SUPERVISED, SEMI-SUPERVISED AND UNSUPERVISED METHODS IN DISCRIMINATIVE LANGUAGE MODELING FOR AUTOMATIC SPEECH RECOGNITION

by

Erinç Dikici

B.S., Telecommunication Engineering, İstanbul Technical University, 2006

M.S., Electrical and Electronics Engineering, Boğaziçi University, 2009

Submitted to the Institute for Graduate Studies in
Science and Engineering in partial fulfillment of
the requirements for the degree of
Doctor of Philosophy

Graduate Program in Electrical and Electronics Engineering

Boğaziçi University

2016

SUPERVISED, SEMI-SUPERVISED AND UNSUPERVISED METHODS IN DISCRIMINATIVE LANGUAGE MODELING FOR AUTOMATIC SPEECH RECOGNITION

APPROVED BY:

Assoc. Prof. Murat Saraçlar .....
(Thesis Supervisor)

Prof. Ethem Alpaydın .....

Prof. Levent Arslan .....

Assoc. Prof. Hakan Erdoğan .....

Assoc. Prof. Engin Erzin .....

DATE OF APPROVAL: 21.06.2016

ACKNOWLEDGEMENTS

I would like to express my sincere gratitude to my supervisor, Assoc. Prof. Murat Saraçlar, for his invaluable guidance and support throughout this thesis study. I always admire his kind attitude, intelligence, patience and confidence in my ability to succeed.

I am grateful to Prof. Ethem Alpaydın, Prof. Levent Arslan, Assoc. Prof. Hakan Erdoğan and Assoc. Prof. Engin Erzin for their valuable ideas and comments from the beginning of this study, and for participating in my thesis evaluation committee. I would also like to thank Assoc. Prof. Brian Roark and Assoc. Prof. Izhak Shafran for accepting me as an exchange student to their study groups at Oregon Health & Science University (OHSU), Center for Spoken Language Understanding (CSLU).

I owe a word of appreciation to Prof. Bülent Sankur, who has supported me throughout my graduate studies and enlightened me with his wisdom both academically and culturally. With this document that officially marks the end of my studentship, I would like to thank all of my teachers who have, from the very beginning, led me to reach this point.

I send my special thanks to Ebru Arısoy for her continuous help with the dataset preparation and baseline ASR setup, to Murat Semerci for optimizing the training algorithms and implementing dimensionality reduction techniques, to Arda Çelebi for sharing his extensive toolbox for the semi-supervised artificial hypothesis generation framework, to Emily Prud'hommeaux for her kind help on machine translation based confusion modeling, and to Haşim Sak for his constructive feedback. This work would not be completed without their generous contributions.

My sincere thanks go to all my past and present colleagues at BÜSİM for their freindship, for sharing my happiness and troubles, and for turning the lab into a warm and fun environment with their presence. I would also like to thank all friends at EE and CmpE for the joyful moments we had.

I remember with respect my dear friends, İsmail Arı and Özgür Dalkılıç, two of the most considerate people I have ever known. I will always cherish our memories and friendship.

This research was supported in part by the Scientific and Technical Research Council of Turkey (TÜBİTAK) project 109E142, Turkish State Planning Organization (DPT) under the TAM project 2007K120610, the Bogazici University Research Fund (BU-BAP) project 14A02D3, and by Turkish Academy of Sciences (TÜBA) as part of Murat Saraçlar's TUBA-GEBİP award. The data used in this study, Bogazici University Turkish Broadcast News Database, was collected as part of the TÜBİTAK project 105E102 and the BU-BAP project 05HA202. My visit to OHSU was supported by TÜBİTAK 109E142 and by OHSU through the NSF project 0964102. Part of the numerical calculations reported in this thesis were performed at TÜBİTAK ULAKBİM, High Performance and Grid Computing Center (TRUBA Resources) and at CSLU clusters.

Finally, I would like to express my deepest gratitude to my dear parents, Ferhan-Ahmet Dikici, who have supported me with their endless love and support throughout my life, and to my dear wife, Nilay Şentürk Dikici, for her enduring love, understanding and patience.

ABSTRACT

SUPERVISED, SEMI-SUPERVISED AND UNSUPERVISED METHODS IN DISCRIMINATIVE LANGUAGE MODELING FOR AUTOMATIC SPEECH RECOGNITION

Discriminative language modeling aims to reduce the error rates by rescoring the output of an automatic speech recognition (ASR) system. Discriminative language model (DLM) training conventionally follows a supervised approach, using acoustic recordings together with their manual transcriptions (reference) as training examples, and the recognition performance is improved with increasing amount of such matched data. In this thesis we investigate the case where matched data for DLM training is limited or not available at all, and explore methods to improve ASR accuracy by incorporating unmatched acoustic and text data that come from separate sources. For semi-supervised training, we utilize weighted finite-state transducer and machine translation based confusion models to generate artificial hypotheses in addition to the real ASR hypotheses. For unsupervised training, we explore target output selection methods to replace the missing reference. We handle discriminative language modeling both as a structured prediction and a reranking problem and employ variants of the perceptron, MIRA and SVM algorithms adapted for both problems. We propose several hypothesis sampling approaches to decrease the complexity of algorithms and to increase the diversity of artificial hypotheses. We obtain significant improvements over baseline ASR accuracy even when there is no transcribed acoustic data available to train the DLM.

