Learning and Nonlinear Models - Journal of the Brazilian Computational Intelligence Society, Vol. 11, Iss. 1, pp. 11–25, 2013.

TREE-BASED STATISTICAL MACHINE TRANSLATION: EXPERIMENTS WITH THE ENGLISH AND BRAZILIAN PORTUGUESE PAIR

Daniel Beck and Helena Caseli
Computer Science Department
Federal University of São Carlos, UFSCar
{daniel_beck, helenacaseli}@dc.ufscar.br

Resumo – Paradigmas baseados em Aprendizagem de Máquina dominam as pesquisas mais recentes em Tradução Automática. O estado-da-arte é baseado em implementações que dependem apenas de métodos estatísticos que coletam todo o conhecimento necessário de corpora paralelos. No entanto, essa falta de conhecimento linguístico explícito os torna incapazes de modelar alguns fenômenos linguísticos. Neste trabalho, são focados modelos que levam em conta a informação sintática das línguas envolvidas no processo de tradução. É seguida uma proposta recente baseada no preprocessamento de corpora paralelos através de analisadores sintáticos e que usa modelos de tradução compostos por Transdutores de Árvores. São realizados experimentos com o par de línguas Inglês e Português Brasileiro, provendo os primeiros resultados conhecidos em Tradução Automática Estatística baseada em sintaxe para esse par. Os resultados mostram que essa proposta é capaz de modelar mais facilmente fenômenos como reordenamentos de longa distância e fornecem direcionamentos para melhorias futuras na construção de modelos de tradução baseados em sintaxe para esse par.

Palavras-chave – Tradução Automática Estatística, Transdutores de Árvores, Aprendizado de Máquina

Abstract – Machine Learning paradigms have dominated recent research in Machine Translation. Current state-of-the-art approaches rely only on statistical methods that gather all necessary knowledge from parallel corpora. However, this lack on explicit linguistic knowledge makes them unable to model some linguistic phenomena. In this work, we focus on models that take into account the syntactic information from the languages involved on the translation process. We follow a novel approach that preprocess parallel corpora using syntactic parsers and uses translation models composed by Tree Transducers. We perform experiments with English and Brazilian Portuguese, providing the first known results in syntax-based Statistical Machine Translation for this language pair. These results show that this approach is able to better model phenomena like long-distance reordering and give directions to future improvements in building syntax-based translation models for this pair.

Keywords – Statistical Machine Translation, Tree Transducers, Machine Learning

1. Introduction

Statistical Machine Translation (SMT) is the process of translating from one natural language to another one using statistical models and machine learning techniques [1]. In the last twenty years, SMT has become the main research focus in Machine Translation, mainly due to the advent of massive parallel data available in the web and the improvement in computational performance. The idea of SMT is to take advantage of this data to automatically build statistical (language and translation) models that infer the necessary linguistic knowledge to do the translation process.

By improving the statistical models and training algorithms, many advances were obtained in SMT since it was first proposed by [2]. Current state-of-the-art SMT systems implement Phrase-based models (PB-SMT), which use phrases[1] as the translation unit [3, 4]. These models do not use any explicit linguistic knowledge, relying only on the implicit knowledge provided by the corpus. In previous work, [5] performed experiments in PB-SMT between Brazilian Portuguese and both English and Spanish languages. The results presented were promising: in some experiments, the PB-SMT systems outperformed rule-based, hand-made systems.

However, in the last years these advances have been decreasing: purely statistical changes have not brought any significant improvements in translation performance. The following example shows a sentence in English translated to Brazilian Portuguese by a PB-SMT system, along with the reference translation made by a human specialist:

Source sentence: `The oldest poems are translated.`

PB-SMT translation: `O mais antigo poemas são traduzidos.`

Reference translation: `Traduzidos poemas mais antigos.`

[1] A phrase in this context is defined as any sequence of words.

As the example shows, the system was not able to correctly model *long-distance reordering* and the number *agreement* between "antigo" and "poemas". Such problems led to new approaches in SMT intended to combine statistical methods with explicit linguistic knowledge.

Some of these approaches incorporate syntactic information by parsing the parallel data before building the translation model. Usually, the problem is modelled by means of *Synchronous Context-Free Grammars* (SCFGs), which are an extension of Context-Free Grammars with rules that expand symbols in two right-hand sides, one for the source language and another one for the target language. The translation (decoding) process consists in using the grammar to parse the source sentence and retrieve its derivation (sequence of rule applications). Then, the same derivation rules are applied in reverse order to generate the translation in the target language. SCFG-based models appear in the literature under different names such as Syntax-based SMT [6] and Syntax-augmented SMT [7].

Recently, tree-based methods have been used to model syntax in SMT. These methods are able to represent the so called *Tree Languages*, which are more powerful than the string languages represented by the SCFGs. In this work, we focus on implementing these models and investigate how they behave on the English-Brazilian Portuguese (en-ptBR) language pair.

The remaining of this text is structured as follows:

- In Section 2, we describe Tree Languages in more detail and one of the most used formalisms to represent them, the Tree Transducers.
- The methods used to learn translation models based on Tree Transducers are explained in Section 3.
- The SMT model used in this work is defined in Section 4.
- Section 5 tells about the decoding process, namely how the SMT model trained is used to translate new sentences.
- Experiments with the en-ptBR language pair are the focus of Section 6.
- In Section 7 we give concluding remarks and directions for future work.

2. Tree Languages and Tree Transducers

A tree language is defined as a set of trees. Similarly to string languages, they are usually represented by a grammar or an automata that generates and/or recognizes them. In syntax-based SMT, the tree languages of interest are the ones obtained by syntactically parsing the parallel corpus used as training data. These form a subtype called the *Regular Tree Languages* (RTL). RTLs are related to string Context-Free Grammars in the following ways:

- The set of possible derivations of a CFG forms a RTL.
- The set of *yields* of a RTL may be recognized by a CFG. The yield of a tree is the sequence of its leaves from left to right.

The greater modelling power of tree languages comes from the fact that the reciprocal of the first item above is not true: some RTLs cannot be represented by CFGs. Figure 1 shows an example: a CFG able to model these two trees must have the rule $S \rightarrow SS$. Since this rule is recursive, this CFG would result in an infinite number of trees, instead of just the two ones shown on the figure.

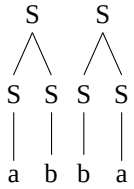


Figure 1: An RTL that can not be modeled by a CFG.

In the context of SMT, theoretically we want to "translate" a tree into another. Those trees are composed by the source and target sentences with its corresponding syntactic analysis. The most widely used formalism to transform trees are the *Tree Transducers* (TTs).

The definition of a TT is analogous to a Finite-State Transducer: while the former transform strings, the latter transform trees. Following the definition proposed by [8], a TT is a quintuple $(\Sigma, \Delta, Q, Q_i, R)$, where:

- Σ is the input alphabet (source language),
- Δ is the output alphabet (target language),
- Q is a set of states,
- $Q_i \in Q$ is the initial state,

- R is a set of rules that maps tree fragments in the input alphabet to fragments in the output alphabet.

