Ant Colony Algorithm Applied to Automatic speech Recognition Graph Decoding

Benjamin Lecouteux and Didier Schwab

GETALP - Laboratory of Informatics of Grenoble (LIG) - Univ. Grenoble Alpes

name.surname@imag.fr

10.21437/Interspeech.2015-479

Abstract

In this article we propose an original approach that allows the decoding of Automatic Speech Recognition Graphs by using a constructive algorithm based on ant colonies. In classical approaches, when a graph is decoded with higher order language models; the algorithm must expand the graph in order to develop each new observed n-gram. This extension process increases the computation time and memory consumption. We propose to use an ant colony algorithm in order to explore ASR graphs with a new language model, without the necessity of expanding it. We first present results based on the TED English corpus where 2-grams graph are decoded with a 4-grams language model. Then, we show that our approach performs better than a conventional Viterbi algorithm when computing time is constrained and allows a highly threaded decoding process with a single graph and a strict control of computation time and memory consumption.

Index Terms: Graph decoding, ant colony algorithm, language model, automatic speech recognition, real-time

1. Introduction

In this article, we introduce a new paradigm for the exploration of decoding graphs in automatic speech recognition (ASR) systems. A preliminary part of this work was presented in a French conference [1]. A problem that arises during this exploration of word graphs is when higher-order language models are applied to them. Indeed, the number of paths grows exponentially with the language model order. For example, the size of a simple word graph decoded with a 2-gram model and expanded to a 4-gram model is tenfold. Several techniques have been successfully applied to overcome the problem. Some, such as compact-expansion [2], are approximation methods while others such as beam-search are based on pruning. Here, we present an alternative method based on a constructive version of an algorithm that is widely used in operational research: an ant colony algorithm. Such algorithms have been successfully applied to the travelling salesman problem or lexical disambiguation for example.

The paper is structured as follows: In the 2^{nd} section, we briefly review decoding techniques for ASR system with large vocabularies and then ant colony algorithms in general. The 3^{th}, 4^{th}, 5^{th} sections present the overall architecture of the system used for this work. Then, we present all the experiments we conducted in section 6^{th} and the analysis of their results, followed by a discussion in the 7^{th} section. We conclude by proposing some interesting improvement perspectives based on the properties of ant colony algorithms.

2. Related Work

2.1. Classical approaches

The principle of ASR systems is to find in a graph the hypothesis that will maximize the probabilities of the acoustic and language models. Generally, heuristics are applied to limit the size of the search space. We can find several approaches to dynamically generate hypotheses:

- synchronous graph algorithms that generate a virtual copy of the graph for all hypotheses and that terminate after a fixed time t. The next word is shared but is associated to several timelines. In this approach, the dynamic alignment algorithm is applied for each word without any conservation of their timelines. Such algorithms are rarely implemented [3].
- reentrant graph algorithms, where a virtual copy of the graph is explored for every linguistic context. The information pertaining to each context is recorded for each path and combined with a new virtual root relative to the history in the language model [4]. Reentrant graph algorithms are widely used and the more frequent implementation is the Viterbi beam-search [5, 6]. Their design based on dynamic programming.
- stack asynchronous algorithms [7] where the principle is to do a deep exploration and prioritize promising hypotheses. This is achieved by expanding the selected hypothesis word by word. Their implementation relies on priority stacks that order the hypotheses to explore. The most commonly used asynchronous method is A^* [8, 9, 10].
- in the field of WFST-based speech recognition, several algorithms have been proposed in order to reduce time and memory problem: [11], [12] propose on the fly composition algorithms while [13] propose to factor the language models into smaller components.

For a more detailed and in-depth review of decoding techniques for large vocabulary ASR systems, readers are advised to refer to [14].

The rescoring of a 2-gram graph with 3-gram language model increases drastically the number of nodes. For example, in our experiments the storage of 2-gram graphs takes up to 500MB against 6GB for expanded 4-gram graphs. In order to remove the language model expansion phase, we propose to use an ant colony algorithm to explore the graph.

