teoricamente, por conta desse erro, esse termo terá um valor baixo pois em um *corpus*, o *token* "foram" na maior parte das vezes virá seguido de substantivos ou sintagmas nominais que estejam no plural[5]. Consequentemente, a probabilidade total da sentença será prejudicada, ainda que esteja gramaticalmente correta. Por conta dessas limitações, modelos que incorporem sintaxe foram e são investigados até hoje.

Gramáticas como modelos de língua

Uma forma de incorporar sintaxe dentro de um modelo de língua é através da utilização de gramáticas, tais como as vistas no capítulo 2. Dada uma gramática, a maneira mais imediata de obter uma probabilidade para uma sentença é utilizando um analisador sintático e calculando a probabilidade de sua árvore sintática mais provável. Nesse caso, a suposição de independência assumida é que cada *token* e_i é dependente apenas das suas respectivas regras na gramática.

Um detalhe importante quando se utiliza um analisador sintático como modelo de língua é

[5] Considerando-se que o *corpus* possui poucos ou nenhum erro gramatical.

que não há interesse em recuperar a árvore sintática da sentença, apenas a sua probabilidade. No caso da análise utilizando TSGs e o método da melhor derivação, esse detalhe torna desnecessária a criação dos não-terminais, apenas a planificação é feita (POST; GILDEA, 2009b). A figura 3.7 mostra um exemplo, aplicado para a mesma situação da figura 2.5.

TSG:	**CFG:**
$S(NP\,VP)$	$S \rightarrow NP\,VP$
$NP(Det(a)\;N(menina))$	$NP \rightarrow a\;menina$
$VP(V(caiu)\;PP(P(da)\;N))$	$VP \rightarrow caiu\;da\;N$
$N(bicicleta)$	$N \rightarrow bicicleta$

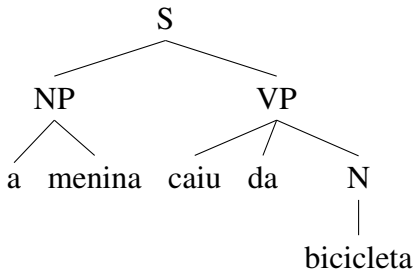


Figura 3.7: Exemplo de derivação de uma TSG desconsiderando a criação de novos não-terminais. Ainda que não seja possível recuperar a derivação original, a probabilidade da árvore permanece a mesma.

Durante muito tempo, as tentativas de incorporar um modelo de língua sintático a um sistema de SMT não obtiveram sucesso (CHARNIAK; KNIGHT; YAMADA, 2003; OCH et al., 2004). Entretanto, recentemente houve avanços significativos: Post (2010) utiliza TSGs obtidas através de amostragem de Gibbs e obtém resultados similares aos modelos de ngramas dentro de um sistema de SMT. Schwartz et al. (2011) utiliza analisadores incrementais incorporados ao algoritmo de decodificação com resultados até melhores quando utilizado em conjunto com o modelo de ngramas.

3.3 Modelos discriminativos

Modelos gerativos são uma forma elegante de se modelar o processo de tradução e correspondem fortemente aos modelos utilizados no espaço de busca da melhor tradução, que é o método utilizado pelos algoritmos de decodificação (LOPEZ, 2008). No entanto, como visto anteriormente, esses modelos precisam assumir condições de independência que na prática sabe-se que não são válidas. É possível relaxar essas condições utilizando um modelo *discriminativo*, que em SMT assume a forma de modelos *log-linear*.

Os modelos *log-linear* dividem o termo $P(e|f)$ em *K funções de feature* $h(e,f)$. Essas funções podem representar qualquer informação a respeito da sentença fonte, alvo ou ambas.

Exemplos de *features* são o tamanho da sentença alvo ou o número de vezes que um certo par de palavras aparece nas duas sentenças. A fórmula completa do modelo *log-linear* é mostrada abaixo:

$$P(e|f) = \frac{exp \sum_{k=1}^{K} \lambda_k h_k(e,f)}{\sum_{e'} exp \sum_{k=1}^{K} \lambda_k h_k(e',f)} \tag{3.6}$$

Substituindo em 3.1, temos:

$$\hat{e} = \underset{e}{argmax} \ \frac{exp \sum_{k=1}^{K} \lambda_k h_k(e,f)}{\sum_{e'} exp \sum_{k=1}^{K} \lambda_k h_k(e',f)} \tag{3.7}$$

Assim como no modelo gerativo, o denominador serve apenas como fator de normalização e pode ser descartado no momento que se busca a melhor tradução:

$$\hat{e} = \underset{e}{argmax} \ exp \sum_{k=1}^{K} \lambda_k h_k(e,f) \tag{3.8}$$

Os coeficientes λ_k definem o peso de cada função: um valor positivo indica correlação direta com o valor de $P(e|f)$, um valor negativo indica correlação inversa e um valor próximo de zero é um indício de que aquela *feature* contribui muito pouco para a obtenção da melhor tradução.

Em grande parte, os trabalhos que utilizam o modelo *log-linear* não são completamente discriminativos. Isso porque dentre as *features* utilizadas estão os modelos de tradução e língua comentados anteriormente, que são parametrizados por si só. Na verdade, o modelo gerativo apresentado na seção 3.2 pode ser visto como um caso especial da equação 3.8 com as seguintes condições:

- $K = 2$
- $\lambda_1 = \lambda_2 = 1$
- $h_1(e,f) = log\ P(f|e)$
- $h_2(e,f) = log\ P(e)$

O que esses trabalhos fazem é, a partir da formalização descrita acima, adicionar novas funções de *feature*. Koehn, Och e Marcu (2003) e Och e Ney (2004) descrevem algumas dessas *features* que se mostraram úteis na desambiguação de candidatos, enquanto Liu, Liu e Lin (2006) e Zhang et al. (2007) adaptam-nas para os modelos TTS e TTT, respectivamente:

Modelo de tradução "inverso": além de utilizar o termo $log\ P(f|e)$ (equivalente ao modelo de tradução gerativo) como *feature*, também é utilizado o termo $log\ P(e|f)$. Ainda que não exista uma motivação formal para o seu uso, estudos empíricos mostram que a performance melhora ao se utilizar as duas direções do modelo de tradução como *features*.

Peso léxico: representa o quão bem cada palavra da sentença fonte traduz na sua respectiva palavra no candidato. A ideia é que em uma boa tradução, cada palavra será a melhor tradução possível da respectiva palavra na sentença fonte. O peso léxico para cada par de palavras é calculado da seguinte forma:

$$w(f|e) = \frac{count(f,e)}{\sum_{f'} count(f',e)} \tag{3.9}$$

O valor da função é então o produto de todos os pesos léxicos dos pares de palavras que foram traduzidos. Os valores de $count(f,e)$ são normalmente obtidos a partir da informação de alinhamento lexical no *corpus* de treinamento.

Tamanho do candidato: devido à utilização dos modelos de tradução e língua como funções, candidatos mais curtos tendem a ter uma probabilidade maior (por conterem menos regras). Essa *feature* tem como objetivo compensar esse fato, penalizando candidatos muito curtos.

Total de regras utilizadas: também com o objetivo de compensar os valores gerados pelos modelos de tradução e língua, o total de regras utilizadas para chegar ao candidato também é definido como uma *feature*.

O grande poder dos modelos discriminativos está na flexibilidade de se adicionar ou remover *features*. Och et al. (2004) apresenta experimentos avaliando uma série delas, como por exemplo a utilização de léxicos externos, modelos de língua baseados em informação de POS, analisadores sintáticos e outras. Mais recentemente, Post (2010) divide um modelo de língua sintático baseado em uma TSG em várias *features* "menores", levando em conta características de cada regra utilizada como tamanho, número de terminais, altura da árvore, entre outras. Em geral, nota-se uma tendência em SMT de se utilizar modelos discriminativos cada vez mais refinados: Chiang, Knight e Wang (2009) apresenta um modelo de tradução hierárquica que usa mais de 10.000 *features*.

4 MÉTODOS DE APRENDIZADO

Seguindo a descrição de Lopez (2008), após a definição e parametrização dos modelos utilizados em um sistema de SMT, deve-se então estimar os parâmetros definidos. É nessa etapa que entra o *corpus* paralelo alinhado lexicalmente, conforme explicado na seção 1.1. Outros recursos linguísticos também podem fazer parte da etapa de treinamento caso sejam necessários para o modelo, como por exemplo um *corpus* monolíngue para modelagem de língua.

Como visto anteriormente, a maior parte dos sistemas de SMT utilizam um modelo discriminativo com submodelos gerativos sendo definidos como *features*. Por conta dessa abordagem híbrida, dois conjuntos de parâmetros devem ser estimados separadamente:

- A maior fração do *corpus* paralelo é utilizada para estimar os parâmetros do modelo de tradução. Essa fração é denominada *corpus de treinamento*. O modelo de língua pode ser estimado pelo mesmo *corpus* usando somente a parte relativa à língua alvo ou ainda usando um outro *corpus*.
- Os parâmetros do modelo discriminativo (coeficientes λ das funções de *feature*) são estimados utilizando uma outra fração do *corpus* paralelo, chamado de *corpus de validação*.

Assim, o pipeline de aprendizado de um modelo de SMT passa inicialmente pelo alinhamento lexical do *corpus* de treinamento. Caso seja um modelo baseado em sintaxe, esse *corpus* também deve vir associado às árvores sintáticas das sentenças fonte e/ou alvo, dependendo da abordagem utilizada (TTS, STT ou TTT). Esse *corpus* é então utilizado para treinar o modelo de tradução e, opcionalmente, o modelo de língua e outras *features*. Após esse treinamento, os parâmetros das funções de *feature* são estimados em uma nova etapa de treinamento usando o *corpus* de validação.

Neste capítulo o foco será nos algoritmos para estimar modelos de tradução baseados em transdutores TTS e modelos discriminativos em geral. O aprendizado de um modelo de língua sintático é feito de forma análoga ao aprendizado de gramáticas explicado na seção 2.3.

4.1 Estimando modelos de tradução

A extração de regras de tradução baseada em transdutores TTS utiliza duas sentenças alinhadas e a árvore sintática de uma delas. Ao contrário de uma TSG, onde basta obter todas as subárvores para extrair as regras possíveis, essa extração não é trivial. Isso porque nem toda subárvore e seu respectivo alinhamento forma uma regra TTS válida. Sendo assim, foi desenvolvido um algoritmo que consegue extrair as regras mínimas para um dado par de sentenças: o GHKM.

4.1.1 GHKM

Proposto inicialmente por Galley et al. (2004), o GHKM é um algoritmo que tem como entrada um *grafo de alinhamento*, que corresponde a um par de sentenças alinhadas lexicalmente junto com a árvore sintática de uma delas (mostrado na figura 4.1). As arestas desse grafo são direcionadas: elas partem da raiz da árvore sintática e vão até os itens lexicais da sentença alinhada.

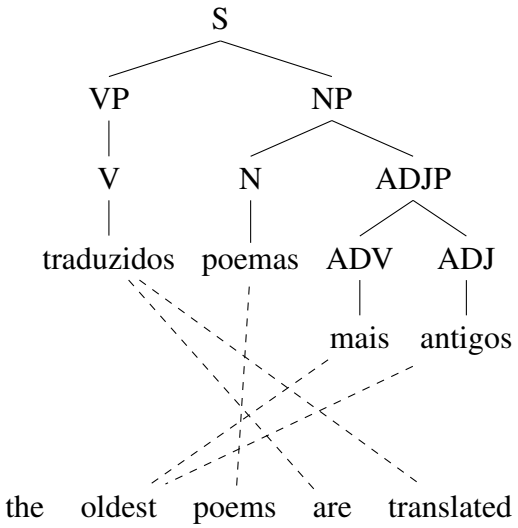


Figura 4.1: Exemplo de grafo de alinhamento

Para determinar esse conjunto de regras, o GHKM procede da seguinte forma:

1) Primeiramente, cada nodo é anotado com seu *span*. O *span* é definido como o conjunto de itens lexicais da sentença alinhada que são alcançáveis a partir do nodo em questão (lembrando que as arestas são direcionadas). Por exemplo, o nodo VP na figura 4.1 tem como *span*: $\{are, translated\}$. Nodos que não estão alinhados (caso de "the" na figura 4.1) são considerados parte apenas dos *spans* de si mesmo e da raiz.

2) Cada nodo do grafo também é anotado com seu *span complementar*. O *span* complementar é definido como a união do *span* complementar do nodo pai e dos *spans* de cada nodo irmão. Por exemplo, o nodo VP na figura 4.1 tem como *span complementar*: {*the*, *oldest*, *poems*}. O *span* complementar do nodo raiz é definido como vazio.

3) Os nodos também são marcados com os *fechos* de seus respectivos *spans*. O fecho é definido como a sequência contígua mínima de itens lexicais que contenha o *span*. Por exemplo, considerando o grafo da figura 4.1, se um *span* é definido como $\{oldest, translated\}$, o seu fecho é $\{oldest, poems, are, translated\}$.

4) Calcula-se o *conjunto de fronteira*: se um nodo n for um símbolo não-terminal, ele faz parte desse conjunto se e somente se $span\_complementar(n) \cap fecho(span(n)) = \emptyset$. Por exemplo, o nodo VP na figura 4.1 é fronteira uma vez que seu *span complementar* ({*the*, *oldest*, *poems*}) não tem nenhuma intersecção com seu fecho ({*are*, *translated*}). Caso seja um terminal, ele fará parte do conjunto somente se for um dos itens lexicais da sentença alvo.