Figure 2 shows an example of a transducer able to translate the English sentence shown in Section 1 (*the oldest poems are translated*) to its reference in Brazilian Portuguese (*traduzidos poemas mais antigos*). Each non-terminal symbol is associated with a state, indicating that the rule can only be applied when the transducer is in that state. The TT of figure 2 has only one state ("q"). Theoretically, it could have more states but when TTs are used as translation models they usally have only one.

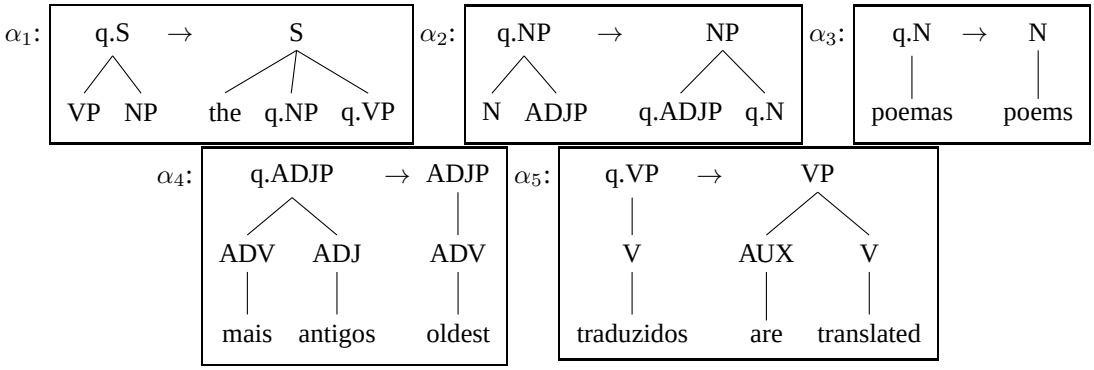


Figure 2: A set of rules in a TT

Rule α_1 on Figure 2 is of great interest in this work since it is able to model long-distance (phrase-level) reordering. While this phenomena is easily represented in a TT, it is prohibitive for phrase-based systems since modeling all possible reorderings is unfeasible due to exponential growth in processing time [9]. Thus, to solve the problem, a PB-SMT system has to constrain these reorderings by limiting the distance between phrases. In TTs, these constraints are motivated by syntactic structure instead of phrase distance.

TTs are commoly used to build translation models when syntactic parsers are available in both languages, an approach known as *Tree-to-Tree*. In this work, we focus on approaches used when syntactic information is available only on one of the languages:

Tree-to-String (TTS): syntactic information is given only on the source language.

String-to-Tree (STT): syntactic information is given only on the target language.

To build translation models for these approaches, we use a similar formalism called the *Tree-to-String* (TTS) transducers.

2.1. Tree-to-String Transducers

While tree transducers have rules with tree fragments on both sides, TTS transducers have rules that map tree fragments to strings. Figure 3 shows an TTS transducer equivalent to the TT of Figure 2. Notice that rule α_5 on the Figure 2 has been broken into two rules. This is due to the fact that TTS transducers only have to follow the syntactic constraints of the source (or target) language instead of both.

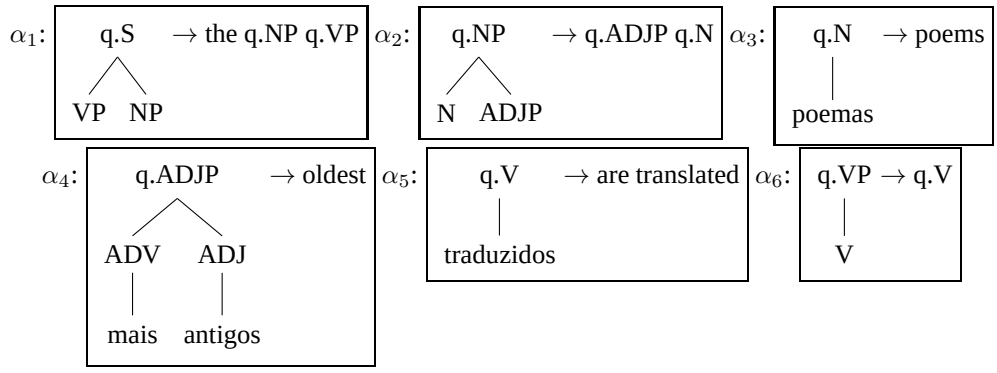


Figure 3: A set of rules in a TTS transducer

When using a TTS transducer in the decoding (translation) process we are not usually interested in retrieving the resulting tree in the target langauge, only the sentence. Because of this, it is possible to transform it into a SCFG and then use the same parsing algorithms used by them in decoding. When the transducer has only one state this transformation is straightforward: the rules are flattened, removing any internal syntactic information. Figure 4 shows the resulting SCFG rules after flattening the transducer of Figure 3.

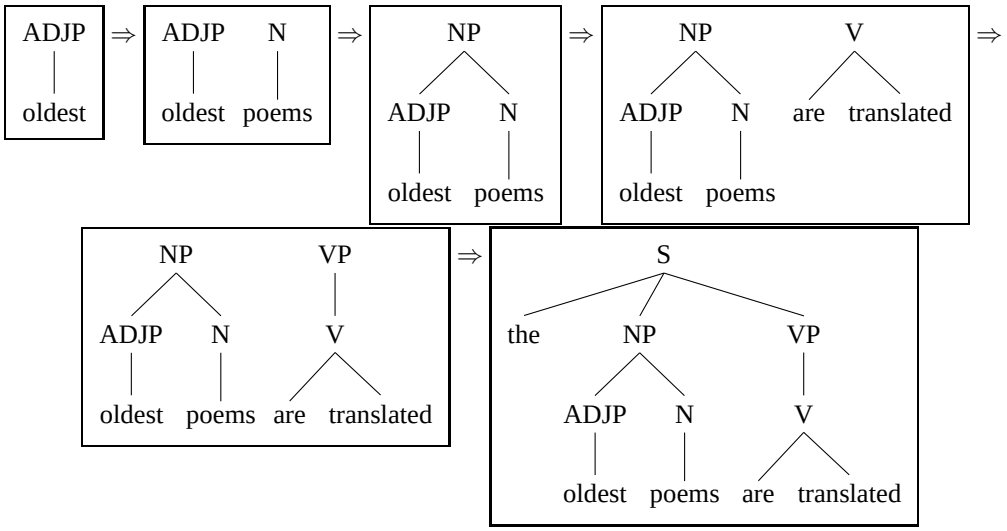
To use the resulting grammar to translate new sentences, we first parse the source sentence and retrieve its derivation (sequence of rule applications). Then, we apply the derivation rules in reverse order to generate the target sentence, as shown on Figure 5.

