# A FLAT DIRECT MODEL FOR SPEECH RECOGNITION

*G. Heigold**

Chair of Computer Science 6
RWTH Aachen University, Aachen, Germany
heigold@cs.rwth-aachen.de

*G. Zweig, X. Li, and P. Nguyen*

Microsoft Research
One Microsoft Way, Redmond, WA 98052, USA
{gzweig,xiaol,panguyen}@microsoft.com

**ABSTRACT**

We introduce a direct model for speech recognition that assumes an unstructured, *i.e., flat* text output. The flat model allows us to model arbitrary attributes and dependences of the output. This is different from the HMMs typically used for speech recognition. This conventional modeling approach is based on sequential data and makes rigid assumptions on the dependences. HMMs have proven to be convenient and appropriate for large vocabulary continuous speech recognition. Our task under consideration, however, is the Windows Live Search for Mobile (WLS4M) task [1]. This is a cellphone application that allows users to interact with web-based information portals. In particular, the set of valid outputs can be considered discrete and finite (although probably large, *i.e.,* unseen events are an issue). Hence, a flat direct model lends itself to this task, making the adding of different knowledge sources and dependences straightforward and cheap. Using *e.g.* HMM posterior, m-gram, and spotter features, significant improvements over the conventional HMM system were observed.

***Index Terms***— maximum entropy, language model, nearest neighbor, voice search, speech recognition

## 1. INTRODUCTION

This work focuses on the definition, implementation, and test of a flat direct model for the Windows Live Search for Mobile (WLS4M) task [1]. Here, a flat model refers to a model with a fully unstructured text output and thus, clearly differs from a sequential and structured model.

The most prominent example for sequential models are HMMs. The special structure of HMMs allow for efficient training and search algorithms, *e.g.* dynamic programming. Furthermore, the modeling on sub-word units makes it possible to recognize unseen events. One of the great disadvantages of HMMs is the adding of dependences, which is possible only to some limited extent and is associated with high modeling and engineering costs. Opposed to the sequential models, the flat models are completely unstructured models. All structural information and dependences are encoded by the features which represent attributes of the flat text output $W$. The differences between a sequential and the flat model are depicted in Fig. 1. The incorporation of all structural information into the features makes this approach very flexible. In particular, adding dependences is straightforward and cheap. Furthermore, the discriminative formulation better fits the nature of pattern recognition. For the flat approach, the set of valid outputs is assumed to be discrete and finite, *e.g.* the 10M entries of the yellow pages. This is an important difference from a typical large vocabulary continuous speech recognition task. In particular, it allows for refined modeling approaches.

*The author performed the work while at Microsoft Research.

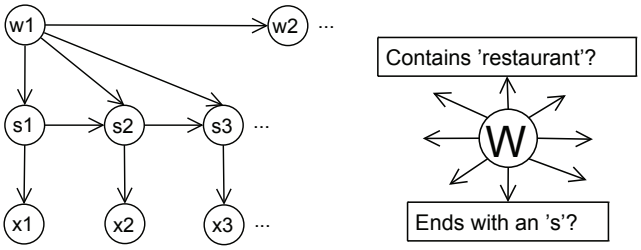


**Fig. 1**. Conventional HMM $vs.$ flat model. Left: dependence graph of a conventional HMM for speech recognition, nodes represent variables $w_n$ (word), $s_t$ (hidden state), and $x_t$ (feature), arcs indicate dependences between variables. Right: flat $W$ with attributes.

Next, we address some of the general concerns about this approach. First, that the computational complexity is prohibitive because arbitrary features are allowed. The complexity, however, can be reduced by suitable search strategies (*e.g.* rescoring and special task described above) and a reasonable choice of features. In addition, the straightforward parallelization of such models also alleviates this problem. Second, it might be not clear how to cope with unseen events, *i.e.,* events $W$ that are not covered by the training data. An unseen event can be recognized if the associated attributes have been observed in training. Hence, solving the generalization problem in this context requires a reasonable granularity of the features, *e.g.* utterance *vs.* word spotters. Finally, it should be noted that this approach does not require a number of features proportional to the number of outputs $W$. In the ideal case, each attribute divides the text output space into two, *e.g.* 25 attributes can discriminate up to 10M events, *cf.* yellow pages.

