Heuristic Search for Non-Bottom-Up Tree Structure Prediction

Andrea Gesmundo
Department of Computer Science
University of Geneva
andrea.gesmundo@unige.ch

James Henderson
Department of Computer Science
University of Geneva
james.henderson@unige.ch

Abstract

State of the art Tree Structures Prediction techniques rely on bottom-up decoding. These approaches allow the use of context-free features and bottom-up features. We discuss the limitations of mainstream techniques in solving common Natural Language Processing tasks. Then we devise a new framework that goes beyond Bottom-up Decoding, and that allows a better integration of contextual features. Furthermore we design a system that addresses these issues and we test it on Hierarchical Machine Translation, a well known tree structure prediction problem. The structure of the proposed system allows the incorporation of non-bottom-up features and relies on a more sophisticated decoding approach. We show that the proposed approach can find better translations using a smaller portion of the search space.

1 Introduction

Tree Structure Prediction (TSP) techniques have become relevant in many Natural Language Processing (NLP) applications, such as Syntactic Parsing, Semantic Role Labeling and Hierarchical Machine Translation (HMT) (Chiang, 2007). HMT approaches have a higher complexity than Phrase-Based Machine Translation techniques, but exploit a more sophisticated reordering model, and can produce translations with higher Syntactic-Semantic quality.

TSP requires as inputs: a weighted grammar, $\mathbf{G}$, and a sequence of symbols or a set of sequences encoded as a Lattice (Chappelier et al., 1999). The input sequence is often a sentence for NLP applications. Tree structures generating the input sequence can be composed using rules, r, from the weighted grammar, $\mathbf{G}$. TSP techniques return as output a tree structure or a set of trees (forest) that generate the input string or lattice. The output forest can be represented compactly as a weighted hypergraph (Klein and Manning, 2001). TSP tasks require finding the tree, $\mathbf{t}$, with the highest score, or the best-k such trees. Mainstream TSP relies on Bottom-up Decoding (BD) techniques.

With this paper we propose a new framework as a generalization of the CKY-like Bottom-up approach. We also design and test an instantiation of this framework, empirically showing that wider contextual information leads to higher accuracy for TSP tasks that rely on non-local features, like HMT.

2 Beyond Bottom-up Decoding

TSP decoding requires scoring candidate trees, $\mathsf{cost}(\mathbf{t})$. Some TSP tasks require only local features. For these cases $\mathsf{cost}(\mathbf{t})$ depends only on the local score of the rules that compose $\mathbf{t}$:

$$\mathsf{cost}(\mathbf{t}) = \sum_{r_i \in \mathbf{t}} \mathsf{cost}(r_i) \qquad (1)$$

This is the case for Context Free Grammars. More complex tasks need non-local features. Those features can be represented by a non-local factor, $\mathsf{nonLocal}(\mathbf{t})$, into the overall $\mathbf{t}$ score:

$$\mathsf{cost}(\mathbf{t}) = \sum_{r_i \in \mathbf{t}} \mathsf{cost}(r_i) + \mathsf{nonLocal}(\mathbf{t}) \qquad (2)$$

Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing, pages 899–908,
Edinburgh, Scotland, UK, July 27–31, 2011.

For example, in HMT the Language Model (LM) is a non-local fundamental feature that approximates the adequacy of the translation with the sum of log-probabilities of composing n-grams.

CKY-like BD approaches build candidate trees in a bottom-up fashion, allowing the use of Dynamic Programming techniques to simplify the search space by mering sub-trees with the same state, and also easing application of pruning techniques (such as Cube Pruning, e.g. Chiang (2007), Gesmundo (2010)). For clarity of presentation and following HMT practice, we will henceforth restrict our focus to binary grammars. Standard CKY works by building objects known as $items$ (Hopkins and Langmead, 2009). Each item, ι, corresponds to a candidate sub-tree. Items are built linking a rule instantiation, r, to two sub-items that represents left context, ι_1, and right context, ι_2; formally: $\iota \equiv \langle\ \iota_1 \succ r \prec \iota_2\ \rangle$. An item is a triple that contains a $span$, a $postcondition$ and a $carry$. The span contains the indexes of the starting and ending input words delimiting the continuous sequence covered by the sub-tree represented by the item. The postcondition is a string that represents r's head non-terminal label, telling us which rules may be applied. The carry, κ, stores extra information required to correctly score the non-local interactions of the item when it will be linked in a broader context (for HMT with LM the carry consists of boundary words that will form new n-grams).

Items, $\iota \equiv \langle \iota_1 \succ r \prec \iota_2 \rangle$, are scored according to the following formula:

$$\begin{aligned} \mathsf{cost}(\iota) &= \mathsf{cost}(r) + \mathsf{cost}(\iota_1) + \mathsf{cost}(\iota_2) \\ &+ \mathsf{interaction}(r, \kappa_1, \kappa_2) \end{aligned} \quad (3)$$

Where: $\mathsf{cost}(r)$ is the cost associated to the weighted rule r; $\mathsf{cost}(\iota_1)$ and $\mathsf{cost}(\iota_2)$ are the costs of the two sub-items computed recursively using formula (3); $\mathsf{interaction}(r, \kappa_1, \kappa_2)$ is the interaction cost between the rule instantiation and the two sub-items. In HMT the interaction cost includes the LM score of new n-grams generated by connecting the childrens' sub-spans with terminals of r. Notice that formula (3) is equal to formula (2) for items that cover the whole input sequence.