$r_1 : S \rightarrow \langle VP_1\ NP_2\ , the\ NP_2\ VP_1 \rangle$
$r_2 : NP \rightarrow \langle N_1\ ADJP_2\ ,\ ADJP_2\ N_1 \rangle$
$r_3 : N \rightarrow \langle poemas\ ,\ poems \rangle$
$r_4 : ADJP \rightarrow \langle mais\ antigos\ ,\ oldest \rangle$
$r_5 : V \rightarrow \langle traduzidos\ ,\ are\ translated \rangle$
$r_6 : VP \rightarrow \langle V_1\ ,\ V_1 \rangle$

Figure 4: Resulting SCFG after transducer flattening

English sentence analysis:

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



Brazilian Portuguese sentence generation:

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

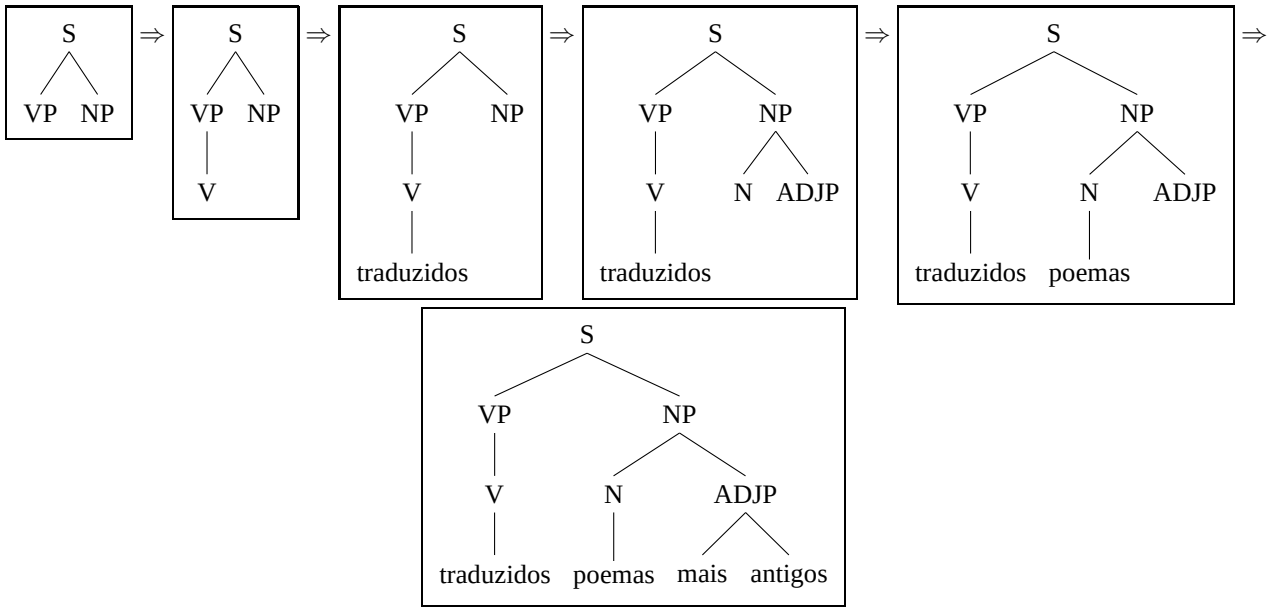


Figure 5: Translation process using a SCFG

3. Transducer Training

In SMT, a parallel corpus is used as input data for building translation models. This corpus is made by a set of sentences in the source language with its corresponding translations in the target language. In most SMT models (including phrase-based ones), a preprocessing step called *lexical alignment* is done before the actual training procedure. This alignment aims to indicate the correspondences between words in each sentence pair. Figure 6 shows an example of a sentence pair that is lexically aligned.

To train a translation model from this preprocessed corpus, it is necessary to 1) extract translation rules and 2) infer their probabilities. In phrase-based models, the rules are extracted straight from the lexical alignment information. For example, from the alignment shown on Figure 6, it is possible to extract a rule translating the word “oldest” into the phrase “mais antigos”. The probabilities are usually calculated by getting the relative frequencies of each rule.

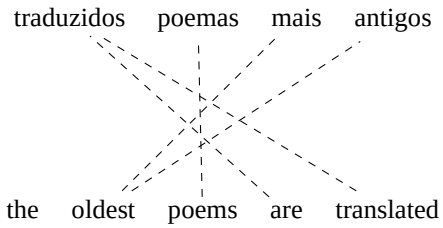


Figure 6: Lexical alignment example

In syntax-based SMT, another corpus preprocessing step takes place: the source language half (TTS models) or the target language half (STT models) is syntactically parsed[2]. This result in a set of structures named *alignment graphs*. As shown in Figure 7, an alignment graph is composed by a lexically aligned sentence pair with one of the sentences annotated with its syntactic tree.

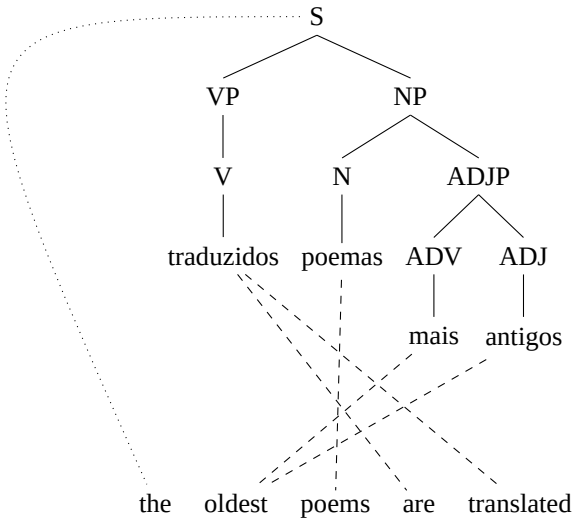


Figure 7: Alignment graph for the pair of sentences in Section 1

3.1. GHKM

To extract TTS rules from the alignment graphs, we use the GHKM algorithm [10]. For each graph in the parallel corpus, GHKM executes the following steps:

1. For each non-terminal symbol in the graph, annotate its *span*. The span is defined as the set of reachable terminals in the target sentence. By definition, non-aligned target terminals are aligned to the graph root symbol.
2. For each non-terminal symbol in the graph, annotate its *complementary span*. The complementary span is defined as the union of the parent complementary span and each sibling span.
3. Non-terminals with null intersection between its span and complementary span are marked as *frontier nodes*.
4. Rules are then extracted from *frontier graphs*, which are non-trivial subgraphs where all its source and sink nodes are frontier nodes or source terminals.

The alignment graph on Figure 8 shows each non-terminal annotated with its span and frontier nodes are marked in bold. Only the ADV and ADJ non-terminals are not frontier since their spans overlap (both reach the same target word “oldest”). The rules extracted from this graph have already been shown in Figure 3.

To obtain the rule probabilities we need to infer the *best* TTS transducer according to the corpus. In other words, we need to infer the transducer $\hat{T}$ that maximizes $P(T|C)$, where C is the input corpus (composed by the alignment graphs). This term is expanded using Bayes’ Rule, resulting in the following formula:

$$\hat{T} = \underset{T}{argmax}\, P(T|C) = \underset{T}{argmax}\, \frac{P(C|T) \times P(T)}{P(C)} \tag{1}$$

[2]In Tree-to-Tree models, both halves are parsed.