5) Com o grafo anotado com seus respectivos nodos pertencentes ao conjunto de fronteira, os *grafos de fronteira mínimos* são extraídos. Um grafo de fronteira mínimo é qualquer subgrafo do grafo de alinhamento que não seja trivial (ou seja, composto por mais de um nodo) que tenha a seguinte propriedade: todos os seus nodos devem possuir arestas chegando e partindo dele *a menos que* faça parte do conjunto de fronteira.

6) Finalmente, cada grafo de fronteira mínimo é transformado em uma regra TTS. Nessa regra, a árvore corresponde à própria árvore sintática contida dentro do grafo de fronteira, enquanto a *string* é composta pelos símbolos presentes nas "folhas" do grafo de fronteira, com não-terminais sendo substituídos por variáveis. A ordem em que esses símbolos aparecem na regra deve obedecer à seguinte restrição: os *spans* de cada símbolo concatenados devem ser igual ao *span* da raiz do grafo.

A figura 4.2 mostra os passos descritos acima para o grafo da figura 4.1 enquanto a figura 4.3 mostra as regras TTS obtidas a partir desse grafo (que nesse caso são as mesmas da figura 3.4. Note que os itens lexicais da sentença fonte não possuem *spans* definidos. Isso porque não existem regras TTS que possuem apenas um símbolo terminal do lado esquerdo, assim esses nodos não são considerados na processo de anotação dos *spans*.

O grafo de alinhamento marcado pelos nodos de fronteira constitui em uma derivação possível em termos de regras TTS (mais especificamente, a derivação *mínima*). Se forem consideradas somente as derivações mínimas para cada grafo, é possível extrair um conjunto de

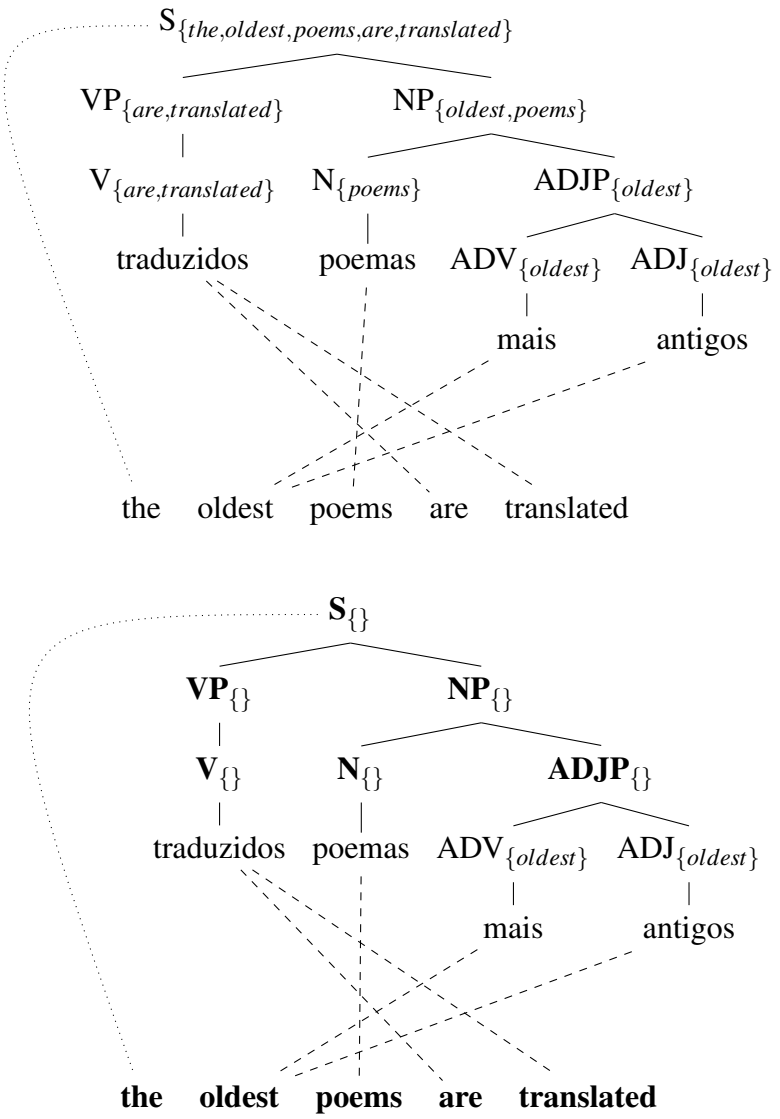


Figura 4.2: Aplicação do GHKM ao grafo da figura 4.1. No primeiro grafo, cada nodo está anotado com seu respectivo *span*. A palavra não alinhada "the" é artificialmente alinhada com o símbolo raiz (representado pela linha pontilhada). No segundo grafo, os nodos estão marcados com o resultado da operação $span\_complementar(n) \cap fecho(span(n))$. Os nodos em negrito são os que fazem parte do conjunto de fronteira.

regras do *corpus* através do MLE, usando contagem de frequências. Além disso, como nesse caso o MLE é aproximado pelo método da melhor derivação, é possível transformar o transdutor TTS resultante em uma SCFG, através da linearização mostrada na figura 3.4. Como no processo de tradução não existe interesse em recuperar a estrutura sintática das sentenças, não são criados novos não-terminais mas isso poderia ser feito caso houvesse esse interesse.

Conforme visto na seção 3.1.3, os transdutores gerados pelo GHKM podem ser utilizados tanto em sistemas TTS quanto STT. Assim, dado um *corpus* de entrada é possível utilizar o algoritmo para gerar modelos de tradução para ambas as direções do respectivo par de línguas

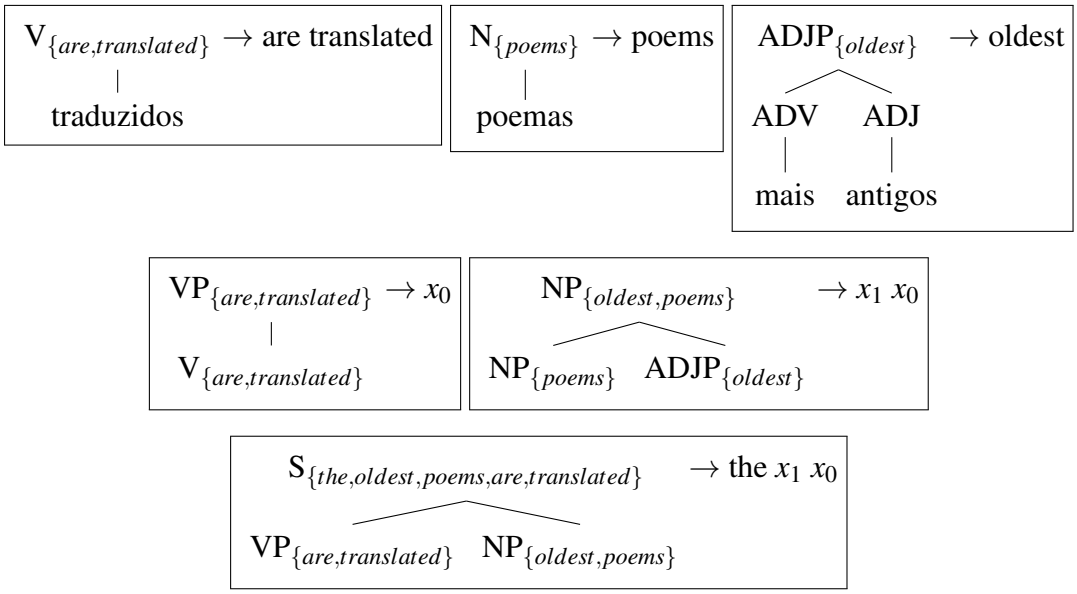


Figura 4.3: Regras TTS obtidas a partir do grafo da figura 4.1. Os *spans* de cada não-terminal estão marcados apenas para clarificação, não fazendo parte dos símbolos em si.

(sendo uma delas usando a abordagem TTS e a outra, a abordagem STT).

4.1.2 Processo Dirichlet e amostragem de Gibbs para transdutores TTS

O GHKM possui como saída as regras *mínimas* de um grafo de alinhamento. A ideia é pressupor que cada grafo foi gerado apenas por regras desse tipo e, posteriormente, calcular as respectivas probabilidades através de contagem de frequências. No entanto, nada impede que um grafo seja gerado por um conjunto de regras *maiores*, que não sejam necessariamente mínimas.

Fazendo um paralelo com o problema do aprendizado de gramáticas monolíngue apresentado na seção 2.3, o GHKM seria equivalente a extrair uma CFG de um *treebank* (ou, de forma equivalente, uma TSG que possui somente regras mínimas, de altura igual a 1). Assim, essas regras mínimas, mesmo sendo regras TTS, tendem a ter problemas semelhantes às regras de uma CFG no sentido de capturarem pouco contexto das sentenças, ainda que num nível menor.

Uma forma de atacar esse problema é aprimorando o GHKM para que ele gere regras maiores, que capturem mais contexto. No entanto, essa ideia passa pelo mesmo problema citado na seção 2.3: o número de regras TTS possíveis de ser extraídas de um grafo de alinhamento é exponencial, exigindo a adoção de heurísticas que limitem o seu tamanho.

Galley et al. (2006) expande o GHKM utilizando o *Expectation-Maximization*: após extrair as regras mínimas, um subconjunto delas contendo somente n terminais do lado esquerdo é combinado entre si, adicionado regras maiores ao transdutor. Com esse conjunto obtido, o

EM é aplicado ao *corpus* para obter as respectivas probabilidades. Os experimentos descritos utilizam valores de n igual a 3 e 4. Entretanto, ainda que a restrição imposta no tamanho das regras permita a realização do EM em um tempo tratável, ele não ataca o problema do *overfitting*. Nesse caso, o que acontece é que o EM tende a dar maior probabilidade para regras maiores, degenerando o transdutor (da mesma forma que acontece com as TSGs em aprendizado monolíngue).

Considerando que os problemas de aprendizado de transdutores TTS são bastante semelhantes aos de aprendizado de TSGs, Cohn e Blunsom (2009) incorporam a combinação do Processo Dirichlet com a amostragem de Gibbs ao GHKM. A ideia é inicialmente anotar os *spans* de todos os nodos com o GHKM e então considerá-los como as variáveis do modelo. Ao visitar um nodo, o processo de amostragem detecta todos os *spans* possíveis daquele nodo e escolhe um deles, podendo ser o mesmo de antes. A escolha é feita proporcionalmente às respectivas posterioris resultantes de cada possibilidade de *span* daquele nodo.

Para detectar o conjunto de possíveis *spans* de um determinado nodo, quatro regras são consideradas:

1) O *span* do nodo raiz deve conter toda a sentença alvo. Ou seja, na prática ele nunca é visitado pelo amostrador.

2) O *span* de um nodo deve estar contido no *span* do nodo pai.

3) O *span* de um nodo deve conter todos os *spans* dos seus respectivos filhos.

4) Os fechos dos *spans* de dois nodos irmãos não podem se sobrepor (ou seja, eles devem ser disjuntos).

Note que no grafo da figura 4.2 os nodos ADV e ADJ violam a regra 4 descrita acima. Isso acontece porque o método original considera que cada item lexical da sentença alvo está alinhado somente com um item lexical da sentença fonte. Por conta dessa restrição, o método foi adaptado para os experimentos realizados neste trabalho: ao invés de visitar todos os nodos, o amostrador visita apenas nodos de fronteira. Assim, não há risco do amostrador visitar um nodo que viole a regra 4.

A figura 4.4 mostra um exemplo em que o amostrador está visitando o nodo NP e avaliando seus possíveis *spans*. Cada um deles define duas regras TTS diferentes, mostradas na figura 4.5. O amostrador então define qual span será selecionado de acordo com suas respectivas probabilidades, dadas nas equações abaixo:

$$\begin{aligned} P(span = span_1) &= P(r_{11}) \times P(r_{12}) \\ P(span = span_2) &= P(r_{21} \times P(r_{22}) \end{aligned} \tag{4.1}$$

onde:

- $span_1$ e $span_2$ são os dois *spans* possíveis para o nodo, mostrados na figura 4.4.
- r_{11} e r_{12} são as duas regras implicadas pelo $span_1$, mostradas na figura 4.5.
- r_{21} e r_{22} são as duas regras implicadas pelo $span_2$, também mostradas na figura 4.5.

Existe também a possibilidade do amostrador trocar o *span* de nodo para o conjunto vazio, caso as regras permitam. Nesse caso, o *span* vazio define apenas uma regra, como mostra a figura 4.6, e a probabilidade desse *span* vazio passa a ser simplesmente a probabilidade da regra.

As probabilidades das regras são calculadas confome a fórmula 2.10, reproduzida abaixo:

$$P(e_i|\vec{e}_{<i}, r_i, \alpha_{r_i}, P_0) = \frac{count(e_i) + \alpha_{r_i} P_0(e_i|r_i)}{count(r_i) + \alpha_{r_i}}$$

A única diferença em relação à sua utilização para aprendizado de TSGs é que e_i representa agora uma regra TTS ao invés de uma árvore elementar. O restante dos termos possui significado análogo.