In many TSP applications, the search space is too large to allow an exhaustive search and therefore pruning techniques must be used. Pruning decisions are based on the score of partial derivations. It is not always possible to compute exactly non-local features while computing the score of partial derivations, since partial derivations miss part of the context. Formula (3) accounts for the interaction between r and sub-items ι_1 and ι_2, but it does not integrate the cost relative to the interaction between the item and the surrounding context. Therefore the item score computed in a bottom-up fashion is an approximation of the score the item has in a broader context. For example, in HMT the LM score for n-grams that partially overlap the item's span cannot be computed exactly since the surrounding words are not known.

Basing pruning decisions on approximated scores can introduce search errors. It is possible to reduce search errors using heuristics based on future cost estimation. In general the estimation of the interaction between ι and the surrounding context is function of the carry, κ. In HMT it is possible to estimate the cost of n-grams that partially overlap ι's span considering the boundary words. We can obtain the heuristic cost for an item, ι, adding to formula (3) the factor, $\mathsf{est}(\kappa)$, for the estimation of interaction with missing context:

$$\mathsf{heuristicCost}(\iota) = \mathsf{cost}(\iota) + \mathsf{est}(\kappa) \quad (4)$$

And use $\mathsf{heuristicCost}(\iota)$ to guide BD pruning decisions. Anyway, even if a good interaction estimation is available, in practice it is not possible to avoid search errors while pruning.

More sophisticated parsing models allow the use of non-bottom-up features within a BD framework. Caraballo and Charniak (1998) present best-first parsing with Figures of Merit that allows conditioning of the heuristic function on statistics of the input string. Corazza et al. (1994), and Klein and Manning (2003) propose an A* parsing algorithm that estimates the upper bound of the parse completion scores using contextual summary features. These models achieve time efficiency and state-of-the-art accuracy for PCFG parsing, but still use a BD framework that doesn't allow the application of a broader class of non-bottom-up contextual features.

In HMT, knowing the sentence-wide context in which a sub-phrase is translated is extremely important. It is obviously important for word choice: as

a simple example consider the translation of the frequent English word "get" into Chinese. The choice of the correct set of ideograms to translate "get" often requires being aware of the presence of particles that can be at any distance within the sentence. In a common English to Chinese dictionary we found 93 different sets of ideograms that could be translations of "get". Sentence-wide context is also important in the choice of word re-ordering: as an example consider the following translations from English to German:

1. EN : *I go home.*
   DE : *Ich gehe nach Hause.*

2. EN : *I say, that I go home.*
   DE : *Ich sage, dass ich nach Hause gehe.*

3. EN : *On Sunday I go home.*
   DE : *Am Sonntag gehe ich nach Hause.*

The English phrase "*I go home*" is translated in German using the same set of words but with different orderings. It is not possible to choose the correct ordering of the phrase without being aware of the context. Thus a bottom-up decoder without context needs to build all translations for "*I go home*", introducing the possibility of pruning errors.

Having shown the importance of contextual features, we define a framework that overcomes the limitations of bottom-up feature approximation.

3 Undirected-CKY Framework

Our aim is to propose a new Framework that overcomes BD limitations allowing a better integration of contextual features. The presented framework can be regarded as a generalization of CKY.

To introduce the new framework let us focus on a detail of CKY BD. The items are created and scored in topological order. The ordering constraint can be formally stated as: *an item covering the span* $[i, j]$ *must be processed after items covering sub spans* $[h, k] | h \geq i, k \leq j$. This ordering constraint implies that full yield information is available when an item is processed, but information about ancestors and siblings is missing. Therefore non-bottom-up context cannot be used because of the ordering constraint. Now let us investigate how the decoding algorithm could change if we remove the ordering constraint.

Removing the ordering constraint would lead to the occurrence of cases in which an item is processed before all child items have been processed. For example, we could imagine to create and score an item, ι, with postcondition X and span $[i, j]$, linking the rule instantiation $r : X \rightarrow AB$ with only the left sub-item, ι_A, while information for the right sub-item, ι_B is still missing. In this case, we can rely on local and partial contextual features to score ι. Afterwards, it is possible to process ι_B using the parent item, ι, as a source of additional information about the parent context and sibling ι_A yield. This approach can avoid search errors in cases where pruning at the parent level can be correctly done using only local and partial yield context, while pruning at the child level needs extra non-bottom-up context to make a better pruning decision. For example, consider the translation of the English sentence "*I run*" into French using the following synchronous grammar:

$$
\begin{aligned}
&r_1 : S \rightarrow X_{\boxed{1}}\, X_{\boxed{2}} \mid X_{\boxed{1}}\, X_{\boxed{2}} \\
&r_2 : X \rightarrow I \mid Je \\
&r_3 : X \rightarrow run \mid course \\
&r_4 : X \rightarrow run \mid courir \\
&r_5 : X \rightarrow run \mid cours \\
&r_6 : X \rightarrow run \mid courons \\
&\vdots
\end{aligned}
$$