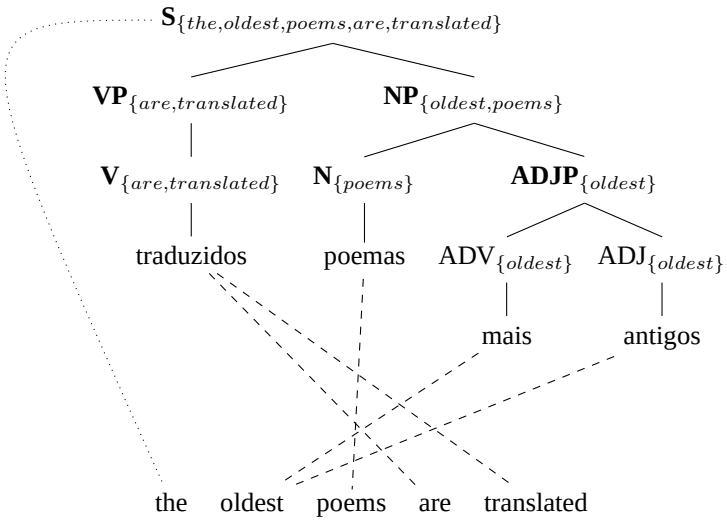


Figure 8: Alignment graph with spans annotated

The term $P(C)$ is constant because the input corpus is always the same. Since it is an maximization formula, it can be discarded:

$$\hat{T} = \underset{T}{argmax}\, P(T|C) = \underset{T}{argmax}\, P(C|T) \times P(T) \tag{2}$$

This results in a *generative process*, where we suppose that a transducer is given with probability $P(T)$ and *generates* the corpus with probability $P(C|T)$. The terms $P(T|C)$, $P(C|T)$ and $P(T)$ are named the *posterior*, the *likelihood* and the *prior*, respectively. When we do not want to make any assumptions about the best transducer we model the prior probability as an uniform distribuition. Therefore, the term $P(T)$ can also be discarded:

$$\hat{T} = \underset{T}{argmax}\, P(T|C) = \underset{T}{argmax}\, P(C|T) \tag{3}$$

The equation above corresponds to the *Maximum Likelihood Estimate* (MLE) statistical method. In GHKM, it can be proved that to obtain the best transducer according to MLE it is enough to simply calculate relative rule frequencies according to its root[3], similar to what PB-SMT uses. This is due to the fact that extracts only the *minimal rules* from the alignment graphs and therefore only one derivation is considered for all alignment graphs.

While it is possible to build a translation model using only the minimal rules extracted by GHKM, these rules tend to capture small contexts in the syntactic structure. To build robust models, bigger rules should be taken into account. Theoretically, the best procedure would extract all rules from the corpus and then infer their probabilities. But it is not feasible to extract all possible rules because they grow exponentially in graph size. Because of this, heuristics are used to limit rule sizes.

To tackle the above problem, [11] expands GHKM in the following way: after extracting minimal rules, a subset containing only n terminal symbols on the left side is combined between them, generating bigger rules. With this new rule set, it becomes impossible to calculate MLE using only the relative frequencies because these new rules result in more than one derivation for the alignment graphs. Instead, the Expectation-Maximization (EM) algorithm is applied to obtain the probabilities. The experiments described in their work used 3 and 4 as values for n. This restriction on rule size permits EM to run in polynomial time.

The main issue with the EM approach is that it tends to overfit the training corpus. Since alignment graphs probabilites are calculated by a product of rule weights EM tends to give higher weights to bigger rules, degenerating the transducer. In light of this problem, [12] proposes another method, where instead of inferring the best transducer $\hat{T}$, they infer the best *sequence of derivations* $\hat{S}$ used in the generative corpus building process, modifying Equation 2 in the following way:

$$\hat{S} = \underset{S}{argmax}\, P(S|C) = \underset{S}{argmax}\, P(C|S) \times P(S) \tag{4}$$

The difference between $\hat{T}$ and $\hat{S}$ is crucial: while for a given transducer T many corpora could be generated, for a given sequence S, only one corpus C is possible. If you change S even in the slightest way (like swapping two rules in the sequence, for example), then the corpus C would be different. The result of this difference is that the likelihood $P(C|S)$ can only be 1 (when the corpus is composed by the sequence) or 0 (when it is not). So, the inferring process is all done by the prior $P(S)$, which is modelled as a *Dirichlet Process (DP)* instead of a uniform distribution.

[3]We define the root of a TTS rule as the root symbol of the rule tree fragment.

3.2. Dirichlet Process

The Dirichlet Process (DP) is defined as a distribution over the infinite space of possible TTS rules[4]. This distribution has the property to take into account all TTS rules used in the generative procees. Formally, the distribution of a rule according to its root r is defined as:

$$\begin{aligned} rule|r &\sim G_{root} \\ G_r|\alpha_r, P_0 &\sim DP(\alpha_r, P_0(\cdot|r)) \end{aligned} \tag{5}$$

where $P_0(\cdot|r)$ (the *base distribution*) is a distribution over the infinite rule set with r as its root and α_r is called the *concentration parameter* of r. Intuitively, the base distribution defines what rules are used in the generative process while the concentration parameter controls the trend to create new rules or reusing existing ones.

Following the procedure used by [12], instead of sampling straight from the DP distribution, we integrate over all possible rule sequences, obtaining a conditional distribution defining the probability of the next rule be generated. Then, the probability of $rule_i$ (i being the position of this rule in the sequence) according to its root r_i is defined as:

$$P(rule_i|\vec{rule}_{<i}, r_i, \alpha_{r_i}, P_0) = \frac{count(rule_i) + \alpha_{r_i} P_0(rule_i|r_i)}{count(r_i) + \alpha_{r_i}} \tag{6}$$

where $\vec{rule}_{<i}$ is the set of all other rules used in the generative process until then, $count(rule_i)$ is the total of times that $rule_i$ shows up in $\vec{rule}_{<i}$ and $count(r_i)$ is the total of times that a rule with root r_i shows up in $\vec{rule}_{<i}$. If α_{r_i} is equal to zero, the formula becomes the relative frequency of the rule. Because of this, intuitively this formula can be understood as a relative frequency that takes into account the base distribution.

Another way to understand the DP is to consider it as a *cache* model, where everytime a new rule is generated, the model chooses one from the cache of already existent rules or creates a new one using the base distribution. As the corpus is generated, the cache becomes bigger and the model tends to choose one from it. This is a phenomenon that actually occurs in natural language: while it is always possible to use new syntactic structures we tend to use already existent ones.

The base distribution defines what kind of rules will appear in the resulting transducer. Since bigger rules tend to overfit the corpus, this distribution is defined in a way that it gives greater probabilites to smaller rules. To achieve this, it is modelled as another generative process in the following way:

- Tree (left side) probability:
  1. If the node is not a *Part-of-Speech* (POS) tag, expand it into n children, where n is sampled from a geometric distribution with parameter β_{child}. If the node is a POS tag, expand it deterministically into 1 child.
  2. Choose the generated symbol according to a uniform distribution over all symbols (non-terminals and terminals).
  3. For each non-terminal generated, choose to expand it or not, according to a parameter β_{expand}. Return to step 1 for each expanded symbol.
- String (right side) probability:
  1. Generate m terminal symbols, where m is sampled from a geometric distribution with parameter β_{term}.
  2. Generate each terminal symbol according to a uniform distribution over all terminal symbols.
  3. Generate the string non-terminals one at a time, putting it in one of the available positions according to a uniform distribution over all available positions.

