ABSTRACT

In natural languages, the words within an utterance are often correlated over large distances. Long-spanning contextual effects of this type cannot be efficiently and robustly captured by the traditional \( N \)-gram approaches of stochastic language modelling. We present a new kind of stochastic grammar — the permugram model. A permugram model is obtained by linear interpolation of a large number of conventional bigram, trigram, or polygram models which operate on different permutations of the input word sequence under consideration. This way, stochastic dependences between word pairs or word triples lying non-contiguous in the input text can be captured simply by choosing the appropriate permutation — this “choice” is of course a random process — that brings the respective word items into touch.

In this paper, we present a new kind of stochastic grammar — the permugram model — which is obtained by linear interpolation of a large number of conventional bigram, trigram, or polygram models which operate on different permutations of the input word sequence under consideration. This way, stochastic dependences between word pairs or word triples lying non-contiguous in the input text can be captured simply by choosing the appropriate permutation — this “choice” is of course a random process — that brings the respective word items into touch.

The remainder of the paper is organized as follows: section 2 reviews the formalism of polygram models. Permutations of the input word order are introduced in the central section 3, most of this part of the paper deals with the definition of local representations of permutations (“configurations”) and a generalization of discrete-output HMM’s (“hidden permutation model”). Finally, sections 4 and 5 will present the experimental results and a conclusion.

1. INTRODUCTION

In natural languages, the words within an utterance are often correlated over large distances; for instance, this is the case as a result of the insertion of subordinate clauses. Another example is the tight grammatical coupling between the parts of discontinuous verbal phrases which are very common in German language:

\[
\begin{array}{|c|c|c|c|}
\hline
\text{Der Einkauf} & \text{fährt} & \text{kommt} & \text{läuft} \\
\text{am Bahnsteig} & \text{zwei} & \text{ab} & \text{an} \\
\text{ein} & \text{ein} & \text{ein} & \text{ein} \\
\hline
\end{array}
\]

Long-spanning contextual effects of this type cannot be efficiently and robustly captured by the traditional \( N \)-gram approaches (see [6] for a survey) of stochastic language modelling. Even for moderate positional distances between grammatically related words, the estimates of higher-order \( N \)-grams are required in order to bridge the interword gaps, and the model runs into the danger of combinatorial explosion and overadaptation. The same argument is true for the polygram interpolation technique [8] as well as other smoothing and backing-off procedures [7, 14], which might avoid the overadaptation effect, but are still restricted to contiguous word histories when estimating conditional word probabilities.

In order to circumvent this problem, one may try to reduce the dimensionality of the parameter space by introducing gaps or wildcards into the word \( N \)-grams under consideration. A simple heuristic is to replace conditional \( N \)-grams by weighted averages of gappy bigrams \( P(w_i | w_{i-1}) \). This approach was followed in [12] and [13], using different approximations for the marginal distribution \( P(w_i | w_{i-1}) \). The notion of gappy bigrams was extended to \( N \)-grams in [4], and in [11] a product expression for the approximation of conditional \( N \)-grams by their marginals was proposed. Another approach is to employ a tree classifier to approximate the required probabilities [1], or to configure the desired statistical (in)dependences into a causal Bayesian network [9].

The joint distribution \( P(w_1, \ldots, w_T) \) for a given sequence \( w = w_1, \ldots, w_T \) of words from a vocabulary \( \mathcal{V} \) of size \( L \) may be written as a conditional decomposition

\[
P(w_1, \ldots, w_T) = \prod_{t=1}^{T} P(w_t | w_1, \ldots, w_{t-1})
\]

where the function \( \#(\cdot) \) counts the frequency of a given word sequence in the training corpus. Unfortunately, even for small \( L \) the above frequency statistics are far from being reliable estimates. Smoothing of these statistics can be achieved by pruning the word histories or by partitioning the vocabulary into word categories. The polygram model [8] does without history pruning and evaluates the conditional \( N \)-gram probabilities by the linear interpolation formula

\[
P(w_t | w_1, \ldots, w_{t-1}) = \rho_1 P(w_t | w_1, \ldots, w_{t-1}) + \ldots + \rho_T P(w_t | w_1) + \rho_0 P_{\text{uniform}}
\]

where \( \rho_1, \ldots, \rho_T \) are iteratively optimized by the estimation-maximization (EM) algorithm [2, 6] using a cross-validation data set.

