An Application of Probabilistic Grammars to Efficient Machine Translation

Paweł Skórzewski

Adam Mickiewicz University
Faculty of Mathematics and Computer Science
ul. Umultowska 87, 61-614 Poznań, Poland
pawel.skorzewski@amu.edu.pl

Abstract. In this paper we present one of the algorithms used to parse probabilistic context-free grammars: the A* parsing algorithm, which is based on the A* graph search method. We show an example of application of the algorithm in an existing machine translation system. The existing CYK-based parser used in the *Translatica* system was modified by applying the A* parsing algorithm in order to examine the possibilities of improving its performance. This paper presents the results of applying the A* algorithm with different heuristic functions and their impact on the performance of the parser.

Keywords: A* algorithm, machine translation, natural language parsing, PCFG, probabilistic grammars

1 Introduction

Context-free grammars are one of the popular and useful tools for modeling natural languages. Probabilistic context-free grammars are the extension of the context-free grammars that assign the probabilities to different sentences.

A chart parsing is a common method of parsing probabilistic context-free grammars. The time complexity of chart parsing algorithms is generally worst-case cubic, but in practice even cubic time can be insufficient, especially when dealing with large grammars and long sentences.

The one of the most important tasks connected with PCFGs is to find the parse which has the highest probability from all the possible parses of the given sentence. Such a parse is called a *Viterbi parse* of the sentence. There is an analogy between finding the Viterbi parse of a given sentence and finding the shortest path between given two points in some weighted multigraph. This is the reason why we can use modified graph search algorithms to parse PCFGs. We may expect that applying the shortest-path-finding algorithms in PCFG parsing can speed-up the process of finding a Viterbi parse of a given sentence.

The best-first parsing is one of the methods of speeding-up the parsing of PCFGs. Best-first method is a greedy strategy described in [1] and [2]. In this method, some figure of merit is used to determine the priority of removing edges from the agenda. The number of edges processed by the parser can be significantly reduced by proper choice of the figure of merit.

References [5] and [6] describe a beam-search strategy. This is also a greedy algorithm, in which only limited number of best parses is tracked at any time. Time required to complete the algorithm can be shortened by reducing the number of tracked parses.

Both algorithms have their advantages but neither best-first strategy nor beam-search method guarantees that the actual Viterbi parse of a given sentence will be find by them.

Klein and Manning in [3] presented a PCFG parsing algorithm based on the A* graph search procedure. The A* algorithm finds the shortest path between two given vertices of the graph using some given heuristic function. The heuristic function is a function defined on the vertices of the graph that estimates the distance between the given vertex and the target vertex. The algorithm tries to find the shortest path by minimizing the sum of the distance covered so far and the value of the heuristic.

The authors showed that the use of A* algorithm with properly chosen heuristic function can considerably shorten the time of finding the Viterbi parse of a given sentence.

2 The A* Parsing Algorithm

2.1 The Idea of the Algorithm

The A* parsing algorithm is a variant of a chart parser. The A* parser operates on data structures called edges. Let $w = w_1 \dots w_n$ be a string to parse. The edge $X[i,j]$ consists of a symbol of the grammar (X) and the pair of numbers (i,j) that denotes the span of this symbol in the parse tree of a given string $w = w_1 \dots w_n$. In other words, the symbol X dominates the substring $w_{i+1} \dots w_j$. The algorithm assigns a score to each edge. The agenda is a structure where the edges waiting to be processed are stored. The other structure, called a chart, stores the edges whose best parses have already been found.