2.2. Ant colony algorithm

The idea of ant colony algorithms comes from biology and from the observation of ant social behavior. Indeed, ants have the ability to collectively find the shortest path between their nest and a source of energy. It has been demonstrated that cooperation inside an ant colony is self-organized and emerges from interactions between individuals.

These interactions are often very simple and allow the colony to solve complex problems. This phenomenon is called swarm intelligence [15, 16] and is increasingly used in computer science, where centralized control systems are often successfully replaced by other types of control based on interactions between simple elements.

LM order LM order	# n-grams	Perplexity	dev WER	test WER
1g	41K	-	-	-
2g	+ 35M	234	22.01%	21.67%
3g	+ 238M	159	18.1%	17.7%
4g	+ 524M	150	17.4%	16.7%

Table 1: *Details on the different language models.*

Artificial ants have first been used for solving the Traveling Salesman Problem [17]. In these algorithms, the environment is usually represented by a graph, in which virtual ants exploit pheromone trails deposited by others, or pseudo-randomly explore the graph.

Ant colony algorithms are a good alternative for the resolution of problems modeled by graphs. They allow a fast and efficient exploration that exhibits performance close to other search methods. Their main advantage is their high adaptivity to changing environments. Readers can refer to [18], [19] or [20] for a state of the art.

Such algorithms have already been applied for natural language processing tasks successfully including Word Sense Disambiguation [21]. The work presented here aims at expanding this paradigm to automatic speech recognition.

3. ASR system

The ASR system we implement is based on the KALDI toolkit [22]. KALDI offers state of the art tools for automatic speech recognition. The acoustic model and phonetization of our system have been trained with the corpora provided by LIUM [23] (118 hours of annotated data). The enrichment of the delta and delta-delta coefficient parameters, as well as several transformations such as LDA and MLLT [24] are applied. Finally, speaker adaptive training is applied on the acoustic models and combined with fMLLR space type parameter adaptation [25] at the speaker level.

The language model is trained on all the data provided for the IWSLT 2010 campaign as well as the training corpora provided by LIUM. A language model was trained for each subcorpus and all models were interpolated by minimizing the perplexity on the development set, without applying any data selection or pruning.

The final perplexity obtained on the dev is 159 for the 3-gram model and 150 for the 4-gram. The amounts of n-grams and the perplexity of the different models used in our experiments are shown in Table 1.

4. Used Corpora and baseline results

The entirety of the experiments were performed on a corpus made available to the community for the IWSLT 2010 campaign: the TED corpus that corresponds to a set of conference talks recorded in English [26]. Details and baseline results are shown in Table 2.

TED	# Sentences	# Words	duration	3-gram WER	4-gram WER
dev	507	18226	4h12	18.1 %	17.4 %
test	1155	28430	7h30	17.7 %	16.7 %

Table 2: *Details on TED corpora used for the experiments.*

We purposely used an initial bi-gram language model in order to not introduce additional information at the linguistic level in the ASR. Then the graphs were rescored without pruning with 3 and 4-gram language models.

4.1. Graph conversion

Although the ASR system we used is based on the FST framework, the tools we used to implement the Ant Colony Algorithm was developed in LIG and uses graphs based on the HTK format. We have thus converted the entirety of the graphs from KALDI in the HTK format, which allowed us to work with that format. The ant decoder can be downloaded at the following address: `http://getalp.imag.fr/static/wsd/INTERSPEECH2015-Ants4ASR/`

5. Proposed approaches

As a first step, we want to validate the approach by simplifying the decoding stages. For this reason, all the work presented is performed on word graphs extracted from the KALDI ASR. We use an initial bi-gram language model, so that we could use the ant colony algorithm to expand the graph to higher n-gram orders.