Em relação à probabilidade base (P_0) de uma regra TTS, Cohn e Blunsom (2009) a define como o produto da probabilidade do seu lado direito (árvore) pela probabilidade do seu lado esquerdo (*string*). Por sua vez, elas são definidas da seguinte forma:

- Probabilidade da árvore:
  - 1) Se o nodo não for um preterminal (etiqueta POS), expande a nodo em n filhos, sendo que n é amostrado de uma distribuição geométrica com parâmetro β_{child}. Caso o nodo seja um pré-terminal ele expande deterministicamente para 1 filho.
  - 2) Escolhe o símbolo gerado de acordo com uma distribuição uniforme relativa aos conjuntos N de não-terminais e T de terminais.
  - 3) Para cada não-terminal gerado, realiza uma escolha entre expandí-lo ou não, de acordo um um parâmetro β_{expand}. Para cada símbolo expandido, retorna ao passo 1.

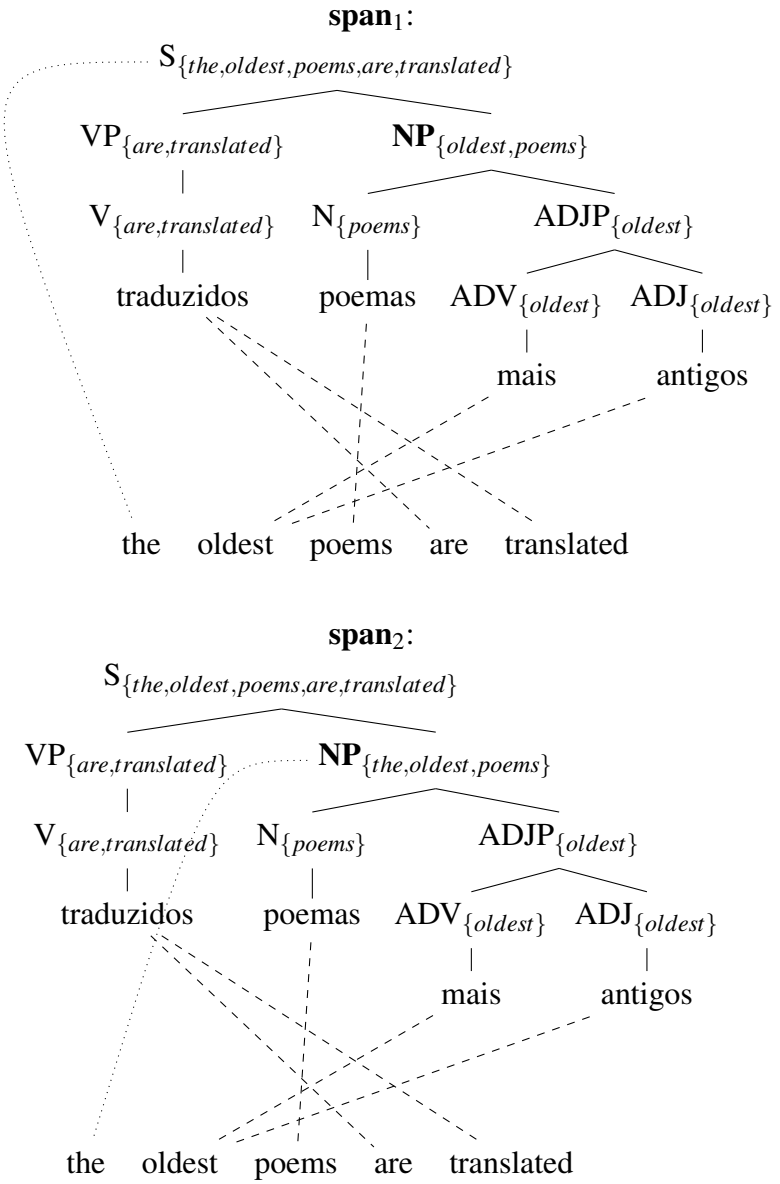


Figura 4.4: Exemplo de possíveis *spans* para um nodo visitado durante o processo de amostragem de Gibbs, no caso o nodo NP. Note que o segundo caso implica um alinhamento entre o nodo NP e o nodo "the", que antes estava alinhado com o nodo raiz, conforme saída do algoritmo GHKM.

- Probabilidade da *string*:
  - 1) Gera m símbolos terminais, onde m é amostrado de uma distribuição geométrica com parâmetro β_{term}.
  - 2) Gera cada terminal de acordo com uma distribuição uniforme sobre T', conjunto de terminais da língua alvo.
  - 3) Gera cada variável existente na string, posicionando uma de cada vez de acordo com uma distribuição uniforme sobre as posições possíveis

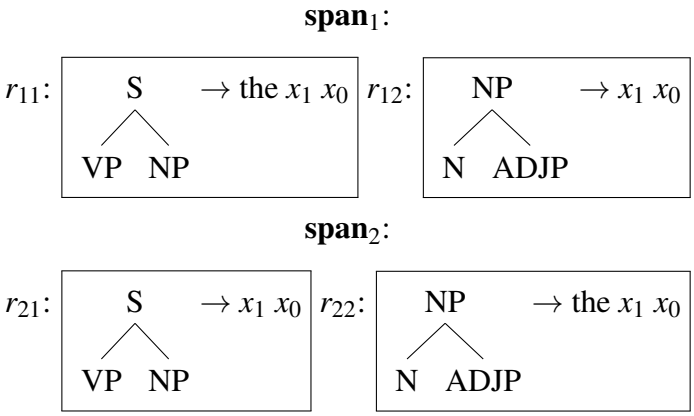


Figura 4.5: Regras implicadas pelos *spans* da figura 4.4

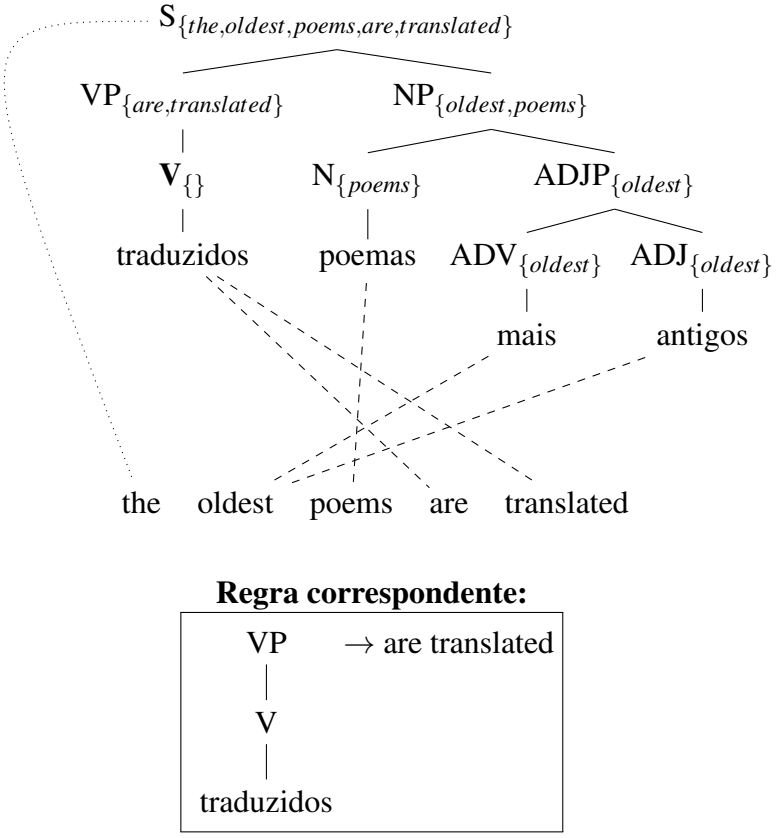
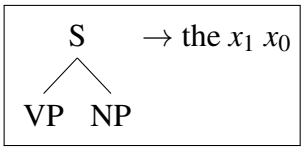


Figura 4.6: Exemplo de *span* vazio e sua regra correspondente. No caso, o nodo analisado é o nodo V. Note que, segundo as regras do amostrador, ele possui 4 *spans* possíveis: vazio, {are}, {translated} e {are,translated}.

Assim como no aprendizado de TSGs, a probabilidade base é definida dessa forma para que regras muito grandes tenham probabilidades menores, prevenindo o *overfitting*. A figura 4.7 mostra um exemplo de como esse cálculo é feito.

Regra: 

Parâmetros:
$\beta_{child} = 0.5$, $\beta_{expand} = 0.5$, $\beta_{term} = 0.5$, $N = 5$, $T' = 10$

Cálculo:
$P(árvore) = P_{geom}(2|\beta_{child}) \times \frac{1}{N} \times (1-\beta_{expand}) \times \frac{1}{N} \times (1-\beta_{expand})$
$P(árvore) = 0.25 \times 0.2 \times 0.5 \times 0.2 \times 0.5 = 0.0025$
$P(string) = P_{geom}(1|\beta_{term}) \times \frac{1}{T'} \times \frac{1}{2} \times \frac{1}{3}$
$P(string) = 0.5 \times 0.1 \times 0.5 \times 0.333 = 0.08325$
$P(regra) = P(árvore) \times P(string) = 2.08125e^{-05}$

Figura 4.7: Exemplo de cálculo da probabilidade base de uma regra TTS

4.2 Estimando parâmetros de funções de feature

Em SMT, os métodos de estimativa de parâmetros em modelos discriminativos são em sua grande maioria baseados na minimização de uma métrica de erro. Formalmente, define-se uma função de erro $E(\hat{e}, e)$ que diz o total de erro entre um candidato $\hat{e}$ e uma tradução de referência e. Assim, os parâmetros λ_1^K são ajustados através de um *corpus* de validação utilizando a seguinte equação:

$$\lambda_1^K = \underset{\hat{\lambda}_1^K}{argmin} \sum_{(\mathbf{e},\mathbf{f}) \in C} E(\underset{\hat{\mathbf{e}}}{argmax}\, P_{\hat{\lambda}_1^K}(\hat{\mathbf{e}}|\mathbf{f}), \mathbf{e}) \tag{4.2}$$

onde:

- O termo $\hat{\lambda}_1^K$ é um conjunto de parâmetros com quaisquer valores.
- O termo $\underset{\hat{\mathbf{e}}}{argmax}\, P_{\hat{\lambda}_1^K}(\hat{\mathbf{e}}|\mathbf{f})$ é a tradução devolvida pelo sistema considerando o conjunto de parâmetros $\hat{\lambda}_1^K$. Como estamos lidando com um modelo *log-linear*, ela é equivalente à função 3.7.
- Os termos **e** e **f** são as sentenças alvo e fonte do *corpus* de validação.

Definir a função $E(\hat{e}, e)$ faz parte do problema de como avaliar automaticamente uma tradução. Essa é uma área em constante pesquisa e várias métricas foram propostas com o passar do tempo. Algumas dessas métricas são mostradas na seção 5.2. No entanto, é importante salientar que os algoritmos de treinamento (apresentados nas subseções a seguir) são independentes da métrica considerada.

4.2.1 Minimum Error-Rate Training

O algoritmo de *Minimum Error-Rate Training* (MERT) (OCH, 2003) foi um dos primeiros propostos para treinamento discriminativo e é utilizado até hoje em sistemas de SMT. A ideia do MERT é minimizar cada parâmetro λ_k enquanto mantém os outros parâmetros constantes. Com isso, a probabilidade de cada tradução é calculada com a seguinte fórmula:

$$P(\mathbf{e}|\mathbf{f}) = \lambda_k h_k(\mathbf{e},\mathbf{f}) + (\sum_{k' \neq k} \lambda_{k'} h_{k'}(\mathbf{e}|\mathbf{f})) \tag{4.3}$$

Pela definição do algoritmo, o segundo termo na fórmula 4.3 é constante, o que torna a função linear em λ_k. Dessa forma, cada candidato pode ser representada por uma linha que relaciona $P(\mathbf{e}|\mathbf{f})$ com λ_k.

O algoritmo é representado na figura 4.8. Para realizar a minimização do parâmetro, vários candidatos são gerados para todas as sentenças do *corpus*. Em cada sentença, são computadas as intersecções das linhas correspondentes às suas traduções, definindo os chamados *pontos críticos* (gráfico 1 da figura 4.8). Cada ponto crítico define um valor para λ_k onde há uma mudança sobre qual a melhor tradução para a sentença em questão, o que define *intervalos* de λ_k. Finalmente, para cada intervalo é calculado o valor da função de erro, resultando no gráfico 2 da figura 4.8.

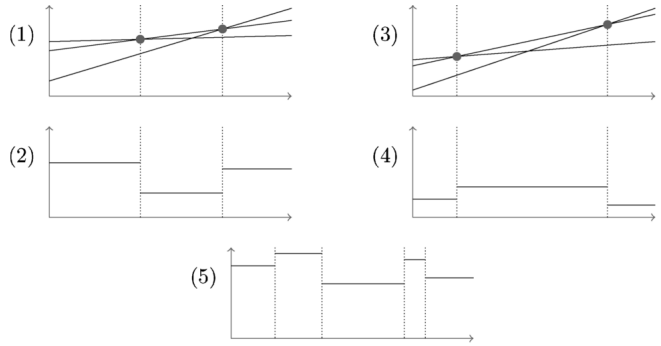


Figura 4.8: Algoritmo MERT exibido de forma gráfica. As linhas do gráfico 1 mostram as traduções hipotéticas para uma das sentenças do *corpus*, juntos com os pontos críticos. Se considerarmos todos os valores máximos para cada valor de λ_k o resultado é a função 3.1. O gráfico 2 mostra os valores de erro para cada intervalo definido pelos pontos críticos. Os gráficos 3 e 4 exibem resultados para outra sentença. Por fim, o gráfico 5 é a soma dos gráficos 2 e 4. Se considerarmos apenas essas duas sentenças de exemplo, é possível obter o valor minimizado para λ_k percorrendo os intervalos até descobrir qual contém o menor erro (retirado de Lopez (2008)).