ÖZET

OTOMATİK KONUŞMA TANIMA İÇİN AYIRICI DİL MODELLEMEDE GÖZETİMLİ, YARI-GÖZETİMLİ VE GÖZETİMSİZ YÖNTEMLER

Ayırıcı dil modelleme bir otomatik konuşma tanıma (OKT) sisteminin çıktılarını yeniden değerlendirerek hata oranlarını azaltmayı amaçlar. Ayırıcı dil modeli (ADM) eğitimi geleneksel olarak akustik kayıtların elle yazılandırılmış referanslar ile birlikte eğitim verisi olarak kullanıldığı gözetimli yöntemle yapılır. Konuşma tanıma başarımı söz konusu eşlenik verinin miktarına bağlı olarak artırılabilir. Bu tezde ADM eğitimi için gerekli eşlenik verinin yetersiz olduğu ya da mevcut olmadığı durumlar incelenmiş ve farklı kaynaklardan gelen akustik ve metinsel veriler kullanılarak OKT doğruluğunun iyileştirilmesine yönelik yöntemler araştırılmıştır. Yarı-gözetimli eğitim için, gerçek OKT hipotezlerine ek olarak yapay hipotezler türetmeyi sağlayan ağırlıklı sonlu durum makinesi ve makine çevirisi tabanlı karışıklık modelleri kullanılmaktadır. Gözetimsiz eğitim için, olmayan referansın yerine geçebilecek üç farklı hedef çıktı seçim yöntemi tanıtılmaktadır. Ayırıcı dil modelleme işlemi hem yapısal kestirim hem de yeniden sıralama problemi olarak ele alınmakta ve eğitim için algılayıcı, MIRA ve destek vektör makinesi algoritmalarının her iki problem için uyarlanmış çeşitleri kullanılmaktadır. Algoritmaların işlemsel karmaşıklığını azaltmak ve türetilen yapay hipotezlerin çeşitliliğini artırmak amacıyla çeşitli hipotez örnekleme yöntemleri önerilmektedir. Yapılan deneyler sonucunda elle yazılandırılmış akustik konuşma verisinin bulunmadığı durumda dahi temel OKT başarımında istatistiksel olarak anlamlı iyileştirmeler elde edilmiştir.

TABLE OF CONTENTS

ACKNOWLEDGEMENTS iii
ABSTRACT v
ÖZET vi
LIST OF FIGURES xi
LIST OF TABLES xii
LIST OF SYMBOLS xvi
LIST OF ACRONYMS/ABBREVIATIONS xviii
1. INTRODUCTION 1
1.1. Challenges in Discriminative Language Modeling 3
1.2. Main Contributions of the Study 5
1.3. Outline of the Thesis 7
2. DISCRIMINATIVE LANGUAGE MODELING FOR ASR 8
2.1. ASR System Architecture 8
2.2. Problem Setting 11
2.3. Language Units 12
2.4. Data Types and Datasets 13
2.5. Baseline System and Preparation of the DLM Data 16
2.6. Training Scenarios 17
2.7. Previous Work 18
3. SUPERVISED DISCRIMINATIVE LANGUAGE MODELING 22
3.1. Linear Model 22
3.2. Training Algorithms 24
3.2.1. Structured Perceptron 25
3.2.2. Ranking Perceptron 27
3.2.3. Margin Infused Relaxed Algorithm (MIRA) 29
3.2.4. Ranking MIRA 31
3.2.5. Support Vector Machine (SVM) 31
3.2.6. Ranking SVM 32
3.3. Testing 33

3.4. Hypothesis Sampling . . . 33
3.5. Experimental Setup . . . 35
3.6. Experimental Results . . . 35
3.6.1. Performance of Training Algorithms . . . 36
3.6.1.1. Structured Perceptron . . . 36
3.6.1.2. Ranking Perceptron . . . 37
3.6.1.3. Margin Infused Relaxed Algorithm (MIRA) . . . 38
3.6.1.4. Ranking MIRA . . . 38
3.6.1.5. Support Vector Machine (SVM) . . . 39
3.6.1.6. Ranking SVM . . . 39
3.6.2. Performance with Different Number of Hypotheses and Features 40
3.6.2.1. Sampling from the N-Best List . . . 40
3.6.2.2. Increasing the Number of Features . . . 41
3.6.2.3. Dimensionality Reduction . . . 42
3.7. Analysis of Results . . . 43
3.7.1. Comparison of CPU Times . . . 44
3.7.2. Comparison of Models . . . 45
3.7.3. Comparison of Test Set Accuracies . . . 45
3.7.4. Comparison of Statistical Significance . . . 48
3.8. Discussion . . . 48
4. SEMI-SUPERVISED DISCRIMINATIVE LANGUAGE MODELING . . . 51
4.1. Generating Artificial Hypotheses via Confusion Modeling . . . 52
4.1.1. Weighted Finite-State Transducer (WFST) Based CM . . . 52
4.1.2. Machine Translation (MT) Based CM . . . 54
4.2. Language Model Reweighting . . . 55
4.3. Hypothesis Sampling . . . 56
4.4. Experimental Setup . . . 56
4.5. Experimental Results . . . 57
4.5.1. Evaluation of WFST-based Confusion Modeling . . . 57
4.5.1.1. Effect of the CM Type . . . 58
4.5.1.2. Effect of the Hypothesis Selection Scheme . . . 59

4.5.1.3. Combining Real Hypotheses with Artificial Hypotheses 60
4.5.2. Performance Comparison of Training Algorithms for WFST-based Confusion Modeling 61
4.5.2.1. Comparison of Training Algorithms Under Real Data . 61
4.5.2.2. Comparison of Training Algorithms Under Artificial Data 62
4.5.2.3. Overall Comparison With Respect to Setup Conditions 62
4.5.2.4. Test Set Performance 63
4.5.3. Comparison of WFST- and MT-based CM 64
4.5.3.1. Effectiveness of Artificial Data 64
4.5.3.2. Combination of WFST and MT Hypotheses 65
4.6. Analysis of Results 66
4.7. Discussion 67
5. UNSUPERVISED DISCRIMINATIVE LANGUAGE MODELING 69
5.1. Choosing the Target Output in the Absence of the Reference 69
5.1.1. 1-best 70
5.1.2. Minimum Bayes Risk 70
5.1.3. Segmental MBR 71
5.2. Experimental Setup 71
5.3. Experimental Results 72
5.3.1. Unsupervised DLM Training 73
5.3.2. Unsupervised CM Training 74
5.3.3. Combination of Data 75
5.3.4. MT-based Confusion Modeling for Unsupervised CM 75
5.4. Data Dependency of DLM Training Scenarios 77
5.4.1. Supervised Training 78
5.4.2. Semi-supervised Training 79
5.4.3. Unsupervised DLM Training 80
5.4.4. Unsupervised CM Training 81
5.4.5. Combination of Methods 83
5.5. Analysis of Results 84
5.5.1. Optimal Data Combination 84