Our general implementation of the algorithm is as follows: Each link of the graph is associated with a pheromone variable initialized at the beginning of each epoch. Our ants are launched from the first node of the graph and in order to leave a node, ants will have to choose a link. This choice is performed with a weighted random selection based on the quantity of pheromone on each arc (Figure 1). With the arrival of an ant at the endpoint, if a solution is found (i.e. the acoustic and linguistic probabilities combination) that is better than the previous best solution found, then pheromone is added on the path by an increment of 1. In the literature, it was shown that with such a simple version of the ant colony, ants tended to converge on a path that does not necessarily represent the optimal solution. It is therefore necessary to use local or global heuristics to avoid this phenomenon. The next section presents some heuristics that bias the choice of ants.

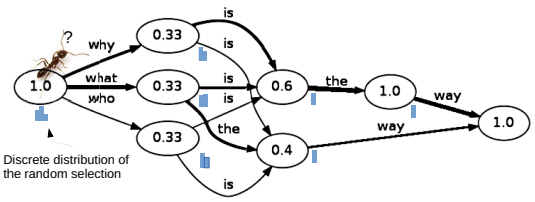

Figure 1: *In order to leave a node, ants will have to choose a link. This choice is performed with a weighted random selection based on the quantity of pheromone on each arc (weighted by the nodes posteriors in our experiments). The thickness of the line expresses the amount of pheromone already deposited.*

5.1. Anamorphic ants

We use different types ants as proposed in [19] using an heuristic exploration biased by the information already present in the initial graph.

Posteriors probabilities $p(a|G)$ are computed on the bi-gram graph for each link a, given the current word graph G. This probability corresponds to the ratio between all the probabilities of all the paths passing through link a and the set of all paths in the graph G: $p(a|G) = \frac{\sum_{C \in G, a \supset C} p(C|G)}{\sum_{C \in G} p(C|G)}$ Where $C \in G$ denotes a full path C in the graph G and $a \supset C$ a link that belongs to the path. This posterior probability is conventionally estimated by using a forward-backward algorithm as shown in [27].

We independently evaluate three variants of the algorithm :

1. Listening ants that are influenced by the acoustic posterior probability, without taking into account the linguistic aspect.
2. Talkative ants that unlike previous ants, are only influenced by the language posterior probability of the next node.
3. Oracles ants, a hybrid model that combines the previous two and that represents the classical posterior probability of the next node.

The various posteriors probabilities are estimated by running Viterbi on the bi-gram graph. For models (1) and (2) we respectively remove the acoustic model or language models and for model (3) we keep the initial score.

The different types of ants are used to guide exploration based on acoustic or linguistic information. In the next section, we present their performance. The detailed algorithm 1 is presented below.

For all of the presented experiments, the number of epochs is set to 5 and the number of ants is proportional to the number of nodes (5 ants/node). The average duration of the decoding with these parameters is approximately 5 seconds per graph (on an Intel Xeon 2.5 Ghz, using a single core).

6. Preliminary experiments and results

Table 3 shows the results obtained with the algorithms presented above. The experiment confirms that orienting starting ants on optimal paths strongly influences the final result. Ants launched on paths that privilege acoustic (model 1) or linguistic information (model 2) perform poorly, but the merged model with more heterogeneous ants (full posterior, model 3) provides meaningful information.

Set	base-line	Ant (1) WER	Ant (2) WER	Ant (3) WER	Base-line time	Ant alg. time
Dev 2g	22.0	28.0	31.0	22.0	1h	2h00
Dev 3g	18.1	27.2	29.2	18.4	13h	2h00
Dev 4g	17.4	26.1	28.7	17.7	18h	2h00
Test 2g	21.6	27.5	30.1	21.6	1h30	2h40
Test 3g	17.7	26.3	28.2	17.9	19h	2h40
Test 4g	16.7	25.3	27.6	17.0	27h	2h40

Table 3: *Decoding results of the ant colony algorithm, ants have WER results that vary between +/- 0.1% during execution (WER scores shown are averaged over 5 runs). The computational time is compared to a Viterbi rescoring algorithm without pruning, as performed in KALDI framework. Ant algorithm decoding time is constant against the language model order.*

These first experiments show that ant colony algorithms are able to expand the graph, but do not reach strictly the performance of baseline results. However, the computational time required with the ant colony algorithm is about 10 times lower than a conventional expansion method: the configurations shown in the Table reach the optimums scores in 1xRT. If we add more iterations, the ants still converge on the same path.

