Semiring Parsing

Arnd Hartmanns

Universität des Saarlandes
arnd.hartmanns@gmail.com

Semiring parsing is a framework for parsing probabilistic grammars, simplifying and unifying the computation of both traditional (non-probabilistic) and probabilistic values. I first give an introduction to probabilistic context-free grammars and then summarize and illustrate Joshua Goodman's fundamental work on semiring parsing.

1 Introduction

Semiring parsing allows a unified way to specify parsers, most notably for probabilistic grammars, that can compute a wide range of both probabilistic and non-probabilistic values like recognition, derivation forests, inner probabilities and Viterbi values by using the operations of a corresponding semiring.

Probabilistic grammars are an attractive formalism for parsing natural language because of the combination of the possibility for broad coverage of language phenomena, sufficient distinction of natural and unnatural interpretations, and efficient parsing. One particular probabilistic grammar formalism is probabilistic context-free grammars (PCFGs), which is a simple extension of the well-known context free grammars. PCFGs will be our formalism of choice.

I will first give a motivation for the use of PCFGs, an overview of their most important properties, and illustrative examples. I will then present the semiring parsing framework, mainly summarising and exemplifying the fundamental paper by Joshua Goodman [1].

1 Natural language parsing problems

When parsing natural language, our ultimate goal is to be able to computationally determine the meaning of a sentence. In contrast to the main use of grammars in basic theoretical computer science lectures, we are not only interested in recognition, i.e. deciding whether a given sentence is syntactically correct, but we also need to analyse its structure, which is ideally given by the structure of the corresponding parse trees.

We know that natural language is inherently complex. Most notably, the meaning of words often depends on the context in which they occur in a sentence, and even complete sentences can be ambiguous. Consider the sentence *I saw the man with the telescope* [2]: The phrase *with the telescope* can be interpreted as specifying the in-

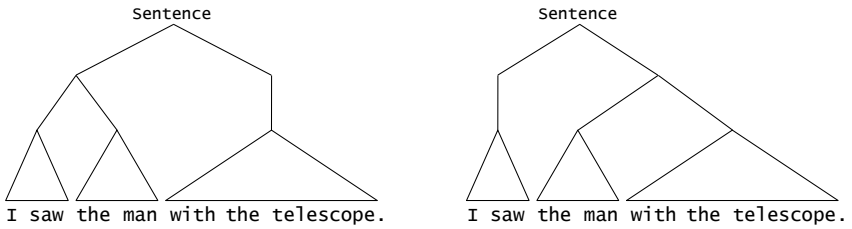


Figure 1: Natural language ambiguities

strument we used to see the man, but also as an attribute of the man, where the man has a telescope.

In this case, the usual interpretation is the one where we use the telescope to see the man. While also valid, the other interpretation is at least less probable in everyday use of the language. On the other hand, if we substitute a rifle for the telescope, thus giving the sentence *I saw the man with the rifle*, the situation will be reversed: In most cases, we expect the man to have a rifle, because seeing with a rifle (though conceivable, e.g. through an attached scope) is unusual.

We thus have a situation where we may get two different but valid interpretations, i.e. two different parse trees (see figure 1), for one sentence. While we might be able to exclude one interpretation in the case of telescopes and the other in the case of rifles by using some strong grammar formalism, e.g. context-sensitive grammars, this would be computationally expensive and prohibit efficient language processing. Because we have already observed that both interpretations are valid, but differ in the amount of occurrences in everyday use, a natural approach to obtain an efficient parser that sufficiently distinguishes the respective interpretations would be to somehow equip parse trees with a probability.

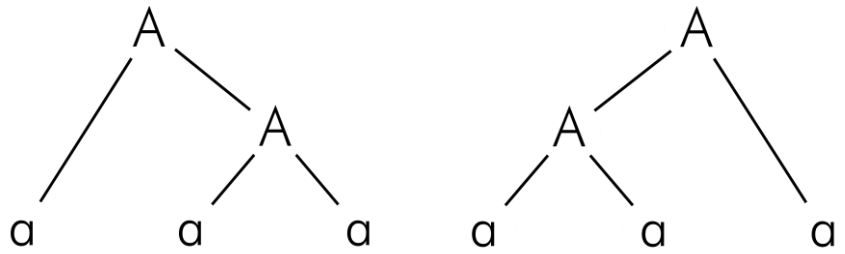
2 Probabilistic Context-Free Grammars

A probabilistic context-free grammar is basically a normal context-free grammar $\mathrm{G} = (\mathrm{N}, \mathrm{T}, \mathrm{S}, \mathrm{R})$ and a probability distribution on the derivations. However, defining this distribution directly, i.e. as a function $\mathbb{E} \to \mathbb{R}[0,1]$ (where $\mathbb{E}$ is the set of all derivations), is complex and conflicts with the grammar's context-free approach: If we know the complete derivation when assigning the probability, we have all possible context at hand.