5.5.2. Effectiveness of Artificial Hypotheses 86
5.5.3. Effect of Using Unmatched Audio and Out-of-Domain Source Text 87
5.6. Discussion 90
6. CONCLUSION 91
REFERENCES 93
APPENDIX A: RELATIONSHIP BETWEEN PerRank AND SVMrank 102

LIST OF FIGURES

Figure 2.1. An example 10-best list. 12

Figure 2.2. Parsing of an example sentence into different language units. . . . 13

Figure 3.1. The generic structured perceptron algorithm. 25

Figure 3.2. The generic ranking perceptron algorithm. 27

Figure 3.3. RPerRank WER (%) on h with respect to the number of epochs. . 38

Figure 3.4. Higher-order n-grams WER (%) on h. 42

Figure 3.5. Count-based thresholding WER (%) on h. 43

Figure 4.1. An example CM for the word “vize”. 53

Figure 4.2. Word error distribution on h 60

Figure 5.1. WPerRank WER (%) on h for different types and amounts of training data. 83

Figure 5.2. WPerRank WER (%) on h for different number of $\mathcal{T}$ and $\mathcal{A}$ data pieces for training. 85

LIST OF TABLES

Table 2.1. Bogazici University Turkish Broadcast News Database. 14

Table 2.2. Other datasets involved in the study. 15

Table 2.3. Training scenarios. 17

Table 3.1. Naming of algorithms with respect to margin functions. 26

Table 3.2. An example of sampling schemes. 34

Table 3.3. Per WER (%) on h. 37

Table 3.4. RPerRank WER (%) on h. 37

Table 3.5. MIRA WER (%) on h. 38

Table 3.6. MIRArank WER (%) on h. 39

Table 3.7. SVM WER (%) on h. 39

Table 3.8. SVMrank WER (%) on h. 40

Table 3.9. Sampling schemes WER (%) on h. 41

Table 3.10. SVMrank training CPU times for fixed C=1000. 44

Table 3.11. Pairwise comparison of models in terms of the number of Zero and NonZero features they use on unigrams. 46

Table 3.12. Feature comparison for Perceptron and MIRA. 47

Table 3.13. WER (%) on e. 47

Table 3.14. p values of MAPSSWE test. 48

Table 3.15. 10-fold cross-validated WER (%) on e. 48

Table 3.16. p values of 10-fold cross-validation t test on e. 49

Table 4.1. WPer WER (%) on h for different CMs and LMs with ASRdist-50. 58

Table 4.2. Sampling from the 1000-best list with morph CM and ASR-LM. . 59

Table 4.3. WPer WER (%) on e for combining real and artificial N-best lists. 61

Table 4.4. Training: $m_{2/2}$ real, WER(%) on h. 62

Table 4.5. Training: $m^{\mathcal{T}}_{2/2}$ artificial, WER(%) on h. 62

Table 4.6. Training: $m^{\mathcal{T}}_{2/2}$ artificial, Rank averages on h. 63

Table 4.7. WER (%) on e. 64

Table 4.8. WER (%) on h. 65

Table 4.9. WER(%) on h and e with GEN-LM. 66

Table 4.10. Cosine similarities between real and artificial hypotheses. 67

Table 4.11. Number of utilized features (GEN-LM). 67

Table 5.1. Baseline and oracle WER (%). 72

Table 5.2. Unsupervised DLM training WER (%)
(Baseline: h 22.9%, e 22.4%). 73

Table 5.3. Unsupervised CM training WER (%)
(Baseline: h 22.9%, e 22.4%). 74

Table 5.4. WPerRank data combination WER (%)
(Baseline: h 22.9%, e 22.4%). 75

Table 5.5. WPerRank WER (%) for different training data types. 76

Table 5.6. N-best combination WPerRank WER (%). 77

Table 5.7. Supervised training WER (%). 78

Table 5.8. Semi-supervised training WER (%) on h. 79

Table 5.9. Combining supervised and semi-supervised setups, WER (%) on h. 80

Table 5.10. Unsupervised DLM training with $m^{\mathcal{A}}_{1/3}$, WER (%) on h. 80

Table 5.11. WPerRank 1-best unsupervised DLM training WER (%). 81

Table 5.12. Unsupervised CM training WER (%) on h. 82

Table 5.13. Similarity of artificial hypotheses to real hypotheses. 86

Table 5.14. KL divergence of artificial examples. 87

Table 5.15. Unsupervised CM training with $m^{\mathcal{A}}$ and t, WER (%) on h. 88