Where: r_1 is a $Glue$ rule and boxed indexes describe the alignment; r_2 translates "I" in the corresponding French pronoun; r_3 translates "run" as a noun; remaining rules translate "run" as one of the possible conjugations of the verb "$courir$". Using only bottom-up features it is not possible to resolve the ambiguity of the word "run". If the beam is not big enough the correct translation could be pruned. Anyway a CKY decoder would give the highest score to the most frequent translation. Instead, if we follow a non bottom-up approach, as described in Figure 1, we can: 1) first translate "I"; 2) Then create an item using r_1 with missing right child; 3) finally choose the correct translation for "run" using r_1 to access a wider context. Notice that with this undirected approach it is possible to reach the correct translation using only beam size of

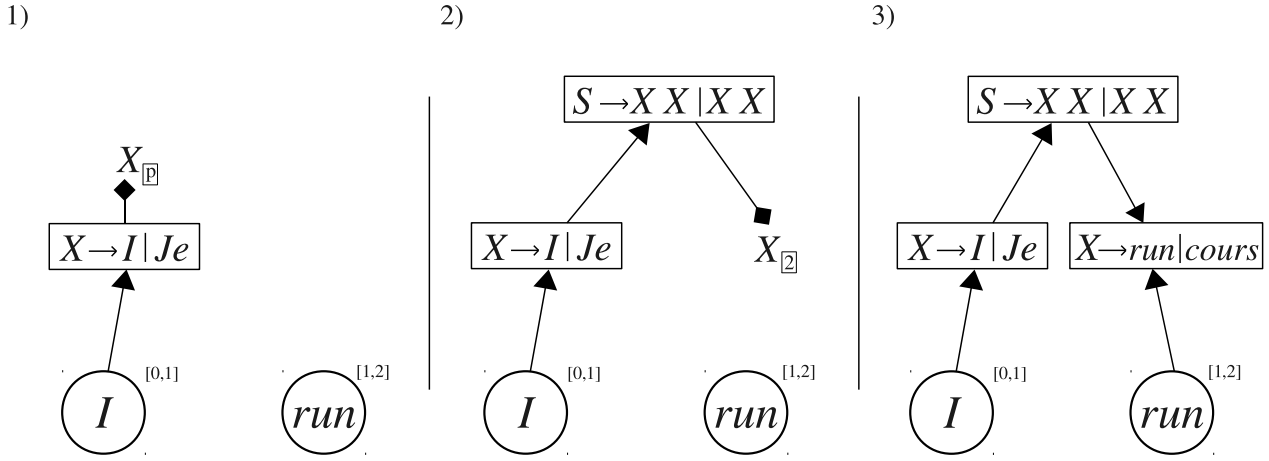

Figure 1: Example of undirected decoding for HMT. The arrows point to the direction in which information is propagated. Notice that the parent link at step 3 is fundamental to correctly disambiguate the translation for "run".

1 and the LM feature.

To formalize Undirected-CKY, we define a generalized item called $undirected\text{-}item$. Undirected-items, $\mathring{\boldsymbol{\imath}}$, are built linking rule instantiations with elements in $\mathcal{L} \equiv \{$left child, right child, parent$\}$; for example: $\mathring{\boldsymbol{\imath}} \equiv \langle \mathring{\boldsymbol{\imath}}_1 \gg r \dot{\vee} \mathring{\boldsymbol{\imath}}_p \rangle$, is built linking r with left child, $\mathring{\boldsymbol{\imath}}_1$, and parent, $\mathring{\boldsymbol{\imath}}_p$. We denote with $\mathcal{L}^+_{\mathring{\boldsymbol{\imath}}}$ the set of links for which the undirected-item, $\mathring{\boldsymbol{\imath}}$, has a connection, and with $\mathcal{L}^-_{\mathring{\boldsymbol{\imath}}}$ the set of missing links. An undirected-item is a triple that contains a span, a carry and an $undirected\text{-}postcondition$. The undirected-postcondition is a set of strings, one string for each of $\mathring{\boldsymbol{\imath}}$'s missing links, $l \in \mathcal{L}^-_{\mathring{\boldsymbol{\imath}}}$. Each string represents the non-terminal related to one of the missing links available for expansion. Bottom-up items can be considered specific cases of undirected-items having $\mathcal{L}^+ = \{$ left child, right child$\}$ and $\mathcal{L}^- = \{$parent$\}$. We can formally describe the steps of the example depicted in Figure 1 with:

$$1)\quad \frac{r_2 : X \rightarrow I|Je\ ,\ terminal : [0,1]}{\mathring{\boldsymbol{\imath}}_1 : [0,1,\{X_{\boxed{1}}\},\kappa_1]}$$

$$2)\quad \frac{r_1 : S \rightarrow X_{\boxed{1}}X_{\boxed{2}}|X_{\boxed{1}}X_{\boxed{2}}\ ,\ \mathring{\boldsymbol{\imath}}_1 : [0,1,\{X_{\boxed{1}}\},\kappa_1]}{\mathring{\boldsymbol{\imath}}_2 : [0,1,\{X_{\boxed{2}}\},\kappa_2)]}$$

$$3)\quad \frac{r_5 : X \rightarrow run|cours, \mathring{\boldsymbol{\imath}}_2 : [\cdots]\ ,\ terminal : [1,2]}{\mathring{\boldsymbol{\imath}}_3 : [0,2,\{\},\kappa_3]}$$

