Generation of Hypergraphs from the N-Best Parsing of 2D-Probabilistic Context-Free Grammars for Mathematical Expression Recognition

Ernesto Noya, Joan Andreu Sánchez and José Miguel Benedí
PRHLT Research Center
Universitat Politècnica de València, Spain
{noya.ernesto, jandreu, jmbenedi}@prhlt.upv.es

***Abstract*—We consider hypergraphs as a tool obtained with bi-dimensional Probabilistic Context-Free Grammars to compactly represent the result of the n-best parse trees for an input image that represents a mathematical expression. More specifically, in this paper we propose: i) an algorithm to compute the N-best parse trees from a 2D-PCFGs, ii) an algorithm to represent the n-best parse trees using a compact representation in the form of hypergraphs, and iii) a formal framework for the development of inference algorithms (*inside* and *outside*) and normalization strategies of hypergraphs.**

***Index Terms*—Mathematical expression recognition, Probabilistic Context-Free Grammars, N-best parse trees, Hypergraphs.**

I. Introduction

Nowadays, large repositories with printed scientific documents are available worldwide from different research fields in Engineering, Computer Science, Physics, etc. Hundreds of thousands of these documents include mathematical expressions (MEs). Locating scientific information in articles and books placed in public[1] or private servers owned by universities, editorials, libraries, etc. is a usual activity that concerns many scientific disciplines. The searching of textual scientific information in these documents is currently a possibility widely exploited by the search engines of the most used web browsers. However, the searching in massive collections of digitized printed scientific documents with MEs queries, is a research area not sufficiently explored.

An appropriate alternative is to develop search engines with queries of MEs according to the rendered image rather than the encoding version of the ME given the ambiguity conveyed by the encoding version. The recently researched methods that address this problem are based on comparing images. These methods are not realistic for searching in massive collections. The reason is that their interpretations are not precise enough because they do not take into account the context nor the structure of the image. Furthermore, if the matching problem is approached in this way, then searching MEs in an electronic document (e.g. pdf documents) resembles the issue of searching a given object in an image (e.g. a car in the street). But a remarkable difference with the previous scenario is that, in the case of MEs, two MEs written with different symbols can have the same meaning, and consequently, they are the same ME. For example, if we have a look at these two MEs:

$$\sum_{i=0}^{\infty} \frac{1}{2^i} \qquad \sum_{j=0}^{\infty} 1/2^j$$

We observe that both of them represent the same concept. In these examples, both i and j are exchanging variables that do not change the meaning of the expression. Note also that comparing both images with traditional image-based retrieval techniques will show some differences because of the misplacements. This last problem, when searching ME in huge collections, can be alleviated by the context around the ME using a language model. Bi-dimensional Probabilistic Context-Free Grammars (2D-PCFG) are a formalism that has demonstrated to be appropriate to address math expressions recognition [1]. In this paper, we consider this formalism for ME parsing.

Recently, a new approach has been introduced for searching words in massive collections of historical handwritten document images [2], [3]. This approach does not require any kind of segmentation, and it does not need to have the complete and error-free transcription of the images [4]. To address the ambiguity problem, this approach uses language models to take advantage of the context. To reduce the search time in the exploitation phase, a two-phase solution is proposed. In the first offline phase, the posterior probabilities of words are calculated from word graphs derived from a handwritten text recognition process. In a second online phase, these posteriors are used for indexing and searching for words in the collection. The fundamental tool in this approach are weighted word graphs computed with finite-state models, in which the weights are carefully processed as probabilities. Each path through the word graph represents a possible transcript and/or segmentation alternative [5]. This approach has been tested in real scenarios and massive repositories have started to include this solution among their services.[2]

In the case of key math expression spotting in printed document images, and by analogy with the previous approach of keyword indexing and search, a two-phase solution was

[1] https://arxiv.org/

[2] http://prhlt-kws.prhlt.upv.es/fcr/

also considered. In this paper, we focus on the first offline phase where the aim is to calculate the posterior probabilities of a math expression image. For this purpose, we define the *hypergraphs* [6], as a compact model derived from the parsing process with 2D-PCFG, similar to word graphs for handwritten document images. The hypergraphs provide a more compact representation of the parsing space represented by the n-best parse trees from a 2D-PCFG for a ME image. Where each of the n-best parse trees represents a possible interpretation of the ME image and therefore, a specific path in the hypergraph. Furthermore, the hypergraphs can generalize the information contained in the n-best parse trees, (as will be discussed in Section IV-B). Specifically, in this paper, we propose to research and extend the theoretical framework introduced in [2], [3], [4] before getting the posterior probabilities of MEs for document images. The contributions of this paper include: i) the generation of the n-best parse trees from a 2D-PCFGs, in a similar way as explained in [7]; ii) the generation of hypergraphs from the n-best parse trees; iii) the inference algorithms necessary to normalize the hypergraph.

II. Related work

ME searching in collections of typeset documents has been researched in the past for datasets of moderate size [8]. Recent papers state the problem of ME detection in datasets with less than 50 scientific documents. Locating isolated MEs seems an easy task, but locating embedded ME presents a considerable challenge since they can be confused with running text, as it is shown in [9]. This last paper does not state the problem of searching ME according to their contents. Searching for ME taking into account their contents requires understanding the ME [10]. Current methods for parsing ME are far from providing results without errors, both lexical and syntactical. Even if an error-free ME recognition system was developed, the possible mark-up language allows the same ME to be represented in several different ways, as we illustrate in the Introduction section.

Searching for information in vast collections of difficult handwritten document images is currently feasible, as some recent researches have demonstrated [4]. Probabilistic indexing has allowed searching in collections with hundreds of thousands of document images. ME searching in the collection of typeset documents can be developed in a similar way, but searching techniques must take into account that the same ME can be represented differently with small variations. 2D-PCFGs is a powerful formalism that can be used for parsing ME [1]. Several parse trees can account for the same ME and therefore, 2D-PCFGs can deal with the ambiguity associated with a ME. Merging several parse trees in a graph structure has been researched in the past [6], [11].