Table 5.16. Unsupervised CM training with a and t, WER (%) on h_2. 89

LIST OF SYMBOLS

a	Acoustic dataset
c	Class of a training instance
d	Dimension of the feature vector
e	Test (evaluation) dataset
$g(\cdot)$	Margin function
h	Validation (held-out) dataset
i, j	Training instance (utterance number)
m	Matched dataset
$m^{\mathcal{A}}$	Acoustic part of the matched dataset
$m^{\mathcal{T}}$	Textual part of the matched dataset
$m_{a/b}$	The a-th piece of a b-piece dataset
n	Number of consecutive units in language modeling (n-gram)
$p(\tilde{y}\|x)$	Recognition score
r	Rank of a hypothesis
t	Textual dataset
$\mathbf{w}$	Discriminative language model vector
w_0	Weight of the recognition score
x	Acoustic (speech) input
y	Reference, source text *or* target output
y_i	Oracle hypothesis for the i-th utterance
$\tilde{y}$	Hypothesis
z_i	Current best hypothesis for the i-th utterance
$\mathcal{A}$	Acoustic data *or* real N-best lists of acoustic data
C	SVM trade-off parameter
CM	Confusion model
$\mathcal{G}$	Generative language model
$\mathbf{GEN}(\cdot)$	Hypothesis generating function
I	Number of training instances (utterances)

K	Cross validation fold number
$\mathcal{L}$	Lexicon
$\mathcal{M}$	Matched data
N	Number of hypotheses for a specific utterance (N-best)
T	Number of iterations
$\mathcal{T}$	Textual data *or* artificial N-best lists of textual data
W	Word sequence
$\tilde{\mathcal{Y}}$	Set of hypotheses (N-best list)
β	SVM class weight
η	Learning rate
γ	Decay rate
ϵ	SVM slack variable
ϕ_0	Recognition score
τ	Margin multiplier
$\Delta(\cdot)$	Levenshtein (edit) distance
$\mathbf{\Phi}(\cdot)$	Feature vector

LIST OF ACRONYMS/ABBREVIATIONS

AM	Acoustic Model
ASR	Automatic Speech Recognition
ASRdist	ASR Distribution Sampling
BLEU	Bilingual Evaluation Understudy
BN	Broadcast News
CBT	Count Based Thresholding
CM	Confusion Model
CPU	Central Processing Unit
DLM	Discriminative Language Model
GCLM	Global Conditional Log-Linear Model
KL	Kullback-Leibler
LDC	Linguistic Data Consortium
LM	Language Model
LVCSR	Large Vocabulary Continuous Speech Recognition
MAPSSWE	Matched Pair Sentence Segment Word Error
MBR	Minimum Bayes Risk
MIRA	Margin-Infused Relaxed Algorithm
MIRArank	Ranking MIRA
MT	Machine Translation
NIST	National Institute of Standards and Technology
PCA	Principal Component Analysis
Per	Structured Perceptron
RC	Rank Clustering
RG	Rank Grouping
RPer	Reciprocal Perceptron
RPerRank	Reciprocal Ranking Perceptron
SegMBR	Segmental MBR
SMT	Statistical Machine Translation

SRILM	SRI Language Model Toolkit
SVM	Support Vector Machine
SVMrank	Ranking SVM
US	Uniform Sampling
WE	Number of Word Errors
WER	Word Error Rate
WFST	Weighted Finite-State Transducer
WPer	WER-sensitive Perceptron
WPerRank	WER-sensitive Ranking Perceptron

1. INTRODUCTION

The aim of automatic speech recognition (ASR) is to transcribe speech into text. An ASR system uses acoustic and linguistic information together to process an input speech utterance and outputs possible transcriptions, called the *hypotheses*. These are typically arranged in an *N-best list* and ordered with respect to their *recognition scores*, i.e., the posterior probabilities assigned by the recognizer. The accuracy of a hypothesis is defined in terms of the *number of word errors (WE)* it has with respect to the *reference*, which is the manual transcription of the utterance. However, the hypothesis having the highest recognition score is not necessarily the most accurate transcription. Discriminative language modeling for ASR is proposed as a post-processing step to determine the most accurate hypothesis in the N-best list by considering other linguistic factors besides the recognition score.

Discriminative language modeling is different from generative language modeling. In generative language modeling the aim is to assign probabilities to sequences of words in the language being modeled. A generative language model (LM), also called a statistical language model, is an integral component of an ASR system, and is used in conjunction with the acoustic model (AM) to generate meaningful transcriptions of an acoustic input. On the other hand, in discriminative language modeling the aim is to discriminate the best transcription of the ASR output from the others, or to simply reorder the ASR outputs with respect to their accuracies. In that sense, discriminative language modeling is not an alternative but a complementary technique to generative language modeling.

The traditional way of training a discriminative language model (DLM) for ASR requires the acoustic speech utterances together with their corresponding reference transcriptions as input examples. The acoustic input is passed through the ASR to obtain the candidate hypotheses, and the reference transcriptions act as the ground truth to determine the accuracy of these hypotheses. This type of training, inspired by the machine learning literature, is called *supervised* training.

The accuracy of discriminative language modeling depends on the amount of training data. The more training examples, the more likely the DLM is to accurately model the language. However, this requires listening to and manually transcribing many hours of recorded speech, which is a costly process in terms of both time and labor. Therefore, the performance of supervised DLM training methods is limited by the extent of data that can be afforded.

One way to facilitate DLM training when the data is inadequate is to make use of a separate text corpus through a process called *confusion modeling*. Most of the time, finding such a corpus is easier than transcribing a large number of spoken utterances. In this scenario, a small number of transcribed speech data is used to build a *confusion model* (CM), which captures the confusions (errors) made by the ASR system. This CM is then used to transform the sentences from the text corpus (the *source text*) into *artificial hypotheses* which look like the real ASR hypotheses. The accuracy of the artificial hypotheses can easily be determined since their reference, the source text, is already known. This process is sometimes referred to as *semi-supervised* training in the literature, because a small number of transcribed data is used to "label" the rest of the training examples.