Related work can be found in [2] where Dynamic Time Warping (DTW) spotters and HMM scores are considered in the maximum entropy framework on an isolated digit recognition task. Single word detectors are investigated in [3]. The detectors are combined in a linear fashion. Some of the features used in our work are similar to discriminative language modeling, *e.g.* [4, 5, 6].

The remainder of the paper is organized as follows. First, the flat model is introduced more formally and discussed in more detail in Sec. 2. The features considered in this work are defined in Sec. 3 and tested in Sec. 4. The paper concludes with a summary on the different improvements.

## 2. FLAT MODEL

In this section, the flat model is defined in a more formal way. The flat model assumes audio input $X$ and text output $W$. The input may be sequential, $X = (x_1, x_2, \dots)$. In contrast to the conventional structural approach, the output $W$ is considered an object without explicit structure, see Fig. 1.

**Table 1**. A simple feature example.

| $\Phi(W)$ | $\Psi(X)$ | $f(W,X)$ |
|---|---|---|
| $W$ contains *food* | *food* has been spoken | word spotter |

**Table 2**. Illustration of generalization issue using word spotters.

| Attributes (word spotters) | |
|---|---|
| {*chinese, food, mexican, restaurant*}⊂W | |
| Seen events $W$ | Unseen events $W$ covered by word spotters |
| *mexican restaurant*<br>*chinese food* | *mexican food*<br>*chinese restaurant* |

In this work, the flat model is implemented by log-linear models

$$p_\Lambda(W|X) \quad = \quad \frac{\exp(\sum_i \lambda_i f_i(W,X))}{\sum_{W'} \exp(\sum_i \lambda_i f_i(W',X))}. \qquad (1)$$

The feature functions $f_i(W,X)$ can represent arbitrary attributes of $W$ and $X$. Here, the features are assumed to be of the form $f(W,X) = \Phi(W)\Psi(X)$, *i.e.,* can be decomposed into a text feature $\Phi(W)$ and an acoustic feature $\Psi(X)$, see Tab. 1 for an example. To achieve good generalization, it is important to find features that are precise (each $X$ is assigned a $W$ in the extreme case) and general at the same time. This tradeoff is illustrated in Tab. 2.

The model parameters are $\Lambda = \{\lambda_i \in \mathbb{R}\}$. These parameters are estimated in the usual way, including L2-norm regularization with some positive regularization constant $C \in \mathbb{R}^+$ (*e.g.* $C = 10^{-8}$) [7]

$$\hat{\Lambda} \quad = \quad \arg\max_\Lambda \left\{ \sum_n \log p_\Lambda(W_n|X_n) - \frac{C}{2} \sum_i \lambda_i^2 \right\}.$$

The optimization of this objective function is performed with Rprop [8]. For the setups under consideration, convergence is reached after about 30 iterations. This corresponds to training times below 20 minutes (without the generation of the spotter features).

## 3. FEATURES

The features considered in this work consist of a text feature $\Phi(W)$ and an acoustic feature $\Psi(X)$, $f(W,X) = \Phi(W)\Psi(X)$. The text features are prior-like and are similar to a discriminative language model [4, 5, 6]. Observe that pure acoustic features cancel in the posterior $p_\Lambda(W|X)$ ($f_i(W,X) \equiv f_i(X)$, *i.e.,* constant factor both in the numerator and denominator in Eq. (1)) and thus, are only effective in combination with (non-trivial) text features. First, we focus on the simpler text features and then, investigate the more complex acoustic (spotter) features.

### 3.1. "Simple" Features

The *m-gram feature* $\Phi_g(W)$ counts the number of occurrences of the m-gram $g$ in $W$. The m-grams can be defined on different levels, *e.g.* word or letter m-grams. Example: the choice $g = aurant$ (6-gram letter) and $W = restaurant$ leads to $\Phi_g(W) = 1$. This type of features is similar to an m-gram language model. The *length feature* returns the length of $W$, $\Phi_{\text{length}}(W) = |W|$.