III. N-best parsing of 2D-probabilistic context-free grammars

In printed mathematical expression recognition (MER), the input is an image or a region of an image. Given an input image, the first step is to consider a possible representation function that maps the image to another representation. In our case, the representation function extracts the set of connected components, $\boldsymbol{x} = \{x_1, \ldots, x_{|\boldsymbol{x}|}\}$, from the input image. Figure 1 shows an image example of an input ME and its set of associated connected components. We pose the MER

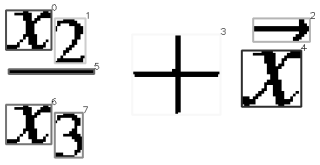

Fig. 1. Example of input image, $\frac{x_2}{x_3} + \vec{x}$, and the set of associated connected components.

as a structural parsing problem, such that the main goal is to obtain the set of symbols and the structure that defines the relationships among them from an input image. Formally, let $\boldsymbol{x}$ be a set of connected components from an input ME. The aim is to obtain the most likely sequence of mathematical symbols $\boldsymbol{s} \in \mathcal{S}$ related among them according to the most likely syntactic parse $\boldsymbol{t} \in \mathcal{T}$ that accounts for $\boldsymbol{x}$, where $\mathcal{S}$ is the set of all possible sequences of (pre-terminals) symbols and $\mathcal{T}$ represents the set of all possible syntactic parses, such that $\boldsymbol{s} = \text{yield}(\boldsymbol{t})$. This can be approximated as follows:

$$\hat{\boldsymbol{t}} = \underset{\boldsymbol{t}\in\mathcal{T}}{\arg\max}\ p(\boldsymbol{t}|\boldsymbol{x}) = \underset{\boldsymbol{t}\in\mathcal{T}}{\arg\max} \sum_{\substack{\boldsymbol{s}\in\mathcal{S} \\ \boldsymbol{s}=\text{yield}(\boldsymbol{t})}} p(\boldsymbol{t}, \boldsymbol{s} \mid \boldsymbol{x})$$

This is approximated as:

$$(\hat{\boldsymbol{t}}, \hat{\boldsymbol{s}}) \approx \underset{\substack{\boldsymbol{t}\in\mathcal{T},\boldsymbol{s}\in\mathcal{S} \\ \boldsymbol{s}=\text{yield}(\boldsymbol{t})}}{\arg\max}\ p(\boldsymbol{s} \mid \boldsymbol{x}) \cdot p(\boldsymbol{t} \mid \boldsymbol{s}) \tag{1}$$

where $p(\boldsymbol{s}|\boldsymbol{x})$ represents the observation (symbol) likelihood and $p(\boldsymbol{t}|\boldsymbol{s})$ represents the structural probability.

Two strategies are often used to address this problem. The first one solves the problem in two steps. In a first step, by calculating the segmentation of the input into mathematical symbols, and a second, by computing the structure that relates all recognized symbols [12]. The second one solves the problem through a fully integrated strategy for computing Eq. (1) where symbol segmentation, symbol recognition, and the structural analysis of the input expression are globally achieved [1]. This second option is considered in this paper. Specifically, we will consider the formulation of Eq. (1) as a search problem.

A. Notation and Problem Formulation

Probabilistic Context-Free Grammars (PCFGs) are a powerful formalism for syntactic pattern recognition. These models are appropriate for capturing long-term dependencies between the different elements in a ME. In this paper, we consider a two-dimensional extension of PCFG, a well-known formalism widely used for MER [13], [1].

Definition 1. *A* Context-Free Grammar *(CFG), G, is a four-tuple $(\mathcal{N}, \Sigma, S, \mathcal{P})$, where $\mathcal{N}$ is a finite set of non-terminal symbols, Σ is a finite set of terminal symbols ($\mathcal{N} \cap \Sigma = \emptyset$), $S \in \mathcal{N}$ is the start symbol of the grammar, and $\mathcal{P}$ is a finite set of rules: $A \to \alpha$, $A \in \mathcal{N}$, $\alpha \in (\mathcal{N} \cup \Sigma)^+$.*

A CFG in Chomsky Normal Form (CNF) is a CFG in which the rules are of the form $A \to BC$ or $A \to a$, where $A, B, C \in \mathcal{N}$ and $a \in \Sigma$.

Definition 2. *A* Probabilistic CFG *(PCFG) is defined as a pair (G, p), where G is a CFG and $p : P \to]0, 1]$ is a probability function of rule application such that $\forall A \in \mathcal{N} : \sum_{i=1}^{n_A} p(A \to \alpha_i) = 1$, where n_A is the number of rules associated with non-terminal symbol A.*

Definition 3. *A* Two-Dimensional PCFG *(2D-PCFG), $\mathcal{G}$, is a generalization of a PCFG, where terminal and non-terminal symbols describe two-dimensional regions. This grammar in CNF results in two types of rules: terminal rules and binary rules.*

First, the terminal rules $A \to a$ represent the mathematical symbols which are ultimately the terminal symbols of 2D-PCFG. Second, the binary rules $A \xrightarrow{r} BC$ have an additional parameter, r, that represents a given spatial relationship, and its interpretation is that regions B and C must be spatially arranged according to the spatial relationship r.

Definition 4. *Let $\mathcal{G}$ be a 2D-PCFG, and let $\boldsymbol{x}$ be a set of connected components. First, we define $\mathcal{T}(A, a)$ as the set of trees whose root is (A, a), where $A \in \mathcal{N}$ is a non-terminal symbol and $a \in \wp(\boldsymbol{x})$ is a certain input span. $T(A, a) \in \mathcal{T}(A, a)$, represents one of the possible trees in $\mathcal{T}(A, a)$. Second, we define a probabilistic function $p : \mathcal{T} \to [0, 1]$, as follows:*

***(I)** If there is a terminal rule $A \to s$ and a set with just one connected component, $\{x_i\}, 1 \le i \le |\boldsymbol{x}|$, then we have the tree $\langle A, \{x_i\}\rangle \in \mathcal{T}(A, \{x_i\})$, and its probability is:*

$$p(\langle A, \{x_i\}\rangle) = \sum_{s \in \Sigma} p(s \mid A)\, p(\{x_i\} \mid s) \approx \frac{1}{|\boldsymbol{x}|} \max_s \left\{ \frac{p(s \mid A)\, p(s \mid \{x_i\})}{p(s)} \right\} \quad (2)$$

where $p(s|A)$ is the probability of the terminal rule, $p(s|\{x_i\})$ is the probability provided by a symbol classifier, and $p(s)$ is the prior probability of the symbols. We consider here all connected components to be equiprobable.