An inferior scenario than the data inadequacy problem mentioned above is to have no transcribed data at all. One practical example of this is ASR for underresourced languages where there are no experts to transcribe the spoken language. Moreover, in applications where the speaker's identity must be kept confidential, listening to the recordings in order to manually transcribe them might not be allowed. In such a case, it is still possible to train a DLM by making use of the ASR outputs of untranscribed speech. Since this necessitates DLM training to be done without any supervision, such a case is called *unsupervised* training.

In the scope of this thesis we explore the supervised, semi-supervised and unsupervised scenarios and propose methods to achieve DLM training under such conditions.

1.1. Challenges in Discriminative Language Modeling

This thesis touches upon several problems of discriminative language modeling, which we briefly explain in this section.

Canonical approaches define discriminative language modeling as a classification problem where the aim is to discriminate the best possible transcription of the ASR output from the others, while optimizing an objective function that is directly related to the word error rate (WER). But the hypotheses that are not the best transcription are not all equally bad, and one of the contributions in this study is to formulate this as a reranking problem which promises to be more informative. In this formulation, each hypothesis has a rank based on an accuracy measure, e.g., the number of word errors, and the aim is to reorder the hypotheses in such a way that more accurate ones are pushed towards the top of the list, as opposed to simply separating the best from the rest. We explore classification and reranking variants of three algorithms, namely the perceptron, the margin infused relaxed algorithm and the support vector machine, and compare their performances to show that reranking is a better fit for DLM training.

The type and amount of data that is used for training the DLM plays an important role on the effectiveness of discriminative language modeling on ASR performance. Supervised DLM training requires an acoustic speech corpus, an in-domain ASR system to generate the hypotheses, and the transcriptions of the spoken utterances as the reference. Since the acoustic data and the transcriptions belong to the same utterances, we call this collection the *matched* data.

Semi-supervised technique eliminates the need for a large number of matched data by generating artificial hypotheses that resemble real ASR N-best lists via confusion modeling. Two important aspects of this setting are construction of the CM and efficiency of the generated hypotheses. We present a complete artificial hypothesis generation framework which includes two types of CMs, one based on weighted finite-state transducers (WFST) and the other on machine translation (MT). We build CMs based on various units of the language to present the relationship between the granularity and efficiency.

By contrast, in the unsupervised case there is absolutely no transcribed text accompanying the acoustic input. Without transcribed text the accuracy of the hypotheses cannot be determined. We present our approaches to choose a word sequence for unsupervised training which will serve as the missing reference text. We explore three methods based on the recognition score and the Minimum Bayes Risk (MBR) criterion. We use this word sequence to achieve DLM training conditions similar to the supervised and semi-supervised setups, by either training the DLM directly using the real hypotheses or building a CM first to generate artificial data for training.

The semi-supervised and unsupervised cases allow for incorporating unmatched acoustic and textual data that are coming from different sources for DLM training. The unmatched sources can be in-domain or out-of-domain. For the availability of in-domain text data that is not accompanied by audio data, two mainstream examples come to mind. The first and classical one is the dictation task where the text comes from existing (in-domain) documents (especially for business, law and medical domains). The second and more recent example is voice-enabled search applications where there is an abundance of written search queries which are in-domain but perhaps following a slightly different style. The source text can also be out-of-domain, as is the case for using newspaper articles as the source text for a broadcast news transcription task. The domain mismatch may result in a variety of different words unseen in the CM training phase.

Data sets used in speech recognition are very large and for each instance, discriminative training uses the hypotheses in the N-best list. As another direction, we investigate whether we can decrease complexity without loss in accuracy. We try two possibilities: (1) With the high-order n-grams that we use as the input features, most will appear rarely in training and would not be informative. A simple count-based thresholding prunes the rare combinations, decreases the input dimensionality considerably and hence the training complexity of the learner that follows it. (2) In the N-best list, it is possible that most of the hypotheses will be very similar and the idea is that we can get the same discriminative power by using a small subset from the list, which will lead to significant saving from computation especially in the case of reranking.

Being an agglutinative language with rich morphological structure, Turkish is a challenging language in terms of obtaining large vocabulary continuous speech recognition (LVCSR) accuracy. The possibility of producing almost infinitely many words from a single stem results in very high WERs, as compared to systems of similar vocabulary sizes in English. It has been shown in previous studies that using sub-lexical units such as statistical morphs, instead of word-based models, help enhance the recognition performance. We extend this idea in confusion modeling to generate more efficient artificial hypotheses that look similar to the real ASR outputs.

1.2. Main Contributions of the Study

This thesis study has been conducted as part of the joint projects numbered NSF #0963898/NSF #0964102/TÜBİTAK 109E142: Semi-Supervised Discriminative Training of Language Models, and the project BU-BAP 14A02D3: Unsupervised Approaches in Discriminative Language Modeling for Automatic Speech Recognition. Parts of the research include contributions of the project partners from Bogazici University Department of Computer Engineering (BU-CmpE) and Oregon Health & Science University Center for Spoken Language Understanding (OHSU-CSLU). The outcomes of the research study have been published in two national and seven international conference proceedings and in two journal articles, cited in chronological order in [1–11]. In line with the challenges mentioned in Section 1.1, the main contributions of this thesis study to discriminative language modeling literature will be summarized in the following list:

(i) Introduction of the reranking approach to DLM training for ASR: We present reranking variants of three algorithms (perceptron, MIRA and SVM)[1] that are popularly used in a classification (structured prediction) setting for DLM training. We show that the reranking approach is more suitable for the discriminative language modeling task for ASR by considering the improvements in WER with this setting. This study has been published in [4].

(ii) Hypothesis sampling approaches: We propose several hypothesis sampling approaches to decrease the algorithmic complexity of training algorithms for the supervised case in [1][2]. The same idea is then adapted to semi-supervised training for selecting a sufficiently diverse subset of artificial hypotheses based on their WER distribution in [2][3].

