USING X-GRAM FOR EFFICIENT SPEECH RECOGNITION

Antonio Bonafonte and José B. Mariño
{antonio|canton}@gps.tsc.upc.es

Universitat Politècnica de Catalunya
C/Jordi Girona 1-3 08034 Barcelona, SPAIN

ABSTRACT

X-grams are a generalization of the n-grams, where the number of previous conditioning words is different for each case and decided from the training data. X-grams reduce perplexity with respect to trigrams and need less number of parameters. In this paper, the representation of the x-grams using finite state automata is considered. This representation leads to a new model, the non-deterministic x-grams, an approximation that is much more efficient, suffering small degradation on the modeling capability. Empirical experiments for a continuous speech recognition task show how, for each ending word, the number of transitions is reduced from 1222 (the size of the lexicon) to around 66.

1. REVIEW OF X-GRAMS

One of the main components in continuous speech recognition is the language model. The model has to estimate the *a priori* probability of each possible sentence. The most extended models are based on n-grams, basically bigrams or trigrams, which estimate the probability of each word taking into account only the previous n-1 words. Recently, the authors have introduced x-grams [1]. The difference between n-grams and x-grams is that, in x-grams, the length is not fixed by the designer of the language model, but found automatically by the algorithm from the training data. For each sequence of given words, the method decides the number of words which really conditionate the probabilities of the incoming words. To decide if a history is relevant or not, the algorithm uses the number of times that the history occurs in the training data and the divergence function [1]. X-grams are estimated using back-off smoothing.

X-grams are more efficient than trigrams in the following sense: x-grams are more compacts (smaller) than trigrams and produce smaller perplexity. This is because some probabilities distributions conditioned by the two previous words do not improve the estimation with respect to the use of only the one previous word. On the other hand, in other cases, the estimation of the probabilities conditioned by the 3, 4 or, in general, the x previous words is significantly better that the estimation considering only the two previous words.

To illustrate this, table 1 shows the perplexity and the number of *histories* considered in trigrams and with x-grams. The task consists in queries to a database with geographical information (rivers, mountains, cities, etc.). The models are trained using 8262 queries (90.000 words) and the perplexity is evaluated using 1147 queries. All the queries in the training and in the test are different. The size of the lexicon is 1222 words.

Model	Perplexity	# histories
trigram	8.3	7526
x-grams	7.8	5915

Table 1: Comparison of trigrams and x-grams.

2. REPRESENTATION OF X-GRAMS

This paper treats about the representation of x-grams. It also applies, as a particular case, to n-grams.

Let's assume that a language model is estimated using only the following data ($ stands for the beginning and the end of the sentence): *abb*

If the language model is a smoothed trigram, the histories that will appear are: *{<$>, <a>, <b>, <$a>, <ab>, <bb>}*

Non seen histories, for instance *<ba>*, are not necessary because the probabilities $p(\cdot/ba)$ are estimated using $p(\cdot/a)$. For instance, the probability of the sentence *aab*:

$$p(\$aab\$) = p(a/\$)\, p(a/\$a)\, p(b/\$aa)\, p(\$/\$aab)$$

is approximated by

$$p(\$aab\$) \approx p(a/\$)\, p(a/\$a)\, p(b/a)\, p(\$/ab)$$

In the case of x-grams, it can happen that some of these histories are considered irrelevant while other longer ones are considered important. Suppose that the retained histories for a x-gram are only: *{<0>, <$>, <a>, <ab>}*

In this case, the probability would be approximated by:

$$p(\$aab\$) \approx p(a/\$)\, p(a/a)\, p(b/a)\, p(\$)$$

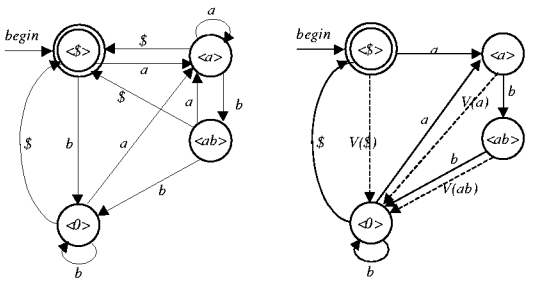
Note that because some bigram histories are not retained (in particular, *<b>*), the unigram history (*<0>*) has to be included (for instance to compute $p(\$)$.)

2.1 Expanded representation.

X-grams (as n-grams) can be represented as a deterministic stochastic finite state automaton (stochastic because it contains probabilities; deterministic because from each state and for each word, there is only one transition labelled with this word). The states of the automaton are identified with the selected histories. The transitions are labelled with words and probabilities of words given the history. Figure 1.a shows the representation of the previous x-gram. It can be seen how there are 4 states (the same that the number of selected histories). Because of the smoothing, each word of the lexicon (and the end of the

This research was supported by CICYT, contract TIC95-0884-C04-02

sentence) can happen after each history. This is why the number



of transitions from each state is three.