***(II)** If there is a binary rule $A \to BC$, a tree $T_1(B, b)$ in $\mathcal{T}(B, b)$ and a tree $T_2(C, c)$ in $\mathcal{T}(C, c)$, such that $b \cap c = \emptyset$ and $a = b \cup c$, then we refer the tree $\langle T(A, a), T_1, T_2\rangle$ as the tree with root A, left subtree T_1, and right subtree T_2, and its probability is approximated as:*

$$p(\langle (A, a), T_1, T_2\rangle) \approx \max_r p(r \mid BC)\, p(BC \mid A)\, p(T_1)\, p(T_2) \quad (3)$$

where $p(BC|A)$ is the probability of the binary rule, and $p(r|B, C)$ is the probability that regions encoded by non-terminals B and C are arranged according to spatial relationship r.

A *parse tree* for $\boldsymbol{x}$ according to $\mathcal{G}$ is a binary tree, $T(S, \boldsymbol{x}) \in \mathcal{T}(S, \boldsymbol{x})$. The *best parse tree* is the parse tree with maximum probability. The N *best parse trees* are the N parse trees with maximum total probability. Finally, let us denote $T^n(A, a)$ as the n-th best tree among those in $\mathcal{T}(A, a)$. The problem is then finding $T^1(S, \boldsymbol{x}), T^2(S, \boldsymbol{x}), \ldots, T^N(S, \boldsymbol{x})$.

B. Best Parse Tree

The computation of the N-best parse tree from 2D-PCFGs for MEs has been designed as a two-step process. First, we calculate the best parse tree for the input ME, and then, taking advantage of the structures created in this first step, we compute the remaining of N-best parse trees.

Following [1], where a parsing algorithm from 2D-PCFGs for on-line MER is presented, we propose a modification of this algorithm to compute the best parse trees based on the following recursive equations.

First, let us define $\gamma^1(A, a)$ is the probability that $A \in \mathcal{N}$ is the 1-best solution of the connected component set a.

Initialization. According to Eq. (2) and given that the symbol classifier can provide several recognition alternatives, with different non-terminal symbols $A, A \in \mathcal{N}$, for every connected component $\{x_i\}, 1 \le i \le |\boldsymbol{x}|$, let us define,

$$\gamma^1(A, \{x_i\}) = \max_{T \in \mathcal{T}^1(A, \{x_i\})} p(T)$$
$$T^1(A, \{x_i\}) = \arg\max_{T \in \mathcal{T}^1(A, \{x_i\})} p(T)$$

The set of trees $\mathcal{T}^1(A, \{x_i\})$ whose roots are of the form $(A, \{x_i\})$ can be define as,

$$\mathcal{T}^1(A, \{x_i\}) = \{\, \langle A, \{x_i\}\rangle : A \to s;\ p(s \mid \{x_i\}) > 0 \,\} \quad (4)$$

Recursion. According to Eq. (3), $\forall A \in \mathcal{N}$ and $\forall a \in \wp(\boldsymbol{x})$, such that $|a| > 1$:

$$\gamma^1(A, a) = \max_{T \in \mathcal{T}^1(A, a)} p(T)$$
$$T^1(A, a) = \arg\max_{T \in \mathcal{T}^1(A, a)} p(T) \quad (5)$$

where, for $a \in \wp(\boldsymbol{x})$ such that $|a| > 1$,

$$\mathcal{T}^1(A, a) = \{\, \langle (A, a), T^1(B, b), T^1(C, c)\rangle : A \to BC;\ b \cap c = \emptyset;\ a = b \cup c \,\} \quad (6)$$

denotes the set of candidate trees with root A that comprises the connected components a.

Assuming the pruning strategies presented in [1], the time complexity of this algorithm is $O(|P||\boldsymbol{x}|^3 \log |x|)$.

C. N-Best Parsing

The second step of the process takes profit of the structures created in the computation of the 1-best parse tree (see Eq. (4) and (6)). For the computation of the N-best parse tree, we follow [7] to compute the recursive enumeration of the best parse trees, that take advantage of the partial order between different candidate trees. Next, we will generalize the recursive equations given in the previous section to compute the N-best parse trees, and then we will propose an algorithm to solve the generalized equations.

As mentioned above, the problem is to obtain $T^n(S, \boldsymbol{x})$ for $1 \leq n \leq N$. For this, we must define $T^n(A, a)$, $1 \leq n \leq N$, $A \in \mathcal{N}$ and $a \in \wp(\boldsymbol{x})$ as the n-th best tree among those that have root A and the set of connected components a. Therefore, $T^n(A, a)$ can be chosen as the best tree different from $T^1(A, a), \ldots, T^{n-1}(A, a)$ in the set $\mathcal{T}^n(A, a)$, which is defined below.

Analogously to the case of 1-best (see Eq. (4)) and taking into account Eq. (2), we can define $\mathcal{T}^n(A, \{x_i\})$ for all $A \in \mathcal{N}$, $1 \leq i \leq |\boldsymbol{x}|$, and $1 < n \leq N$, as,

$$\mathcal{T}^n(A, \{x_i\}) \;=\; \mathcal{T}^{n-1}(A, \{x_i\}) - \{T^{n-1}(A, \{x_i\})\} \qquad (7)$$

Furthermore, for $|a| > 1$ and considering Eq. (6), let us define $\mathcal{T}^n(A, a)$ as,

$$\begin{aligned} \mathcal{T}^n(A, a) = (&\mathcal{T}^{n-1}(A, a) - \{T^{n-1}(A, a)\}) \\ &\cup \{\langle T(A, a), T^{p+1}(B, b), T^q(C, c)\rangle\} \\ &\cup \{\langle T(A, a), T^p(B, b), T^{q+1}(C, c)\rangle\} \end{aligned} \qquad (8)$$

where $A \to BC$; $a, b, c \in \wp(\boldsymbol{x})$; $b \cap c = \emptyset$; $a = b \cup c$ and $1 \leq p, q \leq n$. Values p and q are used to keep track of previous solutions that have been used in the past. That is, if $T^p(B, b)$ has been used in a previous tree, then $T^{p+1}(B, b)$ is needed and it is combined with $T^q(C, c)$.