The scoring function for undirected-items can be obtained generalizing formula (3):

$$\begin{aligned} \mathsf{cost}(\mathring{\boldsymbol{\imath}}) &= \mathsf{cost}(r) \\ &+ \sum_{l \in \mathcal{L}^+} \mathsf{cost}(\mathring{\boldsymbol{\imath}}_l) \\ &+ \mathsf{interaction}(r, \mathcal{L}^+) \end{aligned} \tag{5}$$

In CKY, each span is processed separately in topological order, and the best-k items for each span are selected in sequence according to scoring function (4). In the proposed framework, the selection of undirected-items can be done in any order, for example: in a first step selecting an undirected-item for span s_1, then selecting an undirected-item for span s_2, and in a third step selecting a second undirected-item for s_1, and so on. As in agenda based parsing (Klein and Manning, 2001), all candidate undirected-items can be handled with an unique queue. Allowing the system to decide decoding order based on the candidates' scores, so that candidates with higher confidence can be selected earlier and used as context for candidates with lower confidence.

Having all candidates in the same queue introduces comparability issues. In CKY, candidates are comparable since each span is processed separately and each candidate is scored with the estimation of the yield score. Instead, in the proposed framework,

the unique queue contains candidates relative to different nodes and with different context scope. To ensure comparability, we can associate to candidate undirected-items a heuristic score of the full derivation:

$$\mathsf{heuristicCost}(\mathring{\boldsymbol{\imath}}) = \mathsf{cost}(\mathring{\boldsymbol{\imath}}) + \mathsf{est}(\mathring{\boldsymbol{\imath}}) \qquad (6)$$

Where $\mathsf{est}(\mathring{\boldsymbol{\imath}})$ estimates the cost of the missing branches of the derivation as a function of $\mathring{\boldsymbol{\imath}}'s$ partial structure and carry.

In this framework, the queue can be initialized with a candidate for each rule instantiation. These initializing candidates have no context information and can be scored using only local features. A generic decoding algorithm can loop selecting the candidate undirected-item with the highest score, $\mathring{\boldsymbol{\imath}}$, and then propagating its information to neighboring candidates, which can update using $\mathring{\boldsymbol{\imath}}$ as context. In this general framework the link to the parent node is not treated differently from links to children. While in CKY the information is always passed from children to parent, in Undirected-CKY the information can be propagated in any direction, and any decoding order is allowed.

We can summarize the steps done to generalize CKY into the proposed framework: 1) remove the node ordering constraint; 2) define the scoring of candidates with missing children or parent; 3) use a single candidate queue; 4) handle comparability of candidates from different nodes and/or with different context scope; 5) allow information propagation in any direction.

4 Undirected Decoding

In this section we propose Undirected Decoding (UD), an instantiation of the Undirected-CKY framework presented above. The generic framework introduces many new degrees of freedom that could lead to a higher complexity of the decoder. In our actual instantiation we apply constraints on the initialization step, on the propagation policy, and fix a search beam of k. These constraints allow the system to converge to a solution in practical time, allow the use of dynamic programming techniques to merge items with equivalent states, and gives us the possibility of using non-bottom-up features and testing their relevance.

Algorithm 1 Undirected Decoding

```
1:  function decoder (k) : out-forest
2:  Q ← LeafRules();
3:  while |Q| > 0 do
4:     ı̊ ← PopBest (Q);
5:     if CanPop(ı̊) then
6:        out-forest.Add(ı̊);
7:        if ı̊.HasChildrenLinks() then
8:           for all r ∈ HeadRules(ı̊) do
9:              Ĉ ← NewUndirectedItems(r, ı̊);
10:             for all ĉ ∈ Ĉ do
11:                if CanPop(ĉ) then
12:                   Q.Insert(ĉ);
13:                end if
14:             end for
15:          end for
16:       end if
17:    end if
18: end while
```

