Large-Scale Speech Recognition

Madiha Mubin Chinyere Nwabugwu Tyler O'Neil

Abstract:

This project involved getting a sophisticated speech transcription system, SCARF, running on a large corpus of data and establishing metrics to analyze its performance. Furthermore, by using these metrics we gauged the performance gain of integrating a word detector produced by Andrew Maas' research team. We ran oracle experiments on different ways of integrating word detectors in order to validate our setup, get a feel for the best ways of integrating word detectors into SCARF, and establish a performance bound with which to compare our results. Our oracle experiments showed that a word detector could bring word error from 14.8 down to 10.3 if we create feature weights for every word and detector output pair. This shows there remains work to be done in integrating the real word detector, and provides insight into the best attainable performance.

Data:

We obtained our training data set from Broadcast News Lattices, Linguisitic Data Consortium (LDC) catalog number LDC2011T06. This data set was developed by researchers at Microsoft and John Hopkins University. The lattices were derived from transcripts of approximately four hundred hours of English broadcast news recordings. They were generated using the IBM Attila speech recognition toolkit.

Model:

To train our data set and test performance, we used SCARF[1]. SCARF is a toolkit that uses Segmental Condition Random Fields to do speech recognition. It applies a discriminative model to segment-level features based on acoustic detector inputs. SCARF segments data into chunks and assigns a word to each chunk.

Conditional Random Fields

CRFs estimate conditional distribution $p(y|x)$ with a graphical structure. Let G be a factor graph over Y. Then $P(y|x)$ is a conditional distribution for fixed x.

Let $F = \{\psi_A\}$ for all assignments $\{A\}$ to x. Then, we can write the conditional probability distribution for the model as follows[2]:

$P(\boldsymbol{y}|\boldsymbol{x}) = \frac{1}{Z(x)} \prod_{\psi_A \in G} exp\{\sum_{k=1}^{K(A)} \lambda_{AK} f_{AK}(x_A, y_A)\}$

$Z(\boldsymbol{x}) = \sum_{x,y} \prod_A \psi(x_A, y_{A)}$

- $\psi_A(x_{A,}y_A) = exp\{\sum_{k=1}^{K} \lambda_{Ak} f_{Ak}(x_A, y_A)\}$,
- λ_A= parameter vector (feature weights/bias)
- f_{Ak} = feature function for k^{th} feature using assignment A.

Segmental Conditional Random Fields

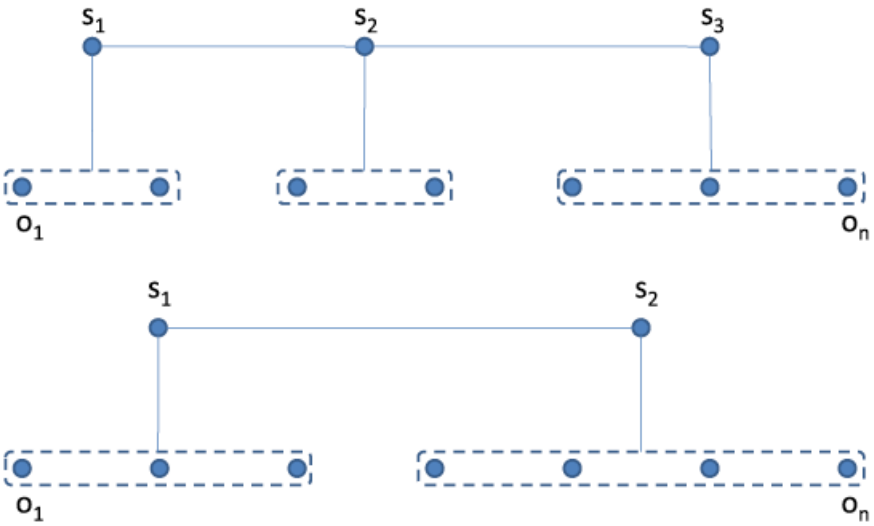

Figure1: Shows a segmental CRFs with two different segmentations.

State:

A state in the model corresponds to states in a finite state representation of a n-gram language model. It is a variable over a derived time span. We do not fix the structure of the model before hand. This is because the goal is to segment the observation sequence into $l \leq N$ chunks, where N is the size of the observation sequence (See Figure 1). Furthermore, these states are labeled with the index of the underlying language model state. So if SCARF state representation is s1 -> s10 -> s5 then the language model state will be s1 := "in", s10 := "the", s5 := "car".