Following [7] and assuming that $\{\langle T(A, a), T_1, T_2\rangle\}$ denotes the empty set if T_1 or T_2 does not exist. Then we have that:

$$\begin{aligned} \gamma^n(A, a)^n &= \max_{T \in \mathcal{T}^n(A,a)} p(T) \\ T^n(A, a) &= \arg\max_{T \in \mathcal{T}^n(A,a)} p(T) \end{aligned} \qquad (9)$$

Therefore, the problem of computing the N-best parse trees for an image consists in using equations (4) and (5) to obtain $T^1(S, \boldsymbol{x})$ and, taking advantage of the calculations made, using equations (7) and (8) to find $T^2(S, \boldsymbol{x}), \ldots, T^N(S, \boldsymbol{x})$. The algorithm that is shown in Figure 2 solves the equations for increasing values of n, recursively starting from the tree $T^n(S, \boldsymbol{x})$. The algorithm makes use of the recursive procedure `NextTree()`, for $n > 1$, and once $T^n(S, \boldsymbol{x}) =$ `NextTree`$(T^{n-1}(S, \boldsymbol{x}), n)$ is available, `NextTree`$(\langle T(A, a), T^p(B, b), T^q(C, c)\rangle, n)$ computes $T^n(S, \boldsymbol{x})$ according to equation (8). This requires two new candidates trees $T^{p+1}(B, b)$ and $T^{q+1}(C, c)$, if any of this candidates has not been computed before we call the function `NexTree` recursively on it. An example of the solutions produced can be seen in Figure 3.

The 2D-CYK algorithm runs in time $O(|P||x|^3 \log |x|)$. The number of different sets $\mathcal{T}(A, a)$ is $O(|a|^2|\Sigma|)$ and, in the worst case, all of them must be initialized in linear time with respect to the size of the set. The computation of the N-best parse trees requires no more than $N|x|$ calls to $NextTree$. The total time required by the whole algorithm to compute the N best parse trees is $O(|P||x|^3 \log |x| + N|x| \log(|x| + N))$. This computational complexity analysis is based on worst case assumptions that could be too pessimistic. In practice, it can be expected that even for large values of N, not all the sets of candidates are initialized and the number of recursive calls would be much lower than $N|x|$.

```
T^1(S, x) = CYK(x)
for n=2 to N:
  T^n(S, x) = NextTree(T^{n-1}(S, x), n)
return {T^1(S, x), T^2(S, x), ..., T^N(S, x)}

def NextTree(⟨T(A, a), T^p(B, b), T^q(C, c)⟩, n):
  T^n(A, a) = T^{n-1}(A, a) − {T^{n-1}(A, a)}
  if |a| > 1:
    if T^{p+1}(B, b) not set:
      NextTree(T^p(B, b), p + 1)
    if T^{p+1}(B, b) != None:
      p = p(BC | A) · p(r | BC) · γ^{p+1}(B, b) · γ^q(C, c)
      if p > 0.0:
        T^n(A, a) = T^n(A, a) ∪
            {⟨T(A, a), T^{p+1}(B, b), T^q(C, c)⟩}
    if T^{q+1}(C, c) not set:
      NextTree(T^q(C, c), q + 1)
    if T^{q+1}(C, c) != None:
      p = p(BC | A) · p(r | BC) · γ^p(B, b) · γ^{q+1}(C, c)
      if p > 0.0:
        T^n(A, a) = T^n(A, a) ∪
            {⟨T(A, a), T^p(B, b), T^{q+1}(C, c)⟩}
  if T^n(A, a) != None:
    γ^n(A, a) = max_{T∈T^n(A,a)} p(T)
    T^n(A, a) = arg max_{T∈T^n(A,a)} p(T)
  else T^n(A, a) = None
```

Fig. 2. Algorithm to generate n-best parse trees for an input expression.

IV. Parsing and hypergraphs

A hypergraph is a generalization of the graph, where the arcs (now called hyperarcs) may connect several nodes at the same time. We use hypergraphs as a representation of the result of an N-best parsing with a given 2D-PCFG. Given a certain parse tree, the non-terminals represent the nodes and the rules represent the hyperarcs. Figure 3 in the left shows the hypergraph associated with the 1-best parse tree for the expression $\frac{x_2}{x_3} + \vec{x}$. Nodes `Term_H`, `LetterMin` and `Digit` are non-terminals of the 2D-PCFG and the hyperarc that relates them is the rule $(Term\_H \to LetterMin\ Digit)$. Each hyperarc has an associated weight that is related to the probability of the rule.

We intend to use hypergraphs to represent the n-best parse trees for a 2D-PCFG. Nodes and hyperarcs may be shared among different parse trees if they represent the same information. Thus, hypergraphs provide a compact representation of the parsing space. As we will see below, this compact representation allows us to develop efficient inference algorithms.

A. Notation

First, we give some preliminary definitions of hypergraphs. For more detailed information see [14].

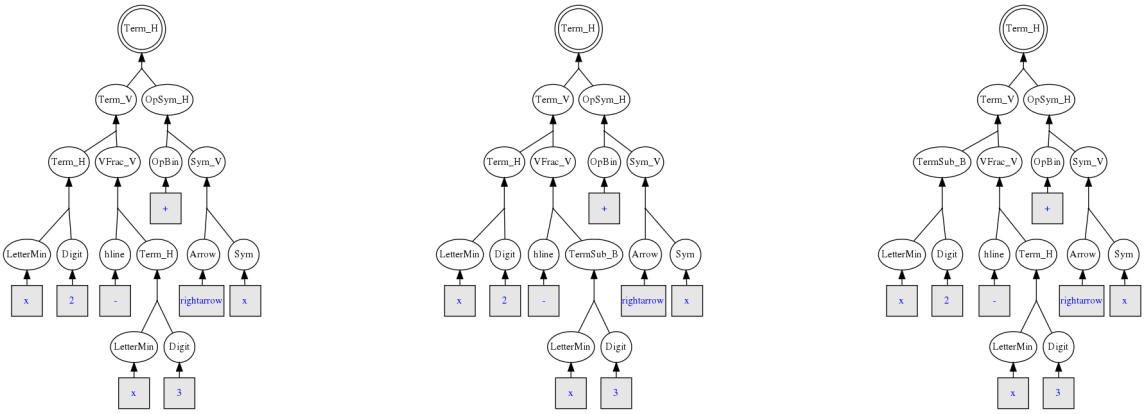

Fig. 3. 3-best parse trees for input expression, $\frac{x_2}{x_3} + \vec{x}$, and the expressions associated with the interpretations of these 3-best solutions.