Since the interpolation weights tend to become very small for higher-order \( N \)-gram statistics if estimated
globally, a dependence of the weights on the actual word history is introduced: we let \( \rho_t = \rho_t(\eta) \) with
\[
\eta = \max \{ \nu \mid \#(w_{-\nu}, \ldots, w_{-1}) \neq 0 \} \tag{4}
\]
In order to limit the storage requirements of the model and to avoid overadaptation to the training set, a suitable upper bound \( N \) for the maximum order of \( N \)-grams to consider in the model should be chosen.

Polygram models have been introduced in [8, 16]; a similar approach using heuristically determined interpolation weights was presented in [15], too.

**3. PERMUTATIONS**

Assume we are going to rearrange the word order in \( w_1, \ldots, w_T \) according to a permutation
\[
\pi : \{1, \ldots, T \} \rightarrow \{1, \ldots, T \}
\tag{5}
\]
The conditional decomposition of \( P(w) \) remains valid after reordering of the input sequence:
\[
P(w) = P_\pi(w) = P(w_{\pi(1)}, \ldots, w_{\pi(T)})
= \prod_{r=1}^T P(w_{\pi(r)} \mid w_{\pi(1)}, \ldots, w_{\pi(r-1)}) \tag{6}
\]
Note that in the above equation the expression \( w_{\pi(r)} \) is a shorthand for \( w_{\pi(1)}, \ldots, w_{\pi(r)} \), denoting the event that word item \( w_{\pi(r)} \) happens to occur in sentence position \( \pi(i) \). In subsequent formulas we will drop the random variable \( \pi \) provided that the subscripts of \( \pi \) and \( \pi \) coincide. For instance, if \( \pi(1) = 3 \) and \( \pi(2) = 3 \), \( P(w_{\pi(2)}) \) refers to the unigram distribution of all words in the fifth sentence position, and \( P(w_{\pi(2)} \mid \pi(1)) \) denotes the statistical dependence between two non-adjacent word items moving backward in time.

As a matter of fact, the identity \( P(w) = P_\pi(w) \) becomes inapplicable as soon as the permuted \( N \)-gram probabilities are replaced by the polygram-like interpolation rule (3). Consequently, it is tempting to formulate a (perfect) permigram model by the linear combination
\[
P(w) = \sum_{\pi \in \mathcal{P}} \lambda_{\pi} P_\pi(w) \tag{7}
\]
of permutation-dependent joint probability estimates, ranging over the set \( \mathcal{P} \) of all possible permutations of \( \{1, \ldots, T\} \). Observe that our permigram formula in fact incorporates conditional bigram probabilities \( P(w_{i,j}) \), for each possible pair \( i, j \) of relative word positions; the same is true for trigrams, and tetragrams, and so forth. Consequently, stochastic dependences between word pairs or triples lying widely separated in the input text can already be modelled without the need of higher-order statistics of the word generation process.

The essential challenge in permgram modelling is to check the combinatorial explosion caused by the vast amount of theoretically possible sentence permutations. We shall particularize in the remainder of this section how this task has been solved by selecting an appropriate subset of \( \mathcal{P} \) and rearranging its elements in a probabilistic finite state network.

The key idea comes from the observation that sentence probabilities with respect to similar permutations usually share several of their product terms, too. This fact is best exploited by recombining the partial probability products of competing permutations in a dynamic programing manner.

**3.1. Coinciding local probabilities**

At first glance it may appear that the local probability scores of permutations \( \pi \) and \( \pi \) at time \( t \) just coincide if and only if the actual as well as all past positions of \( \pi \) and \( \pi \) coincide, i.e., if \( \pi(s) = \pi(s) \) for each \( s \leq t \) is valid. Fortunately, two formal properties of our stochastic language models, limited model order and homogeneity, allow a much larger degree of coincidence.

A model order of \( N \) restricts the word history of conditional probabilities to the last \( N - 1 \) input positions; “last” refers to the process time \( t \) rather than word order. Accordingly, the condition \( \pi(s) = \pi(s) \) for \( t - N < s \leq t \) is now sufficient for coincidence.

Homogeneity in word position assumes that only relative word positions matter; for instance, we do not want to distinguish between probabilities \( P(w_2 \mid w_1 = w_1) \) and \( P(w_2 = w_2) \); this assumption is used in traditional \( N \)-gram models, too. Homogeneity in process time states complete independence of \( t \) which is already implicit in our notation; in other words, the value \( P(c_1 \mid \pi = w_1) \) is independent of the time \( t \) when \( \pi \) has reached sentence positions \( 6 \) or \( 7 \).