O processo repete-se para todas as sentenças do *corpus*, gerando uma função de erro para cada uma delas. Para determinar o valor ideal de λ_k elas são somadas (gráfico 5 da figura 4.8)

e então busca-se o intervalo que contém o menor erro. O novo valor de λ_k será o ponto médio desse intervalo.

O algoritmo não garante que se encontre um mínimo global para o vetor de parâmetros λ_1^K. Portanto, para evitar que se busque um mínimo local muito fraco, ele é repetido diversas vezes, cada vez gerando aleatoriamente um novo vetor de entrada $\hat{\lambda}_1^K$. Se o resultado da minimização de $\hat{\lambda}_1^K$ resultar num modelo com erro menor que o vetor anterior (λ_1^K), ele é atualizado com os novos valores. Caso contrário, mantém-se os valores antigos.

5 DECODIFICAÇÃO E AVALIAÇÃO

Obtido o modelo com todos os parâmetros estimados, é possível realizar a tradução de novas sentenças não presentes no *corpus* de treinamento. Essa etapa é chamada de decodificação e consiste em encontrar a sentença $\hat{e}$ que resolve a equação 3.1, replicada abaixo:

$$\hat{e} = \underset{e}{argmax}\, P(e|f)$$

O espaço de possíveis traduções, ainda que restrito pelo modelo e pelos parâmetros estimados, em geral é bastante amplo. Por conta disso, heurísticas de poda como o *beam search* (KOEHN; OCH; MARCU, 2003) são aplicadas na maior parte das implementações existentes. A ideia do *beam search* é descartar uma parte das traduções parciais, mantendo sempre um número fixo de hipóteses (o *beam*) cada vez que uma regra é aplicada. Heurísticas como essa são fundamentais para que os sistemas de SMT possam ser utilizados na prática.

O algoritmo de decodificação em si é diretamente ligado ao formalismo utilizado na modelagem do sistema de SMT. Na seção 5.1 será visto como ela é feita para modelos baseados em sintaxe, em especial para os modelos baseados em SCFGs. Entretanto, ressalta-se que o algoritmo também pode ser usado para transdutores TTS, bastando linearizá-los conforme mostrado na seção 3.1.3.

Métricas automáticas para avaliação de traduções são o tópico da seção 5.2. Ainda que a avaliação não faça parte diretamente da descrição proposta por Lopez (2008), essas métricas são importantes tanto para a realização de treinamentos baseados em métricas de erro (como o MERT, por exemplo) quanto para auxiliar a comparação entre dois sistemas distintos. Essas métricas se baseiam na ideia de comparar o resultado devolvido pelo tradutor com uma tradução de referência, feita por um especialista. Dessa forma, quanto mais esse resultado se aproximar da referência, maior será o valor dessas métricas e, supostamente, melhor será a tradução.

5.1 Decodificação em modelos baseados em sintaxe

No caso de SCFGs, o algoritmo de decodificação é muito semelhante ao de análise sintática. Caso o sistema seja composto apenas pelo modelo de tradução, basta realizar o processo descrito na figura 3.1. Se houver mais de uma derivação possível, considera-se a mais provável (a que possuir a maior probabilidade).

No entanto, como visto anteriormente, um sistema de SMT não é composto apenas pelo modelo de tradução: ele também possui um modelo de língua e, se for um sistema que use a abordagem discriminativa, poderá ter outras *features* também. Nesse caso, o decodificador obtém todas as derivações possíveis e gera uma *floresta de tradução* ou *hipergrafo* (KLEIN; MANNING, 2001; HUANG; CHIANG, 2005).

Uma floresta de tradução é uma forma compacta de representar todas as derivações possíveis para a sentença fonte. Cada aplicação de regra é representada por uma *hiperaresta*, que é uma aresta que parte de vários nodos *cauda* e chega em um nodo *cabeça*. A construção da floresta se baseia na ideia de que derivações de uma sentença possuem regras em comum, que podem ser representadas por apenas uma hiperaresta. Assim, essas regras comuns são compactadas, como mostra a figura 5.1.

Em uma floresta de tradução, cada nodo possui uma probabilidade, correspondente ao valor da formula 3.8[1]. Caso ele seja cabeça de alguma hiperaresta (que é o caso de todos os nodos não-folha), o cálculo também leva em conta os valores dos nodos cauda dessa hiperaresta. No caso de haver mais de uma hiperaresta, normalmente é considerado somente o valor mais alto, correspondente à melhor derivação realizada até o momento.[2]

O decodificador procede fazendo o cálculo para todos os nodos da floresta até chegar à raiz. A partir dela, obtém-se a melhor derivação existente naquela floresta e então a sentença alvo é gerada, num processo similar ao da figura 3.1.

Teoricamente, a construção da floresta possui a mesma complexidade do problema de análise sintática: $O(n^3)$ onde n é o tamanho da sentença[3]. No entanto, no modelo de SMT, o cálculo das probabilidades não leva em conta apenas a regra da SCFG utilizada na tradução mas sim todo o conjunto de *features* do modelo. Em especial, a utilização de um modelo de língua baseado em ngramas traz problemas pois faz com que o número de termos a ser calculado

[1] Lembrando que o modelo gerativo pode ser visto como um caso específico do modelo *log-linear*.

[2] Esse processo é similar ao algoritmo de Viterbi para obtenção da melhor derivação. É possível também somar os valores das hiperarestas, resultando em um algoritmo similar ao Inside(MANNING; SCHüTZE, 2000).

[3] Considerando algoritmos que realizam a binarização da gramática, seja de forma explícita (CKY) ou implícita (Earley).

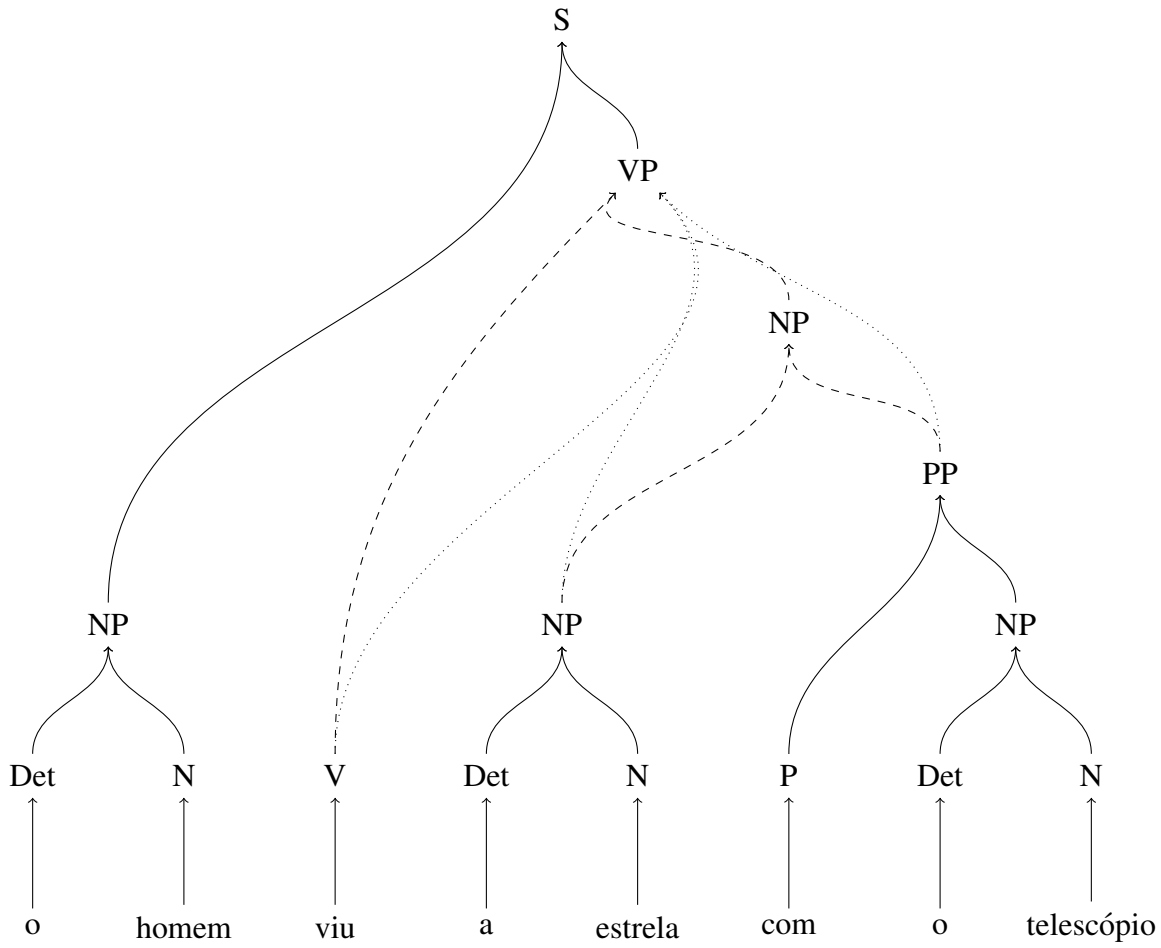


Figura 5.1: Exemplo de uma floresta de tradução representando duas derivações diferentes. Cada seta corresponde a uma hiperaresta. A hiperaresta pontilhada representa uma derivação enquanto as duas hiperarestas tracejadas representam a outra. As hiperarestas sólidas correspondem a aplicações de regras presentes em ambas as derivações, que foram compactadas.

em cada hiperaresta exploda rapidamente. Ainda que essa explosão seja polinomial, na prática ela torna o processo de decodificação muito lento e justifica a adoção de heurísticas de poda da floresta de tradução. Além do já citado *beam search*, uma outra heurística muito utilizada é o *cube pruning* (CHIANG, 2007), que consiste em retirar da floresta termos que sejam menores do que um certo valor limite β. Avaliação de heurísticas desse tipo estão fora do escopo desse trabalho mas vale salientar que o *cube pruning* tornou-se parte da configuração padrão da maior parte dos decodificadores disponíveis, sendo utilizado em todos os experimentos realizados com SCFGs nesse trabalho.

5.1.1 *Reranking*

Os algoritmos de decodificação em geral devolvem a melhor tradução possível para uma sentença fonte. No entanto, nada impede esses algoritmos de gerar as melhores k traduções

possíveis. Esse tipo de saída é interessante quando existem *features* no sistema de SMT que só podem ser calculadas em nível de sentença. Além disso, listas de k melhores traduções podem servir de auxílio em ferramentas de pós-edição utilizadas pelo usuário final.

Teoricamente, todas as *features* de um sistema podem ser calculadas no nível de sentença. Esse tipo de abordagem inclusive possui a vantagem de facilitar os algoritmos de decodificação e torná-los mais rápidos. No entanto, levar em conta as *features* em níveis menores (por exemplo, no nível de cada nodo de uma floresta de tradução) permite que o decodificador faça melhores decisões ao devolver a melhor ou k melhores traduções. *Features* mais simples como tamanho da sentença e total de regras utilizadas são fáceis de serem integradas ao algoritmo, enquanto soluções para a integração de modelos de língua baseados em ngramas surgem na forma de heurísticas como o *cube pruning*. Mas esse tipo de integração nem sempre é possível para todas as *features*.

Modelos de língua sintáticos são um exemplo: a integração deles em algoritmos de decodificação de uma forma genérica é um problema em aberto. Algumas soluções específicas existem: Post (2010) define um algoritmo de decodificação baseado em ITGs que integra um modelo de língua sintático baseado em uma TSG. No entanto, esse algoritmo leva em conta a propriedade das ITGs de serem binarizáveis, o que não ocorre com todas as SCFGs.

Devido a essa limitação, neste trabalho os modelos de língua sintáticos são utilizados como uma ferramenta de *reranking*, que consiste no reordenamento de listas de k melhores traduções geradas pelo decodificador. A partir desse reordenamento, considera-se a melhor tradução gerada pelo sistema aquela que estiver no topo da lista.

5.2 Métricas automáticas de avaliação de sistemas de MT

A forma mais precisa de verificar a performance de um sistema de MT é através da avaliação manual feita por um ou mais especialistas (tradutores humanos). No entanto, essa forma de avaliação costuma ser cara e demorada, ainda mais quando se deseja realizar um treinamento via minimização de erro (como o MERT, por exemplo). Esses problemas levaram à criação de diversas métricas para avaliação automática. As mais utilizadas pela comunidade são a BLEU e a NIST. Ambas se baseiam na comparação entre o resultado fornecido pelo sistema de MT avaliado e uma ou mais traduções de referência, feita previamente por humanos.

5.2.1 BLEU

A métrica BLEU (*Bilingual Evaluation Understudy*) (PAPINENI et al., 2001) é a mais utilizada nos trabalhos de SMT, seja na avaliação em si ou na sua aplicação em modelos treinados através de MERT. O seu estabelecimento como métrica automática mais usada se deve tanto a apresentar alta correlação com avaliações humanas quanto pela sua fácil implementação.