Definition 5. *A weight directed hypergraph (hereinafter hypergraph) $\mathcal{H}$ is a pair $(\mathcal{V}, \mathcal{E})$ where $\mathcal{V}$ is a set of nodes (or vertices) and $\mathcal{E}$ is a set of directed hyperarcs (or hyperedges).*

Since we want to use the hypergraphs as a compact representation of the n-best parse trees for a 2D-PCFG, $\mathcal{G} = (\mathcal{N}, \Sigma, S, \mathcal{P})$, we re-define the notions of node and hyperarc.

Definition 6. *A node $v \in \mathcal{V}$ is a pair $(n(v), s(v))$ where $n(v)$ is the node tag and $s(v)$ is the span (or position information) associated with this node.*

In our case, the span, $s(v) = a$, represents the set of connected components associated to the node tag, $n(v)$. Therefore, if $n(v) = A \in \mathcal{N}$ (non-terminal symbol) then v is an *internal node*. On the other hand, if $n(v) = A \in \Sigma$ (terminal symbol) then v is an *leaf node*.

Definition 7. *Given $\mathcal{H}$, a hyperarc $e \in \mathcal{E}$ is a tuple $(H(e), T(e),\ t(e), p(e))$ where the tail $T(e)$ and head $H(e)$ are subsets of $\mathcal{V}$, $t(e)$ is a transcription associated with the hyperarc and $p(e)$ is a score (or weight).*

Given that 2D-PCFGs considered are in *Chomsky Normal Form*, the head $H(e)$ contains exactly one (internal) node and the tail $T(e)$ two (internal) nodes to model binary rules ($A \rightarrow BC$), or one (leaf) node to model unary rules ($A \rightarrow x$). This is a particular case of hyperarc *B-arc* defined in [14]. In our case, $t(e)$ is the LaTeX transcript representing the semantics associated with the hyperarc. Finally, $p(e)$ is the arc score. In our case, the arc score represents a probability. Depending on whether e represents a unary or binary rule, its probability is obtained from Eq. (2) or Eq. (3) respectively,

$$\forall e \in \mathcal{E}:\ n(H(e)) = A;\ n(T(e)) = (s);\ s(H(e)) = \{x_i\};$$
$$p(e) = \frac{p(s \mid A)\ p(s \mid \{x_i\})\ p(\{x_i\})}{p(s)} \tag{10}$$

$$\forall e \in \mathcal{E}:\ n(H(e) = A;\ n(T(e)) = (B, C);\ s(H(e)) = a$$
$$p(e) = p(r \mid BC)\ p(BC \mid A) \tag{11}$$

Definition 8. *A complete tree $\boldsymbol{t}$ of a hypergraph $\mathcal{H}$ is a sequence of hyperarcs in which there is a root node (and only one) that completely covers the input ME represented by $\boldsymbol{x}$.*

Let $\psi(\boldsymbol{t})$ be the set of all hyperarcs that make up $\boldsymbol{t}$ in $\mathcal{H}$, there must be a hyperarc, and only one, $e \in \psi(t)$ that accomplishes $n(H(e)) = S \in \mathcal{N}$, where S is the axiom of the 2D-PCFG; and $s(H(e)) = \boldsymbol{x}$, where $\boldsymbol{x}$ is the complete set of connected components that represents the input ME. The probability of a tree is the product of the probabilities of all the hyperarcs that compose it. Thus, for a given set of connected components $\boldsymbol{x}$, the joint probability $p(\boldsymbol{x}, \boldsymbol{t})$ can be approximated from a hypergraph $\mathcal{H}$ as:

$$p(\boldsymbol{t}, \boldsymbol{x}) \approx \prod_{e \in \psi(\boldsymbol{t})} p(e) \stackrel{\text{def}}{=} p_{\mathcal{H}}(\boldsymbol{t}, \boldsymbol{x}) \tag{12}$$

A hypergraph $\mathcal{H}$ should typically contain the majority of the most probable decoding hypotheses considered in the maximization of Eq. (9), including the best hypothesis. Therefore, the unconditional probability of $\boldsymbol{x}$ can be approximated by the accumulated probability of all complete trees represented in $\mathcal{H}$:

$$p(\boldsymbol{x}) \approx \sum_{\boldsymbol{t}} p_{\mathcal{H}}(\boldsymbol{t}, \boldsymbol{x}) \stackrel{\text{def}}{=} p_{\mathcal{H}}(\boldsymbol{x}) \tag{13}$$

This expression can be very efficiently computed using dynamic programming, as we will see in Sec. IV-C.

B. Generation of hypergraphs

Once the n-best parse trees for an input expression are generated, in order to get the complete hypergraph $\mathcal{H} = (\mathcal{V}, \mathcal{E})$ we need to go through each tree, saving all nodes and rules in a graph structure, merging the terminals and pre-terminals representing the same sets of connected components. From the N-best parse trees for the input ME, represented by $\boldsymbol{x}$, algorithm in Figure 4 computes the set of nodes and hyperarcs of the hypergraph $\mathcal{H}$

Figure 5 shows the hypergraph obtained by applying the algorithm of Figure 4, only for the 3-best parse trees of expression, $\frac{x_2}{x_3} + \vec{x}$, shown in Figure 3. It should be noted that

```
Compute:  T^1(S,x),...,T^n(S,x)
V={}   // Nodes
Hu={}  // unary hyperarcs
Hb={}  // binary hyperarcs
for all T^i(S,x) in T^1(S,x),...,T^N(S,x):
  addTree(T^i(S,x))
return (V,[Hu+Hb])
def addTree(⟨T(A,a),T^p(B,b),T^q(C,c)⟩):
  i = addNode(A,a)
  if |a| == 1:
    j = addNode(s,a)
    if (i(A) → j("s"),p) not in Hu:
      Hu[|Hu|] = (i(A) → j("s"),p))
  else:
    j = addNode(B,b)
    k = addNode(C,c)
    if (i(A) → j(B)k(C),p)) not in Hb:
      Hb[|Hb|] = (i(A) → j(B)k(C),p))
    addTree(T^p(B,b))
    addTree(T^q(C,c))
def addNode(X,x):
  for i = 0...|V|:
    if (X,x) in V[i]:
      return i
  V[|V|] = (X,x)
  return |V| − 1
```

Fig. 4. Algorithm to generate a hypergraph using the n-best parse trees.

the true solution is not among these 3-best parse trees (in fact it is the 4-best parse tree). However, as also it happens with the word graphs [2], the hypergraphs allow to generalize to the n-best parse trees. As can be seen in Figure 5, the hypergraph generated only with the 3-best parse trees also contains the true solution.