Figure 1: Representation of a x-gram using a finite state automaton. a) On the left, expanded representation; b) On the right, compact representation using null-transitions.

The advantages of this representation are the following: the number of states depends on the number of selected histories, independently of the unseen or unselected histories. Furthermore, the representation is very simple and very efficient to analyse input text: for each word, a transition has to be followed.

The main disadvantage is the size: as there is as many transitions as words, the size is prohibitive even for moderate lexicon sizes. For instance, for the geographical task, the number of states is 5915 and the lexicon size is 1222 (plus the end of sentence). Therefore, the number of transition in the automaton is 7.2 millions. (If four bytes are used to define the end of the transition and four more for the probability, then the memory required to store the automaton would be more than 55 megabytes).

2.2 Compacted representation.

The representation of x-gram can be compacted if the back-off smoothing is not applied during the estimation of the model, but when using the model. For each history, *h*, the probabilities of each unseen word following *h*, are obtained by multiplying the factor *V(h)* with the probabilities of the unseen words following the back-off history. *V(h)* is computed as *Q(h)/T(h)*, were *Q(h)* is the reserved mass of probability for unseen words leaving state *<h>* and *T(h)* is the value which makes that the sum of the probabilities is one. For instance, in the previous example, only symbol *b* is seen after history *h* = *<ab>*. The probabilities *p(a/ab)* and *p($/ab)* are estimated using the smoothing in the following form:

$$p(a/ab) \approx V(ab)\, p(a/b)$$

$$p(\$/ab) \approx V(ab)\, p(\$/b)$$

In fact, because the history *<b>* has not been selected, the probabilities are estimated as:

$$p(a/ab) \approx V'(ab)\, p(a)$$

$$p(\$/ab) \approx V'(ab)\, p(\$)$$

The representation is enormously simplified if null transitions are allowed. Figure 1.b shows the x-gram of the example using this representation.

For the geographical task, the number of states is 5916: the 5915 that appear in table 1, plus the unigram state that has to be included. The number of transitions (including null transitions) is 30,544, which compares favourably with the 7.2 millions of the expanded representation. The representation is reduced by a factor greater than 200.

The advantage of this representation resides in its compactness. However, the representation has also some disadvantages.

First, the representation is not a formal automaton. In an automaton, when a state is accessed, all the transitions can be followed, as is the case of the automaton of figure 1.a. However, in the automaton of figure 2, the valid transitions depend on the way how the state was accessed. For instance, from state *<0>*, the transitions that can be followed are:

a,$ if state *<0>* has been accessed from *<a>* or *<ab>*

b,$* if state *<0>* has been accessed from *<$>

This makes the use of the automaton more inefficient. For instance, to compute the probability of a sentence, the algorithm looks at each word and looks for a non-null transition from the state. If there is not non-null transition, then goes to a new state following the null transition and looks for a non-null transition from the new state. If x-grams are used, for some cases, several null transitions have to be followed before a non-null transition with the desired word is found.

We use x-grams in speech recognition using the Viterbi algorithm. For each frame (time), the search space is composed by the states of the automaton. Each time that an explored path arrives to the end of a word (associated to a history), the path has to be ramified into as many paths as words in the lexicon, and for each word, the probability of the words has to be applied. In the expanded representation of figure 1.a, the successors are explicitly expressed in the automaton. However, if the compact representation of figure 1.b is used, then the successors have to be expanded dynamically to proceed with the recognition process. Note that if the x-grams are used in this way, we have a compact representation in terms of memory but not in the recognition search. In the geographical task, the 30544 transitions are in fact expanded, as soon as they are needed, as if there were 7.2 millions of transitions.

Now, let's assume that the representation of figure 1.b is a formal automaton. It would mean that when a state is accessed, all the transitions leaving the state can be followed. In this case the number of transitions will be absolutely 30544 and the inter-word recognition search will be much more efficient.

To illustrate this point, let's consider two hypotheses in the recognition search (two paths in the trellis) with probability of finishing in frame *t* different of zero:

Hypothesis A: Sentence: <..., w_a, w> Probability: p_a
Hypothesis B: Sentence: <..., w_b, w> Probability: p_b

For each one of the words of the lexicon, *w'*, with transitions leading to the same state, the following comparison has to be done:

$$p_a \cdot p(w'/ w_a w) \; >? \quad p_b \cdot p(w'/ w_b w) \qquad (1)$$

In practical situations, for most of the words of the vocabulary, the probability of the new word is based on the probability of the back-off method. Let's assume for simplicity in the notation that the back-off state is the bigram one, labeled with <*w*>. Equation (1) can be reformulated as:

$$p_a V(w_a w) \cdot p(w'/ w) \; >? \; p_b \cdot V(w_b w)\, p(w'/ w) \qquad (2)$$

where $V(w_a w)$ and $V(w_b w)$ are the back-off weights.