The *rank feature* indicates if hypothesis $V$ has rank $r$, and is bounded above by $r_{max}$: $\Psi_{Vr}(X) = \delta(\min\{\text{rank}(V), r_{max}\}, r)$, where $\delta(i,j)$ denotes the Kronecker delta which is one for $i = j$ and zero otherwise. Similar features were used for the inverse rank. There were two sources of obtaining hypotheses. The type features indicate which source the hypothesis comes from. The *HMM posterior feature* of hypothesis $V$ is the HMM posterior normalized over the n-best list, *i.e.,* $\Psi_V(X) = p(V|X)$. Finally, the DTW feature for hypothesis $V$ is the DTW distance between this instance and the closest template. More formally, $\Psi_V(X) = \min_{Y \in \mathcal{T}_V}\{d(X,Y)\}$ where $\mathcal{T}_V$ denotes the set of templates of $V$ [2, 9]. Keep in mind that in general, $V$ is not the same as the text output $W$, *e.g.* pronunciation variants or parts of $W$.

The text and acoustic features introduced in the last two subsections can be combined to create more complex features. The *confusion feature* is defined as the product of the utterance text feature (1-gram feature on utterance level) and the rank 1 feature, $f_{VV'}(W,X) = \delta(W,V) \cdot \delta(\text{rank}(V',1))$. The acoustic data $X$ enters this feature implicitly through the rank feature. This feature is used to describe typical confusion pairs in the n-best lists. Another example of a combined feature is the *spotter feature*. The spotter feature consists of an m-gram text feature and some acoustic feature to indicate the acoustic confidence of the m-gram under consideration. In the simplest case, this might be the HMM posterior feature. This rather *ad hoc* choice shall be replaced with the DTW distance in the next section to refine the spotter feature.

### 3.2. Nearest Neighbor DTW Spotters

Next, we focus on Nearest Neighbor (NN) DTW spotter features. Like other NN approaches and in contrast to other spotter features (*e.g.* GMM-based spotters), these features have the advantage to be completely parameter-free and to show good asymptotic behavior. Here, we investigate utterance and m-gram word spotters. The utterance spotters detect a certain utterance, *i.e.,* indicate whether some utterance has been spoken or not. In addition to the detection problem, m-gram word spotters also need to find the segmentation of the m-gram in the utterance. But we believe that the coverage of the data is better for m-gram word spotters.

First, we study the utterance spotters because of their simplicity. Then, the utterance spotters are extended to the m-gram word spotters. A critical issue in practice is the intrinsically high complexity of the NN DTW spotter features. The complexity arises from the warped distance calculation and the search of the nearest neighbor(s). The first source of complexity is solved by (variants of) the dynamic time warping algorithm and the NN search is made feasible by different optimization strategies discussed in Sec. 3.2.3.

#### *3.2.1. Utterance spotters*

The Dynamic Time Warping (DTW) is a common technique to calculate the distance between two real-valued sequences of different length, $X_{hyp}$ and $X_{tpl}$. The warped distance can be efficiently calculated using dynamic programming, *e.g.* [2, 9]. For the experiments, we used symmetric transition constraints. The complexity of this algorithm is $\mathcal{O}(|X_{tpl}| \cdot |X_{hyp}|)$, *i.e.,* it is basically quadratic in the length of the sequences.

Cheating experiments using the oracle distance

$$\Psi_V(X) \quad = \quad 1 - \delta(V, \text{transcription})$$

suggested that m-gram word spotters are probably more suitable for this task. Note that the absolute difference between the correct and the competing hypotheses is irrelevant for log-linear models. This is because the scaling of the feature is compensated by the respective model parameter $\lambda_i$ in Eq. (1).