The course of the algorithm is following: The agenda and the chart are initialized by the set of edges representing the words of the parsed sentence. Now the scores of the edges in the agenda are calculated in order to determine the priority of edges. The edge $e = X[i,j]$ which has the highest priority from among the edges of the agenda is chosen and removed from the agenda. If there exists an edge $Y[k,i]$ in the chart and a rule $Z \to YX$ in the set of rules, the new edge $Z[i,j]$ is created in the chart and inserted into the agenda. Similarly, if there exists an edge $Y[j,k]$ in the chart and a rule $Z \to YX$, the new edge $Z[k,j]$ is formed in the chart and inserted into the agenda. To sum up, the chosen edge is combined with the edges from the chart to form new edges (if possible according to the rules of the grammar) and these newly formed edges are put into the agenda. The whole procedure, called a *turn of the parser*, is repeated until no edges remain in the agenda or the root of the grammar is reached (ie. removed form the agenda and constructed into the chart).

The important thing that distinguishes the A* parsing algorithm from among other chart parsing algorithms is the function used to prioritize edges in the agenda.

2.2 The Viterbi Inside and Outside Scores

Consider the probabilistic context-free grammar $G = (V, T, R, S, P)$ and a string $w = w_1 \dots w_n$ we want to parse. We define the *Viterbi inside score* $\beta_{G,w}(e)$ of an edge $e = X[i,j]$ as the maximum of the log-probabilities of the derivation $X \Rightarrow^* w_{i+1} \dots w_j$. Similarly we define the *Viterbi outside score* $\alpha_{G,w}(e)$ of an edge $e = X[i,j]$ as the maximum of the log-probabilities of the derivation $X \Rightarrow^* w_1 \dots w_i X w_{j+1} \dots w_n$. We can see the Viterbi inside score as the log-probability of the best inside parse of a given edge and the Viterbi outside score as the log-probability of the best outside parse. We will drop the grammar and parsed string indicators and write simply $\beta(e)$ and $\alpha(e)$ instead of $\beta_{G,w}(e)$ and $\alpha_{G,w}(e)$ if it doesn't lead to confusion.

We can estimate Viterbi inside and outside scores $\beta(e)$ and $\alpha(e)$ by estimates $b(e)$ and $a(e)$ that fulfil the following conditions. The estimate $b(e)$ represents the log-probability of the best inside parse of the edge e found so far. At the beginning $b(e) = -\infty$ and never decreases during the course of the algorithm. When the edge e is removed from the agenda, the estimate $b(e)$ should be equal to the value of $\beta(e)$. The estimate $a(e)$ should be admissible, ie. $a(e) \geq \alpha(e)$.

Note that the value of $\beta(e)$ can play the role of the distance between starting vertex and the actual vertex (the distance covered so far) from the classic graph A* algorithm. The estimate $a(e)$, if admissible, can be used as a heuristic function. So we can use the sum $\beta(e) + a(e)$ to prioritize edges in the agenda if the value of $\beta(e) + a(e)$ never increases during the course of the algorithm.

There are various heuristic fuctions that can be used in the A* parsing algorithm. Reference [3] presents heuristics based on context summary estimates and on grammar projection estimates.

2.3 The Context Summary Esimates

The context summary estimates use the knowledge about the neighbourhood of the considered symbol: the label and the number of the symbols on the left and on the right. The value of such estimate is calculated as the maximum of the log-probabilities of the possible outside derivations that satisfy some given condition.

Let $G = (V, T, R, S, P)$ be a probabilistic context-free grammar and $w = w_1 w_2 \dots w_n$ be a string of terminates.

No information about the symbol's neighborhood results in the NULL estimate, constantly equal zero:

$$a_{G,w}^{\mathrm{NULL}}(X[i,j]) \;=\; \max_{u \in T^*} \log \mathbf{P}(S \Rightarrow^* u) \;=\; 0. \tag{1}$$

The opposite is the TRUE estimate, which assumes having the complete information about the context and thus equals the exact value of Viterbi outside score α:

$$a_{G,w}^{\mathrm{TRUE}}(X[i,j]) \;=\; \log \mathbf{P}(S \Rightarrow^* w_1 \dots w_i X w_{j+1} \dots w_n) \;=\; \alpha_{G,w}(X[i,j])\,. \tag{2}$$