In the literature, it has been shown for example that heuristics that influence the initial distribution of pheromone on the nodes had a significant bearing on the evolution of the anthill.

Moreover [17] shows that the major potential pitfall for ant colony algorithms is its convergence to local optima. One way to avoid such phenomena is to use several dimensions that can be combined *a posteriori*, as shown in [19]. The mixed model with three types of ants was an embodiment of this heuristic and showed its effectiveness.

Algorithm 1: *Ant colony algorithm*

AntColonyAlgorithm()
- 5 ants/node in our experiments;
- $Init\ N_{ants}$;
- 10 epochs in our experiments;
- $Init\ N_{Epochs}$;
- $Epoch \leftarrow 0$;
- $BestScore \leftarrow -inf$;
- $BestPaths \leftarrow \{\}$;
- **foreach** *Node* $N \in Graph$ **do**
  - $Compute\ posteriors\ \phi(N)$
- **end**
- **foreach** *Node* $N \in Graph$ **do**
  - $Ph_N \leftarrow 1$
- **end**
- **while** *Epoch* $< N_{Epoch}$ **do**
  - evaporation of pheromones (empirically determined);
  - **foreach** *Node* $N \in Graph$ **do**
    - $Ph_N \leftarrow Ph_N * 0.6$
  - **end**
  - **foreach** *Node* $N \in BestPaths[0...Epoch]$ **do**
    - $Ph_N \leftarrow Ph_N + 1$
  - **end**
  - $C_{ant} \leftarrow 0$;
  - **while** $C_{ant} < N_{ant}$ **do**
    - $Path \leftarrow \{\}$;
    - $Path = FindPath(FirstNode)$;
    - **if** $Score(Path) > BestScore$ **then**
      - $BestScore \leftarrow Score(Path)$;
      - $BestPaths[Epoch] \leftarrow Path$;
      - **foreach** *Node* $N \in BestPaths[Epoch]$ **do**
        - $Ph_N \leftarrow Ph_N + 1$
      - **end**
    - **end**
    - $C_{ant} \leftarrow C_{ant} + 1$;
  - **end**
- **end**

FindPath(Node)
- This function explore recursively the graph
- $NextNode = SelectNextNode(Node)$;
- **if** $NextNode = EndNode$ **then**
  - $return\ Node$
- **end**
- $Path \leftarrow NextNode + FindPath(NextNode)$;
- $return\ Path$;

SelectNextNode(Node)
- select randomly an output Link of the node according a discrete distribution based on pheromones Ph and posteriors $\phi()$ combination;
- $NextNode$ = $select(\{Links\}, \{Ph_{Link} * \phi(NexNode)\})$;
- $return\ NextNode$;

Score(Path)
- $P_n(W)$ is the n-gram language model and $P(X)$ the acoustic model.;
- $Score$ = $\sum_{p=begin}^{endpath} (logP_n(W_p|w_{p-n}..w_{p-1}) + logP(X_p))$;
- $return\ Score$;

7. Discussion on the ant colony algorithm

7.1. Behavior analysis

In order to understand the evolution of ants we conducted several decoding: we experimented different number of ants relative to the amount of nodes in each graph (left Figure 2). We observe that with about 10 ants per node, the WER remains stable without converging to the optimal solution. This is explained by the fact that ants tend to reinforce a path corresponding to a local optimum.