The PCFG approach is thus to use a function $p\colon \mathrm{R} \to \mathbb{R}[0,1]$ to assign a probability to every rule of the grammar. Intuitively, this makes the decision which rule to use when replacing some nonterminal $A \in \mathrm{N}$ a probabilistic one. In order to obtain a valid probability distribution for this case, we have to make sure that

$$\forall A \in \mathrm{N}\colon \sum_{(A \to \alpha) \in \mathrm{R}} p(A \to \alpha) = 1\,. \tag{1}$$

Grammar rules: $A \to Aa$, $A \to aA$, $A \to aa$
Rule probabilities: $p(A \to Aa) = 0.4$,
$p(A \to aA) = 0.1$,
$p(A \to aa) = 0.5$
Derivations:



$P = 0.1 \times 0.5 = 0.05$ $P = 0.4 \times 0.5 = 0.2$

Figure 2: PCFG derivation probabilities example

At this point, we can use p to obtain the probability distribution on derivations, which we wanted to define in a context-free way: A derivation E as a list of grammar rules is, in the context of p, a series of decisions, so we just have to multiply the grammar rule probabilities to obtain the probability of the derivation:

$$P(E) = \prod_{(A \to \alpha) \in E} p(A \to \alpha) \tag{2}$$

Figure 2 shows a simplified example of a grammar that creates both possible interpretations for the example sentence from section 1 (except that the English sentence fragments have been simplified to a single nonterminal a), but also assigns proper probabilities.

Usually, the rule probabilities will not be assigned by hand. There exist both some large databases of sentences hand-annotated with correct derivation trees and algorithms to compute the probabilities of a given grammar such that the most probable results of the grammar match those given in the database.

2.1 Values of Interest

Aside from properties such as recognition or derivation forests, PCFGs, when used with a fixed input sentence $w_1 \dots w_n$, have a set of interesting probabilistic properties. In the previous section, I have already shown how to compute the probability of a certain derivation; this can easily be extended to the **inside probability**, which is the probability of covering a range of the input starting from a specified nonterminal,

$$\text{inside}(i, A, j) = \Pr\left(A \to^* w_i \dots w_{j-1}\right), \tag{3}$$

and can be computed simply as the sum of the probabilities of all derivations of $w_i \dots w_j$ starting with A.

Another class of interesting values are the **Viterbi** value, the Viterbi derivation, and the Viterbi n-best derivations. All of these make a statement about the most probable derivations: The Viterbi value is the probability of the most probable derivation, while the Viterbi derivation is this derivation itself. Inevitably, the Viterbi n-best derivations are the n most probable derivations for the given input range and starting nonterminal.

Finally, the **outside probability** is, intuitively, the probability of everything surrounding a certain nonterminal, and thus in some way dual to the inside probability. Formally:

$$\text{outside}(i, A, j) = \Pr\left(\text{S} \to^* w_1 \dots w_{i-1}\, A\, w_j \dots w_n\right). \tag{4}$$

While the outside probability is usually more difficult to compute than the other values, there is a convenient relationship to the inside probability when we multiply both [3]:

$$\begin{aligned} &\text{inside}(i, A, j) \times \text{outside}(i, A, j) \\ &= \Pr\left(A \to^* w_i \dots w_{j-1}\right) \times \Pr\left(\text{S} \to^* w_1 \dots w_{i-1}\, A\, w_j \dots w_n\right) \\ &\quad = \Pr\left(\text{S} \to^* w_1 \dots w_{i-1}\, A\, w_j \dots w_n \to^* w_1 \dots w_n\right). \end{aligned} \tag{5}$$

The product of inside and outside probability is therefore the probability of covering the whole input by using a specified nonterminal for a certain range.

For the example in figure 2 and input aaa, we thus have $\text{inside}(1, A, 4) = 0.25$, $\text{viterbi}(1, A, 4) = 0.2$ and $\text{viterbi-derivation}(1, A, 4) = \{\langle A \to Aa, A \to aa \rangle\}$.

3 Semiring Parsing

At this point, we know what (a special form of) probabilistic grammars are, and what values we might want to compute for a given grammar and input. What we do not know yet is how to actually perform these computations. Right now, one might be tempted to just develop an algorithm for inside probabilities, an algorithm for Viterbi values, and one for outside probabilities. In fact, this was basically what happened before semiring parsing: Independent and sometimes quite elaborate algorithms were developed for every value class.