The SX estimate is an example of a context summary estimate. It specifies the label of the current symbol and the number of terminals to the left and to the right:

$$a_{G,w}^{\mathrm{SX}}(X[i,j]) \;=\; \max_{u_1,\dots,u_i,v_1,\dots,v_{n-j} \in T} \log \mathbf{P}(S \Rightarrow^* u_1 \dots u_i X v_1 \dots v_{n-j})\;, \tag{3}$$

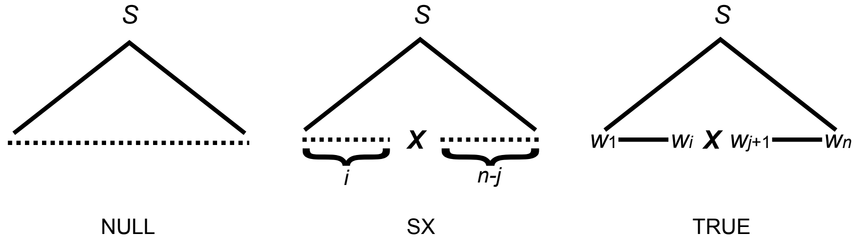

Fig. 1. Examples of context summary estimates.

ie. the value of the estimate depends on X, i and j.

The symbolic presentation of concepts of different context summary estimates are presented on the Figure 1.

2.4 The Grammar Projection Estimates

The grammar projection estimates use the entire context of the current symbol and a reduced grammar (with smaller number of symbols and reduced set of rules). We define the grammar projection $G' = (V', T', R', S', P')$ of a given probabilistic context-free grammar $G = (V, T, R, S, P)$ as follows. Let N be an arbitrary set of symbols and let $\pi \colon V \cup T \rightarrow N$ be a function. Then:

– $V' := \{\pi(A) \colon A \in V\} \subseteq N$,
– $T' := \{\pi(a) \colon A \in T\} \subseteq N$ and $T' \cap V' = \emptyset$,
– $R' := \{\hat{\pi}(r) \colon r \in R\}$, where

$$\hat{\pi} \colon R \rightarrow R' \ , \tag{4}$$

$$\hat{\pi}(A \rightarrow X_1 \dots X_k) := \pi(A) \rightarrow \pi(X_1) \dots \pi(X_k) \ , \tag{5}$$

– $S' := \pi(S)$,
– P' is determined as

$$P'(r') := \max_{r : \hat{\pi}(r) = r'} P(r) \ . \tag{6}$$

Note that the resulted grammar is a weighted context-free grammar but do not have to be a probabilistic context-free grammar (the probabilities of the rules with the same symbol on the left do not have to sum to 1). More precisely, for every fixed $A \in V'$:

$$\sum_{A \rightarrow \zeta \in R'} P'(A \rightarrow \zeta) \geq 1 \ . \tag{7}$$

If we compare it with the condition that helds for PCFGs:

$$\sum_{A \rightarrow \zeta \in R} P(A \rightarrow \zeta) = 1 \ , \tag{8}$$

we can see that the following inequation holds for each production $r \in R$:

$$P'(\hat{\pi}(r)) \geq P(r). \tag{9}$$

This relation enables us to use $a(e) = \alpha_{G',w}(\pi(X)[i,j])$ as an outside estimate of $\alpha_{G,w}(X[i,j])$, bacause it guarantees that

$$\alpha_{G',w}(\pi(X)[i,j]) \geq \alpha_{G,w}(X[i,j]) \tag{10}$$

for each edge $X[i,j]$.

There exists a variety of grammar projection estimates, from the NULL projection estimate (based on the constant projection) to the TRUE projection estimate (corresponding to the identity projection).

3 The Application of A* Parsing Algorithm in the *Translatica*'s Parser

3.1 *Translatica* Machine Translation System

Translatica is a rule-based machine translation system that can translate between following pairs of lanuages:

- Polish and English,
- Polish and Russian,
- Polish and German,
- Polish and French.

The translation of more language pairs is in development.

The German language parser of the *Translatica* system is a CYK-based parser that uses a weighted context-free grammar. The rules and their weights for this grammar have been extracted automatically from the TüBa treebank[1].