A BLEU é composta de dois fatores principais: a *precisão modificada de n-gramas* e o *fator de brevidade* (*brevity penalty* - BP). A precisão em si é uma métrica que diz quantas vezes um n-grama de um candidato a tradução aparece na tradução de referência. No entanto, um sistema pode gerar vários n-gramas "prováveis", resultando numa tradução ruim mas com alta precisão, como mostrado abaixo:

Candidato: the the the the the the the

Referência: the cat is on the mat

Precisão de unigramas: 7/7

Por conta disso, a BLEU utiliza o conceito de precisão *modificada*, que considera que os n-gramas devem aparecer na mesma quantidade. No exemplo acima, como há apenas 2 "the" na referência, o resultado da precisão modificada de unigramas é 2/7.

Generalizando, é possível definir a precisão modificada de n-gramas P_n como:

$$P_n = \frac{\sum_{n-grama \in candidato} Count_{clip}(n-grama)}{\sum_{n-grama \in candidato} Count(n-grama)} \tag{5.1}$$

onde:

- $Count_{clip}(n-grama)$ é o total de co-ocorrências do n-grama no candidato e na referência.
- $Count(n-grama)$ é o total de ocorrências do n-grama no candidato.

O segundo fator, o BP, leva em conta o fato de que candidatos pequenos demais podem exibir altos valores de precisão modificada. Isso é mostrado no exemplo abaixo:

Candidato: of the

Referência: it is the guiding principle which guarantees the military forces always being under the command of the party

$P_1 = 2/2$

$P_2 = 1/1$

O BP possui, então, a função de prevenir que candidatos assim sejam considerados boas traduções. Ele é definido como:

$$BP = \begin{cases} 1 & \text{se } c > r \\ e^{1-r/c} & \text{se } c \leq r \end{cases} \tag{5.2}$$

onde:

- r = número de palavras na referência.
- c = número de palavras no candidato.

Por fim, a métrica BLEU como um todo é definida como:

$$BLEU = BP \times \exp \sum_{n=1}^{N} w_n(logP_n) \tag{5.3}$$

onde:

- N = limite superior dos n-gramas a serem considerados.
- w_n = peso da precisão modificada do n-grama.

Em geral, utiliza-se o valor 4 para N e $1/N$ para w_n (o que resulta numa distribuição uniforme de valores de precisão modificada). São esses valores que Papineni et al. (2001) utilizam em seus testes de correlação.

5.2.2 NIST

A métrica NIST (DODDINGTON, 2002) é uma variação da BLEU. A principal diferença é que a NIST busca considerar o quão "informativo" um n-grama é: se esse n-grama aparece com

uma frequência menor no texto, ele possui um peso maior. Esse fator de informação é definido através da fórmula abaixo:

$$Info(w_1 \dots w_n) = log_2 \left(\frac{\text{ocorrências de } w_1 \dots w_{n-1}}{\text{ocorrências de } w_1 \dots w_n} \right) \tag{5.4}$$

onde $w_1 \dots w_n$ representam as palavras que compõem o n-grama e as ocorrências são consideradas na tradução de referência.

Além desse fator de informação, a NIST faz uma média aritmética sobre os valores de precisão modificada ao invés da média geométrica definida na BLEU. Por fim, o BP também é alterado com o intuito de minimizar o impacto de mudanças pequenas no tamanho de um candidato. Com isso, a métrica NIST fica definida da seguinte forma:

$$NIST = \sum_{n=1}^{N} \left\{ \frac{Count_{clip}(n-grama) \times Info(n-grama)}{Count(n-grama)} \right\} \times \exp\left\{ \beta log_2 \left[min\left(\frac{c}{r}, 1 \right) \right] \right\} \tag{5.5}$$

Todos os termos acima são definidos como na fórmula da BLEU. O fator β é definido como sendo igual a 0,5 quando c/r é igual a 2/3. Os testes de correlação feitos por Doddington (2002) utilizam $N = 5$.

6 EXPERIMENTOS COM O PAR DE LÍNGUAS INGLÊS E PORTUGUÊS DO BRASIL

Neste capítulo serão apresentados experimentos utilizando o par de línguas inglês e português do Brasil (en-ptBR). Inicialmente, a seção 6.1 mostra os *corpora* e as ferramentas utilizadas nesse trabalho. Os experimentos propriamente ditos são descritos na seção 6.2 e por fim, na seção 6.3 os resultados obtidos são avaliados e discutidos.

6.1 Recursos utilizados

6.1.1 Corpora

Neste trabalho foram utilizados três *corpora*: um paralelo en-ptBR e dois monolíngues em ptBR. Todos estão disponíveis na Internet para download.

PesquisaFAPESP

O PesquisaFAPESP[1] é um *corpus* paralelo composto por 646 artigos científicos da revista Pesquisa FAPESP. Esses artigos foram originalmente escritos em ptBR e posteriormente foram traduzidos para o inglês. O *corpus* já se encontra alinhado sentencialmente, sendo composto por 17.397 pares de sentenças. O PesquisaFAPESP foi utilizado para treinamento dos modelos de tradução e também para avaliação.

[1] Disponível em `www.nilc.icmc.usp.br/lacioweb`

CETENFolha

O CETENFolha[2] é um corpus monolíngue em ptBR, composto por textos do jornal Folha de São Paulo no ano de 1994, possuindo no total 1.597.807 sentenças. Esse *corpus* foi utilizado na criação de um modelo de língua para o ptBR baseado em trigramas.

Bosque

O Bosque[3] é um *treebank* composto no total por 9.368 sentenças em português analisadas sintaticamente e corrigido por linguistas. Nesse trabalho, foi utilizada a fração do Bosque descrita por Beck e Caseli (2012), que corresponde às 4.184 sentenças escritas em português do Brasil. Essa fração foi utilizada para a criação de um modelo de língua para o português do Brasil baseado em uma TSG.

6.1.2 Ferramentas

A maior parte das tarefas descritas nos experimentos foram realizadas por ferramentas disponíveis na Internet.

Analisadores sintáticos

A criação de modelos de tradução baseados em sintaxe passa pela anotação do *corpus* paralelo com suas respectivas árvores sintáticas. Para isso, foram utilizados dois analisadores sintáticos: o Berkeley Parser[4] (PETROV et al., 2006; PETROV; KLEIN, 2007) para o inglês e o LX-Parser[5] (SILVA et al., 2010) para o português do Brasil. O primeiro é um dos analisadores considerado estado-da-arte para o inglês enquanto o segundo é uma implementação do analisador de Stanford (KLEIN; MANNING, 2003) para o português de Portugal[6].

Alinhador lexical

Para realizar o alinhamento lexical foi utilizado o Berkeley Aligner[7] (LIANG; TASKAR; KLEIN, 2006). Tradicionalmente, os trabalhos de SMT utilizam um outro alinhador,

[2]Disponível em `www.linguateca.pt/cetenfolha`
[3]Disponível em `www.linguateca.pt/floresta/corpus.html`
[4]Disponível em `code.google.com/p/berkeleyparser`
[5]Disponível em `lxcenter.di.fc.ul.pt/tools/pt/conteudo/LXParser.html`
[6]Silva et al. (2010) cita que a versão disponiblizada é uma implementação do analisador Berkeley. No entanto, a página que disponibiliza o LX-Parser pede como dependência o download do analisador Stanford.
[7]Disponível em `code.google.com/p/berkeleyaligner`

o GIZA++[8] (OCH; NEY, 2003). No entanto, o Berkeley apresenta resultados similares ao GIZA++ tanto em avaliação intrínseca quanto extrínseca (LIANG; TASKAR; KLEIN, 2006), começando inclusive a ser incorporado aos toolkits de SMT existentes. Por conta disso, foi escolhido o Berkeley por esse apresentar maior facilidade de uso e integração com o restante das ferramentas.

Toolkits de SMT

Para a realização dos experimentos com os modelos de tradução sintáticos, foi utilizado o cdec[9] (DYER et al., 2010) um decodificador para SCFGs. O cdec implementa a decodificação através de florestas de tradução e realiza o *rescoring* de cada nodo da floresta de acordo com diversas funções de *feature* que podem ser designadas pelo modelo. Além disso, ele possui um módulo para realização do MERT e também um *script* de avaliação utilizando as métricas BLEU e NIST.

Os experimentos com SMT baseada em frases, utilizados como *baseline*, foram realizados com o *toolkit* Moses[10] (KOEHN et al., 2007). O Moses também permite a especificação de funções de *feature* e o treinamento através de MERT.

Outras ferramentas

Os experimentos com *reranking* e modelos de língua sintáticos foram realizados através de uma implementação do algoritmo CKY[11] utilizada em Johnson (1998). A extração de transdutores TTS via GHKM foi feita através de uma ferramenta de mesmo nome disponível para download[12].

Para determinar se as diferenças entre os valores de BLEU e NIST dos modelos implementados são estatisticamente significativas, foi utilizada a ferramenta de *bootstrapping* descrita por Zhang, Vogel e Waibel (2004), considerando-se 95% de confiança.[13] Assim, sempre que um resultado for relatado como estatisticamente significativo (ou significante) isso indica que ele passou no teste usando *bootstrapping* com 95% de confiança.

Finalmente, a maior parte dos programas e *scripts* necessários para as etapas que não são

[8] Disponível em `code.google.com/p/giza-pp`
[9] Disponível em `cdec-decoder.org`
[10] Disponível em `www.statmt.org/moses`
[11] Disponível em `web.science.mq.edu.au/~mjohnson/code/cky.tbz`
[12] Disponível em `www.assembla.com/code/ghkm/subversion/nodes`
[13] Disponível em `projectile.sv.cmu.edu/research/public/tools/bootStrap/tutorial.htm`

feitas pelas ferramentas citadas[14] foram escritos na linguagem Python e utilizando o *toolkit* NLTK[15] (BIRD; KLEIN; LOPER, 2009).

6.2 Descrição dos experimentos

6.2.1 Preprocessamento dos corpora

O *corpus* paralelo como um todo passou por duas etapas de preprocessamento:

Tokenização: sinais de pontuação como pontos e vírgulas foram separados, tornando-se *tokens* próprios. As contrações como "don't" no inglês e "na" no português também foram separadas em *tokens* distintos. A tokenização resultante foi obtida através dos analisadores sintáticos utilizados (Berkeley e LX-Parser).

***Downcasing*:** todas as letras maiúsculas existentes foram transformadas em minúsculas.

Normalização de aspas e parênteses: na maior parte das sentenças, os sinais de aspas se encontram na forma de dois apóstrofos ou dois acentos. Esses foram transformados em um símbolo único de aspas duplas (''). Além disso, os símbolos de abre e fecha parênteses foram substituídos pelas expressões "-LRB-" e "-RRB-", respectivamente. Essas normalizações foram necessárias para que esses símbolos fossem corretamente tratados pelos analisadores sintáticos.

Após essa etapa, o *corpus* foi dividido em três:

Treinamento (80%): utilizado para obter os modelos de tradução e de língua.

Validação (10%): utilizado para estimar os parâmetros das funções de *feature* via MERT.

Teste (10%): utilizado para avaliação dos modelos.

O *corpus* de treinamento foi alinhado lexicalmente pelo alinhador Berkeley utilizando as configurações padrão (5 iterações em cada direção). Cada metade correspondente a uma das línguas também serviu de entrada para os respectivos analisadores sintáticos, obtendo assim as árvores de cada sentença.

Em relação aos outros dois *corpora* foi feita a tokenização do CETENFolha mantendo o mesmo método usado para o PesquisaFAPESP. O Bosque, por ser um *treebank*, já se encontra tokenizado.

[14] Disponíveis em `bitbucket.org/beckdaniel/pynus-nltk`
[15] Disponível em `www.nltk.org`

6.2.2 Treinamento

Obtido os *corpora* preprocessados, foram treinados 6 modelos de tradução utilizando o *corpus* de treinamento:

- GHKM-TTS en-ptBR
- GHKM-TTS ptBR-en
- GHKM-STT en-ptBR
- GHKM-STT ptBR-en
- PB-SMT en-ptBR
- PB-SMT ptBR-en

Para a realização dos experimentos com modelos de tradução utilizando amostragem de Gibbs, foi utilizado um subconjunto do *corpus* de treinamento com sentenças com 40 tokens ou menos.[16] O *corpus* de treinamento resultante possui em torno de 10300 sentenças, variando conforme a direção da tradução utilizada, e serviu de entrada para os seguintes modelos de tradução:

- GHKM-TTS-40 en-ptBR
- Gibbs-TTS-40 en-ptBR
- GHKM-TTS-40 ptBR-en
- Gibbs-TTS-40 ptBR-en

Para os experimentos que investigam a influência do tamanho do *corpus*, foram treinados os seguintes modelos, utilizando frações correspondentes a 75%, 50% e 25% do total de sentenças do *corpus* de treinamento:

- GHKM-TTS-75% en-ptBR
- GHKM-TTS-50% en-ptBR
- GHKM-TTS-25% en-ptBR
- PB-SMT-75% en-ptBR

[16] Essa restrição foi necessária devido à implementação utilizada do amostrador.