Rule probability is then calculated by multiplying the tree and string probabilites. Figure 9 shows an example of how this calculation is made.

3.3. Gibbs sampling

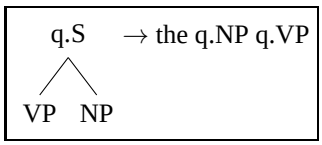
Using a DP prior, we could apply EM to maximize Equation 4. But this would require the enumeration of all possible derivations and, as explained in Section 3.1, this is unfeasible due to exponential growth. The solution proposed by [12] is to sample from the possible derivations using a procedure called *Gibbs sampling*.

The Gibbs sampling works by changing one of the model *variables* while keeping the other variables constant. In the transducer inference problem, the variables are the node spans defined by GHKM. The idea is to iteratively change those spans, aiming to maximize Equation 4 and find the best transducer.

The sampler visits each frontier node in each alignment graph in the corpus in random order. For each of these nodes, it finds what are all its possible spans and choose one of them to be the new span (it can be the same as before). This choice is made in proportion to the resulting posteriors of each span. To find which are the possible spans for a given node, some rules are considered:

[4]For a generalized explanation about the DP, we refer the reader to Appendix A in [13].

TTS rule:



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

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

Figure 9: Example of rule probability calculation according to a base distribution. N is the number of non-terminal symbols and T' is the number of terminal symbols. In this example they are arbitrarily set to 5 and 10, respectively. In a realistic setting, N would be defined by the symbols used in the syntactic parser and T' by the languages' vocabulary.

1. The root node span must cover all words in the target sentence.
2. The span of a node must be a subspan of his parent node.
3. The span of a node must contain all span of its children.
4. Spans of sibling nodes can not overlap.

Figure 10 shows an example where the sampler is visiting the node NP and evaluating its possible spans (in this case, there are two of them). Each one defines two different TTS rules, shown on Figure 11. The sampler then defines which span will be chosen according to the posterior of each resulting transducer.

In theory, to calculate each posterior it would be necessary to take into account the probabilities of each rule in all the alignment graphs. But the sampler does not need the true posterior values, only their *proportion*. So, the sampler only uses the probabilities of the rules inferred by the possible spans, since all the other rules do not change. In the example shown on Figure 10, the sampler chooses the span according to the following values:

$$\begin{aligned} P(span = span_1) &= P(r_{11}) \times P(r_{12}) \\ P(span = span_2) &= P(r_{21} \times P(r_{22}) \end{aligned} \quad (7)$$

where:

- $span_1$ and $span_2$ are the two possible *spans* for the node, shown on Figure 10.
- r_{11} and r_{12} are the two TTS rules implied by $span_1$, shown on Figure 11.
- r_{21} and r_{22} are the two TTS rules implied by $span_2$, also shown on Figure 11.

4. SMT model

To use a trained TTS transducer to translate new sentences we need to insert it into a SMT model. In SMT, the translation task is defined as an optimization process: the goal is to find the best target translation $\hat{t}$ that maximizes the probability $P(t|s)$, where s is the source sentence. Following previous works in discriminative SMT modeling [4, 14], we use a log-linear model to define $P(t|s)$:

$$\hat{t} = \underset{t}{argmax}\ P(t|s) = \underset{t}{argmax}\ exp \sum_{k=1}^{K} \lambda_k h_k(t, s) \quad (8)$$

In this model, the $h_k(t, s)$ terms represent *feature functions* that may take into account: the source sentence, the target candidate translation or both. The λ_k terms in equation 8 are the functions *coefficients*. In our experiments, we use the following feature functions (similar to the ones used by [14]):

- $P(s|t)$: translation model, based on a TTS transducer.
- $P(t)$: a ngram-based target language model used to determine the fluency of the candidate translation. Ngram-based language models are considered state-of-the-art for SMT [1].
- $length(t)$: the length of the target candidate translation.
- $rules(t, s)$: the total number of rules used when translating the sentence.

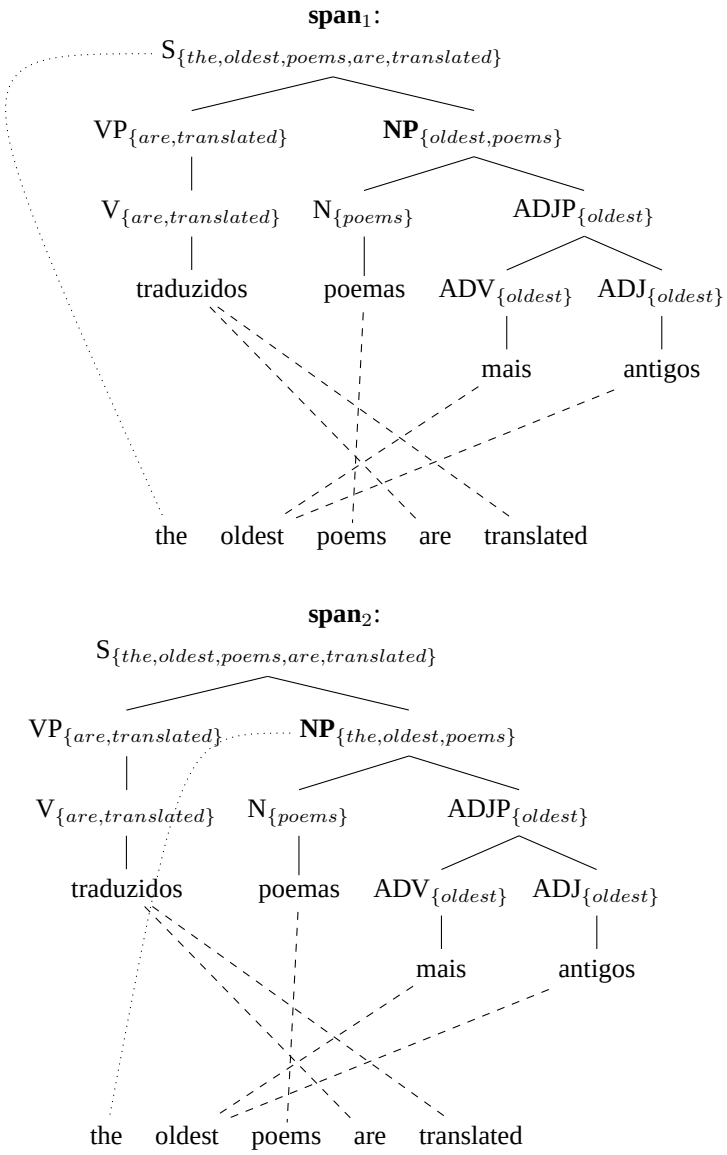


Figure 10: Possible spans for a node visited by the Gibbs sampler, in this case the NP node. The second span implies an alignment between the node and the word "the" which before was aligned with the root node, according to GHKM output.