Algorithm 1 summarizes the UD approach. The beam size, k, is given as input. At $line$ 2 the queue of undirected-item candidates, $\mathbf{Q}$, is initialized with only leafs rules. At $line$ 3 the loop starts, it will terminate when $\mathbf{Q}$ is empty. At $line$ 4 the candidate with highest score, $\mathring{\boldsymbol{\imath}}$, is popped from $\mathbf{Q}$. $line$ 5 checks if $\mathring{\boldsymbol{\imath}}$ is within the beam width: if $\mathring{\boldsymbol{\imath}}$ has a span for which k candidates were already popped, then $\mathring{\boldsymbol{\imath}}$ is dropped and a new iteration is begun. Otherwise $\mathring{\boldsymbol{\imath}}$ is added to the out-forest at $line$ 6. From $line$ 7 to $line$ 10 the algorithm deals with the generation of new candidate undirected-items. $line$ 7 checks if $\mathring{\boldsymbol{\imath}}$ has both children links, if not a new decoding iteration is begun. $line$ 8 loops over the rule instantiations, r, that can use $\mathring{\boldsymbol{\imath}}$ as child. At $line$ 9, the set of new candidates, $\hat{\mathbf{C}}$, is built linking r with $\mathring{\boldsymbol{\imath}}$ and any context already available in the out-forest. Finally, between $line$ 10 and $line$ 12, each element $\hat{\mathbf{c}}$ in $\hat{\mathbf{C}}$ is inserted in $\mathbf{Q}$ after checking that $\hat{\mathbf{c}}$ is within the beam width: if $\hat{\mathbf{c}}$ has a span for which k candidates were already popped it doesn't make sense to insert it in $\mathbf{Q}$ since it will be surely discarded at $line$ 5.

In more detail, the function NewUndirectedItems$(r, \mathring{\boldsymbol{\imath}})$ at $line$ 9 creates new undirected-items linking r using: 1) $\mathring{\boldsymbol{\imath}}$ as child; 2) (optionally) as other child any other undirected-item that has already been inserted in the out-forest and

doesn't have a missing child and matches missing span coverage; 3) and using as parent context the best undirected-item with missing child link that has been incorporated in the out-forest and can expand the missing child link using r. In our current method, only the best possible parent context is used because it only provides context for ranking candidates, as discussed at the end of this section. In contrast, a different candidate is generated for each possible other child in 2), as well as for the case where no other child is included in the undirected-item.

We can make some general observations on the Undirected Decoding Algorithm. Notice that, the **if** statement at $line$ 7 and the way new undirected-items are created at $line$ 9, enforce that each undirected-item covers a contiguous span. An undirected-item that is missing a child link cannot be used as child context but can be used as parent context since it is added to the out-forest at $line$ 6 before the **if** statement at $line$ 7. Furthermore, the **if** statements at $line$ 5 and $line$ 11 check that no more than k candidates are selected for each span, but the algorithm does not require the the selection of exactly k candidates per span as in CKY.

The queue of candidates, $\mathbf{Q}$, is ordered according to the heuristic cost of formula (6). The score of the candidate partial structure is accounted for with factor $\mathsf{cost}(\mathring{\boldsymbol{\imath}})$ computed according to formula (5). The factor $\mathsf{est}(\mathring{\boldsymbol{\imath}})$ accounts for the estimation of the missing part of the derivation. We compute this factor with the following formula:

$$\mathsf{est}(\mathring{\boldsymbol{\imath}}) = \sum_{l \in \mathcal{L}^{-}_{\mathring{\boldsymbol{\imath}}}} \Big(\mathsf{localCost}(\mathring{\boldsymbol{\imath}}, l) + \mathsf{contextEst}(\mathring{\boldsymbol{\imath}}, l) \Big) \tag{7}$$

For each missing link, $l \in \mathcal{L}^{-}_{\mathring{\boldsymbol{\imath}}}$, we estimate the cost of the corresponding derivation branch with two factors: $\mathsf{localCost}(\mathring{\boldsymbol{\imath}}, l)$ that computes the context-free score of the branch with highest score that could be attached to l; and $\mathsf{contextEst}(\mathring{\boldsymbol{\imath}}, l)$ that estimates the contextual score of the branch and its interaction with $\mathring{\boldsymbol{\imath}}$. Because our model is implemented in the Forest Rescoring framework (e.g. Huang and Chiang (2007), Dyer et al. (2010), Li et al. (2009)), $\mathsf{localCost}(\mathring{\boldsymbol{\imath}}, l)$ can be efficiently computed exactly. In HMT it is possible to exhaustively represent and search the context-free-forest (ignoring the LM), which is done in the Forest Rescoring framework before our task of decoding with the LM. We exploit this context-free-forest to compute $\mathsf{localCost}(\mathring{\boldsymbol{\imath}}, l)$: for missing child links the $\mathsf{localCost}(\cdot)$ is the Inside score computed using the $(max,\ +)$ semiring (also known as the Viterbi score), and for missing parent links the $\mathsf{localCost}(\cdot)$ is the corresponding Outside score. The factor $\mathsf{contextEst}(\cdot)$ estimates the LM score of the words generated by the missing branch and their interaction with the span covered by $\mathring{\boldsymbol{\imath}}$. To compute the expected interaction cost we use the boundary words information contained in $\mathring{\boldsymbol{\imath}}$'s carry as done in BD. To estimate the LM cost of the missing branch we use an estimation function, conditioned on the missing span length, whose parameters are tuned on held-out data with gradient descent, using the search score as objective function.