**3.2. Configurations**

A configuration as defined below is meant to describe the situation we encounter when the conditional \( N \)-gram probability of a word \( w_{\pi(t)} \) is to be computed. This includes information about the head position \( \pi(t) \), the sentence positions of the last \( N - 1 \) history words, and markers indicating which sentence positions have already been processed and which have not.

We define an \( N \)-gram configuration to be a finite string \( c = [c_1, \ldots, c_N] \) from the alphabet \( \{1, \ldots, N, \emptyset\} \). The items \( c_k \) describe the state of processing of particular sentence positions from left to right in the natural word order. Each of the numbers \( 1, \ldots, N \) have to appear exactly once in the string; \( N \) denotes the head word position, whereas each \( \emptyset \) refers to the significant history positions. The marker \( \emptyset \) is attached to places which have not yet been reached by the word production process; any number (including zero) of \( \emptyset \)'s may appear in \( c \), and trailing \( \emptyset \)'s are omitted. The marker \( \emptyset \) indicates positions which have been encountered in a very early phase of the sentence generation process and which have now left the scope of \( N \)-gram memory. Just like the open positions \( \emptyset \), an arbitrary number of closed (or forgotten) positions \( \emptyset \) may occur; however, this time we will drop leading occurrences of \( \emptyset \)-markers.

It is the latter arrangement that explicates the homogeneity assumptions discussed above, and which makes the absolute reference positions of the markers \( c_k \) implicit, because we cannot figure out the number of pruned positions to the left of \( c_1 \). The example configurations
\[
\begin{array}{c}
1 \ 2 \ 3 \\
2 \ 3 \ 1
\end{array}
\tag{8}
\]
denote conditional probabilities of type \( P(w_{i,j} \mid w_{i+1}, \ldots, w_{j-1}) \) or \( P(w_{i,j} \mid w_{i,j+1}) \), respectively. The former is a standard trigram probability. The latter one is non-standard since it incorporates gaps and order inversion; moreover, we are informed that all words left to \( w_{i+1} \) as well as \( w_{j+1} \) have been processed and forgotten, and words \( w_{i,j} \) as well as \( w_{i,j+1} \) and beyond have not yet been visited.

**3.3. Hidden permutation models (HIM)**

By means of configurations, which represent the local probability contribution of word permutations, we are able to define a doubly stochastic process that generates sentences according to a convex combination of permuted polygram models. Of course, only a restricted class of subsets of all possible permutations can be realized in a finite state process of manageable size.

A hidden permutation model (HIM) consists of a set \( \{S_1, \ldots, S_M\} \) of states \( S \), with associated configurations \( \xi(S) \). The non-deterministic sentence production process starts in \( S_1 \) (with the associated start configuration \( \xi(S_1) = [1] \) and reads from \( S \) to \( S \), with probability \( a_1 \). The state identity at time \( t \) is hidden to the observer, but using the configuration attached to the state
and its corresponding permutation, an open word position \( w_{o(t)} \) is filled according to the selected conditional (non-standard) N-gram distribution.

It is another important feature of the H/M that state transitions \( i \rightarrow j \) are illegal unless \( \sigma(S) \) is a possible successor of \( \sigma(S) \). The successor relation between configurations can be summarized as follows: we may expand an N-gram configuration towards an \((N+1)\)-gram configuration by replacing one of its \( \square \)'s by the new lead marker \((N+1)\); note that there are infinitely many implicit \( \square \)'s in the right hand side continuation of the configuration string. From the above trigram examples, the tetragrams

\[
\begin{align*}
1 & 2 & 3 & 4 \\
2 & 4 & 1 & 3
\end{align*}
\]

(9)
can be deduced. Moreover, it is possible to deduce the configuration order in the same step. A reduction from \((N+1)\) to \( N \) is technically achieved by subtracting 1 from all positive \( c_i \)'s, then replacing the unique item \( c_i = 0 \) by \( c_i = \square \) and finally removing leading \( \square \)'s. For the standard tetragram above, successive order reductions lead to the standard configurations

\[
\begin{align*}
1 & 2 & 3 & 4 \\
1 & 2 & 3 & 1 \\
1 & 2 & 1 & 3 \\
1 & 2 & 1 & 1 \; \text{and} \; 1 & 2 & 1 & 0
\end{align*}
\]