Input: $w_1 \ldots w_n$
Goal item: $[1, \mathrm{S}, n+1]$

Rules: $$\frac{(A \to w_i) \in R}{[i, A, i+1]}, \qquad \frac{(A \to BC) \in R \quad [i, B, m] \quad [m, C, k]}{[i, A, k]}$$

Recognition:

$$(A \to w_i) \in R \Rightarrow V[i, A, i+1] = \text{true}$$
$$(A \to BC) \in R \Rightarrow V[i, A, k] = V[i, A, k] \vee (V[i, B, m] \wedge V[m, C, k])$$
$$\text{success} = V[1, \mathrm{S}, n+1]$$

Inside probabilities:

$$(A \to w_i) \in R \Rightarrow V[i, A, i+1] = p(A \to w_i)$$
$$(A \to BC) \in R \Rightarrow V[i, A, k] = V[i, A, k] + \big(V[i, B, m] \times V[m, C, k] \times p(A \to BC)\big)$$
$$\text{p\_inside} = V[1, \mathrm{S}, n+1]$$

Figure 3: From CKY recognition to inside probabilities

Semiring parsing greatly simplifies the algorithm development: Given a single correct, item-based (or: deductive) parser description, a semiring parser can compute most of the desired values without fundamental changes in the algorithms.

3.1 From Recognition to Inside Probabilities

To illustrate the way semiring parsing is able to achieve this improvement, let us first compare the computation of recognition and inside probabilities with a CKY-style parser.

An abstract representation of the CKY recognition algorithm is given in figure 3: Being able to prove an item $[i, A, j]$ corresponds to being able to derive $w_i \ldots w_j$ starting with nonterminal A; the algorithm itself is then just an implementation of the inference rule semantics.

Note the close correspondence between the items of the CKY parser and the way we defined the inside probability in section 2.1. This will allow a simple modification of the recognition algorithm to turn it into an algorithm for inside probability computation: substituting $+$ and $\times$ for $\vee$ and $\wedge$ and using the rule probability as the "value of a rule" where we previously used true to merely represent the existence of a rule instance. The modified algorithm is also shown in figure 3.

The key idea of semiring parsing is to generalise this substitution such that we can use any complete semiring's operations with many different item-based parsers.

3.2 Semirings

A semiring is a set R with two operations $\oplus$, called addition, and $\otimes$, called multiplication. $\oplus$ has to be associative and commutative, while $\otimes$ has to be associative and distribute over $\oplus$. We also assume identity elements 0 and 1 for $\oplus$ and $\otimes$, respec-

tively, with the expected properties. Intuitively, therefore, a semiring is a ring without the inverse elements for addition.

We might be interested in the computation of sums over an infinite number of elements x, $\bigoplus_{i=1}^{\infty} x$. If infinite sums are well-defined for a particular semiring, we call it *complete*.

If we recall that the set of integers $\mathbb{Z}$ with the usual addition and multiplication operators $+$ and $\times$ is a ring, it is easy to see that by removing all negative numbers, and thus the inverse elements for addition, we obtain a semiring:

$$\langle \mathbb{N}[0,\infty], +, \times, 0, 1 \rangle \text{ is a semiring.} \tag{6}$$

Another semiring would be the interval from 0 to 1 of the real numbers. In this case, we can still use the usual multiplication operator, but this semiring would not be closed under the usual addition operator. We can, however, use the max function instead of addition:

$$\langle \mathbb{R}[0,1], \max, \times, 0, 1 \rangle \text{ is a semiring.} \tag{7}$$

3.3 Grammar and Item Derivations

In order to use arbitrary semirings to compute different kinds of values with a given parser description, we will have to formalise the notion of values introduced informally in section 3.1. The value of a grammar derivation will serve as a formalisation of our intent, while the value of an item derivation according to a parser will be a description of the actual computations to be performed by a semiring parser.

A grammar derivation E corresponds to the parse trees we usually obtain when deriving grammatical sentences. E is a list of grammar rules, $E = e_1 \dots e_m$, that encodes a tree whose leaves are terminals of the input and whose inner nodes are grammar rules. We can obtain this tree from the derivation by agreeing on the convention to use, for example, leftmost derivations.

An item derivation tree $D = \langle I, D_1 \dots D_l \rangle$ represents one possible proof (or computation) of an item I given a certain input: An inner node and its children represent an instance of one of the parser's inference rules, while the leaves of the tree are rules of the underlying grammar.