- PB-SMT-50% en-ptBR
- PB-SMT-25% en-ptBR

Também foram treinados os seguintes modelos de língua, utilizando o *corpus* de treinamento (FAPESP), além do CETENFolha e do Bosque:

- en:
  - FAPESP-trigramas
  - FAPESP-TSG (sintático)
- ptBR:
  - FAPESP-trigramas
  - FAPESP-TSG
  - CETENFolha-trigramas
  - Bosque-TSG

Esses (sub)modelos foram então inseridos em diversos modelos discriminativos definidos pelas seguintes *features*:

- Modelo de tradução
- Modelo de língua
- Peso léxico da sentença fonte
- Peso léxico da sentença alvo
- Total de regras utilizadas
  - No caso dos modelos GHKM e Gibbs, elas foram separadas pela sua aridade (número de símbolos do lado direito)
- Tamanho da sentença alvo
- Regras *pass-through*, no caso dos modelos GHKM e Gibbs: regras adicionadas em tempo real pelo decodificador para o tratamento de palavras não existentes no modelo de tradução
- Distorção, no caso dos modelos PB-SMT

O treinamento via MERT foi realizado pelo Moses, no caso dos modelos baseados em frases, e pelo cdec, no caso dos modelos sintáticos. Ambos utilizaram o corpus de validação e a métrica de avaliação utilizada foi a BLEU. Foram realizadas iterações suficientes até que a mudança nos resultados de BLEU fosse pouco significativa.

6.2.3 Comparação entre os modelos GHKM e PB-SMT

Os primeiros experimentos comparam modelos treinados via GHKM com o estado-da-arte (PB-SMT). Inicialmente foi utilizada a abordagem TTS. Os resultados são mostrados na tabela 6.1. Em ambas as direções do par de línguas, o GHKM mostrou-se significativamente pior que o baseline.

	BLEU	NIST
GHKM-TTS en-ptBR	0.2745	7.2783
PB-SMT en-ptBR	0.3898	8.7376
GHKM-TTS ptBR-en	0.0946	3.8034
PB-SMT ptBR-en	0.4001	9.1309

Tabela 6.1: Comparação entre GHKM e PB-SMT

Uma das razões principais para o pior desempenho do GHKM é o fato dele ter realizado muitos reordenamentos onde não deveria, como mostrado no exemplo abaixo, para o par en-ptBR:

Sentença fonte: `several factors contribute towards the complexity of the process , in any language .`

Referência: `vários fatores contribuem para a complexidade de o processo , em qualquer idioma .`

Hipótese PB-SMT: `vários fatores contribuir para a complexidade de o processo , em qualquer linguagem .`

Hipótese GHKM: `contribuir para a complexidade de o processo de vários fatores , em qualquer linguagem .`

Nesse caso, o reordenamento realizado pelo GHKM causou um problema de adequação em relação ao significado original (no caso do trecho “processo de vários fatores”).

No entanto, a rigidez das métricas também prejudicou um pouco a avaliação do GHKM, como no exemplo abaixo:

Original: `and their vocabulary is very limited .`

Referência: `e seu vocabulário é muito limitado .`

Hipótese PB-SMT: `e o vocabulário é muito limitada .`

Hipótese GHKM: `e é muito limitada seu vocabulário .`

O reordenamento nesse caso não prejudicou o sentido da sentença. Mas o fato da hipótese gerada pelo PB-SMT possuir um trigrama em comum com a referência ("vocabulário é muito") fez com que ela tivesse um valor maior para ambas as métricas.

Ainda que eventualmente as métricas possam ter prejudicado a avaliação, o que se observou em relação ao GHKM é que muitos dos reordenamentos realizados são de fato erros no processo de tradução. Uma das razões possíveis para o GHKM proceder dessa forma é o fato dele ser um modelo relativamente simples de tradução, que captura pouca informação de contexto. Refinamentos, tanto do modelo quanto dos algoritmos de aprendizagem, poderão melhorar as gramáticas inferidas e diminuir a diferença entre as métricas. No entanto, as ferramentas de preprocessamento também influem nos resultados. Em especial, erros presentes nos analisadores sintáticos podem se propagar para o modelo de tradução. Outra possibilidade também pode ser o tamanho reduzido do *corpus*. Se por um lado, adicionar informações de sintaxe teoricamente deveria gerar um modelo de tradução melhor, por outro lado é necessário que o *corpus* de treinamento seja grande o suficiente para que haja estatísticas suficientes dessas informações.

Para verificar essas hipóteses, foram feitos outros dois experimentos, um com o objetivo de investigar a influência do analisador sintático e outro com o objetivo de investigar a influência do tamanho do corpus.

6.2.4 Influência do analisador sintático

Na tabela 6.1 nota-se que os valores de BLEU e NIST para PB-SMT são bastante similares para as duas direções do par. No entanto, o mesmo não se refletiu para os modelos GHKM: os valores para ptBR-en são muito mais baixos que para en-ptBR. Considerando-se que a maior diferença entre os dois modelos é o analisador sintático utilizado (Berkeley no caso en-ptBR e LX-Parser no caso ptBR-en), foram realizados experimentos invertendo a abordagem utilizada para o GHKM, passando para a STT. Com isso, os analisadores também são utilizados de forma inversa. Assim, se houver grande discrepância entre os desempenhos dos analisadores, deverá

haver novamente uma grande diferença entre os valores de BLEU e NIST (dessa vez, em prol da direção ptBR-en).

A tabela 6.2 confirma essa hipótese: a inversão da abordagem traz uma melhora estatisticamente significativa nos valores para a direção ptBR-en e, ao mesmo tempo, piora os resultados para en-ptBR. A sentença abaixo mostra um exemplo de piora para a direção en-ptBR:

Sentença fonte: `several factors contribute towards the complexity of the process , in any language .`

Referência: `vários fatores contribuem para a complexidade de o processo , em qualquer idioma .`

Hipótese TTS: `contribuir para a complexidade de o processo de vários fatores , em qualquer linguagem .`

Hipótese STT: `alguns contribuir para a complexidade de os fatores de o processo de o brasil , in any language .`

A utilização do LX-Parser nesse caso gerou uma gramática com regras espúrias (como no caso do acréscimo da palavra "brasil") e também com menor cobertura (observada pelo fato do trecho "in any language" não ter sido traduzido). Esses resultados demonstram que diferenças nos métodos de análise sintática interferem de maneira sensível na criação dos modelos GHKM.

	BLEU	NIST
GHKM-TTS en-ptBR	0.2745	7.2783
GHKM-STT en-ptBR	0.0872	3.1267
GHKM-TTS ptBR-en	0.0946	3.8024
GHKM-STT ptBR-en	0.1739	6.1405

Tabela 6.2: Comparação entre abordagens TTS e STT

6.2.5 Influência do tamanho do corpus

Neste experimento, foram criados três novos *corpora* de treinamento e validação, com 75%, 50% e 25% das sentenças, respectivamente. Para este experimento foi considerada somente a direção en-ptBR. A hipótese é que se o tamanho do *corpus* tiver importância maior para o modelo GHKM do que para o modelo PB-SMT, então a piora nos valores para o GHKM deverá ser mais abrupta do que para o PB-SMT. Os resultados de BLEU e NIST são mostrados na tabela 6.3 e nos gráficos das figura 6.1 e 6.2. Todas as diferenças apresentadas se mostraram

estatisticamente significantes, tanto aquelas entre tamanho de *corpus* diferentes, quanto entre as abordagens utilizadas.

	BLEU	NIST
GHKM-TTS en-ptBR	0.2745	7.2783
GHKM-TTS-75% en-ptBR	0.2841	7.3936
GHKM-TTS-50% en-ptBR	0.2565	7.3611
GHKM-TTS-25% en-ptBR	0.2550	7.3158
PB-SMT en-ptBR	0.3898	8.7376
PB-SMT-75% en-ptBR	0.3847	8.6392
PB-SMT-50% en-ptBR	0.3794	8.5470
PB-SMT-25% en-ptBR	0.3607	8.2450

Tabela 6.3: Comparação entre GHKM e PB-SMT variando o tamanho do *corpus*

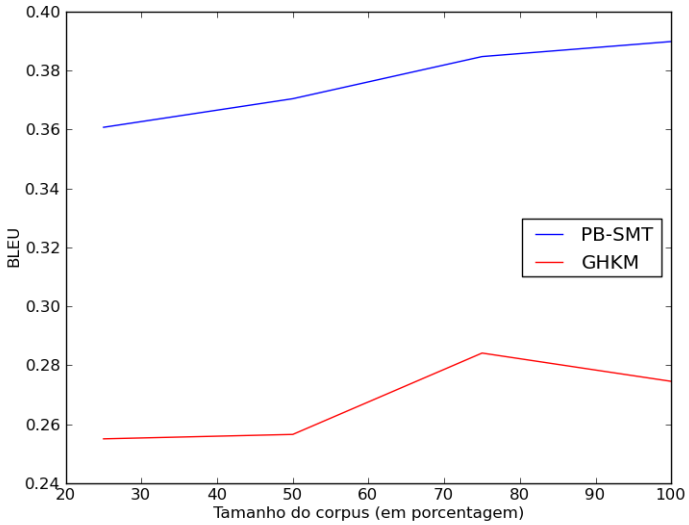


Figura 6.1: Valores de BLEU para GHKM e PB-SMT variando o tamanho do *corpus*

Como esperado, houve queda nos resultados para o modelo PB-SMT. Com o GHKM, também houve piora a partir do *corpus* com 75% das sentenças mas curiosamente, o valor de BLEU para esse modelo foi melhor do que considerando o *corpus* inteiro. Com os valores de NIST, a diferença é mais surpreendente: todos os modelos treinados em uma fração do *corpus* foram melhores do que considerando todas as sentenças.

Uma das causas possíveis para esse comportamento é o *overfitting*: os pesos obtidos pelo MERT podem não estar generalizando o suficiente para o *corpus* de teste. Clark et al. (2011) cita esse problema e mostra alguns resultados que o corroboram. Observa-se que nesse experimento, o *corpus* de validação também foi fracionado nos respectivos tamanhos. Assim, uma possibilidade é o modelo completo (com todas as sentenças) estar memorizando o *corpus* de validação por este ser grande demais.

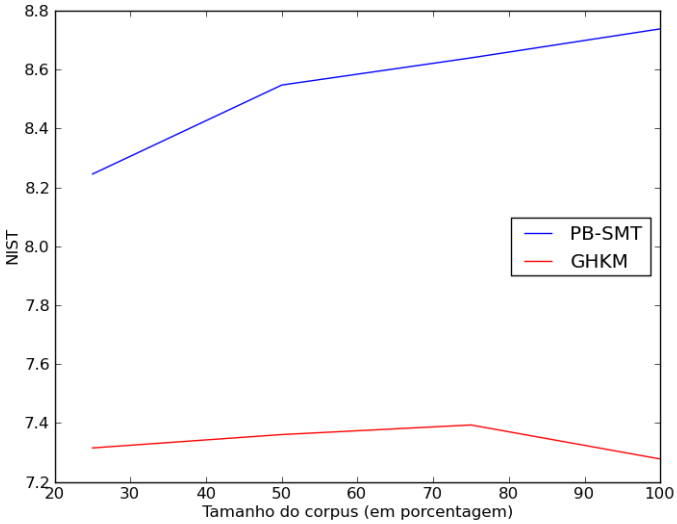


Figura 6.2: Valores de NIST para GHKM e PB-SMT variando o tamanho do *corpus*

Em relação à hipótese inicial, observa-se que a curva de scores BLEU é mais acentuada para o GHKM mas a curva de scores NIST tende a ser mais acentuada para PB-SMT. Somando-se a essas observações a questão do *overfitting* citada anteriormente, não é possível afirmar que há influência do tamanho do *corpus* em relação ao GHKM. Experimentos mais refinados, com *corpora* maiores e também buscando tamanhos menores (mas ainda relativos) do conjunto de sentenças usado para validação podem trazer resultados mais significativos.

6.2.6 Comparação entre os modelos GHKM e Gibbs

Para investigar se o processo de amostragem de Gibbs resulta em modelos de tradução melhores, foram comparados os dois modelos para as duas direções do par de línguas. Os parâmetros utilizados no treinamento via amostragem de Gibbs foram os mesmos utilizados por Cohn e Blunsom (2009): o parâmetro de concentração α foi setado em 100000, os parâmetros da distribuição base β_{expand}, β_{child} e β_{term} foram setados todos em 0.5 e foram realizadas 300 iterações pelo amostrador, sendo que o resultado da última iteração foi utilizado para extração das regras TTS[17].

Os resultados da tabela 6.4 mostram que essa hipótese se confirma para a direção en-ptBR, com diferenças significativas estatisticamente. No caso da direção ptBR-en, houve melhora no valor de BLEU mas piora no valor de NIST. Apesar de ambas diferenças serem estatisticamente significativas, essa divergência nos valores impossibilita que se tire conclusões a respeito dessa

[17]Conforme citado na seção 4.1.2, o amostrador foi adaptado para visitar somente nodos de fronteira pois os alinhamentos gerados pelo Berkeley violam as regras originalmente propostas por Cohn e Blunsom (2009).

direção.