We adapted the current CYK-based parser so that the A* methods could be used to imrove its performance and speed.

3.2 The Implementation of Context Summary Based Heuristics

We chose to implement the SX heuristic because (as described in [3]) it was expected that it would bring the significant edge savings with relatively little precomputation required.

Each turn of the parser begins with finding the sum $\beta(e) + a(e)$ for each edge e in the agenda. If it is the fist time for the edge e to find the value of the sum, it is calculated and stored in the memory. Otherwise, the result is retrieved from the memory. This process, called memoization, prevents calculating the value of the heuristic for the same arguments twice and thus saves the time and speeds up the calculation.

We also implemented the NULL heuristic — for comparational reasons. As constatntly equal zero, the NULL heuristic is easy to implement.

[1] `http://www.sfs.uni-tuebingen.de/en/tuebadz.shtml`

Table 1. The average translation times for the test set of 100 German sentences. The column labeled *relative* shows the ratio of the test time to the time of parsing without A* methods.

threshold	10000		50000	
time	absolute	relative	absolute	relative
without A*	36 s	1.00	1 min 9 s	1.00
with A*, SX heuristic	52 s	1.44	2 min 2 s	1.77
with A*, NULL heuristic	37 s	1.03	1 min 20 s	1.16

Table 2. The average translation times for the test set of 500 German sentences. The column labeled *relative* shows the ratio of the test time to the time of parsing without A* methods.

threshold	10000		50000	
time	absolute	relative	absolute	relative
without A*	2 m 56 s	1.00	5 min 52 s	1.00
with A*, SX heuristic	4 m 32 s	1.55	11 min 50 s	2.02
with A*, NULL heuristic	3 m 6 s	1.06	6 min 55 s	1.18

3.3 The Implementation of Grammar Projection Based Heuristic: the Attribute Grammar Projection

The grammar used in *Translatica*'s German parser is a kind of weighted attribute grammar. In the attribue grammar each rule is accompanied by a set of attribute expressions. The rule can be used only if the expressions associated with the rule are satisfied by the actual values of symbols' attributes.

By considering all possibile values of attributes and fixing the values of selected attributes of the rules and the symbols we can obtain the context-free grammar. Thus the probabilistic attribute grammar can be projected to a probabilistic context-free grammar. In the *Translatica*'s parser we have implemented the projection obtained by fixing the most common attributes (eg. the tense).

4 The Evaluation of the Solutions

The number of turns of *Translatica*'s parser can be limited. The maximal number of parser turns is called a *threshold* and can be arbitrarily set before running the parser. The default value of *Translatica*'s German parser is set to 50000.

In order to compare the performance of the old *Translatica*'s parser and the new A* parser, we conducted various tests. We have done a series of machine translation on the two sets of example sentences on various topics (a set of 100 sentences and a set of 500 sentences) using two different thresholds: 10000 and 50000. We used three different parsers:

– the “old” *Translatica*'s parser—without A*,
– the A* parser with SX heuristic,
– the A* parser with NULL heuristic.

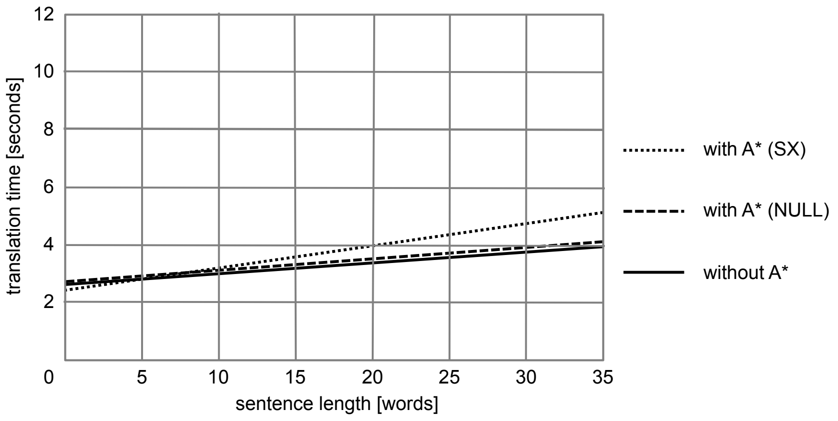

Fig. 2. The relationship between the lengths of individual sentences and the times of their translation for the threshold of 10000.