C. Algorithms

An efficient way to calculate the total probability of Eq. (13) is to generalize the well-known Inside and Outside algorithms for PCFGs [15] and extend them for 2D-PCFGs and hypergraphs.

1) INSIDE *Algorithm for hypergraphs:*

Given $\mathcal{H}$ and the set of connected components $\boldsymbol{x}$, associated with the input ME, for each of the nodes of $v \in \mathcal{V}$ of $\mathcal{H}$, we can define,

$$\alpha_{\mathcal{H}}(A,a) \stackrel{\text{def}}{=} p(A \stackrel{*}{\Rightarrow} a), \tag{14}$$

as the probability that $A = n(v)$ is a solution of the set of connected components, $a = s(v)$.

Initialization. Given equations (2) and (10) for unary hyperarcs, $\forall e \in \mathcal{E}$ with $H(e) = v \in \mathcal{V}$ and $T(e) = s \in \Sigma$, and considering that $n(v) = A \in \mathcal{N}$; $s(v) = \{x_i\}$ and $1 \leq i \leq |\boldsymbol{x}|$, we can define,

$$\alpha_{\mathcal{H}}(A,\{x_i\}) = \sum_{e \in \mathcal{E}} p(e)$$

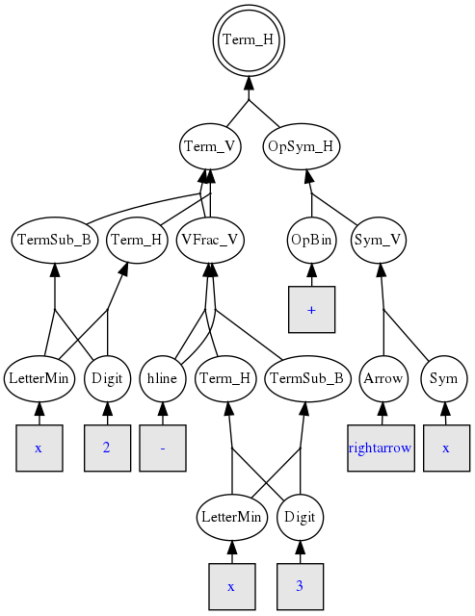

Fig. 5. Hypergraph obtained from the 3-best parse trees (Figure 3) for the input expression, $\frac{x_2}{x_3} + \vec{x}$.

Recursion. Given equations (3) and (11) for the binary hyperarcs, $\forall e \in \mathcal{E}$ with $H(e) = v$ and $T(e) = (m,n)$: $v, m, n \in \mathcal{V}$, and considering that $n(v) = A$, $n(m) = B$ and $n(n) = C$ are the node tags and $s(v) = a$, $s(m) = b$, and $s(n) = c$ are the node spans, that must satisfy $b \cap c = \emptyset$ and $b \cup c = a$, we can define,

$$\alpha_{\mathcal{H}}(A,a) = \sum_{e \in \mathcal{E}} \alpha_{\mathcal{H}}(B,b)\ \alpha_{\mathcal{H}}(C,c)\ p(e) \tag{15}$$

Finally, the probability $p_{\mathcal{H}}(z)$, defined in Eq. (13), can be computed as,

$$p_{\mathcal{H}}(z) = \alpha_{\mathcal{H}}(S,\boldsymbol{x}) \tag{16}$$

where S it is the axiom of grammar and $\boldsymbol{x}$ is the set of connected components of the input ME.

2) OUTSIDE *Algorithm for Hypergraphs:*

Similarly, given $\mathcal{H}$ and the set of connected components $\boldsymbol{x}$, associated with the input ME, for each of the nodes of $v \in \mathcal{V}$ of $\mathcal{H}$, we can define,

$$\beta_{\mathcal{H}}(A,a) \stackrel{\text{def}}{=} p(S \stackrel{*}{\Rightarrow} g\,A\,u), \tag{17}$$

as the probability that $A = n(v)$ is a solution of the set of connected components, $a = s(v)$, and the remaining of $\mathcal{H}$ correctly explains the connected components that are not in a, where g and u are two disjoint sets of connected components that meet that $g \cup a \cup u = \boldsymbol{x}$.

Initialization. For the (only) root node of $\mathcal{H}$, where its tag is the axiom S of the grammar, and $\boldsymbol{x}$ is the complete set of connected components associated with the input ME, we define,

$$\beta_{\mathcal{H}}(S,\boldsymbol{x}) = 1$$

Recursion. For every pair of binary arcs $e, e' \in \mathcal{E}$ in the hypergraph $\mathcal{H}$ that satisfy respectively $H(e) = H(e') = m$, $T(e) = (v, n)$ and $T(e') = (n, v)$ with $v, m, n \in \mathcal{V}$. In addition, $n(v) = A$, $n(m) = B$ and $n(n) = C$ are the node tags and $s(v) = a$, $s(m) = b$ and $s(n) = c$ are the node spans, that must satisfy $a \cap c = \emptyset$ and $a \cup c = b$, we can define.

$$\beta_{\mathcal{H}}(A, a) = \sum_{e \in \mathcal{E}} \beta_{\mathcal{H}}(B, b)\ \alpha_{\mathcal{H}}(C, c)\ p(e) + \quad (18)$$

$$\sum_{e' \in \mathcal{E}} \beta_{\mathcal{H}}(B, b)\ \alpha_{\mathcal{H}}(C, c)\ p(e') \quad (19)$$

Finally, the probability $p_{\mathcal{H}}(z)$, defined in Eq. 13, can be also computed as,

$$p_{\mathcal{H}}(\boldsymbol{x}) = \sum_{e \in \mathcal{E}} \beta_{\mathcal{H}}(A, \{x_i\})\ \alpha_{\mathcal{H}}(A, \{x_i\}) \quad (20)$$

For $1 \leq i \leq |\boldsymbol{x}|$, where $H(e) = v$ with $n(v) = A$ and $s(v) = \{x_i\}$.

D. Hypergraph Normalization

In this section will address the issue of normalization of hyperarcs of hypergraph. To do this we start by defining the tree posterior probability associated with a hypergraph $\mathcal{H}$.