#### *3.2.2. M-gram word spotters*

The approach for m-gram word spotters is similar to that for utterance spotters but with the additional difficulty of determining the segmentation of the m-gram. The m-gram word segmentation directly follows from a time alignment, for instance. Here, we implemented an integrated approach that provides both the DTW distance

**Table 3**. Speed-ups (pruning does not change DTW distances).

| Description | Fct. |
|---|---|
| Discard non-speech frames at utterance boundaries | 2 |
| Consider each 10th template (except for extreme counts), consider frequently seen spotters | 10 |
| Prune DTW partial paths with score worse than currently best total score, cache distances to avoid duplicate calculations | 10 |
| Compiler intrinsics | 2 |

**Table 4**. Corpus statistics.

| Corpus | Period | #Utt. | Audio data [h] |
|---|---|---|---|
| Train | Oct 2007 - Feb 2008 | 550k | ≈350 |
| Dev | Mar 2008 - May 2008 | 310k | ≈200 |
| Test | Feb 2008 - Mar 2008 | 21k | ≈10 |

and the optimal segmentation. To avoid an explicit search over all valid segmentations with complexity $\mathcal{O}(|X_{hyp}|^2)$, we introduce a slightly modified version of the above DTW distance calculation. A special feature $\star$ is added at the beginning and at the end of $X_{hyp}$, resulting in the augmented input vector $(\star, X_{hyp}, \star)$. The feature $\star$ has the property to match any other feature, *i.e.,* always provides zero distance. The complexity of the original algorithm remains unchanged. As a consequence, an explicit segmentation is only needed for the templates but not the hypotheses. Furthermore, this integrated approach does not require models for events that we are not interested in, *e.g.* non-speech models. It also avoids the combinatorial complexity that the concatenation of sub-utterance templates would introduce [9].

#### *3.2.3. Speed-ups*

Finally and as pointed out at the beginning of this subsection, several speed-ups were required to make this approach feasible. On the one hand, the amount of data to process can be reduced. On the other hand, the distance calculation can be optimized. Tab. 3 provides an overview of the different techniques and the respective speed-up factors used to make the DTW distance calculation faster. With these speed-ups, the computation time on a single CPU and for all data amounts to a few days/months for the utterance/m-gram spotters.

## 4. EXPERIMENTAL RESULTS

The different features defined in the last section were tested on a voice search task.

### 4.1. Windows Live Search for Mobile (WLS4M)

The experiments were done on the voice search task WLS4M [1]. This application consists of look-ups of the yellow pages via a speech interface. The user is prompted with a list of hypotheses and then, can confirm the input by a click.

The training and test data are the users' histories from different periods, see Tab. 4. To reduce text normalization issues, the mangled text was used, *e.g.* words were concatenated and the text was all lower case. Transcriptions were available for the test but not for the training and development data. For this reason, we used the click hypothesis as the true transcription for the training. Gender-dependent GHMMs using conventional MFCC features (projected to 36 dimensions by HLDA) served as baseline model. The MFCC features were projected down to 36 dimensions by HLDA. The GHMM consisted of about 3,000 senones and 16 Gaussians per mixture. This GHMM baseline system yields about 40% utterance error rate on the test corpus. The flat models were used for rescoring (no direct indexing) this baseline in a second pass. There were no parameters to tune on the development data. So, this data was used as additional training data.

**Table 5**. Test utterance error rates for simple features, '6g' stands for '6-gram letters'.

| Setup | Add. #Feat. | Test Utt. Err. [%] Actual | Total |
|---|---|---|---|
| GHMM baseline | - | 17.4 | 39.6 |
| + type + rank + 6g w/o post | 50k | 15.0 | 37.8 |
| + type + rank + 6g w/ post | 50k | 14.2 | 37.2 |
| + HMM posterior | 1 | 13.4 | 36.7 |
| + confusion w/ post | 100 | 13.4 | 36.6 |
| + length | 1 | 13.5 | 36.7 |
| Oracle | - | 0.0 | 26.9 |

**Table 6**. Accuracy and HMM overlap of utterance DTW.

| | Test Acc. [%] | HMM overlap [%] |
|---|---|---|
| k=1 | 72.6 | 64.8 |
| k=1, scaled | 71.9 | 64.4 |
| k=10, $\beta = 10^{-4}$ | 73.8 | 65.8 |
| GHMM | 71.0 | - |