It can be observed that the result of question (2) seems independent of the value of *w'*. If this was the case, the comparison could be done just once for all the words *w'* with important saving in computation.

The same would happen if the back-off state would be the unigram <*0*>. For instance, if state <*w*> was not included in the x-gram, then equation (2) would become:

$$p_a V'(w_a w) \cdot p(w') \; >? \; p_b \cdot V'(w_b w)\, p(w') \qquad (3)$$

which seems to be independent of both *w* and *w'*. Again, if this was the case, the recognition algorithm should select the best $p_h V(h)$ and then propagate the transitions leaving the unigram state to all the words.

However, equation (2) and (3) are not really independent of *w'*. They are independent of *w'* as far as there is not a direct transition labeled *w'* leaving state <*h*>. Only in this case, the probability $p(w'/h)$ relies in $V(h)\, p(w')$ (unigram case). This is the reason why all the transitions have to be considered during the recognition and not only the small proportion that is needed to represent the automaton. Equation (1) has to be used for every pair *h,w'* leading to the same state.

3. NON-DETERMINISTIC X-GRAMS

To speed up the recognition process, we are interested in consider the representation of figure 1.b as if it was a formal automaton. In this case, all the transitions leaving the states can be followed independently of how the state was accessed and the number of transitions to follow during recognition would be the same that the number used to represent the automaton. In this case the automaton is not deterministic: if a non-null transition labeled with *w* leaves the state <*h*>, then other transitions labeled with *w* can be found following the null transition. In this sections we want to evaluate the degradation that suffers the model if this new view of the automaton is considered. If the degradation were small, then this new model would be preferred for speech recognition.

To evaluate the performance, the perplexity in the test-set will be used. First, it is needed to normalize the probabilities so that the probabilities leaving each state sum one. If probabilities are not normalized, as new paths are now allowed, the sum of probabilities would be greater than one and there would be no sense in computing the perplexity.

If the automaton is deterministic (compact representation) the sum of non-null transitions is 1-$Q(h)$. For each state, assuming than all the other states accomplish the stochastic restriction, we have to impose

$$f_1 (1-Q(h)) + f_2 V(h) = 1 \qquad (4)$$

In equation (4), f_1 is the normalization factor for direct words (seen histories) and f_2 is the normalization factor for unseen words. Two options have been tried: in the first one, we impose $f_1 = f_2$: all the probabilities are affected by the same factor. In the second one, f_1 is imposed to be 1 assuming that the probability of seen words is better estimated than the unseen probabilities. Table 2 shows the perplexity for these two cases compared with deterministic automaton. The fact that the automaton is not deterministic introduces a difficulty to interpret the perplexity as a measure of quality of the model when the Viterbi algorithm is used to recognize. When computing the perplexity, the probability of all the paths through the automaton which produce the same sentence are added. However, the approximation imposed by the Viterbi algorithm makes that during recognition, only the probability of one of these paths is considered. In table 2, the perplexity using the Viterbi approximation (best path) is also included. It is not claimed that this measure of the perplexity is better than the real perplexity to predict the performance of the models in speech recognition. Instead, the number is given as a measure of the ambiguity of the model.

x-gram	PP	PP (vit.)	# states	trans/state
deterministic.	7.8	7.8	5915	1223
non det. $f_1=f_2$	7.8	8.5	5915	5.2
non det. $f_1=1$	7.9	8.3	5915	5.2

Table 2: Comparison of deterministic vs. non deterministic x-grams. The table shows, the perplexity, the perplexity computed using the Viterbi approximation, the number of selected histories (states) and the number of transitions per state which should be followed during recognition.

It can be seen in the table how non deterministic x-grams are as good models as the deterministic ones and their complexity during recognition is much smaller. Even when the PP(vit) is considered, the result is similar to deterministic trigrams but with less complexity in number of states and in number of transitions.

In order to reduce the ambiguity of the model some experiments have been performed where only the most ambiguous nodes have been expanded. The conclusion is that many states have to be expanded to get significant changes in the perplexity. For instance, in one experiment we get: PP = 7.8 and PP (vit) = 8.0,

but the number of transitions were reduced only by a factor 3 (compared with a factor 230 for the cases in table 2).