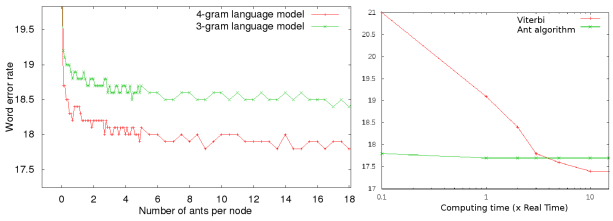

Figure 2: *Evolution of rescoring according the number of ant by node (on the dev set) during 1 run and comparison between Viterbi beam-search and ant colonies algorithm: Computing time against performance, on the dev set with a 4-gram language model. With time constraints the ant algorithm is more efficient than a conventional extension..*

Despite this convergence, we observed that each new random seed generates new solutions. Table 4 shows the selection of the best hypothesis (i.e. best score AM+LM) among 100 different runs. The results show that hypotheses change for each run but are not always complementary.

Set	Viterbi no prune	Viterbi pruned for 1xRT	Ant 1 run WER	10 runs WER	100 runs WER
Dev 3g	18.1	20.2	18.5	18.4	18.3
Dev 4g	17.4	19.1	17.8	17.7	17.7
Tst 3g	17.7	19.8	18.0	17.9	17.8
Tst 4g	16.7	18.5	17.2	17.0	17.0
Mem.3g	1 GB	200MB	100MB	=	=
Mem.4g	3 GB	500MB	100MB	=	=
Time3g	32h	4h40	0h30	4h40	45h
Time4g	45h	4h40	0h30	4h40	45h
xRT	7&10	1	0.1	1	10

Table 4: *Decoding analysis of the ant colony algorithm, using several runs. We report the decoding time (dev+test), the memory footprint (average for one graph) and the WER compared to Viterbi beam-search baseline. We introduce a pruned Viterbi beam-search in order to compare decoding time vs performance.*

As shown in Table 4, this experiment shows that our algorithm is sometimes trapped in wrong areas of the graph. However in the case of constrained computing time, ant colony algorithm outperforms classical expansion algorithm. The Table shows also memory footprint (average for one graph) for the various systems and the computing time for the whole decodings: the results show that these two ressources consumption are independant of the language model size.

In order to make a meaningful comparison with a classical expansion (Viterbi beam-search), we fixed the computing time and limited empirically the beam-width. Results are shown in Table4. The right Figure 2 shows the evolution of WER against the decoding time for the 4-gram rescoring task (dev set).

This result suggests that ant colony algorithms are more effective when rigid pruning are carried out. However the trend is reversed starting from 4xRT. Ant colony algorithm can not always find the exact solution, but with time constraints the algorithm is more efficient than a conventional extension. These experiments highligths a potential disadvantage: the proposed search algorithm include non-determinism and no guarantee of optimality (which Viterbi search with pruning also doesn't have).

7.2. Discussion and perspectives

The parallelization of the ant colony algorithms is trivial. In fact, ants can evolve independently of each other in the search for new paths. Parallelization is also possible at a higher level: that of ant colonies. Different colonies operate on copies of a graph and exchange information by merging copies (i.e. by summing the amount of pheromone present on each node). Finally, the computational time is easily controlled: the number of evaluations is known in advance and the time depends only on the depth of the graph.

Another important factor is the amount of memory needed, which is bounded by the space required to store the graph in memory. As there are no physical extensions, memory usage is constant. In our experiments, the algorithm uses on average about 100MB against several GBs for a conventional expansion.

The evaluation of the solution is performed when ant arrives at the final node of the graph. The number of evaluations is relatively low, so it is easy to add extra information to refine the solution (semantic language model based on neural network etc.). An application of ants colony will also be applied to ASR systems using continuous space language models. Current methods are not integrated: they propose to rescore the graphs or n-best list [28]. Ant colonies can integrate a continuous space model directly in the decoder (the evaluation of an hypothesis is global).