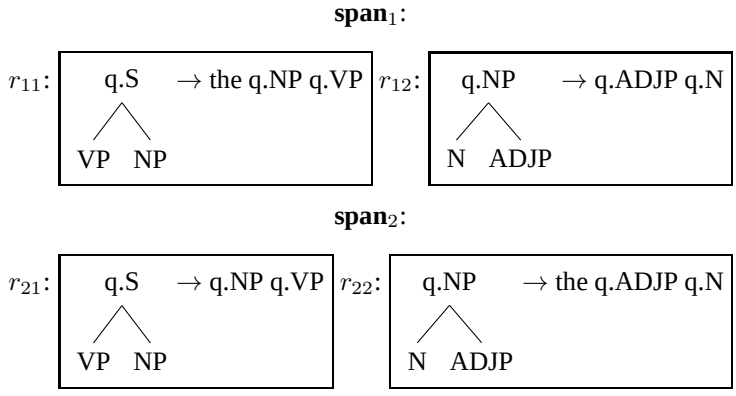


Figure 11: TTS rules implied by the spans of Figure 10

- $lex(t|s)$: lexical weight with respect to the target sentence.
- $lex(s|t)$: lexical weight with respect to the source sentence.

The lexical weight is a feature function which tries to model how good each word in a TTS rule translates into its corresponding

string in the rule. It is defined by the following formula:

$$lex(t|s) = \frac{count(t, s)}{\sum_{t'} count(t', s)} \tag{9}$$

where t' means every possible target word aligned with s. The count values are gathered from the lexical alignment in the training corpus.

To obtain the λ_k values we use Minimum Error-Rate Training (MERT) [15]. The MERT algorithm uses a validation corpus (which should be different from the training corpus) with reference translations and infer λ_k values that minimizes a given error metric. For our experiments, we use BLEU [16] as the error metric.

5. Decoding

In the decoding process we flatten the TTS rules and transform it into a SCFG, as explained in Section 2.1. If the SMT model is composed only by the translation model, it is possible to translate new sentences by just parsing the input sentence using the best derivation. But our model is composed by several feature functions which should be considered when finding the best translation. In this case, it is easier for the decoder to first finds all possible derivations for the input sentence according to the translation model, building a *translation forest* or *hypergraph* [17, 18].

A translation forest is a compact representation of all possible derivations for a source sentence. Each rule application is represented by an *hyperedge*, which is an edge that starts in many nodes (the *tail* nodes) and ends in one node (the *head* node). Since some derivations have common rules, these are "compressed" into one hyperedge, as shown on Figure 12.

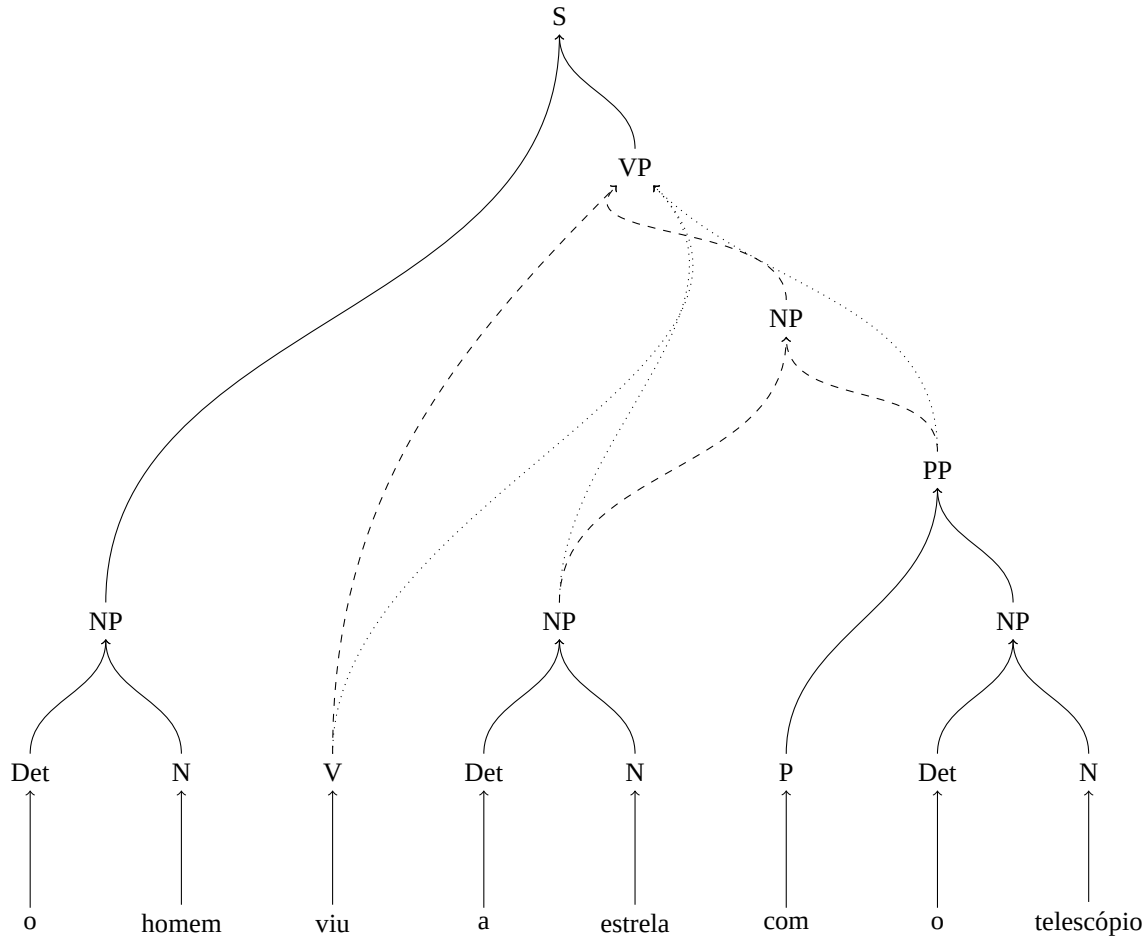


Figure 12: A translation forest representing two possible derivations. The dotted hyperedge shows one derivation while the dashed hyperedge shows another one. Solid hyperedges corresponds to rules common to both derivations.

Each node in a translation forest has a probability. The decoder then *rescores* each node according to the log-linear model. If it is a head node of an hyperedge, the calculation also take into account the probabilities of the hyperedge tail nodes. When there is more than one hyperedge, only the bigger value is considered, which corresponds to best derivation until then[5]. Each node is rescored by the decoder in a bottom-up process. When it reaches the root, the best derivation is obtained and the target sentence is generated.

[5]This process is analogous to the Viterbi algorithm used to obtain the best derivation.

The forest building process is relatively fast ($O(n^3)$, where n is the source sentence size) but the rescoring process tends to be very slow due to the additional features used in the log-linear model. To tackle this issue, a number of heuristics are used to prune the forest when the rescoring process takes place. The most used are the *beam search* [3] and the *cube pruning* [19].