(iii) Artificial hypothesis generation pipeline: We propose a novel three-step artificial hypothesis generation pipeline for semi-supervised DLM training, consisting of confusion modeling, LM rescoring and hypothesis sampling components[4]. The architecture is first introduced in [2], and the performances of training algorithms with respect to different types of the CM, LM, and different sampling schemes are compared in [3].

(iv) Unsupervised DLM training by reranking: We adapt the reranking approach for unsupervised DLM training and compare three target output selection schemes in [7].

(v) Unsupervised confusion modeling: We propose an unsupervised confusion modeling approach which adopts the target output selection scheme to CM training. This approach allows for training of discriminative models with acoustic and language data coming from completely different domains. This study has been presented in [8].

(vi) Introduction of MT-based confusion modeling: We introduce a new confusion modeling scheme based on the MT framework, which is a phrase-based approach as opposed to the context-independent WFST approach presented earlier. We investigate the use of MT-based artificial hypotheses for the semi-supervised case in [6][5], and extend it to the unsupervised CM case in [9].

(vii) Performance analysis with respect to data types, sources and sizes: We provide a detailed analysis of supervised, semi-supervised and unsupervised discriminative language modeling performance with respect to the types (matched, acoustic, textual), sources (in-domain, out-of-domain), and amounts of data they require. This study has been published in [11].

1.3. Outline of the Thesis

This thesis is organized as follows: In Chapter 2, theoretical and practical background on discriminative language modeling for ASR is presented. This includes a brief introduction to ASR and the discriminative language modeling problem, explanation of the language units, data types and datasets used in this study, information about the baseline system and the training scenarios, and a literature review on discriminative language modeling.

Chapters 3, 4 and 5 include the novel contributions of this study. In Chapter 3 we investigate the supervised DLM training scenario. We present the structure of the model, the training algorithms and the hypothesis sampling approaches we propose. In Chapter 4 we explore the semi-supervised DLM training scenario and present our pipeline for confusion modeling and artificial hypothesis generation. In Chapter 5 we focus on the unsupervised DLM training scenario, where we present different techniques to replace the missing reference. Each of these chapters include sections that explain the experimental setup, results of the experiments and a discussion of the outcomes.

Chapter 6 concludes this thesis with a summary of achievements, an overall discussion and future directions.

[1]The adaptation and coding of perceptron and MIRA for reranking were done in collaboration with Murat Semerci and Prof. Ethem Alpaydın (BU-CmpE).

[2]The application of dimensionality reduction approaches (online-PCA and count-based thresholding) in this study was done by Murat Semerci and Prof. Ethem Alpaydın (BU-CmpE).

[3]The tool for sampling artificial hypotheses by WER analysis was prepared by Arda Çelebi (BU-CmpE).

[4]The toolbox for training WFST-based CMs and generating artificial hypotheses was prepared by Arda Çelebi (BU-CmpE).

[5]This study has been conducted in collaboration with Dr. Emily Prud'hommeaux and Prof. Brian Roark during the author's visit to OHSU-CSLU as an exchange researcher.

2. DISCRIMINATIVE LANGUAGE MODELING FOR ASR

Before we explore the methods on how to train a DLM, in this chapter we present the setting which discriminative language modeling is based upon. We begin with a brief summary of the architecture of an ASR system, which creates the input data for discriminative language modeling. We then explain the need for discriminative language modeling. Different language units and data types can be involved in discriminative language modeling, and we touch upon these issues in the sections that follow. We then explain how the baseline system is constructed, and present the training scenarios which will be explored in detail in the following chapters. We conclude this chapter with a review of previous studies on the subject.

2.1. ASR System Architecture

An ASR system is used to automatically transcribe speech into text. ASR systems model speech as a well-formulated statistical process whose parameters are estimated from a corpus of audio samples and corresponding manual transcriptions. In recognition, the system analyses the input speech signal and tries to find the most likely word sequence that could have been uttered. In mathematical terms, this corresponds to

$$\hat{W} = \underset{W}{\operatorname{argmax}}\, P(W|A) \tag{2.1}$$

Here, A is the sequence of feature vectors obtained from the acoustic signal, W is a particular word sequence of the language being spoken, $P(\cdot)$ is the probability function, and $\hat{W}$ is the most likely word sequence. Using Bayes' formula, Equation 2.1 can be written as

$$\hat{W} = \underset{W}{\operatorname{argmax}}\, \frac{P(A|W)P(W)}{P(A)} \tag{2.2}$$

The denominator $P(A)$ does not affect the result of the argmax operation and thus can be omitted, resulting in

$$\hat{W} = \underset{W}{\operatorname{argmax}}\, P(A|W)P(W) \tag{2.3}$$

Equation 2.3 is considered to be the fundamental equation of ASR. Let us now investigate the three components of this equation and the names given to these components in the ASR system architecture:

$P(A|W)$ is called the acoustic model (AM). An AM assigns probabilities to acoustic features A for a particular word sequence W. The most common acoustic features for processing speech are the Mel-Frequency Cepstral Coefficients (MFCC), Linear Predictive Coding Coefficients (LPC) and Perceptual Linear Prediction Coefficients (PLP). Temporal dynamics of the speech signal can also be represented by the first (Δ) and second ($\Delta\Delta$) derivatives of these features. In this study we use the MFCCs together with Δ and $\Delta\Delta$s as the acoustic features.

A popular structure to represent the AM is the Hidden Markov Model. HMMs are used to describe a time-varying process such as the articulation of a phoneme which can also be affected by its neighboring phonemes by co-articulation. In a typical left-to-right triphone model, each phone is represented by three states representing the beginning, progression and end of the articulation. There are recurrent connections that allow for temporal dynamics of staying in the same state or making a transition to the next, each of which is associated with a probability. Each state has also an output probability distribution over the feature vectors, which is generally modeled by a mixture of Gaussians.