Definition 9. *Given a hypergraph $\mathcal{H}$, the tree posterior probability $p(\boldsymbol{t}|\boldsymbol{x})$ can be approximately computed as:*

$$p(\boldsymbol{t}|\boldsymbol{x}) \approx \frac{p_{\mathcal{H}}(\boldsymbol{t}, \boldsymbol{x})}{p_{\mathcal{H}}(\boldsymbol{x})} \stackrel{def}{=} p_{\mathcal{H}}(\boldsymbol{t}|\boldsymbol{x}) \quad (21)$$

Similarly as introduced in [2] for the case of word lattices, the scores (or probabilities) of hyperarcs of a hypergraph can be normalized in several ways. For the purposes of this paper, we need the hyperarcs scores to be normalized as follows:

Definition 10. *For a given hyperarc $e \in \mathcal{E}$, its normalized score, $\varphi(e)$, is defined as the sum of posterior probabilities of all the complete parsing trees which include arc e. That is:*

$$\varphi(e) \stackrel{def}{=} \sum_{\boldsymbol{t}: e \in \psi(\boldsymbol{t})} p_{\mathcal{H}}(\boldsymbol{t}|\boldsymbol{x}) \quad (22)$$

An example of normalized hypergraph can be seen in Figure 5. For a hypergraph normalized in this way, the following properties can be shown:

Theorem 1. *Given a hypergraph $\mathcal{H} = (\mathcal{V}, \mathcal{E})$, the normalized score of a hyperarc $\varphi(e) : e \in \mathcal{E}$ can be efficiently computed by the expression,*

$$\varphi(e) = \frac{\alpha_{\mathcal{H}}(A, a)\ \beta_{\mathcal{H}}(A, a)}{\alpha_{\mathcal{H}}(S, \boldsymbol{x})} \quad (23)$$

Proof: Given $\mathcal{H}$, $\forall e \in \mathcal{E}$, Eq. (22) indicates that the sum must be made for all the parse trees $\boldsymbol{t}$ in $\mathcal{H}$, where the hyperarc e appears in a certain position. Therefore, considering Eq. (21) and Eq. (16), then Eq. (22) can be written as:

$$\sum_{\boldsymbol{t}: e \in \psi(\boldsymbol{t})} p_{\mathcal{H}}(\boldsymbol{t}|\boldsymbol{x}) = \frac{1}{\alpha_{\mathcal{H}}(S, \boldsymbol{x})} \sum_{\boldsymbol{t}: e \in \psi(\boldsymbol{t})} p_{\mathcal{H}}(\boldsymbol{t}, \boldsymbol{x}) \quad (24)$$

where $p_{\mathcal{H}}(\boldsymbol{t}, \boldsymbol{x})$ is the product of probabilities of the hyperarcs of the parse tree $\boldsymbol{t}$, with the restriction that the hyperarc e is present in $\boldsymbol{t}$. Given $H(e) = v$ is the head node, where $n(v) = A$ is the node tag and $s(v) = a$ is the span associated with this node, we have that:

$$\sum_{\boldsymbol{t}: e \in \psi(\boldsymbol{t})} p_{\mathcal{H}}(\boldsymbol{t}, \boldsymbol{x}) = p(S \stackrel{*}{\Rightarrow} g\, A\, u)\ p(A \stackrel{*}{\Rightarrow} a)$$

where $g \cap a \cap u = \emptyset$ and $g \cup a \cup u = \boldsymbol{x}$. Considering the definitions of inside probability (Eq. (14)) and outside probability (Eq. (17)) and arc normalized score (Eq. 22), the Eq. (24) can be rewritten as,

$$\sum_{\boldsymbol{t}: e \in \Psi(\boldsymbol{t})} p_{\mathcal{H}}(\boldsymbol{t}|\boldsymbol{x}) = \frac{\alpha_{\mathcal{H}}(A, a) \cdot \beta_{\mathcal{H}}(A, a)}{\alpha_{\mathcal{H}}(S, \boldsymbol{x})} = \varphi(e)$$

■

V. Experimental evaluation of the algorithms

We developed a MER system based on 2D-PCFG that computes the hypergraph for an input image using the equations and algorithms previously presented . In this section, we study the time behaviour of the algorithms depending on the size of the MEs, the spatial relationships and the number of N-best trees generated when creating the hypergraph.

In order to evaluate the effect that both the size of the ME and the number of relationships among connected components have on the execution time, we synthesized two different types of MEs for illustrating the best and worst-case scenarios. Figures 6 and 7 were made using simple MEs, where only a horizontal spatial relationship was considered. Figure 8 was made using a more complex template where the ME grows both horizontally and vertically giving a more pessimistic case for execution time. When increasing the number of parse trees generated for a ME, we can study the effect in small expressions that may have a small number of different parse trees, and long expressions with many more possible trees.[3]

Figure 6 shows the mean execution time when computing the 1-best tree on MEs of increasing length. The ordinate axis represents the execution time in seconds on a logarithmic scale and the abscissa axis represents the size $|\boldsymbol{x}|$ of the input MEs. The plot shows the theoretical time complexity in blue, and the behaviour in practice with a black dotted line.

Figure 7 shows the effect of increasing the number of parse trees to generate, N, on simple MEs of sizes $10, 25$ and 50. The ordinate axis represents the execution time in seconds and the abscissa axis represents the number of parse trees that are generated, both in logarithmic scale. The time for computing the 1-best solution is excluded from this plot.

Figure 8 shows the mean execution time when computing the 1-best tree on MEs of increasing size for complex MEs. The ordinate axis represents the execution time in seconds on a logarithmic scale and the abscissa axis represents the size $|x|$

[3]All the experiments have been run on a 3 GHz Intel(R) Core(TM) i5-6600K CPU computer running under Ubuntu 18.04. The algorithm has been implemented in C++ and compiled with g++ 7.5.0.

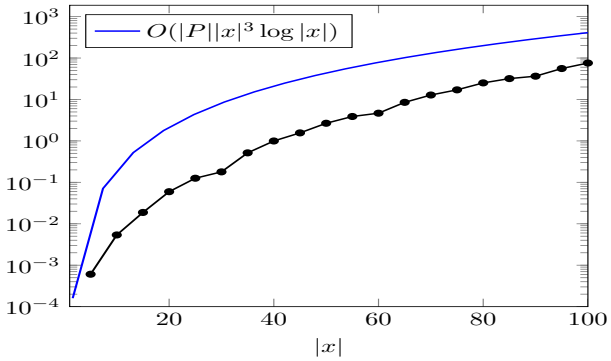

Fig. 6. 1-best for $|x|$=[5, 100] on horizontal relationships.