6. Experiments

To evaluate how TTS transducers behave in *en-ptBR* and *ptBR-en* translation, four experiments were performed using the PesquisaFAPESP corpus[6]. We used the corpus version 1, composed by 646 articles and 17.397 sentence pairs in English and Brazilian Portuguese[7]. For training, validation (via MERT) and testing purposes, we broke the corpus in sets of 80%, 10%, 10%, respectively.

We also used the additional tools:

- Berkeley Parser[8] [20, 21] and LX-Parser[9] [22] for parsing the *en* and *ptBR* training sentences, respectively.
- Berkeley Aligner[10] [23] for corpus lexical alignment.
- Moses[11] toolkit [24] for experiments with PB-SMT models and cdec[12] toolkit [25] for decoding with GHKM models.
- SRILM[13] [26] for building the ngram-based language models.
- BLEU [16] and NIST [27] metrics to evaluate the results and bootstrapping tool[14] [28] to assess statistical difference. All the results presented in this paper are statistical significant considering a confidence level of 95%.

6.1. GHKM vs. PB-SMT

We first compared the models generated by GHKM with PB-SMT models. The results, presented in Table 1 show that GHKM models are outperformed by current state-to-the-art models in both language pair directions. This decrease in performance was mainly due to spurious phrase reorderings made by GHKM, as shown in the following example, where the phrase "vários fatores" was wrongly reordered to the middle of the sentence:

Source sentence: several factors contribute towards the complexity of the process , in any language .

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

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

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

Table 1: Results from comparing GHKM and PB-SMT models

	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

[6] PesquisaFAPESP corpus is composed by a selection of articles from the Pesquisa FAPESP scientific magazine (http://revistapesquisa.fapesp.br/)
[7] Available at www.nilc.icmc.usp.br/lacioweb
[8] Available at code.google.com/p/berkeleyparser
[9] Available at lxcenter.di.fc.ul.pt/tools/pt/conteudo/LXParser.html
[10] Available at code.google.com/p/berkeleyaligner
[11] Available at www.statmt.org/moses
[12] Available at cdec-decoder.org
[13] Available at www.speech.sri.com/projects/srilm
[14] Available at projectile.sv.cmu.edu/research

6.2. TTS vs. STT

In Table 1, the BLEU and NIST scores are very similar for both language pair directions when using PB-SMT models but very different when using GHKM models: scores for *ptBR-en* are much lower. Since the main difference between each direction is the parser used (Berkeley for *en-ptBR* and LX-Parser for *ptBR-en*) we also made experiments using GHKM to build STT models and therefore changing the parser used in each pair.

Results presented in Table 2 confirm that the parser indeed has a serious impact on the model performance: scores for *en-ptBR* were much lower when using the LX-Parser. The example below shows the same sentence from the first experiment but evaluated in respect to the TTS or STT approach:

Source sentence: `several factors contribute towards the complexity of the process , in any language .`

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

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

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

The transducer generated by the STT approach in this case has spurious rules (like the one which added the word “brasil”) and lower lexical coverage (observed by the untranslated phrase “in any language”).

Table 2: Results from comparing TTS and STT approaches

	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

6.3. Language model influence

The previous experiments for the *en-ptBR* direction used the *ptBR* portion of PesquisaFAPESP for training the language model. To investigate how more robust language models improve GHKM results, we also performed experiments using the CETENFolha corpus[15]. This corpus is composed of 1.597.807 Brazilian Portuguese sentences extracted from “Folha de São Paulo” newspaper.

Table 3 brings the scores obtained when using each language model in GHKM and PB-SMT. As expected, both translation models improved when using CETENFolha to train the language model but the improvement in GHKM is larger than the one for PB-SMT. We credit this to the reordering power of GHKM, since it allows a more diverse set of candidate translations for language model disambiguation, unlike PB-SMT models.

Table 3: Results from comparing PesquisaFAPESP and CETENFolha as language models

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

The bigger language model provided by CETENFolha helped to prevent the spurious reorderings occurred in the first experiment. Using the same example shown in Section 6.1, we notice that in this case the phrase “vários fatores” was not reordered:

Source sentence: `several factors contribute towards the complexity of the process , in any language .`

[15] Available at `www.linguateca.pt/cetenfolha`

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

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

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

6.4. GHKM vc. Gibbs sampling

Finally, we also compared the models extracted from plain GHKM and the one with spans modified by a Gibbs sampler. Our hypothesis is that running the sampler results in better translation models. In this experiment, we used a subset of the training corpus with 40 words or less[16]. This resulted in a corpus with about 10300 sentences.

The parameters used in our experiments are the same used by [12]: the concentration parameter α was set to 100000, the base distribution parameters β_{expand}, β_{child} and β_{term} were all set to 0.5 and 300 iterations were made by the sampler. The last iteration was used to extract the TTS rules.

The results shown on Table 4 confirms our hypothesis for the en-ptBR direction. In the case of the ptBR-en direction, the BLEU scores were higher but the NIST scores were lower. Although both differences are statistically significant, it is not possible to infer conclusions because of this divergence on the scores.

Table 4: Results from comparing GHKM and Gibbs sampling algorithms

	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

Comparing the hypotheses generated by plain GHKM and the Gibbs sampler for the en-ptBR direction, we noted that the rules extracted by the sampler had more discerning power of when to do a long-distance reordering. The sentence below shows an example of a spurious reordering made by plain GHKM that was avoided by the Gibbs sampler.

Source sentence: `precision in diagnosis`

Reference: `precisão em o diagnóstico`

GHKM hypothesis: `em o diagnóstico de a precisão`

Gibbs hypothesis: `a precisão de o diagnóstico`

6.5. Discussion

Our experiments have given evidence that GHKM models are able to do long-distance reorderings that the PB-SMT models were unable to do, as shown on the example sentences given in the subsections above. The lower scores obtained when comparing the GHKM and the state-of-the-art models show that the former are actually doing *too many* reorderings, even when they are not necessary. We believe that the GHKM models by itselves do not take into account enough context to prevent those spurious reorderings, even though TTS transducers have the power to model these contexts. We propose two solutions to tackle this problem:

- Improve the GHKM algorithm by changing the alignment graphs before rule extraction. We use a generative model based on a Dirichlet Process and a Gibbs sampler to modify the graphs and extract rules with bigger contexts. We have shown that better models are achieved this way for the en-ptBR direction but results are still unconclusive for the ptBR-en direction.
- Use language models trained on bigger corpora. One possible cause of the spurious reorderings made by the GHKM models is its ability to generate a more diverse set of hypothesis in the decoding process when comparing to the state-of-the-art. More robust language models can help to alleviate this issue by having more power to select correct reorderings.

Another conclusion obtained by our experiments is that the syntactic parser used when preprocessing the parallel corpus have a large influence in translation performance. There have been some discussion in the literature about which approach (TTS or STT) is better when building translation models and our results show that this decision should take into account the available parsers for the language pair of choice. For the en-ptBR language pair, current state-of-the-art parsers for English tend to give better translation models in either translation direction.