4. SPEECH RECOGNITION USING NON-DETERMINISTIC X-GRAMS

The result of the previous section is very optimistic. As the number of transitions per state is reduced by a factor 230, we could think that the inter-word effort in the recognition search would also be reduced by the same factor. This would be the case if the search was complete, but usually, the search is driven in a beam and all the states are not equally used. For instance, if each time, only the end of one word is hypothesized, then the number of transition will be (at least) the lexicon size, and the complexity of the search when non deterministic x-grams are used would be the same than when deterministic x-grams are used. In this section, the saving of using non-deterministic x-grams is analyzed using speech recognition experiments.

Non-deterministic x-grams have been evaluated using a Viterbi-based recognition algorithm. The system includes broad silence/speech detection. The language model is the non-deterministic x-gram presented in section 3, preceded by an optional silence model. Each word of the lexicon is represented as a sequence of sublexical units optionally followed by a silence model. Each model is represented by HMM (continuous, semicontinuous or discrete). During the search the lexicon is organized linearly. The search effort is limited using beam search. The beam search was adjusted so that there were no significant search errors.

The use of non-deterministic x-grams is useful only in the case that several words end simultaneously. So, the first evaluation consist in counting the number of frames where there are ending words and the mean number of ending words in these ending frames. The experiments showed that 93% of the frames have ending words. For these frames, the mean number of ending words is 65.

If deterministic x-grams were used, for each ending frame, the number of transitions that should be propagated would be 1222/ending word (the lexicon size). If non-deterministic x-grams are used the mean number of transitions for each ending word happens to be 31.7. The saving is not as optimistic as predicted in section 3 (230), but still quite significant (38). Furthermore, the transitions have not to be expanded dynamically, speeding up the recognition process.

Look-ahead.

In large vocabularies, it is not feasible to propagate all the words. Tree organization of the lexicon or look-ahead search is required. Our system uses look ahead which significantly reduces the number of evaluated transitions. The look-ahead search is organized in the following way. First, the non-null transitions leaving each state are sorted by the probability of the transition (off-line). Second, for each frame, the probability that each phone begins is estimated using the next four frames. This estimation uses simplified one-state HMM, which makes this estimation very fast. The best probability that a phone begins in this frame is also registered simultaneously (*MaxLA*). The probability that a word begins in the frame, *LA(w)*, is directly the probability estimated for the first phone of the word.

The algorithm first makes the intra-word search (inside the lexicon) and then the inter-word search. When the intra-word search is being performed, the ending word with greater probability is registered. Let's be $<h^*>$ the state associated to this best ending word and p_{h^*} the accumulated probability. Then, the look-ahead beam is defined using a minimum probability. Only the paths with higher probability are propagated. The minimum probability p_{min} is defined as follows:

$$p_{min} = max\ \{p(w/h^*)\ LA(w)\}\ \ p_{h^*}/\Delta_{LA}$$

The maximization is made for all the words (transitions) leaving the state $<h^*>$. Δ_{LA} defines the look-ahead beam width. The value of p_{min} is an initialization, which is updated when doing the inter-word search.

The inter-word recursion is as follows:

For each ending word (associated to the state $<h>$)
  For each transition associated with word w
    if ($p_h p(w/h)\ LA(w) > p_{min}$) then propagate
    if ($p_h p(w/h)\ MaxLA < p_{min}$) then go to next ending word.

As the transitions are sorted, the second condition guarantees that the rest of transitions leaving $<h>$ will not accomplish the first condition. Therefore, these transitions do not have to be evaluated. The experimentation reveals that using this organization, instead of considering 31.7 transitions per ending word, only 4.0 transitions need to be considered.

5. SUMMARY

In this paper, the x-grams have been reviewed making emphasis in representations of the x-grams as finite state automata. This representation suggests a new model, the non-deterministic x-gram, which greatly reduces the number of inter-word transitions that have to be followed in the speech recognition search. The experimentation shows that the loss in terms of perplexity of this new model is very small while the number of transitions is reduced by a factor of 30 in a task of geographical queries. Another advantage of the representation is that because we use a formal automaton, standard transformations of automatons can be done. For instance, in the geographical task, we have found very useful to substitute in the training text each name of river, mountain, length number, etc. by a generic label and, afterwards, to expand the nodes labeled with generic labels using a hand-coded representation. The input and the output of the expansion algorithm are just finite state automata.

The paper is based in x-grams because they have proven to be better than trigrams. However, the methodology presented is also valid for bigrams and trigrams.

6. REFERENCES

1. A. Bonafonte et al., *"Language Modeling Using X-grams"*, International Conference on Spoken Language Processing, ICSLP-96, pp. 394-397, Philadelphia, 1996.