### 4.2. "Simple" Features

Tab. 5 gives an overview of the relative contributions from the different features in Sec. 3.1. We distinguish between the 'Total' and the 'Actual' utterance error rate. The first refers to the utterance error rate on the complete test data. Utterances for which the transcription is not in the list of hypotheses always lead to recognition errors in rescoring (*cf.* the oracle error rate of the n-best lists). In our case, this concerns approximately a fourth of the test utterances. For this reason, the second error is introduced. It is the utterance error rate limited to the test utterances that can be correctly recognized in the ideal case, *i.e.,* the oracle error rate on this subset is 0%, see Tab. 5. The baseline is the conventional GHMM system. The type and rank features are always included in the flat model to guarantee the baseline performance. The effect of these two features, however, is marginal. On this setup, the confusion features did not help. Using these features upon the '5-gram letters w/post' system (suboptimal, not shown in the tables), reduces the error rate from 15.0% to 14.3%. We conclude from this that the optimal m-gram letter features are sufficiently general to capture this effect, making these confusion features redundant.

### 4.3. Utterance Spotters

First, the quality of the DTW distances is checked. This was done by rescoring the n-best lists with these DTW distances. Obviously, this is possible only for utterances that are covered by the spotters. 2,500 spotters cover 54% of the test transcriptions and 16% of the test hypotheses. For this reason, the utterance accuracies in Tab. 6 are not directly comparable with the above error rates. The number of nearest neighbors considered for the distance calculation is denoted by $k$. For $k > 1$, the distance is smoothed over the $k$ nearest neighbors, $\sum_{Y \text{ is } k \text{ NN}} \exp(-\beta d(X, Y))$. Furthermore, unscaled (*i.e.,* no whitening matrix) features were used unless otherwise stated ('scaled'). Our interest is not in the classification using the isolated DTW distances but rather in using these distances in combination with other features, *e.g.* HMM scores. Hence, the overlap of the DTW with the HMM system is probably more meaningful measure in this context. The HMM overlap in Tab. 6 is defined to be the probability that an utterance is correctly recognized both by the HMM

**Table 7**. Spotter features. First block: utterance spotters on simple setup, $\hat{d}$ the denotes normalized distance $d/\max\{|X_{tpl}|, |X_{hyp}|\}$. Second block: utterance spotters on best setup. Third block: 2-gram word spotters on best setup.

| Setup | Add. #Feat. | Test Utt. Err. [%] Actual | Total |
|---|---|---|---|
| GHMM baseline | - | 17.4 | 39.6 |
| +type+rank+HMM posterior | 15 | 15.8 | 38.4 |
| +utterances,1 | 2.5k | 14.6 | 37.6 |
| +utterances,$d$ | | 14.0 | 37.1 |
| +utterances,$\hat{d}$ | | 13.7 | 36.9 |
| +6-gram letters w/ post | 50k | 13.6 | 36.8 |
| +utterances,1 | 2.5k | 13.3 | 36.6 |
| +utterances,HMM post | | 13.5 | 36.7 |
| +utterances,$\hat{d}$ | | 13.0 | 36.3 |
| +dev data | | 12.8 | 36.2 |
| +2-gram words,1 | 10k | 13.4 | 36.7 |
| +2-gram words,$d$ | | 13.2 | 36.5 |
| +dev data | | 12.9 | 36.3 |

and the DTW system. This overlap corresponds to 4% difference of the test error rate and reduces to 2% for the best setup in Tab 5.

Next, these DTW distances with $k = 1$ and without scaling were used as features on a simple setup without the 6-gram letters, see first block in Tab. 7. The experiments clearly demonstrate that the effect is not only due to the text feature ('utterances,1') and that the normalization of the distances helps ('utterances,$d$' *vs.* 'utterances,$d/\max\{|X_{tpl}|, |X_{hyp}|\}$'). Later experiments have shown that using $\exp(-\beta d)$ instead of $d$ as features performs comparably but appears to be less sensitive to the normalization. Similar experiments were conducted on the best setup from Tab. 5, and are shown in the second block of Tab. 7. Unfortunately, we used slightly different 6-gram letter features here such that the results are only consistent within the subsections. The utterance spotters help on this much better setup as well although the improvement is significantly smaller than on the suboptimal setup. In addition, we replaced the DTW distances with the HMM posteriors ('utterances,HMM post') to make sure that the same effect cannot be achieved by much simpler acoustic "confidences". Finally, we added the data from the development corpus to the training data ('+dev data'). Further tuning of the DTW setup has not shown any improvements so far, *e.g.* using more utterance spotters or considering more than only the nearest neighbor.