[16]This restriction was due to the Gibbs sampler implementation used

7. Conclusions and Future Work

In this paper we presented the first experiments carried out on TTS and STT translation between English and Brazilian Portuguese using tree transducers. The results obtained show that those models have potential to improve the state-of-the-art since they are able to model long-distance reordering between phrases. Better translation and language models may be able to deal with the spurious reorderings issue encountered.

The analysis shown in this paper leads to interesting future research directions. In particular, we believe that pursuing the following points may lead to substantial improvements:

Parser influence: Since the syntactic parser has a large influence in translation performance, it could be an interesting work to investigate in deeper which parser features improve the resulting translation models. This investigation could lead to modified or specialized parsers for SMT.

Refined features: There is a current tendency in SMT for purely discriminative models. [29] presents a model with more than 10.000 fine-grained features. Another research direction is to combine this approach with the use of syntactic information.

Bigger corpora: Recently, [30] published a new version of PesquisaFAPESP, with more than 180.000 sentences. Redoing the experiments shown in this paper with this new version is a natural extension of this work.

8. Acknowledgments

We thank the support of CAPES and FAPESP (projects 2010/03807-4 and 2010/07517-0).

REFERÊNCIAS

[1] A. Lopez. “Statistical machine translation”. *ACM Computing Surveys*, vol. 40, no. 3, pp. 1–49, 2008.

[2] P. Brown, J. Cocke, S. Pietra and V. Pietra. “A statistical approach to machine translation”. *Computational Linguistics*, vol. 16, no. 2, pp. 79–85, 1990.

[3] P. Koehn, F. J. Och and D. Marcu. “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*, pp. 48–54, 2003.

[4] F. J. Och and H. Ney. “The Alignment Template Approach to Statistical Machine Translation”. *Computational Linguistics*, vol. 30, no. 4, pp. 417–449, 2004.

[5] W. F. Aziz, T. A. S. Pardo and I. Paraboni. “Statistical phrase-based machine translation: Experiments with Brazilian Portuguese”. In *Encontro Nacional de Inteligência Artificial (ENIA-2009)*, 2009.

[6] K. Yamada and K. Knight. “A Syntax-based Statistical Translation Model”. In *ACL ’01 Proceedings of the 39th Annual Meeting on Association for Computational Linguistics*, pp. 523–530, 2001.

[7] A. Zollmann and A. Venugopal. “Syntax augmented machine translation via chart parsing”. In *Proceedings of the Workshop on Statistical Machine Translation*, pp. 138–141. Association for Computational Linguistics, 2006.

[8] J. Graehl, K. Knight and J. May. “Training Tree Transducers”. *Computational Linguistics*, vol. 34, pp. 391–427, 2008.

[9] K. Knight. “Decoding complexity in word-replacement translation models”. *Computational Linguistics*, pp. 1–10, 1999.

[10] M. Galley, M. Hopkins, K. Knight and D. Marcu. “What’s in a translation rule?” In *Proceedings of the Human Language Technology and North American Association for Computational Linguistics Conference (HLT/NAACL 2004)*, pp. 273–280, 2004.

[11] M. Galley, J. Graehl, K. Knight, D. Marcu, S. DeNeefe, W. Wang and I. Thayer. “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*, pp. 961–968, 2006.

[12] T. Cohn and P. Blunsom. “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*, pp. 352–361, Morristown, NJ, USA, 2009. Association for Computational Linguistics.

[13] S. Goldwater, T. L. Griffiths and M. Johnson. “A Bayesian framework for word segmentation: Exploring the effects of context”. *Cognition*, vol. 112, no. 1, pp. 21–54, July 2009.

[14] Y. Liu, Q. Liu and S. Lin. “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*, pp. 609–616, 2006.

[15] F. Och. "Minimum error rate training in statistical machine translation". In *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics-Volume 1*, number July, pp. 160–167. Association for Computational Linguistics, 2003.

[16] K. Papineni, S. Roukos, T. Ward and W.-J. Zhu. "Bleu: a method for automatic evaluation of machine translation". In *ACL*, pp. 311–318, 2001.

[17] D. Klein and C. Manning. "Parsing and hypergraphs". In *Proceedings of the International Workshop on Parsing Technologies*, pp. 351–372, 2001.

[18] L. Huang and D. Chiang. "Better k-best parsing". In *Proceedings of the Ninth International Workshop on Parsing Technology*, pp. 53–64. Association for Computational Linguistics, 2005.

[19] D. Chiang. "Hierarchical Phrase-Based Translation". *Computational Linguistics*, vol. 33, no. 2, pp. 201–228, June 2007.

[20] S. Petrov, L. Barrett, R. Thibaux and D. Klein. "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*, number July, pp. 433–440, Morristown, NJ, USA, 2006. Association for Computational Linguistics.

[21] S. Petrov and D. Klein. "Improved Inference for Unlexicalized Parsing". In *Proceedings of HLT-NAACL'2007*, 2007.

[22] J. a. Silva, A. Branco, S. Castro and R. Reis. "Out-of-the-Box Robust Parsing of Portuguese". In *International Conference on Computational Processing of the Portuguese Language - PROPOR 2010*, 2010.

[23] P. Liang, B. Taskar and D. Klein. "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*, pp. 104–111. Association for Computational Linguistics, 2006.

[24] P. Koehn, H. Hoang, A. Birch, C. Callison-Burch, M. Federico, N. Bertoldi, B. Cowan, W. Shen, C. Moran, R. Zens, C. Dyer, O. Bojar, A. Constantin and E. Herbst. "Moses: Open source toolkit for statistical machine translation". In *Proceedings of the 45th Annual Meeting of the ACL on Interactive Poster and Demonstration Sessions*, pp. 177–180, 2007.

[25] C. Dyer, J. Weese, H. Setiawan, A. Lopez, F. Ture, V. Eidelman, J. Ganitkevitch, P. Blunsom and P. Resnik. "cdec: A decoder, alignment, and learning framework for finite-state and context-free translation models". In *Proceedings of the ACL 2010 System Demonstrations*, pp. 7–12. Association for Computational Linguistics, 2010.

[26] A. Stolcke. "SRILM - An Extensible Language Modeling Toolkit". In *Proceedings of the International Conference on Spoken Language Processing*, pp. 901–904, 2002.

[27] G. Doddington. "Automatic evaluation of machine translation quality using n-gram co-occurrence statistics". In *Proceedings of the Second International Conference on Human Language Technology Research*, pp. 128–132, 2002.

[28] Y. Zhang, S. Vogel and A. Waibel. "Interpreting BLEU/NIST scores: How much improvement do we need to have a better system". In *Proceedings of LREC*, 2004.

[29] D. Chiang, K. Knight and W. Wang. "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*, pp. 218–226. Association for Computational Linguistics, 2009.

[30] W. Aziz and L. Specia. "Fully Automatic Compilation of Portuguese-English and Portuguese-Spanish Parallel Corpora". In *8th Brazilian Symposium in Information and Human Language Technology (STIL-2011)*, 2011.