To show that UD leads to better results than BD, the two algorithms are compared in the same search space. Therefore we ensure that candidates embedded in the UD out-forest would have the same score if they were scored from BD. We don't need to worry about differences derived from the missing context estimation factor, $\mathsf{est}(\cdot)$, since this factor is only considered while sorting the queue, $\mathbf{Q}$, according to the $\mathsf{heuristicCost}(\cdot)$. Also, we don't have to worry about candidates that are scored with no missing child and no parent link, because in that case scoring function (3) for BD is equivalent to scoring function (5) for UD. Instead, for candidates that are scored with parent link, we remove the parent link factor from the $\mathsf{cost}(\cdot)$ function when inserting the candidate into the out forest. And for the candidates that are scored with a missing child, we adjust the score once the link to the missing child is created in the out-forest. In this way UD and BD score the same derivation with the same score and can be regarded as two ways to explore the same search space.

5 Experiments

In this section we test the algorithm presented, and empirically show that it produces better translations searching a smaller portion of the search space.

We implemented UD on top of a widely-used HMT open-source system, cdec (Dyer et al., 2010). We compare with cdec Cube Pruning BD. The ex-

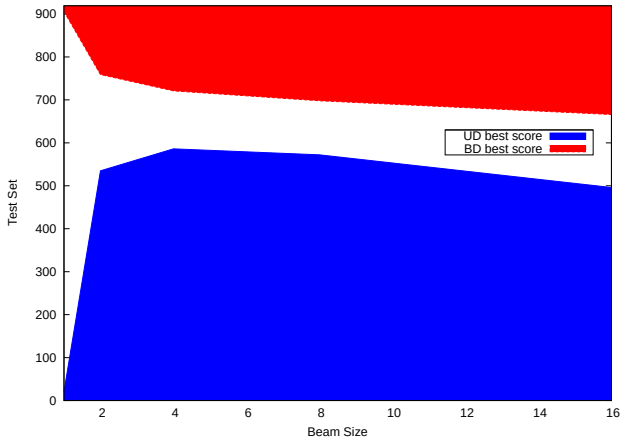

Figure 2: Comparison of the quality of the translations.

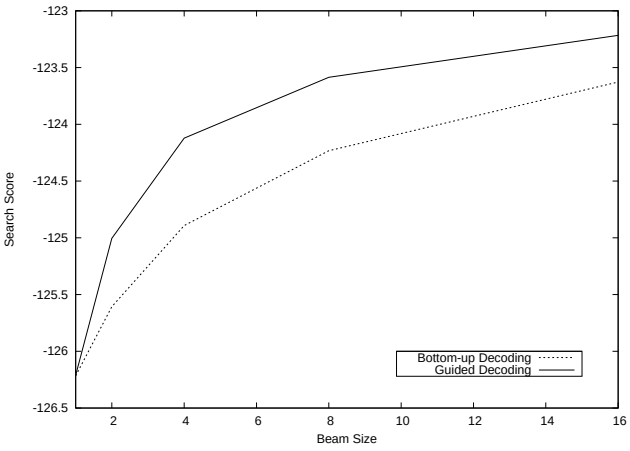

Figure 3: Search score evolution for BD and UD.

periments are executed on the NIST MT03 Chinese-English parallel corpus. The training corpus contains 239k sentence pairs with 6.9M Chinese words and 8.9M English words. We use a hierarchical phrase-based translation grammar extracted using a suffix array rule extractor (Lopez, 2007). The NIST-03 test set is used for decoding, it has 919 sentence pairs. The experiments can be reproduced on an average desktop computer. Since we compare two different decoding strategies that rely on the same training technique, the evaluation is primarily based on search errors rather than on BLEU. We compare the two systems on a variety of beam sizes between 1 and 16.

Figure 2 reports a comparison of the translation quality for the two systems in relation to the beam size. The blue area represents the portion of sentences for which UD found a better translation. The white area represents the portion of sentences for which the two systems found a translation with the same search score. With beam 1 the two systems obviously have a similar behavior, since both the systems stop investigating the candidates for a node after having selected the best candidate immediately available. For beams 2-4, UD has a clear advantage. In this range UD finds a better translation for two thirds of the sentences. With beam 4, we observe that UD is able to find a better translation for 63.76% of the sentences, instead BD is able to find a better translation for only 21.54% of the sentences. For searches that employ a beam bigger than 8, we notice that the UD advantage slightly decreases, and the number of sentences with equivalent translation slowly increases. We can understand this behavior considering that as the beam increases the two systems get closer to exhaustive search. Anyway with this experiment UD shows a consistent accuracy advantage over BD.

Figure 3 plots the search score variation for different beam sizes. We can see that UD search leads to an average search score that is consistently better than the one computed for BD. Undirected Decoding improves the average search score by 0.411 for beam 16. The search score is the logarithm of a probability. This variation corresponds to a relative gain of 50.83% in terms of probability. For beams greater than 8 we see that the two curves keep a monotonic ascendant behavior while converging to exhaustive search.

Figure 4 shows the BLEU score variation. Again we can see the consistent improvement of UD over BD. In the graph we report also the performance obtained using BD with beam 32. BD reaches BLEU score of 32.07 with beam 32 while UD reaches 32.38 with beam 16: UD reaches a clearly higher BLEU score using half the beam size. The difference is even more impressive if we consider that UD reaches a BLEU of 32.19 with beam 4.