(10)
the non-standard tetragram reduces to

\[
\begin{align*}
1 & 2 & 3 & 4 \\
1 & 2 & 3 & 1 \\
1 & 2 & 1 & 3 \\
1 & 2 & 0 & 3 \; \text{and} \; 1 & 2 & 0 & 2
\end{align*}
\]

(11)
To summarize, we call the \( N \)-gram configuration \( \sigma \) a legal successor of the \( N \)-gram configuration \( \sigma' \) if and only if \( \sigma \) results from one expansion of \( \sigma' \) as well as \( N+1 \rightarrow N \) subsequent order reductions. By the shift \( \tau(\sigma, \sigma') \) we denote the total number of closed position markers \( \square \) that have been extinguished from the intermediate configuration strings during that transformation. The shifts of standard trigrams are, for example, \( \tau([123], [1234]) = 0 \), \( \tau([123], [123]) = 1 \), and so forth.

If \( t \) was the absolute word position pointed to by the first entry \( c_1 \) of \( \sigma \), then \( t' = t + \tau(\sigma, \sigma') \) is the absolute position related to (the onset of) configuration \( \sigma' \). Assuming that the start configuration \( \theta(S_0) \) of the model points to the initial sentence position \( u_1 \), a consistent sentence-configuration alignment is guaranteed during the entire H/M word production process.

Quite similar to H/M's, the probability that the H/MM produces a word sequence \( w_n \) can be obtained by forward recursions; additionally, ML estimates of the transition probabilities \( \alpha(r, i) \) and \( \beta(r, i) \) are provided by running the EM algorithm (for further details consult [3]). A probability distribution based on an H/MM as defined above will be called a permgram language model.

3.4. Sentence boundaries
In order to capture the particular word statistics at or near sentence boundaries, auxiliary vocabulary items for the out-of-sentence positions are introduced, and every input sentence is expanded into an infinite word series \( \bar{w}_n = \bar{w}_1 \bar{w}_2 \ldots \bar{w}_n \bar{w}_n+1 \bar{w}_n+2 \bar{w}_n+3 \bar{w}_n+4 \ldots \) with respect to the permgram model, the additional boundary markers \( \$ \), \( \# \) are treated just like ordinary words.

Formally, the conditional decomposition \( P(w | \bar{w}) \) turns into an infinite probability product for the enlarged word sequences. However, it is easily shown that all but a finite number of factors may be omitted from the product; actually, each conditional probability of the form

\[
P(c_i = 1 | \bar{w} \sigma_n) = \rho_i o(t) r_{i,t} \quad 1 \leq r \leq N
\]

assumes the value 1 if only \( w = \bar{s}_{t-r+1} \) is valid and 0 else.

4. EXPERIMENTAL RESULTS
In order to compare the permgram approach with traditional N-gram models, test set perplexities have been computed for two German language modelling tasks, using a collection of standard as well as non-standard models.

4.1. Data
Two German text corpora served as basis for our experiments: the Interley corpus is made up of train timetable inquiries, whilst the Verbmobil data contain transcriptions of face-to-face negotiation dialogs [17]. The total number of dialog turns, words, and the vocabulary size of both text collections is given in Table 1.

<table>
<thead>
<tr>
<th>Corpus</th>
<th># turns</th>
<th># words</th>
<th>Vocabulary size</th>
</tr>
</thead>
<tbody>
<tr>
<td>Interley</td>
<td>2553</td>
<td>13,368</td>
<td>271</td>
</tr>
<tr>
<td>Verbmobil</td>
<td>2031</td>
<td>65,578</td>
<td>2112</td>
</tr>
</tbody>
</table>

Table 1. Statistics of test corpora used in the experiments

Each data set is divided into three disjoint subsets: the training data (\( \approx 80\% \)) are used to estimate all non-standard N-gram counts, the cross-validation data (\( \approx 10\% \)) are fed into the EM algorithm in order to optimize the permutation weights \( a_j \) and the configuration-dependent interpolation parameters \( \rho_i o(t), \# \), and the test data (\( \approx 10\% \)) serve to compute the test set perplexities.