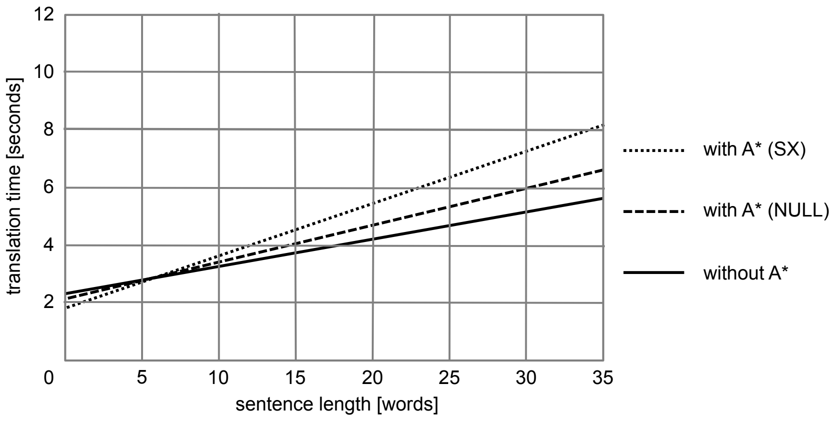

Fig. 3. The relationship between the lengths of individual sentences and the times of their translation for the threshold of 50000.

The average times of translation of the set of 100 example German sentences is shown in Table 1. The average times of translation of the set of 500 example German sentences (100 sentences repeated 5 times) is shown in Table 2.

The relationship between the length of the sentence and the time of its translation is presented on Figures 2 and 3.

The implementation of the A* algorithm has not brought the acceleration of the translation process in *Translatica*'s parser. Indeed, the A* parser implemented in *Translatica* turned out to be slightly slower than the old CYK-based *Translatica*'s parser.

The quality of the translation with the A* algorithm was comparable to the translation without the A* algorithm. Most sentences were translated identically. Many translations differed no more than in a single word. There were some sentences that had

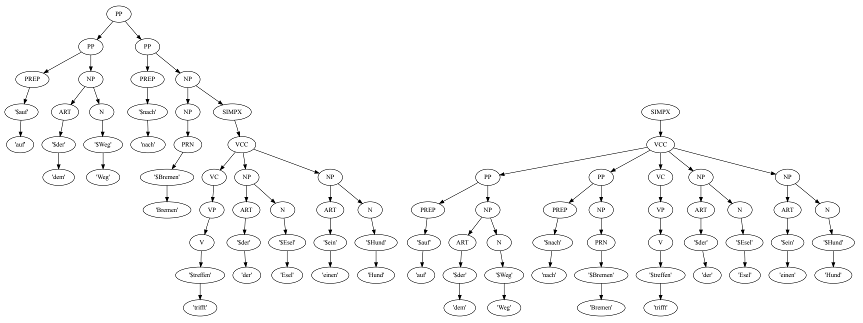

Fig. 4. Comparison of results of parsing the German sentence *Auf dem Weg nach Bremen trifft der Esel einen Hund* when the threshold is set to 10000 — without the A* algorithm (*left*) and using the A* algorithm (*right*).