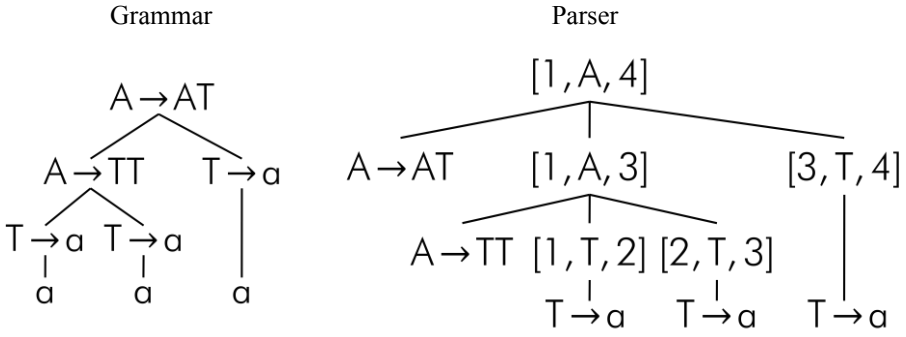


Figure 4: Example grammar and item derivations

An example of a grammar derivation (tree) and an item derivation tree for the input aaa and a Chomsky-normal version of the grammar given in figure 2 can be found in figure 4.

3.4 Semiring Parsing Computations

Recall the computation of the inside probability in section 2.1: We already know how to compute this particular value given a grammar derivation. For arbitrary semirings, we denote the value of a rule e (which, for example, might be a probability) with $R(e)$ and just use whatever multiplication operator there is in the semiring instead of $\times$. We can then extend this to the value of an input sentence by computing the sum over the values of all possible derivations $E_1 \ldots E_k$ of the input:

$$V_G(E) = \bigotimes_{i=1}^{m} R(e_i) \tag{8}$$

$$V_G = \bigoplus_{j=1}^{k} V_G(E_j) \tag{9}$$

At this point, we can specify a semiring parser: Given a parser and an input sentence, we know from section 3.3 that this gives us an item derivation tree where the leaves represent the grammar rules we used in an instance of one of the parser's inference rules; these inference rule instances are represented at the inner nodes. We now want to define the value of an item derivation; naturally, this is the product of all the leaves' values, which can also be written recursively:

$$V(D) = \bigotimes_{d \text{ leaf}} R(d) = \begin{cases} R(d) & \text{(leaf node)} \\ \bigotimes_{i=1}^{l} V(D_i) & \text{(inner node)} \end{cases} \tag{10}$$

This recursive computation closely corresponds to the way the inside probabilities were computed in the modified CKY algorithm in section 3.1; the main difference is that in 3.1, the inside probability was computed for items (most importantly, the goal item) and not only item derivation trees.

Recognition:	$\langle \{\text{true}, \text{false}\}, \vee, \wedge, \text{false}, \text{true} \rangle$
Derivation number:	$\langle \mathbb{N}[0,\infty], +, \times, 0, 1 \rangle$
Derivation forest:	$\langle 2^{\mathbb{E}}, \cup, \cdot, \emptyset, \{\langle\rangle\} \rangle$
Inside probability:	$\langle \mathbb{R}[0,\infty], +, \times, 0, 1 \rangle$
Viterbi:	$\langle \mathbb{R}[0,1], \max, \times, 0, 1 \rangle$
Viterbi-derivation:	$\langle \mathbb{R}[0,1] \times 2^{\mathbb{E}}, \max_{\text{Vit}}, \times_{\text{Vit}}, \langle 0, \emptyset \rangle, \langle 1, \{\langle\rangle\} \rangle \rangle$
Viterbi-n-best:	$\langle \{\text{topn}(X) \mid X \in 2^{\mathbb{R}[0,1] \times \mathbb{E}}\}, \max_{\text{Vit-n}}, \times_{\text{Vit-n}}, \emptyset, \{\langle 1, \{\langle\rangle\} \rangle\} \rangle$

Figure 5: Useful semirings

However, extending the value computations to items is easy: Analogously to the input sentence value computation, we just sum all the values of the item derivation trees $D_1 \ldots D_k$ headed by an item x:

$$V(x) = \bigoplus_{j=1}^{k} V(D_j) \tag{11}$$

It can further be shown [1] that equations (10) and (11) can be combined to a much simpler form that does not internally rely on the notion of item derivation trees, but only needs the parser's inference rules and is thus much easier to compute:

$$V(x) = \bigoplus_{\underset{x}{a_1 \ldots a_k}} \bigotimes_{i=1}^{k} V(a_i) \tag{12}$$

Of course, the actual computations of a semiring parser also have to be done in the right order such that items that are premises of other items are computed first. To achieve this, Goodman uses a notion of ordered buckets that the items are put into; for details, see [1].