Features:

There are transition features functions whose weights are learned to produce a discriminatively trained language model. Feature functions relate a word to an observation span.

Vertical edges correspond to observations from a state and so the feature function will be $f_k(s_v, o_v)$ for k^{th} feature for the v^{th} state node in the graph. Horizontal edges correspond to transtion from one state to another, so the feature function will be $g_d(s_l^e, s_r^e)$ for the d^{th} feature between s_l^e and s_r^e in the graph. For simplicity, we will combine horizontal and vertical feature functions to be over triples $f_k(s_l^e, s_r^e, o(e))$, where $o(e)$ corresponds to observations in chunk associated with s_r^e for a given state segmentation.

We will use the variable $\boldsymbol{q}$ to denote a possible segmentation of the observation sequence $\boldsymbol{o}$. The size of $\boldsymbol{q}$ should be equal to the size of the state chain $\boldsymbol{s}$. Since we do not have segmentation of the training data we have to consider all segmentations for a state sequence $\boldsymbol{s}$. The conditional probability distribution is defined as:

$$P(\boldsymbol{s}|\boldsymbol{o}) = \frac{\sum_q exp\{\sum_{e \in q,k} \lambda_k f_k(s_l^e, s_r^e, o(e))\}}{\sum_{c \in S} \sum_q exp\{\sum_{e \in q,k} \lambda_k f_k(c_l^e, c_r^e, o(e))\}}$$

We take the log of the this distribution function to compute the log likelihood of a state chain using all possible segmentations given observation sequence. We compute the derivative with respect to variable λ to learn it using gradient descent. If we have K features, then we are learning K variables. SCARF adds L2 regularization penalty to the log likelihood objective function.

External Features:

SCARF's training stage generates feature functions in a variety of different forms. The ones that we were concerned with were existence features, expectation features, external features, and lexicalized external features. **Existence features** are simply a function of whether or not a given detector unit occurs within a time interval. For instance, if SCARF is deciding between 'WITH' and 'WIT', and a phoneme detector has a 'TH' output as an existence feature, SCARF will use this information to favor 'WITH.' **Expectation features** are similar to existence features, but are generalized to learn correct-accept, false-reject, and false accept cases. So in the previous example, if the phoneme detector output was encoded as an expectation feature and during training the feature was always a false-accept, we may favor 'WIT' instead. Expectation features and existence features can be used on a single-word or ngram level. **External features** are annotated directly to the lattice graph, and a global weight is used to favor different paths through the graph. Finally, **lexicalized external features** are the same as regular external features, except a different weight is learned for each different word in the dictionary.

Oracle Experiments:

Besides confirming that SCARF is setup with reasonable starting parameters, oracle experiments are a useful tool to see what performance gains are attainable with additional features. For instance, even if our features were the output of 100% accurate classifiers, we still would not get a word error rate of 0% due to being outweighed by the language model or other suboptimal features, so it is useful to see what this lower bound on word error rate is to decide what features we want to use, and gauge performance. The feature that we spent the most time working on was word detection. Given a time interval in an audio stream, a detector would output the probability that a common word is in the interval for one hundred very common words. For instance, given a time interval, it would output P(WITH)=.1, P(THE)=.4, etc.

Best Possible Performance:

The SCARF algorithm described is very computationally intensive, so SCARF uses constraints that are output by another speech system. In our case, it was the IBM Attila Speech Recognition toolkit. These constraints come in the form of a lattice graph where nodes are times and edges are the words that span them. SCARF then uses these constraints to find the best possible path of words to form the transcript. If these constraints do not contain the ground truth transcript, then SCARF could never be expected to perform well (garbage-in, garbage-out). By comparing the ground truth transcript with these constraints, we analyzed the best possible performance SCARF could attain.

The problem of evaluating the consistency between the constraining lattice graph and the ground truth reduced a general one: given a graph and a desired path, find the path that best matches the desired path (measured by levenshtein distance). We used a modified form of Djikstra's Algorithm that kept nodes as tuples with the following structure: current node in the desired path, current node in the path, levenshtein distance accumulated so far. Then the nodes were greedily chosen to minimize the levenshtein distance. This algorithm produced the same results as brute force on small lattices, and scaled to the massive size of our training set. We found the best SCARF could ever be expected to do with our training error was a word error rate of 8.9.