	BLEU	NIST
GHKM-TTS-40 en-ptBR	0.2037	5.7239
Gibbs-TTS-40 en-ptBR	0.2079	5.7521
GHKM-TTS-40 ptBR-en	0.1013	3.7845
Gibbs-TTS-40 ptBR-en	0.1031	3.7726

Tabela 6.4: Comparação entre GHKM e Gibbs

Comparando-se as hipóteses geradas pelo GHKM e pelo Gibbs para a direção en-ptBR, foi observado que a gramática gerada pelo Gibbs possibilitou um melhor discernimento quanto à utilização das regras de reordenamento. A sentença abaixo mostra um exemplo onde houve um reordenamento errado feito pelo GHKM e que foi evitado pelo Gibbs:

Sentença fonte: `precision in diagnosis`

Referência: `precisão em o diagnóstico`

Hipótese GHKM: `em o diagnóstico de a precisão`

Hipótese Gibbs: `a precisão de o diagnóstico`

6.2.7 Influência do modelo de língua baseado em trigramas

Os modelos citados até agora utilizaram o próprio *corpus* FAPESP para criação do modelo de língua baseado em trigramas. Para observar qual a influência de utilizar um *corpus* maior, também foi treinado um modelo de trigramas utilizando o CETENFolha. A ideia, aqui, é verificar se algum modelo de tradução (PB-SMT ou GHKM) se beneficia mais de modelos de língua mais robustos. Os resultados, exibidos na tabela 6.5, mostram que os modelos GHKM têm uma melhora um pouco maior do que os modelos PB-SMT. Ambas as diferenças entre valores de BLEU e NIST são estatisticamente significantes.

	BLEU	NIST
GHKM-TTS en-ptBR c/ PesquisaFAPESP	0.2745	7.2783
GHKM-TTS en-ptBR c/ CETENFolha	0.3132	7.7660
PB-SMT en-ptBR c/ PesquisaFAPESP	0.3898	8.7376
PB-SMT en-ptBR c/ CETENFolha	0.4220	9.0958
Diferença GHKM-TTS	0.0387	0.4877
Diferença PB-SMT	0.0322	0.3582

Tabela 6.5: Comparação entre modelos que usam o PesquisaFAPESP e o CETENFolha como modelos de língua

O exemplo abaixo, com a mesma sentença da seção 6.2.4, mostra como um modelo de língua maior conseguiu prevenir o GHKM de gerar uma hipótese com um reordenamento errado:

Sentença fonte: `several factors contribute towards the complexity of the process , in any language .`

Referência: `vários fatores contribuem para a complexidade de o processo , em qualquer idioma .`

Hipótese PesquisaFAPESP: `contribuir para a complexidade de o processo de vários fatores , em qualquer linguagem .`

Hipótese CETENFolha: `vários fatores que contribuem para a complexidade de o processo , em qualquer língua .`

6.2.8 Influência da adição de *features*

Para verificar o quanto as *features* citadas na seção 6.2.2 interferem no resultado, também foram realizados experimentos utilizando apenas os modelos de tradução GHKM, sem as *features*. A tabela 6.6 compara os dois tipos de modelos. No caso da direção en-ptBR, observou-se aumento significativo nos valores de BLEU e NIST, mostrando que a adição das *features* ajuda a obter melhores traduções.

No caso da direção ptBR-en, no entanto, os resultados foram inversos: a adição das *features* junto com o MERT pioraram os resultados. Uma hipótese para este comportamento é novamente a questão do *overfitting*: o modelo pode estar memorizando o *corpus* de validação e não generalizando o suficiente para sentenças desconhecidas. Outro fenômeno observado é a baixa cobertura da gramática, resultando em muitas palavras não traduzidas, como mostra o exemplo abaixo:

Sentença fonte: `lingüistas e engenheiros de a unicamp formulam um sistema de fala com jeito brasileiro`

Referência: `linguists and engineers from unicamp formulate a speech system with a brazilian accent`

Hipótese sem *features*: `lingüistas and engineers of the unicamp formulam a system fala com jeito brasileiro`

Hipótese com *features*: `a system of lingüistas unicamp and engineers of the formulam fala com jeito brasileiro`

A grande quantidade de palavras não traduzidas influi diretamente nos valores de BLEU e, consequentemente, no algoritmo de MERT. Experimentos com *corpora* maiores (e consequentemente, gramáticas com maior cobertura) podem trazer maiores indícios sobre esse comportamento do MERT.

	BLEU	NIST
GHKM-TTS en-ptBR sem *features*	0.2212	7.1231
GHKM-TTS en-ptBR com *features*	0.2745	7.2783
GHKM-STT en-ptBR sem *features*	0.0349	1.6531
GHKM-STT en-ptBR com *features*	0.0872	3.1267
GHKM-TTS ptBR-en sem *features*	0.1147	3.8837
GHKM-TTS ptBR-en com *features*	0.0946	3.8024
GHKM-STT ptBR-en sem *features*	0.2189	7.0646
GHKM-STT ptBR-en com *features*	0.1739	6.1405

Tabela 6.6: Comparação entre modelos GHKM com e sem adição de *features*

6.2.9 *Reranking* utilizando modelos de língua sintáticos

Os experimentos com *reranking* com modelos de língua baseados em TSGs foram divididos em duas partes. O primeiro experimento buscou observar a diferença entre construir esse modelo a partir do próprio PesquisaFAPESP analisado automaticamente pelo LX-Parser e o Bosque. A hipótese é de que o Bosque, por ter sido corrigido manualmente, geraria um modelo mais robusto, que melhor desambiguaria as listas de melhores traduções. Para o treinamento de ambos os modelos, foram utilizados os parâmetros descritos por Beck e Caseli (2012): o parâmetro de concentração α foi setado em 100, o parâmetro da distribuição base β foi setado em 0.7 e foram realizadas 1000 iterações pelo amostrador, sendo que a gramática resultante foi extraída da última iteração.

O segundo experimento comparou as melhores traduções geradas por sistemas GHKM e PB-SMT com e sem o *reranking*, visando a observar se esse processo está de fato melhorando os resultados dos sistemas.

As listas de *k* melhores traduções foram obtidas da seguinte forma: para o PB-SMT, foram coletadas as 100 melhores traduções de cada sentença. Como elas não são necessariamente únicas (ou seja, as listas possuem traduções iguais), elas foram agrupadas, gerando uma lista com traduções únicas. No caso dos modelos GHKM, o cdec já possui a opção de gerar listas de traduções únicas. Nesse caso foram utilizadas as 10 melhores.

Os resultados do primeiro experimento aparecem na tabela 6.7. Tanto para o PB-SMT quanto para o GHKM o Bosque mostrou-se *pior* do que o PesquisaFAPESP. Provavelmente a razão disso é o tamanho dos dois *corpora*: o Bosque possui 4184 sentenças enquanto a fração de treinamento do PesquisaFAPESP possui 13913 sentenças, três vezes mais. Por conta disso, o Bosque possui um léxico menor, tornando a análise mais difícil especialmente quanto há palavras desconhecidas no *corpus* de teste. Assim, o fato do PesquisaFAPESP possuir um léxico maior acaba sendo mais importante do que eventuais erros de análise sintática gerados pelo LX-Parser.

	BLEU	NIST
GHKM-TTS en-ptBR c/ PesquisaFAPESP	0.2850	7.4306
GHKM-TTS en-ptBR c/ Bosque	0.2812	7.3832
PB-SMT en-ptBR c/ PesquisaFAPESP	0.3898	8.7737
PB-SMT en-ptBR c/ Bosque	0.3884	8.7555

Tabela 6.7: Comparação entre modelos de língua utilizando o Bosque e o PesquisaFAPESP

A tabela 6.8 mostra os resultados do segundo experimento. Os modelos de língua utilizados foram treinados com o PesquisaFAPESP. No caso dos modelos GHKM, o *reranking* melhorou os resultados para ambas as direções do par de línguas. A diferença entre os valores, ainda que pequena, é estatisticamente significativa. Isso mostra que integrar um modelo de língua sintático a um modelo de tradução sintático tem o potencial de gerar traduções melhores, como mostra o exemplo abaixo:

Sentença fonte: `agora , estamos realizando a análise química de as bebidas .`

Referência: `we are now carrying out a chemical analysis of the drinks .`

Hipótese s/ *reranking*: `now , are carrying , analytical chemistry , , beverages .`

Hipótese c/ *reranking*: `now , we carrying , the chemical analysis of drink .`

Para os modelos PB-SMT, houve uma piora nos valores no caso da direção ptBR-en. No caso da direção contrária, houve melhora estatisticamente significativa somente no valores de NIST. O fato de haver melhorias nas traduções do GHKM mas não do PB-SMT pode estar ligado às listas de k melhores traduções. A ideia de um modelo de língua sintático é dar um

valor maior para os candidatos que sigam as regras de sintaxe embutidas no modelo (como regras de reordenamento, por exemplo). Se candidatos assim não estiverem presentes na lista de melhores traduções (o que pode estar acontecendo no caso do PB-SMT), o modelo de língua sintático perde bastante da sua capacidade de desambiguação.

	BLEU	NIST
GHKM-TTS en-ptBR s/ *reranking*	0.2745	7.2783
GHKM-TTS en-ptBR c/ *reranking*	0.2850	7.4306
GHKM-STT ptBR-en s/ *reranking*	0.1739	6.1405
GHKM-STT ptBR-en c/ *reranking*	0.1793	6.2198
PB-SMT en-ptBR s/ *reranking*	0.3898	8.7376
PB-SMT en-ptBR c/ *reranking*	0.3898	8.7737
PB-SMT ptBR-en s/ *reranking*	0.4001	9.1309
PB-SMT ptBR-en c/ *reranking*	0.3941	9.1253

Tabela 6.8: Comparação entre os modelos de tradução com e sem *reranking*

6.3 Análise dos resultados

Com exceção dos resultados de PB-SMT para a direção ptBR-en da tabela 6.8, todas as diferenças entre os valores de BLEU e NIST em análise em cada experimento se mostraram estatisticamente significativas com 95% de confiança. Dessa forma, é possível chegar a uma série de conclusões relativas aos experimentos realizados e ao par de línguas utilizado:

1) Modelos GHKM ainda não ultrapassam o estado-da-arte: melhores algoritmos de inferência de transdutores devem ser investigados. No entanto, o GHKM pode servir como ponto de partida para esses algoritmos.

2) O desempenho dos analisadores sintáticos influi mais do que a abordagem utilizada para o modelo de tradução: os resultados indicam que, devido à grande interferência do desempenho dos analisadores, vale a pena investigar tanto a abordagem TTS quanto STT na hora de utilizar um modelo de tradução sintático.

3) Para a direção en-ptBR, utilizar amostragem de Gibbs gera um modelo de tradução melhor do que utilizar somente o GHKM.

4) Modelos de língua baseados em trigramas tendem a ter influência maior em modelos de tradução sintáticos: o fato de haver melhoria considerável quando se alterou o tamanho do modelo de língua mostra que eles também são importantes mesmo em sistemas de SMT baseada em sintaxe.

5) Modelos de língua sintáticos melhoram os modelos de tradução sintáticos: mesmo utilizando a abordagem de *reranking*, que possui pouco poder de desambiguação em relação a uma integração completa, os resultados obtidos foram melhores. Uma melhor integração desses modelos de língua ao processo de decodificação poderá ajudar ainda mais a desambiguação das traduções.

Além disso, os resultados mostrados nas seções 6.2.5 e 6.2.8 sugerem que a estimativa de parâmetros via MERT está sendo prejudicada ou pela baixa cobertura da gramática ou pelo *overfitting* em relação ao *corpus* de validação. No futuro, pretende-se investigar mais profundamente essa etapa, variando os parâmetros do algoritmo e também realizando experimentos com *corpora* de maior tamanho.

7 CONSIDERAÇÕES FINAIS

Neste capítulo são apresentadas as considerações finais relativas ao projeto. As principais contribuições desse trabalho são apresentadas na seção 7.1 enquanto a seção 7.2 mostra as possibilidades de trabalhos futuros.

7.1 Contribuições

Este projeto teve como objetivo investigar o uso de informação sintática na tradução automática entre as línguas inglês e português do Brasil. O trabalho foi focado na utilização de gramáticas de árvores e autômatos TTS, formalismos bastante promissores na área. Foi definida uma série de modelos utilizando esses formalismos e experimentos foram realizados com o objetivo de investigar várias características desses modelos.

Nesse sentido, pode-se concluir que os experimentos realizados forneceram vários indícios sobre o desempenho dos modelos baseados em sintaxe na tradução entre os idiomas português do Brasil e inglês. Em especial, as conclusões apresentadas na seção 6.3 fornecem subsídios e direcionamentos para pesquisas futuras. Conclui-se, portanto, que o uso de informação sintática na tradução automática é promissor. Além disso, especificamente relacionado ao objetivo desse trabalho que é investigar como a informação sintática pode ser usada na tradução automática, vale ressaltar as seguintes contribuições:

- É importante ter em mente que o desempenho dos analisadores sintáticos tem impacto relevante na qualidade do modelo de tradução gerado a partir de ároves sintáticas analisadas automaticamente. Nos experimentos descritos nesse trabalho, o impacto foi negativamente maior na análise sintática do português do Brasil usando o LX-Parser;
- Sugere-se o uso de GHKM com a amostragem de Gibbs;
- Sugere-se o *reranking* com modelos de língua sintáticos aplicados à saída do modelo de

tradução baseado em sintaxe.

No decorrer do projeto, foi também desenvolvido um conjunto de programas e scripts para a inferência de TSGs a partir de *treebanks* utilizando Processo Dirichlet e amostragem de Gibbs. Esse conjunto está disponível na Internet[1] e poderá ser utilizado por toda a comunidade científica em pesquisas futuras.