3.5 Useful semirings

We have already seen the operations that a semiring parser uses to compute recognition, namely Boolean conjunction and disjunction on the set containing true and false, and to compute inside probabilities, addition and multiplication on the natural numbers including zero and infinity. In the latter semiring, infinity has to be included to allow the value of an unbounded infinite sum to be specified.

Not surprisingly, the Viterbi value, the Viterbi derivation and the Viterbi n-best derivations can also be obtained using an appropriate semiring; see figure 5 for a listing. The Viterbi semiring closely resembles the inside probability semiring, except for the use of max as addition operator – after all, we are only interested in the probability of a single derivation, namely the most probable one – and the range of the real numbers used. In fact, during a semiring parser's calculations with the inside probability semiring, values greater than 1 will never occur, but for the semiring to be closed under addition, these values have to be included nonetheless.

Aside from semirings for the probabilistic values of a grammar and an input sentence, there are also semirings for the traditional non-probabilistic values: we already know the recognition semiring, but there also exist appropriate semirings for the derivation number and the derivation forest (see figure 5). The derivation forest semiring is perhaps the most unusual of the ones we are concerned with as its operations have no direct connection to the usual numeric operations used in the other semirings (except for recognition, which is intuitive).

The Viterbi-derivation semiring is, in essence, a product construction of the Viterbi and the derivation forest semirings. For a definition of the functions used and an explanation of the Viterbi-n-best semiring, see [1].

3.6 Infinite sums

In section 3.2, we mentioned infinite sums and complete semirings, where these sums are well-defined. Infinite sums will occur during semiring parsing whenever there is a loop in the grammar, be it a simple one such as a rule $A \to A$ or a more complex one involving many rules. A semiring parser can detect these loops, because when performing the ordering, there will be so-called looping buckets of inference rules. In this case, an item may have an infinite number of derivations, the sum of whose values has to be computed.

While the necessity to calculate infinite sums, especially when arbitrary semirings are involved, might seem rather intimidating, these computations are, in fact, quite simple for many complete semirings. Still, the precise algorithm depends on the semiring and cannot be dealt with in a general way on the level of the semiring parser. The construction of semiring parsing fortunately allows the concrete algorithms to be treated as a parameter just like the semiring itself.

The necessary infinite sums can be calculated precisely for all the semirings in figure 5 except for the inside probability semiring, where it has to be approximated. I will just give simple intuitions for the calculations for some of the easier semirings; for all details and semirings, again, consult [1].

For the recognition semiring, we observe that one occurrence of true in the sum means that the result value will also be true, since the additive operator is $\vee$. In the case of the derivation number semiring, we can set the value of an item to infinity whenever it is provable and on or depending on a loop, since it will then have infinitely many derivations. To compute an infinite sum in the Viterbi semiring, we can simply "discard" all loops – and therefore avoid "true" infinite sums – because values in a loop will only be multiplied by some probability in $[0,1]$, which means they can only get lower, and will consequently be discarded by the $\max$ operation.

4 Summary

We have seen that parsing and specifically analysing natural language presents several difficulties. Probabilistic grammars, and in particular probabilistic context-free grammars, offer a way to deal with these difficulties in a both appropriate and efficient way by assigning a probability to every interpretation.

Probabilistic context-free grammars, when used on a given input, have some interesting values that are useful during parsing as well as when trying to improve the rule probabilities with some derivation database. In order to compute these values, historically, specialized algorithms were developed for each case. These were often substantially different, in part also because of infinite sum computations being interweaved with the algorithm.

Semiring parsing now allows us to specify an abstract, item-based parser and use this single specification to compute most of the interesting values by simply substituting different semirings. Even the computation of infinite sums, while semiring-specific, is factored out of the main semiring parser and can therefore be handled independently of the parsing algorithm.

Semiring parsing can thus be applied to many parsers of different styles – even parsers for mildly context-sensitive grammar formalisms such as tree adjoining grammars (TAGs) – and is, all in all, a substantial improvement for the development of probabilistic parsing.

References

[1] Goodman, J.T.: Semiring parsing. Computational Linguistics 25(4) (1999) 573–605
[2] Geman, S., Johnson, M.: Probability and statistics in computational linguistics, a brief review. In Johnson, M., Khudanpur, S., Ostendorf, M., Rosenfeld, R., eds.: Mathematical foundations of speech and language processing. Volume 138 of IMA Volumes in Mathematics and its Applications. Springer-Verlag, New York (2003) 1–26
[3] Goodman, J.T.: Parsing inside-out. PhD thesis (1998) Adviser-Stuart Shieber.