4.2. Permgram configurations
The standard polygram (or interpolated N-gram) model according to eqs. (1), (2), (3) is a special case of a permgram model; the H/MM’s for bigrams and trigrams are shown in Figure 1. The numbers outside the configuration boxes indicate the sentence position shift related to an H/MM transition. Both models describe a deterministic word production process, moving through the word sequence in original order (with identity permutation); from the beginning of the sentence, the configuration order is increased by one from position to position until the model order \( N \) (2 and 3 in the examples) is arrived at.

\[
\begin{align*}
1 & 2 & 3 \\
1 & 2 & 3 \\
1 & 2 & 3 \\
1 & 2 & 3
\end{align*}
\]

Figure 1. H/MM representation of bigrams (l) and trigrams (r)

Perhaps the simplest way to characterize (a generic set of) permutations of arbitrary length with finite means is to choose a periodic repetition in time, consisting of multiple copies of a small local index permutation pattern. An obvious candidate is the meandric pattern shown in Figure 2.

\[
\begin{align*}
1 & 2 & 3
\end{align*}
\]

Figure 2. Meandric permutation pattern

The dashed box delimits exactly one period of the meandric permutation. Beginning with the leftmost word position \( t \), \( k \) places are skipped and position \( t + k + 1 \) is occupied. Subsequently, each of the places skipped before is processed stepwise from left to right until the entire sequence of word positions \( t + 1 \) through \( t + k \) has been covered. The H/MM corresponding to a meandric permutation with context order \( N = 2 \) and gap size \( k = 2 \) is found in Figure 3. We denote permgram models of this type by \( M_N^\lambda \).

In order to allow more flexibility in word order rearrangement, mixture models of meandric forms were defined. Particularly, we considered permgram models \( M_N^\lambda \) which are the convex combination of the permuted N-gram models \( M_N^\lambda \), \( \lambda = 1, \ldots, k \) together with the standard N-gram. Note that the H/MM of this mixture is easily constructed by connecting in parallel the H/MM’s of all the submodels \( M_N^\lambda \).

The most complex H/MM structure that has been tested allows the production process to choose freely any sequence of meander periods. This model is built by providing possible transitions from each \( M_N^\lambda \) offset towards...
each $M^N_2$ onset in the above mixture HMM; we will refer to this "connected meander" model as $I^N_2$.

### 4.3. Perplexity figures

Test set perplexities were calculated for models with bigram ($N=2$) and trigram ($N=3$) context. Besides the standard interpolated $N$-grams, single meander ($M^N_2$), mixture meander ($\Sigma^N_2$), and connected meander ($\Pi^N_2$) models have been run, the maximum gap size $k$ ranging between 1 and 6. The two rightmost columns of Table 2 leave $\kappa$ unspecified; actually, results for the best performing $\kappa$ are presented.

![Figure 3. The HMM structure of meander $M^N_2$.](image)

Table 2. Test set perplexities of selected permutagram models

<table>
<thead>
<tr>
<th>Intensity</th>
<th>$N$</th>
<th>$\Lambda$-gram</th>
<th>$M^N_2$</th>
<th>$\Sigma^N_2$</th>
<th>$\Pi^N_2$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$N=2$</td>
<td></td>
<td>26.4</td>
<td>38.6</td>
<td>24.7</td>
<td>25.1</td>
</tr>
<tr>
<td>$N=3$</td>
<td></td>
<td>25.5</td>
<td>29.5</td>
<td>25.2</td>
<td>23.1</td>
</tr>
</tbody>
</table>

For the Verbmobil corpus with its extremely long dialog turns, the periodically repeated patterns of the $\Sigma^N_2$ components do not provide sufficient flexibility to capture the intrinsic dependency structure. Much more could be gained when applying the connected meanders, which in turn showed little (additional) effect for the Intercity corpus.

### 5. CONCLUSION AND FUTURE WORK

We presented a new kind of stochastic grammar — the permutagram model. A permutagram model is obtained by linear interpolation of a large number of conventional bigram, trigram, or polygram models which operate on different permutations of the input word sequence under consideration. Using the permutagram model, we achieved test set perplexity reductions of $\approx 10\%$ compared with interpolated $N$-gram models, depending on the application.

In spite of this success, we feel that more dramatic improvements were possible if sentence scores could be computed in a decision-directed fashion, i.e., by evaluating the HMM using the Viterbi algorithm instead of Baum-Welch forward decoding. Under this condition, the permutagram model would decide on the best-fitting word permutation for each input sequence rather than averaging over the entire subspace spanned by the HMM.

Unfortunately, the Viterbi permutagram scores do no longer form a valid distribution, and the missing renormalization coefficient is not accessible. Hence, an experimental assessment of Viterbi permutagrams on the basis of perplexity is impossible. We envisage automatic subcorpus classification [10] as an appropriate testbed for unnormalizable language models.

### REFERENCES