In Figure 5 we plot the percentage reduction of the size of the hypergraphs generated by UD compared to those generated by BD. The size reduction grows quickly for both nodes and edges. This is due to the fact that BD, using Cube Pruning, must select k candidates for each node. Instead, UD is not obliged to

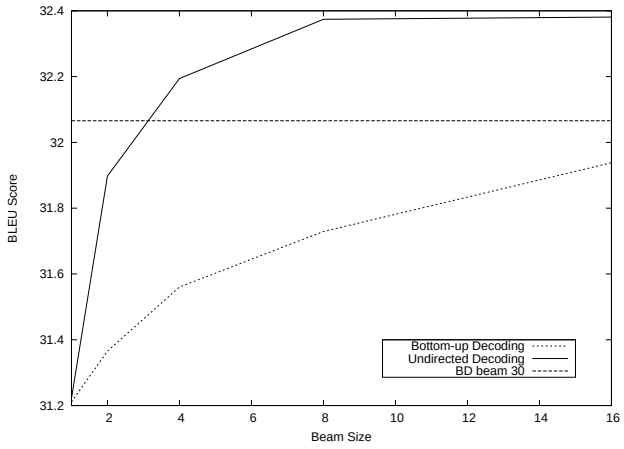

Figure 4: BLEU score evolution for BD and UD.

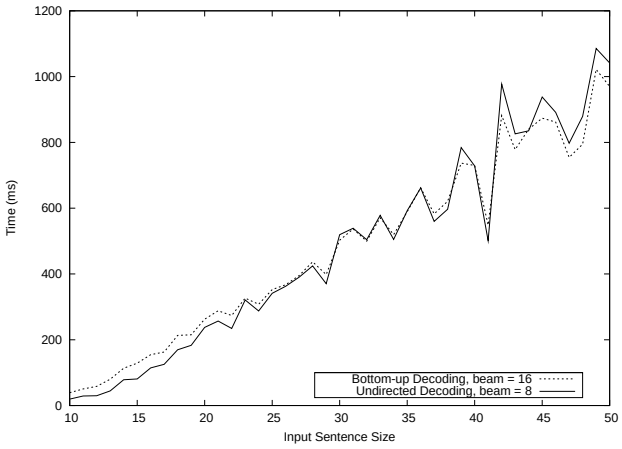

Figure 6: Time comparison between BD and UD.

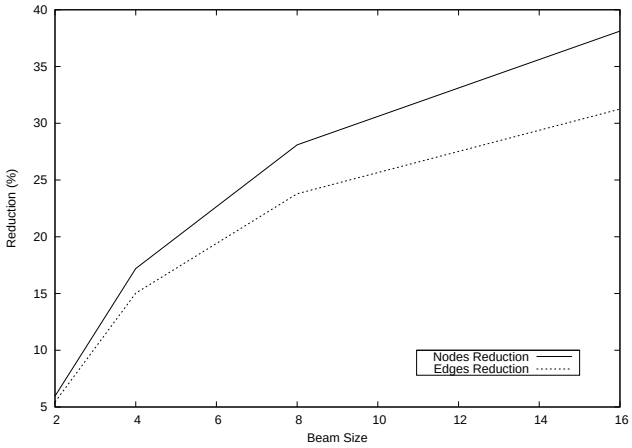

Figure 5: Percentage of reduction of the size of the hypergraph produced by UD.

select k candidates per $\mathbf{f}$-node. As we can see from Algorithm 1, the decoding loop terminates when the queue of candidates is empty, and the statements at $line\ 5$ and $line\ 11$ ensure that no more than k candidates are selected per $\mathbf{f}$-node, but nothing requires the selection of k elements, and some bad candidates may not be generated due to the sophisticated propagation strategy. The number of derivations that a hypergraph represents is exponential in the number of nodes and edges composing the structure. With beam 16, the hypergraphs produced by UD contain on average 4.6k fewer translations. Therefore UD is able to find better translations even if exploring a smaller portion of the search space.

Figure 6 reports the time comparison between BD and UD with respect to sentence length. The sentence length is measured with the number of ideogram groups appearing in the source Chinese sentences. We compare BD with beam of 16 and UD with beam of 8, so that we compare two systems with comparable search score. We can notice that for short sentences UD is faster, while for longer sentences UD becomes slower. To understand this result consider that for simple sentences UD can rely on the advantage of exploring a smaller search space. While, for longer sentences, the amount of candidates considered during decoding grows exponentially with the size of the sentence, and UD needs to maintain an unique queue whose size is not bounded by the beam size k, as for the queues used in BD's Cube Pruning. It may be possible to address this issue with more efficient handling of the queue.

In conclusion we can assert that, even if exploring a smaller portion of the search space, UD finds often a translation that is better than the one found with standard BD. UD's higher accuracy is due to its sophisticated search strategy that allows a more efficient integration of contextual features. This set of experiments show the validity of the UD approach and empirically confirm our intuition about the BD's inadequacy in solving tasks that rely on fundamental contextual features.

6 Future Work

In the proposed framework the link to the parent node is not treated differently from links to child nodes, the information in the hypergraph can be propagated in any direction. Then the Derivation

Hypergraph can be regarded as a non-directed graph. In this setting we could imagine applying message passing algorithms from graphical model theory (Koller and Friedman, 2010).