### 4.4. Cheating Experiments

The cheating experiment using oracle distances in Tab. 8 ('+3k utterances') indicates that the potential for further improvement of these utterance spotters is limited. To find a more promising setup, we performed some more cheating experiments, see Tab. 8. Without regularization and using all utterance spotters of the training data yields 0% error rate on the training data. However, the generalization ability is poor. The results are more balanced including the regularization term but do not improve over the limited number of utterance spotters. This is why we resorted to sub-utterance spotters, *e.g.* m-gram words. These m-gram word spotters seem to have more potential than the utterance spotters. The results for the m-gram word spotters shown in Tab. 8 are for a trimmed setup.

### 4.5. M-Gram Word Spotters

First results for m-gram word spotters can be found in Tab. 7. Unfortunately, the improvement for m-gram word spotters over the utterance spotters observed in the cheating experiments does not carry over to the real DTW distances. The setup for the m-gram DTW distances is probably not optimal yet. The error rates also suggest that the m-gram word spotter features might suffer from overfitting (the training and test error rates are even more unbalanced for 3-gram word spotters).

**Table 8**. Cheating experiments for spotters.

| Setup | Train Utt. Err. [%] | Test Utt. Err. [%] Actual | Total |
|---|---|---|---|
| type+rank+6g+HMM post | 13.1 | 13.6 | 36.8 |
| +3k utterances | 11.5 | 12.4 | 35.9 |
| + all 150k utt. (w/o reg.) | 0.0 | 13.8 | 36.9 |
| +10k 2-gram words | 9.8 | 11.2 | 35.1 |
| +14k 3-gram words | 9.3 | 11.0 | 34.9 |

## 5. SUMMARY

This work is considered a first step towards a flat direct model for speech recognition. Here, we have tested the effect of combining m-gram features, HMM scores, DTW spotters, etc. The relative improvement of these features over the GHMM baseline is 9% on the test data. Each the text features (*i.e.,* the m-gram features related with a discriminative language model) and the acoustic features (HMM posteriors, spotters) contribute approximately 50%. However, the effect of the DTW spotters is marginal compared with the best setup without spotters. Finally, it should be repeated that this is only the first step in this unconstrained framework. Future work will include the consideration of the flat model in a first pass decoding rather than in a second pass rescoring. Additional features will be investigated as well.

## 6. REFERENCES


[1] A. Acero, N. Bernstein, R. Chambers, Y.C. Ju, X. Li, J. Odell, P. Nguyen, O. Scholz, and G. Zweig, "Live search for mobile: Web services by voice on the cellphone," in *ICASSP*, 2008.

[2] S. Axelrod and B. Maison, "Combination of hidden Markov models with dynamic time warping for speech recognition," in *ICASSP*, 2004.

[3] C. Ma and C.-H. Lee, "A study on word detector design and knowledge-based pruning and rescoring," in *Interspeech*, 2007.

[4] R. Rosenfeld, "A maximum entropy approach to adaptive statistical language modeling," *Computer, Speech and Language*, vol. 10, pp. 187–228, 1996.

[5] B. Roark, M. Saraclar, M. Collins, and M. Johnson, "Discriminative language modeling with conditional random fields and the perceptron algorithm," in *ACL*, 2004.

[6] X. Li, Y.-C. Ju, G. Zweig, and A. Acero, "Language modeling for voice search: a machine translation approach," in *ICASSP*, 2008.

[7] A. Gunawardana, M. Mahajan, A. Acero, and J.C. Platt, "Hidden conditional random fields for phone classification," in *ICASSP*, 2008.

[8] M. Riedmiller and H. Braun, "A direct adaptive method for faster backpropagation learning: The Rprop algorithm," in *Proc. of the IEEE Int. Conf. on Neural Networks*, 1993.

[9] M. De Wachter, M. Matton, K. Demuynck, P. Wambacq, R. Cools, and D. Van Compernolle, "Template-based continuous speech recognition," *IEEE Trans. on Audio, Speech, and Language Processing*, vol. 15, no. 4, pp. 1377–1390, 2007.