We wish to explore and analyze this ant colony paradigm more in depth when applied to ASR. Future developments will result in the parallel use of ants with different behaviors. Thus, a selection of well performing and behaving individuals will be performed along the exploration.

It would also be interesting to analyze in detail how the graphs are explored: this aspect will allow to optimize exploration strategies further. Finally, in the longer term, we would like to apply ant colony algorithms throughout the ASR process: from the phoneme lattice to the language model expansion.

8. Conclusion

In this article, we presented a new paradigm to expand word graphs in automatic speech recognition systems. This paradigm based on ant colony algorithms for the exploration of the graph, removes the need to dynamically expand the graph: the memory footprint becomes independant of the language model size. Our results show that the paradigm works: the presented algorithm performs better than a conventional Viterbi algorithm when computing time is constrained. Without constraints, ant algorithm and classical expansion are close. This work offers to some interesting research avenues and in the short-term we want to refine the exploration heuristics in order to remove local optimums. Another very interesting aspect is the fact that the ASR hypotheses are scored globally: introducing continuous space language model directly in the decoder would be interesting. Moreover the algorithm is easy to parallelize with a low memory footprint: this work opens up interesting perspectives in terms of memory use and the exploitation of parallelization opportunities that modern microprocessors offer.

9. References

[1] Benjamin Lecouteux and Didier Schwab, "Reconnaissance automatique de la parole laide de colonies de fourmis," in *Journées d'édude de la parole (JEP)*, 2014.

[2] Fuliang Weng, Andreas Stolcke, and Ananth Sankar, "Efficient lattice representation and generation," in *In Proc. of ICSLP*, 1998, pp. 2531–2534.

[3] S. Ortmanns and H. Ney, "The time-conditioned approach in dynamic programming search for continuous speech recognition," *IEEE Transactions on Acoustics Speech and Signal Processing*, vol. 8, no. 6, pp. 676–687, 2000.

[4] H. Ney, R. Haeb-Umbach, B.-H. Tran, and M. Oerder, "Improvements in beam search for 10000-word continuous speech recognition," in *Proc. IEEE International Conference on Acoustics, Speech, and Signal Processing ICASSP-92*, 1992, vol. 1, pp. 9–12 vol.1.

[5] V. Steinbiss, B. Tran, and H. Ney, "Improvements in beam search," in *ICSLP*, 2014.

[6] G. Antoniol, F. Brugnara, M. Cettolo, and Marcello Federico, "Language model representations for beam-search decoding," in *Acoustics, Speech, and Signal Processing, 1995. ICASSP-95., 1995 International Conference on*, May 1995, vol. 1, pp. 588–591 vol.1.

[7] F. Jelinek, "A fast sequential decoding algorithm using a stack," *IBM J. Res. Develop.*, vol. 13, 1969.

[8] D.B. Paul, "Algorithms for an optimal A* search and linearizing the search in the stack decoder," in *Proc. International Conference on Acoustics, Speech, and Signal Processing ICASSP-91*, 1991, pp. 693–696 vol. 1.

[9] S. Renals and M.M. Hochberg, "Start-synchronous search for large vocabulary continuous speech recognition," *Speech and Audio Processing, IEEE Transactions on*, vol. 7, no. 5, pp. 542–553, Sep 1999.

[10] Pascal Nocera, Georges Linarès, Dominique Massonié, and Loïc Lefort, "Phoneme lattice based A* search algorithm for speech recognition," in *TSD*, 2002, pp. 301–308.

[11] Takaaki Hori and Atsushi Nakamura, "Generalized fast on-the-fly composition algorithm for wfst-based speech recognition.," in *INTERSPEECH*. Citeseer, 2005, pp. 557–560.

[12] D. Caseiro and I. Trancoso, "A specialized on-the-fly algorithm for lexicon and language model composition," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 14, no. 4, pp. 1281–1291, 2006.