Furthermore, considering that the proposed framework lets the system decide the decoding order, we could design a system that explicitly learns to infer the decoding order at training time. Similar ideas have been successfully tried: Shen et al. (2010) and Gesmundo (2011) investigate the Guided Learning framework, that dynamically incorporates the tasks of learning the order of inference and training the local classifier.

7 Conclusion

With this paper we investigate the limitations of Bottom-up parsing techniques, widely used in Tree Structures Prediction, focusing on Hierarchical Machine Translation. We devise a framework that allows a better integration of non-bottom-up features. Compared to a state of the art HMT decoder the presented system produces higher quality translations searching a smaller portion of the search space, empirically showing that the bottom-up approximation of contextual features is a limitation for NLP tasks like HMT.

Acknowledgments

This work was partly funded by Swiss NSF grant CRSI22_127510 and European Community FP7 grant 216594 (CLASSiC, www.classic-project.org).

References

Sharon A. Caraballo and Eugene Charniak. 1998. New figures of merit for best-first probabilistic chart parsing, Computational Linguistics, 24:275-298.

J. C. Chappelier and M. Rajman and R. Arages and A. Rozenknop. 1999. Lattice Parsing for Speech Recognition. In Proceedings of TALN 1999, Cargse, France.

David Chiang. 2007. Hierarchical phrase-based translation. Computational Linguistics, 33(2):201-228, 2007.

Anna Corazza, Renato De Mori, Roberto Gretter and Giorgio Satta. 1994. Optimal Probabilistic Evaluation Functions for Search Controlled by Stochastic Context-Free Grammars. IEEE Transactions on Pattern Analysis and Machine Intelligence, 16(10):1018-1027.

Chris Dyer, Adam Lopez, Juri Ganitkevitch, Johnathan Weese, Ferhan Ture, Phil Blunsom, Hendra Setiawan, Vladimir Eidelman, and Philip Resnik. 2010. cdec: A Decoder, Alignment, and Learning framework for finite-state and context-free translation models. In Proceedings of the Conference of the Association of Computational Linguistics 2010, Uppsala, Sweden.

Andrea Gesmundo and James Henderson 2010. Faster Cube Pruning. Proceedings of the seventh International Workshop on Spoken Language Translation (IWSLT), Paris, France.

Andrea Gesmundo 2011. Bidirectional Sequence Classification for Tagging Tasks with Guided Learning. Proceedings of TALN 2011, Montpellier, France.

Mark Hopkins and Greg Langmead 2009. Cube pruning as heuristic search. Proceedings of the Conference on Empirical Methods in Natural Language Processing 2009, Singapore.

Liang Huang and David Chiang. 2007. Forest rescoring: Faster decoding with integrated language models. In Proceedings of the Conference of the Association of Computational Linguistics 2007, Prague, Czech Republic.

Dan Klein and Christopher D. Manning. 2001 Parsing and Hypergraphs, In Proceedings of the International Workshop on Parsing Technologies 2001, Beijing, China.

Dan Klein and Christopher D. Manning. 2003 A* Parsing: Fast Exact Viterbi Parse Selection, In Proceedings of the Conference of the North American Association for Computational Linguistics 2003, Edmonton, Canada.

Daphne Koller and Nir Friedman. 2010. Probabilistic Graphical Models: Principles and Techniques. The MIT Press, Cambridge, Massachusetts.

Shankar Kumar, Wolfgang Macherey, Chris Dyer, and Franz Och. 2009. Efficient Minimum Error Rate Training and Minimum Bayes-Risk decoding for translation hypergraphs and lattices, In Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP, Suntec, Singapore.

Zhifei Li, Chris Callison-Burch, Chris Dyer, Juri Ganitkevitch, Sanjeev Khudanpur, Lane Schwartz, Wren N. G. Thornton, Jonathan Weese, and Omar F. Zaidan. 2009. Joshua: An Open Source Toolkit for Parsing-based Machine Translation. In Proceedings of the Workshop on Statistical Machine Translation 2009, Athens, Greece.

Adam Lopez. 2007. Hierarchical Phrase-Based Translation with Suffix Arrays. In Proceedings of the Conference on Empirical Methods in Natural Language Processing 2007, Prague, Czech Republic.

Haitao Mi, Liang Huang and Qun Liu. 2008. Forest-Based Translation. In Proceedings of the Conference of the Association of Computational Linguistics 2008, Columbus, OH.

Libin Shen, Giorgio Satta and Aravind Joshi. 2007. Guided Learning for Bidirectional Sequence Classification. In Proceedings of the Conference of the Association of Computational Linguistics 2007, Prague, Czech Republic.

Andreas Stolcke. 2002. SRILM - An extensible language modeling toolkit. In Proceedings of the International Conference on Spoken Language Processing 2002, Denver, CO.

Andreas Zollmann and Ashish Venugopal. 2006. Syntax augmented machine translation via chart parsing, Proceedings of the Workshop on Statistical Machine Translation, New York City, New York.