Evaluation:

Sclite:

In order to evaluate the output of our speech recognition system we are using NIST Sclite tool[4]. Sclite is a part of the NIST Scoring Toolkit. It is a flexible dynamic programming alignment engine used to align errorful hypothesized tests to the correct reference texts and generate scoring reports. We were able to correctly run Sclite to evaluate performance of the SCARF output after writing a wrapper around Sclite that would convert ground truth files and SCARF output into file formats that Sclite can process. For comparing SCARF output to the ground truth, we used Word Error Rate.

Results and Discussion:

Oracle Experiment Results:

Top 100 words

WER	Expectation Order			
Existence Order	0	1	2	3
0	14.8	10.6	10.6	10.6
1	11.3	10.5	10.5	10.5
2	11.3	10.5	10.5	10.5
3	11.3	10.5	10.5	10.5

All words:

WER	Expectation Order			
Existence Order	0	1	2	3
0	14.8	10.6	10.6	10.6
1	10.3	10.6	10.6	10.6
2	10.3	10.6	10.6	10.6
3	10.3	10.6	10.6	10.6

Table1: Shows results of expectation and existence features for a perfect detector for (a) Top 100 words, (b) All words.

	Lexicalized	Nonlexicalized
1 Annotation	10.8	10.8
100 Annotations	10.3	-

Table2: Shows results of using lexicalized external features and plain external features for perfect detector.

Our first oracle experiments were with expectation and existence features. See Table1. Generally, mixing expectation and existence features did not improve results. This was expected because they encode similar information. When both are present, the expectation features overpower the existence features. It also seemed as though we reached a cap on the amount of weight the expectation features would be assigned. However, in practice, this should not be an issue because we won't reach that cap with real detectors.

For expectation and existence features, we tested an oracle detector for the top 100 words and all words. The difference in performance was very small. Since adding additional words to our detector only slightly increases performance, we see that adding more words to any real detector would probably not be expected to improve performance significantly.

For testing lexicalized external features and nonlexicalized external features, we had two different ways of encoding the output of the detector: first we could try annotating the lattice with the probability of the most likely word it is in the top 100, or we could try annotating the lattice with the probabilities for all 100 words. The second approach would require a tremendous amount of computation because it would require training 100 feature weights for every word in the dictionary. For instance, on a four core, 2.40GHz Intel E560, it could take over a day to train on our corpus. Despite the computational drawback, we found that only lexicalized features with all 100 words improved results over the expectation features. See Table2.

After trying all these possible ways of encoding the output of our detector streams, we found that the best upper bound on performance could be achieved by creating a separate weight for every combination of word and word probability. However, because training with this technique can be prohibitively computationally expensive, expectation features perform comparably well (10.3 WER vs 10.6 WER).

Integrating Actual Detector:

Without detector output	15.2 WER
100 Feature annotations	15.1 WER

Table3: Shows results of using lexicalized external features with a non-perfect detector. Note that the 15.2 WER is different from the baseline 14.8 above. This is because we used a subset of our dataset to produce these results.

Using a detector produced by Andrew Maas' research team, results were generated for lexicalized features. See Table3. Our initial results were much worse than the 15.2 above, so we used cross validation on the L2 normalization parameter to get that number to 15.1, which is a very slight improvement. We believe that the number can be lowered even more by running SCARF for more iterations (we are not convinced SCARF is converging, even after running it for two days). Other approaches that could be tried are feature scaling or encoding the features in a different way.

Acknowledgements:

We would like to thank Andrew Maas for providing input data for this project and for useful discussions and suggestions.

References:

[1] Geoffrey Zweig, Patrick Nguyen, et al., "Speech Recognition with Segmental Conditional Random Fields: Final Report from the 2010 JHU Summer Workshop*", Microsoft Research Publications.*

[2] http://www.cs.umass.edu/~mccallum/papers/crf-tutorial.pdf

[3] Mark Gales and Steve Young, "The Application of Hidden Markov Models in Speech Recognition", *Foundation and Trends in Signal Processing*.

[4] NIST Sclite: http://www.icsi.berkeley.edu/Speech/docs/sctk-1.2/sclite.htm