[13] H.J.G.A. Dolfing and I.L. Hetherington, "Incremental language models for speech recognition using finite-state transducers," in *Automatic Speech Recognition and Understanding, 2001. ASRU '01. IEEE Workshop on*, 2001, pp. 194–197.

[14] Xavier L. Aubert, "An overview of decoding techniques for large vocabulary continuous speech recognition," *Computer Speech & Language*, vol. 16, no. 1, pp. 89–114, 2002.

[15] Eric Bonabeau, David Corne, and Riccardo Poli, "Swarm intelligence: the state of the art special issue of natural computing," *Natural Computing*, vol. 9, no. 3, pp. 655–657, 2010.

[16] Eric Bonabeau, David W. Corne, Joshua D. Knowles, and Riccardo Poli, "Swarm intelligence theory: A snapshot of the state of the art," *Theor. Comput. Sci.*, vol. 411, no. 21, pp. 2081–2083, 2010.

[17] Marco Dorigo and Luca Gambardella, "Ant colony system: A cooperative learning approach to the traveling salesman problem," *IEEE Transactions on Evolutionary Computation*, vol. 1, pp. 53–66, 1997.

[18] Dorigo and Stützle, *Ant Colony Optimization*, MIT-Press, 2004.

[19] Boris Bontoux and Dominique Feillet, "Ant colony optimization for the traveling purchaser problem," *Comput. Oper. Res.*, vol. 35, no. 2, pp. 628–637, Feb. 2008.

[20] Marco Dorigo, Mauro Birattari, Christian Blum, Anders Lyhne Christensen, Andries Petrus Engelbrecht, Roderich Groß, and Thomas Stützle, "Ants 2012 special issue - editorial," *Swarm Intelligence*, vol. 7, no. 2-3, pp. 79–81, 2013.

[21] Didier Schwab, Jérôme Goulian, Andon Tchechmedjiev, and Hervé Blanchon, "Ant colony algorithm for the unsupervised word sense disambiguation of texts: Comparison and evaluation," in *COLING*, 2012, pp. 2389–2404.

[22] Daniel Povey, Arnab Ghoshal, Gilles Boulianne, Lukas Burget, Ondrej Glembek Nagendra Goel Mirko Hannemann Petr Schwarz, Jan Silovsky, Georg Stemmer, and Karel Vesely, "The kaldi speech recognition toolkit," in *IEEE 2011 Workshop on Automatic Speech Recognition and Understanding*, IEEE Signal Processing Society, Ed., December 2011, vol. IEEE Catalog No. : CFP11SRW-USB.

[23] A. Rousseau, P. Deléglise, and Y. Estève, "TED-LIUM: an automatic speech recognition dedicated corpus," in *LREC 2012*, 2012.

[24] R. A. Gopinath, "Maximum likelihood modeling with gaussian distributions for classification," in *Proceedings of ICASSP*, 1998, pp. 661–664.

[25] M.J.F. Gales, "Maximum likelihood linear transformations for hmm-based speech recognition," *Computer Speech and Language*, vol. 12, pp. 75–98, 1998.

[26] Michael Paul, Marcello Federico, and Sebastian, "Overview of the iwslt 2010 evaluation campaign," in *IWSLT 2010*, 2010.

[27] F. Wessel, R. Schlüter, and H. Ney, "Using posterior word probabilities for improved speech recognition," in *Proc. IEEE International Conference on Acoustics, Speech, and Signal Processing ICASSP '00*, R. Schluter, Ed., 2000, vol. 3, pp. 1587–1590 vol.3.

[28] Anoop Deoras, Tomáš Mikolov, and Kenneth Church, "A fast re-scoring strategy to capture long-distance dependencies," in *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, Stroudsburg, PA, USA, 2011, EMNLP '11, pp. 1116–1127, Association for Computational Linguistics.