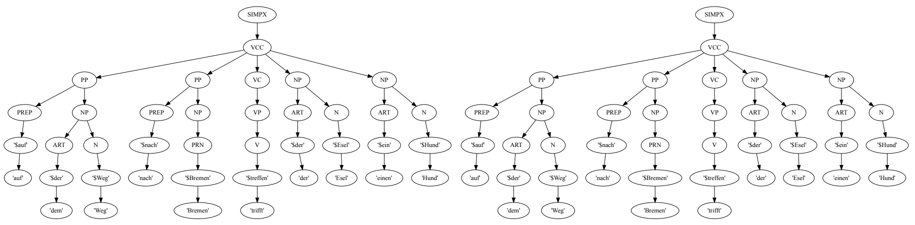

Fig. 5. Comparison of results of parsing the German sentence *Auf dem Weg nach Bremen trifft der Esel einen Hund* when the threshold is set to 50000 — without the A* algorithm (*left*) and using the A* algorithm (*right*).

been translated identically (and correctly) by 10000-threshold A* algorithm, 50000-threshold A* algorithm and 50000-threshold non-A* algorithm, and yet their translation by 10000-threshold non-A* algorithm was broken. The example of parse trees of such sentence is presented on Figures 4 and 5. The existence of sentence translated in such manner may indicate that the number of steps needed to find the best parse using the A* algorithm is smaller than using the parser not equipped with A* tools. The slower spped of the A* parser may result from the implementational matters.

5 Conclusions and Future Work

The implementation of the A* parsing algorithm in *Translatica*'s German parser has not brought the expected performance improvement. There are a few possible reasons for this conclusion.

First, *Translatica*'s parser is not a proper probabilistic context-free grammar. Indeed, it is not even the weighted context-free grammar since it has no special start symbol. Each symbol of the *Translatica*'s grammar can play the role of its start symbol if it is convenient. The input sentence is parsed using the bottom-up method. The parser builds its parse tree upwards until there is no possibility to continue. The last used edge's label becomes *de facto* a start symbol. This approach enables parser to parse the incomplete sentences, but on the other hand, it makes the calculation of the heuristic difficult. Without one designated start symbol, or with many possible start symbols, the calculated heuristic becomes unreliable.

Another reason is that it is not possible to implement the A* parsing algorithm in *Translatica*'s parser without major modification of its structure.

Nevertheless, the attempt to implement the A* algorithm in an existing CYK-based parsed was an innovative and interesting experiment. It brought the experience and large knowledge about probabilistic grammars, A* parsing and various implementational issues.

We expect that future research of A* parsing implementation will bring the desired results: the faster and better translation. The use of modern algorithms and tools to speed up the machine translation is an important matter so the studies on the implementation of modern algorithms in *Translatica*'s parser will be continued. We will focus on efforts to improve the translation performance and to implement the fully functional A* parser.

References

1. Caraballo, S.A., Charniak, E.: New Figures of Merit for Best-First Probabilistic Chart Parsing. Computational Linguistics 24, pp. 275-298 (1998)
2. Charniak, E., Goldwater, S., Johnson, M.: Edge-Based Best-First Chart Parsing. In Proceedings of the Sixth Workshop on Very Large Corpora, pp. 127-133 (1998)
3. Klein, D., Manning, C.D.: A* Parsing: Fast Exact Viterbi Parse Selection. In Proceedings of the Human Language Technology Conference and the North American Association for Computational Linguistics (HLT-NAACL), pp. 119-126 (2003)
4. Manning, C.D., Schütze, H.: Foundations of Statistical Natural Language Processing. Prentice Hall, New Jersey (2000)
5. Ratnaparkhi, A: Learning to Parse Natural Language with Maximum Entropy Models. Machine Learning 34, pp. 151-175 (1999)
6. Roark, B.: Probabilistic Top-Down Parsing and Language Modeling. Computational Linguistics 27, pp. 249-276 (2001)
7. Skórzewski, P.: Efektywny parsing języka naturalnego przy użyciu gramatyk probabilistycznych (Efficient Natural Language Parsing Using Probabilistic Grammars). Master Thesis on Adam Mickiewicz University, Poznań (2010)
8. Skórzewski, P.: Effective Natural Language Parsing With Probabilistic Grammars). In Proceedings of Computational Linguistics – Applications, pp. 175-178. Wisła (2010)