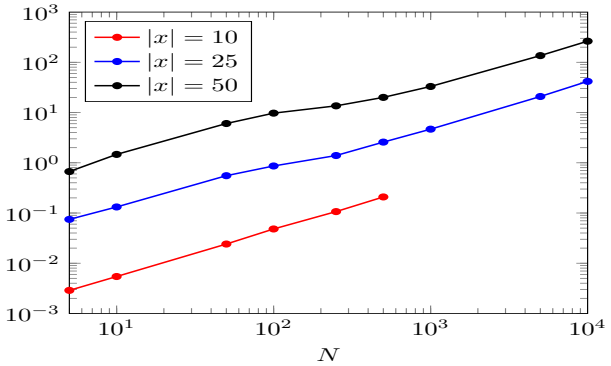

Fig. 7. N-best execution time for different values of N on expressions of different sizes $|x|$. The red expression of $|x| = 10$ has a maximum of 500 hypothesis.

of the input ME. As in the previous experiment, the plot in blue represents the theoretical time complexity and the behaviour in practice is shown as a black dotted line.

In these experiments, we can see that the 1-best parsing algorithm is the most expensive procedure. The results for generating the N-best solutions have a more linear scaling. In realistic MEs of length $|x| = 25$ the time needed to compute the first 100 trees is between 1-2 sec. making it usable in real-world applications.

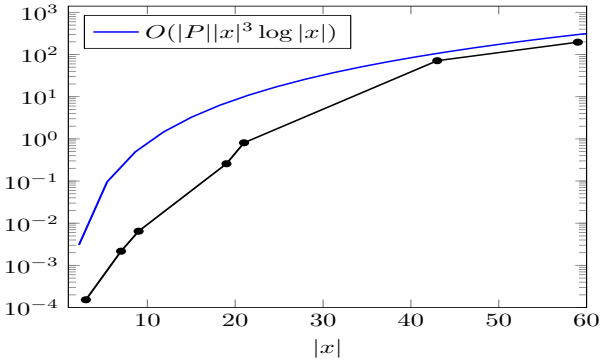

Fig. 8. 1-best for $|x| = \{3, 7, 9, 19, 21, 43, 59\}$ on horizontal and vertical relationships.

VI. Conclusion

In this paper, we have presented a proposal for generating hypergraphs from the N-Best parsing of 2D-PCFGs for ME recognition. Hypergraphs are an efficient interface between mathematical expression recognition and the indexing and searching systems of mathematical expressions. More specifically, in this paper we propose an algorithm to compute the N-best parse trees from a 2D-PCFGs, an algorithm to represent the n-best parse trees using a compact representation in the form of hypergraphs, and a formal framework for the development of inference algorithms (*inside* and *outside*) and normalization strategies of hypergraphs. In addition, some preliminary experiments have been reported to check the behavior of the proposed algorithms. In the future, we plan to apply these results on indexing and searching problems of mathematical expressions in large collections of digitized images.

Acknowledgment

This work has been partially supported by the Ministerio de Ciencia y Tecnología under the grant TIN2017-91452-EXP (IBEM) and by the Generalitat Valenciana under the grant PROMETEO/2019/121 (DeepPattern).

References

[1] F. Álvaro, J. A. Sánchez, and J. M. Benedí, "An integrated grammar-based approach for mathematical expression recognition," *Pattern Recognition*, vol. 51, pp. 135–147, 2016.

[2] A. Toselli, E. Vidal, V. Romero, and V. Frinken, "HMM word graph based keyword spotting in handwritten document images," *Information Sciences*, vol. 370, pp. 497–518, 2016.

[3] A. H. Toselli, V. Romero, E. Vidal, and J. Sánchez, "Making two vast historical manuscript collections searchable and extracting meaningful textual features through large-scale probabilistic indexing," in *2019 15th IAPR Int. Conf. on Document Analysis and Recognition (ICDAR)*, 2019.

[4] E. Vidal, "Text search and information retrieval in large historical collections of untranscribed manuscripts"," Invited key note talk at International Conference on Document Analysis and Recognition (ICDAR), 2019.

[5] S. Ortmanns, H. Ney, and X. Aubert, "A word graph algorithm for large vocabulary continuous speech recognition," *Computer Speech and Language*, vol. 11, no. 1, pp. 43–72, 1997.

[6] D. Klein and C. D. Manning, "Parsing and hypergraphs," in *In IWPT*. Association for Computational Linguistics, 2001, pp. 123–134.

[7] V. Jiménez and A. Marzal, "Computation of the n best parse trees for weighted and stochastic context-free grammars," in *Advances in Pattern Recognition*, F. Ferri, J. Iñesta, A. Amin, and P. Pudil, Eds. Springer Berlin Heidelberg, 2000, pp. 183–192.

[8] R. Zanibbi and D. Blostein, "Recognition and retrieval of mathematical expressions," *International Journal of Document Analysis and Recognition*, vol. 15, pp. 331–357, 2012.

[9] M. Mahdavi, R. Zanibbi, H. Mouchère, C. Viard-Gaudin, and U. Garain, "ICDAR 2019 CROHME + TFD: Competition on recognition of handwritten mathematical expressions and typeset formula detection," in *International Conference on Document Analysis and Recognition*, 2019.

[10] W. Zhong, S. Rohatgi, J. Wu, C. Giles, , and R. Zanibbi, "Accelerating substructure similarity search for formula retrieval," in *Proc. European Conference on Information Retrieval*, 2020.

[11] L. Huang and D. Chiang, "Better k-best parsing," in *In IWPT*. Association for Computational Linguistics, Oct. 2005, pp. 53–64.

[12] R. Zanibbi, D. Blostein, and J. Cordy, "Recognizing mathematical expressions using tree transformation," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 24, no. 11, pp. 1–13, 2002.

[13] A.-M. Awal, H. Mouchere, and C. Viard-Gaudin, "A global learning approach for an online handwritten mathematical expression recognition system," *Pattern Recognition Letters*, vol. 35, no. 0, pp. 68 – 77, 2014.

[14] G. Gallo, G. Longo, S. Pallottino, and S. Nguyen, "Directed hypergraphs and applications," *Discrete Appl. Math.*, vol. 42, no. 2–3, pp. 177–201, 1993.

[15] C. D. Manning and H. Schütze, *Foundations of Statistical Natural Language Processing*. MIT Press, 1999.