7.2 Trabalhos futuros

Os modelos implementados e avaliados nesse trabalho apresentam grande potencial de expansão em vários aspectos. Destacam-se algumas dessas possibilidades a seguir:

Experimentos com *corpora* maiores: recentemente, Aziz e Specia (2011) publicaram uma nova versão do PesquisaFAPESP, possuindo em torno de 180.000 sentenças. Repetir os experimentos descritos nas seções 6.2.5 e 6.2.8 com essa nova versão podem trazer melhores indícios sobre a influência do tamanho do *corpus* na geração do modelo de tradução e também sobre a questão do MERT diminuir a acurácia do modelo para a direção ptBR-en.

Formalismos e modelagem: o foco desse projeto está na representação de sintaxe através de árvores de constituintes. No entanto, eles podem ser aplicados também para árvores de dependência. Essas possuem a vantagem de poderem ser aprendidas a partir de *corpora* em texto puro, sem necessitar de *treebanks*. A utilização de outros tipos de gramáticas de árvores, como por exemplo as *Tree Adjoining Grammars* (TAGs) (JOSHI; SCHABES, 1997) também é um ponto interessante de ser explorado.

Utilização de *features* refinadas: há uma tendência em SMT de se investigar modelos puramente discriminativos, sem a utilização de submodelos gerativos como *features*. Esse tipo de abordagem é ortogonal à utilização de sintaxe e a combinação dessas abordagens pode trazer direções interessantes de pesquisa.

Métodos de aprendizado: ainda que o GHKM seja um modelo simples de inferência de transdutores TTS, ele pode servir de base para a elaboração de métodos mais avançados, que consigam realizar a extração de regras que descrevam melhor os fenômenos sintáticos do par de línguas em questão.

[1] `bitbucket.org/beckdaniel/pynus-nltk`

Integração de modelos de língua sintáticos à decodificação: esse é um problema ainda em aberto mas os resultados obtidos nesse trabalho trazem indícios que é interessante investigar métodos que visem a uma maior integração desses modelos aos algoritmos de decodificação. Com isso, seria possível realizar a desambiguação em níveis menores que o da sentença, podendo trazer resultados ainda melhores.

Investigação da influência dos analisadores sintáticos: nesse trabalho, observou-se que o analisador sintático utilizado influi bastante nos resultados. Sendo assim, um ponto a ser investigado são as características desses analisadores que fazem eles serem melhores ou piores que outros dentro da construção de um modelo de tradução sintático. Assim, seria possível adaptar esses analisadores, tornado-os mais adequados para a tarefa de SMT.

REFERÊNCIAS BIBLIOGRÁFICAS

AZIZ, W.; SPECIA, L. Fully Automatic Compilation of Portuguese-English and Portuguese-Spanish Parallel Corpora. In: *8th Brazilian Symposium in Information and Human Language Technology (STIL-2011)*. [S.l.: s.n.], 2011.

BECK, D. E.; CASELI, H. D. M. Bayesian Induction of Syntactic Language Models for Brazilian Portuguese. In: *10th International Conference on Processing of the Portuguese Language - PROPOR*. [S.l.: s.n.], 2012. p. 157–167.

BIRD, S.; KLEIN, E.; LOPER, E. *Natural Language Processing with Python*. O'Reilly Media, 2009. Disponível em: <http://nltk.org/book>.

BOD, R. Do all fragments count? *Natural Language Engineering*, p. 1–20, 2003.

BROWN, P. et al. A statistical approach to machine translation. *Computational Linguistics*, v. 16, n. 2, p. 79–85, 1990.

BROWN, P. et al. The mathematics of statistical machine translation: Parameter estimation. *Computational linguistics*, MIT Press, v. 19, n. 2, p. 263–311, 1993.

CHARNIAK, E. Statistical parsing with a context-free grammar and word statistics. In: *Proceedings of the National Conference on Artificial Intelligence*. [S.l.: s.n.], 1997. p. 598–603.

CHARNIAK, E.; KNIGHT, K.; YAMADA, K. Syntax-based language models for statistical machine translation. In: *MT Summit IX*. [S.l.: s.n.], 2003. p. 40–46.

CHIANG, D. Hierarchical Phrase-Based Translation. *Computational Linguistics*, MIT Press, v. 33, n. 2, p. 201–228, jun. 2007. ISSN 0891-2017.

CHIANG, D.; KNIGHT, K.; WANG, W. 11,001 new features for statistical machine translation. In: *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*. [S.l.]: Association for Computational Linguistics, 2009. p. 218–226.

CLARK, J. et al. Better hypothesis testing for statistical machine translation: Controlling for optimizer instability. In: *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*. [S.l.]: Association for Computational Linguistics, 2011. p. 176–181.

COHN, T.; BLUNSOM, P. A Bayesian model of syntax-directed tree to string grammar induction. In: *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing Volume 1 - EMNLP '09*. Morristown, NJ, USA: Association for Computational Linguistics, 2009. p. 352–361. ISBN 9781932432596.

COHN, T.; BLUNSOM, P.; GOLDWATER, S. Inducing tree-substitution grammars. *The Journal of Machine Learning*, v. 11, p. 3053–3096, 2010.

COHN, T.; GOLDWATER, S.; BLUNSOM, P. Inducing compact but accurate tree-substitution grammars. In: *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*. Morristown, NJ, USA: Association for Computational Linguistics, 2009. p. 548–556. ISBN 9781932432411.

COLLINS, M. *Head-driven statistical models for natural language parsing*. Tese (Doutorado) — University of Pennsylvania, 1999.

COMON, H. et al. Tree automata techniques and applications. *Available on: http://www. grappa. univ-lille3. fr/tata*, v. 10, 1997.

DENERO, J.; BOUCHARD-CôTé, A.; KLEIN, D. Sampling alignment structure under a Bayesian translation model. In: *Proceedings of the Conference on Empirical Methods in Natural Language Processing*. [S.l.]: Association for Computational Linguistics, 2008. p. 314–323.

DODDINGTON, G. Automatic evaluation of machine translation quality using n-gram co-occurrence statistics. In: *Proceedings of the second international conference on Human Language Technology Research*. [S.l.: s.n.], 2002. p. 128–132.

DYER, C. et al. cdec: A decoder, alignment, and learning framework for finite-state and context-free translation models. In: *Proceedings of the ACL 2010 System Demonstrations*. [S.l.]: Association for Computational Linguistics, 2010. p. 7–12.

EISNER, J. Learning non-isomorphic tree mappings for machine translation. In: *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics-Volume 2*. Morristown, NJ, USA: Association for Computational Linguistics, 2003. p. 205–208. ISBN 0111456789.

GALLEY, M. et al. Scalable inference and training of context-rich syntactic translation models. In: *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the ACL - ACL '06*. [S.l.: s.n.], 2006. p. 961–968.

GALLEY, M. et al. What's in a translation rule? In: *Proceedings of the Human Language Technology and North American Association for Computational Linguistics Conference (HLT/NAACL 2004)*. [S.l.: s.n.], 2004. p. 273–280.

GOLDWATER, S.; GRIFFITHS, T. L.; JOHNSON, M. A Bayesian framework for word segmentation: Exploring the effects of context. *Cognition*, v. 112, n. 1, p. 21–54, jul. 2009. ISSN 1873-7838.

GRAEHL, J.; KNIGHT, K.; MAY, J. Training Tree Transducers. *Computational Linguistics*, v. 34, p. 391–427, 2008.

HUANG, L.; CHIANG, D. Better k-best parsing. In: *Proceedings of the Ninth International Workshop on Parsing Technology*. [S.l.]: Association for Computational Linguistics, 2005. p. 53–64.

HUTCHINS, J. Machine translation: A concise history. *Computer aided translation: theory and practice.*, p. 1–21, 2007.

JOHNSON, M. PCFG models of linguistic tree representations. *Computational Linguistics*, MIT Press, v. 24, n. 4, p. 613–632, 1998.

JOSHI, A.; SCHABES, Y. Tree-adjoining grammars. *Handbook of Formal Languages, Beyond Words*, v. 3, p. 69–123, 1997.

JURAFSKY, D.; MARTIN, J. H. *Speech and Language Processing*. [S.l.: s.n.], 2000.

KLEIN, D.; MANNING, C. Parsing and hypergraphs. In: *Proceedings of the International Workshop on Parsing Technologies*. [S.l.: s.n.], 2001. p. 351–372.

KLEIN, D.; MANNING, C. Accurate unlexicalized parsing. In: *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics-Volume 1*. [S.l.]: Association for Computational Linguistics, 2003. p. 423–430.

KNIGHT, K. Decoding complexity in word-replacement translation models. *Computational Linguistics*, p. 1–10, 1999.

KOEHN, P. et al. Moses: Open source toolkit for statistical machine translation. In: *Proceedings of the 45th Annual Meeting of the ACL on Interactive Poster and Demonstration Sessions*. [S.l.: s.n.], 2007. p. 177–180.

KOEHN, P.; OCH, F. J.; MARCU, D. Statistical phrase-based translation. In: *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology - NAACL '03*. [S.l.: s.n.], 2003. p. 48–54.

LIANG, P.; TASKAR, B.; KLEIN, D. Alignment by agreement. In: *Proceedings of the main conference on Human Language Technology Conference of the North American Chapter of the Association of Computational Linguistics*. [S.l.]: Association for Computational Linguistics, 2006. p. 104–111.

LIU, Y.; LIU, Q.; LIN, S. Tree-to-string alignment template for statistical machine translation. In: *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the ACL - ACL '06*. [S.l.: s.n.], 2006. p. 609–616.

LOPEZ, A. Statistical machine translation. *ACM Computing Surveys*, v. 40, n. 3, p. 1–49, 2008. ISSN 03600300.

MANNING, C. D.; SCHüTZE, H. *Foundations of Statistical Natural Language Processing*. [S.l.: s.n.], 2000. 277–279 p. ISSN 0891-2017.

MARCU, D.; WONG, W. A phrase-based, joint probability model for statistical machine translation. In: *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP-2002)*. [S.l.: s.n.], 2002.

OCH, F. Minimum error rate training in statistical machine translation. In: *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics-Volume 1*. [S.l.]: Association for Computational Linguistics, 2003. p. 160–167.

OCH, F. et al. A smorgasbord of features for statistical machine translation. In: *Proceedings of HLT-NAACL*. [S.l.: s.n.], 2004. p. 161–168.

OCH, F.; NEY, H. A systematic comparison of various statistical alignment models. *Computational linguistics*, MIT Press, v. 29, n. 1, p. 19–51, mar. 2003. ISSN 0891-2017.

OCH, F. J.; NEY, H. The Alignment Template Approach to Statistical Machine Translation. *Computational Linguistics*, v. 30, n. 4, p. 417–449, 2004. ISSN 0891-2017.

PAPINENI, K. et al. Bleu: a method for automatic evaluation of machine translation. In: *ACL*. [S.l.: s.n.], 2001. p. 311–318.

PETROV, S. et al. Learning accurate, compact, and interpretable tree annotation. In: *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics*. Morristown, NJ, USA: Association for Computational Linguistics, 2006. p. 433–440.

PETROV, S.; KLEIN, D. Improved Inference for Unlexicalized Parsing. In: *Proceedings of HLT-NAACL'2007*. [S.l.: s.n.], 2007.

POST, M. *Syntax-based Language Models for Statistical Machine Translation*. Tese (Doutorado) — University of Rochester, 2010.

POST, M.; GILDEA, D. Bayesian learning of a tree substitution grammar. In: *Proceedings of the ACL-IJCNLP 2009 Conference Short Papers*. Morristown, NJ, USA: Association for Computational Linguistics, 2009. p. 45–48.

POST, M.; GILDEA, D. Language modeling with tree substitution grammars. In: *NIPS workshop on Grammar Induction, Representation of Language, and Language Learning*. [S.l.: s.n.], 2009. p. 1–8.

SCHWARTZ, L. et al. Incremental syntactic language models for phrase-based translation. In: *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics, 2011. p. 620–631. Disponível em: <http://www.mt-archive.info/ACL-2011-Schwartz.pdf>.

SILVA, J. a. et al. Out-of-the-Box Robust Parsing of Portuguese. In: *International Conference on Computational Processing of the Portuguese Language - PROPOR 2010*. [S.l.: s.n.], 2010.

SIMA'AN, K. Computational complexity of probabilistic disambiguation by means of tree-grammars. In: *Proceedings of the 16th conference on Computational linguistics*. Morristown, NJ, USA: Association for Computational Linguistics, 1996. p. 1175–1180.

TEH, Y.; JORDAN, M.; BEAL, M. Hierarchical dirichlet processes. *Journal of the American Statistical*, p. 1–30, 2006.

WU, D. Stochastic inversion transduction grammars and bilingual parsing of parallel corpora. *Computational Linguistics*, MIT Press, v. 23, n. 3, p. 377–403, 1997.

YAMADA, K.; KNIGHT, K. A Syntax-based Statistical Translation Model. In: *ACL '01 Proceedings of the 39th Annual Meeting on Association for Computational Linguistics*. [S.l.: s.n.], 2001. p. 523–530.

ZHANG, M. et al. A Tree-to-Tree Alignment-based Model for Statistical Machine Translation. In: *Machine Translation*. [S.l.: s.n.], 2007.

ZHANG, Y.; VOGEL, S.; WAIBEL, A. Interpreting BLEU/NIST scores: How much improvement do we need to have a better system. In: *Proceedings of LREC*. [S.l.: s.n.], 2004.

ZOLLMANN, A.; VENUGOPAL, A. Syntax augmented machine translation via chart parsing. In: *Proceedings of the Workshop on Statistical Machine Translation*. [S.l.]: Association for Computational Linguistics, 2006. p. 138–141.