Other structures which have become increasingly popular to represent the AM are Artificial Neural Networks (ANN) and their adaptations such as Deep Neural Networks (DNN) and Recurrent Neural Networks (RNN). A DNN is an ANN with multiple hidden layers of units between the input and output layers. DNNs can model non-linear relationships, which enables composition of features from lower layers, thus giving them

the potential of representing complex patterns of speech data [12]. RNNs can further provide the network with an internal memory which allows it to incorporate dynamic temporal behavior [13].

$P(W)$, the language model (LM), is also called the statistical or generative LM to distinguish it from the DLM. An LM assigns probabilities to sequence of words in the language being recognized. The LM is typically represented using an n-gram formulation, where n denotes the depth (memory) of the model:

$$P(W) = P(w_1, ..., w_n) = \prod_i P(w_i|w_{i\text{-}1}, ..., w_1) \approx \prod_i P(w_i|w_{i\text{-}1}, ..., w_{i\text{-}1\text{-}n}) \qquad (2.4)$$

The estimation of probabilities in Equation 2.1 are determined by analyzing large text corpora (millions of words of text). Probabilities of unseen sequences in the training corpora are estimated using a variety of smoothing methods [14].

The creation of language models also depends on the underlying vocabulary. One can use individual word-forms (full-forms) as separate entities for the language model, or a sub-word form. Full-forms work very well for languages with low levels of inflection (e.g. English), but encounter difficulties when used for languages with high inflection (e.g. Turkish). The unit for language modeling thus is determined by the structure of the language under consideration and the amount and types of training data available for model creation. We consider different language units used in this study in Section 2.3.

argmax_W is the decoder which produces the most likely sequence of words with respect to the trained AM and LM. The most common approach is a time-synchronous search using the Viterbi algorithm [15], in which words and their pronunciations are aligned to audio features in a dynamic process. Due to the large size of the vocabulary and the resulting possible word combinations, in practice, the search is only carried out on a subspace of all possible word sequences via pruning the search space along the way to keep it manageable. The network that is generated as a result of this process is called

the *lattice*. During search, the acoustic and language model scores are applied jointly to yield scores for the paths of this lattice, which are called the hypotheses. N top scoring hypotheses of the lattice are returned in an N-best list, and the highest-scoring hypothesis is produced as the final decoding result. However, the other hypotheses in the N-best list can also serve as valuable information, which is the main point of focus in discriminative language modeling.

Word error rate (WER) measures the actual performance of the speech recognizer in terms of correctly and incorrectly recognized words. To calculate the WER, the speech recognizer's output is aligned word-by-word with the manual transcription using the edit (Levenshtein) distance which considers three types of misalignment errors. The types of errors considered are: insertion errors (I) where an additional word has been recognized which does not have a corresponding word in the reference, deletion errors (D) where a word in the reference does not have a corresponding word in the transcript and substitution errors (S) where one word in the reference was mis-recognized as another word in the transcript. Depending on specific requirements, the different types of errors may be associated with different costs and the least-cost aligned path may be different due to the weighting. Finally, WER is defined by:

$$WER = \frac{I + D + S}{W} \tag{2.5}$$

where W is the total number of words in the reference text.

2.2. Problem Setting

Being the final stage of ASR, discriminative language modeling can be viewed as a complementary method to baseline generative language modeling. A DLM is trained on the outputs of ASR and the reference transcriptions. Figure 2.1 shows an example 10-best list of an English ASR system for the utterance "This is a test sentence".

The hypotheses in Figure 2.1 denote the paths in the decoding lattice with the highest recognition scores, which is a combination of acoustic and language model

Reference: This is a test sentence

Hypothesis	WE	Rec.Score
This is a guest sentence	1	-1.801
This is the best sentence	2	-2.207
This is a best sentence	1	-2.503
This is a test sentence	0	-3.042
This is a test sense	1	-3.234
This is a guest sense	2	-3.367
That is the a guest sense	4	-4.623
This is the guest sense	3	-5.326
This is the guest sentence	2	-6.231
That is the a guest sentence	3	-7.257

Figure 2.1. An example 10-best list.

scores for that particular transcription, represented in negative log-likelihood. The second column denotes the number of word errors (WE) for each hypothesis, which is calculated by aligning the hypothesis with the reference using the edit distance as mentioned in Section 2.1.

In an N-best list, the hypothesis with the highest recognition score is called the 1-best. As it can be seen, the 1-best of the list in Figure 2.1 has one word error (i.e., "guest" instead of "test"). However, the correct transcription is in fact present in the N-best list, although it has a lower recognition score. The aim of discriminative language modeling is to make the necessary arrangements in the N-best list such that this transcription is returned by the ASR system as the recognition output. Please note that the correct transcription may not occur in the N-best list. In that case, the hypothesis with the least number of errors shall be returned.

2.3. Language Units

Despite the fact that the success of an ASR system is measured in Word Error Rate (WER), the language itself can be modeled at different levels, which corresponds to choosing a language unit to represent the hypotheses in the language being modeled.

In this thesis, we deal with Turkish ASR. Turkish is an agglutinative language of the Altaic family. It has a highly inflectional morphology which causes high out-of-vocabulary rates in ASR. In order to compensate for this, in our experiments we build our models on sub-word language units, instead of words. Going from shorter to longer, these language units are characters, syllables, morphemes and morphs. Morphs are statistically derived pieces of a word which are similar to morphemes. We use the Morfessor algorithm [16] which analyses a training text to determine the morphs statistically.

Figure 2.2 shows an example Turkish sentence parsed into different language units used in this study.

Language Unit	Parse
Word	Iyi akşamlar sayın seyirciler
Morph	Iyi ak +şam +lar say +ı n seyirci +ler
Morpheme	Iyi akşam +lar say +ın seyir +ci +ler
Syllable	I +yi ak +şam +lar sa +yın se +yir +ci +ler
Character	I y i a k ş a m l a r s a y ı n s e y i r c i l e r

Figure 2.2. Parsing of an example sentence into different language units.

Previous experiments have shown that morphs provide higher accuracies and thus they are more suitable and effective for the agglutinative nature of Turkish [2, 17, 18].

2.4. Data Types and Datasets

The data used in this study can be classified into three different types: The first type is acoustic data which has been manually transcribed. We will call this type the "matched" data and denote it with the symbol $\mathcal{M}$. The data is passed through the recognizer to form the real ASR output hypotheses, and the transcriptions are used as the reference text. The second type, $\mathcal{A}$, stands for acoustic data that is not manually transcribed. Finally the third type, $\mathcal{T}$, stands for source text data that is not accompanied by any recording. Note that $\mathcal{A}$ and $\mathcal{T}$ are not matched and may be coming from another corpus than $\mathcal{M}$.

Table 2.1. Bogazici University Turkish Broadcast News Database.

Dataset	Duration	Number of utt./sent.	Number of words
Training m	188h	105K	1.4M
Held-out h	3.1h	1.9K	23K
Evaluation e	3.3h	1.8K	23K

Throughout this thesis, we use discriminative language modeling techniques to correct ASR errors in a Turkish broadcast news (BN) transcription task. We utilize three different datasets throughout this thesis, which will be explained in the following paragraphs.

The main data corpus we use is the Bogazici University Turkish Broadcast News Database, which consists of approximately 194 hours of speech recorded from TV and radio news channels, all of which have been manually transcribed[6]. Part of this dataset has been published by LDC in 2011 [19]. The database is divided into three disjoint sets: a training set for building the models, a held-out set for parameter validation and a test set for algorithmic performance evaluation. We will denote the training set as m, the held-out set as h and the test set as e in this thesis. Table 2.1 shows the size of these sets in terms of their duration, the number of utterances/sentences and the number of words.

Note that not all experiments on this thesis contain the same amount of training data. We use different partitions of the m dataset, by dividing it into two, three, or more pieces, with equal number of utterances. We denote the way of partitioning by a fraction subscript below the dataset symbol. For instance, $m_{1/2}$ denotes the first half of m, whereas $m_{2/3}$ contains the second piece of one-thirds of m. In this regard, $m_{1/2} + m_{2/2} = m$, or $m_{1/3} + m_{2/3} + m_{3/3} = m$.

[6]This dataset was collected and transcribed in the scope of the projects TÜBİTAK 105E102 and BU-BAP 05HA202.

Apart from the main corpus, we also utilize two more corpora in this study. The second corpus is an extension of the Bogazici University Turkish Broadcast News Database, and contains another 310 hours of speech collected at a later time period, this time untranscribed. We will use the symbol a to denote this dataset. Since a was collected at a later stage than h and e, some words in h and e that did not occur in m may already exist in a, which might be deceiving in terms of the WER computation. In order to resemble a real life usage scenario and to make consistent performance comparisons of the two datasets, we introduce a new held-out (h_2) and a new test (e_2) set for experiments in which a is included in DLM training.

Our third and final dataset is a source text corpus of 500K sentences (4.7M words) randomly selected from a larger set (184M words) collected from major Turkish news portals. We use the letter t to refer to this dataset. Table 2.2 contains information on the datasets a, h_2, e_2 and t.

Table 2.2. Other datasets involved in the study.

Dataset	Duration	Number of utt./sent.	Number of words
Training a	310h	288K	2.2M
Held-out h_2	7.2h	5.8K	56K
Evaluation e_2	13.5h	9.9K	95K
Training t	n/a	500K	4.7M

Note that we denote the data types by uppercase letters and the actual datasets by lowercase letters. In our experiments, we use whole or parts of the set m as the matched data $\mathcal{M}$. As the acoustic type $\mathcal{A}$ we have two sources: the acoustic component of the set m, to be denoted by $m^{\mathcal{A}}$, and the set a. Similarly as $\mathcal{T}$ we use either m's textual component (transcriptions), $m^{\mathcal{T}}$, or the set t.

2.5. Baseline System and Preparation of the DLM Data

In this study we use the same baseline ASR system as in [17]. The acoustic model component of the ASR system is composed of a decision-tree state clustered cross-word triphone AM, trained using m. The language model component is a 4-gram morph generative LM with interpolated modified Kneser-Ney smoothing and entropy-based pruning, trained using $m^{\mathcal{T}}$ and the general text corpus of 184M words mentioned earlier. The models built from these two sources were linearly interpolated to reduce the effect of out-of-domain data. The first-pass lattice outputs were rescored with unpruned LMs to compensate for the effect of pruning. The AM is prepared using the AT&T tools [20, 21] and the generative LM is prepared using the SRILM [22] toolkit.

The DLM training data is composed of real ASR N-best lists obtained from the sets $m^{\mathcal{A}}$ or a, and artificial N-best lists obtained from the sets $m^{\mathcal{T}}$ or t. The N-best lists of a were created by decoding the acoustic data by the baseline ASR system and selecting the most probable N paths from the output lattice. Preparation of the real N-best lists of set m deserves special attention, since m was already used in training of the baseline system. The following standard training procedure was applied to alleviate overtraining of the LM [23]: $m^{\mathcal{A}}$ was divided into K-folds, and utterances in each fold were decoded with the baseline AM trained on all utterances and an LM trained on the remaining $K-1$ folds. AM training was not controlled in the same way since baseline AM training is more expensive and less prone to overtraining than n-gram LM training. The artificial N-best lists of set m^{T} or t were obtained using the confusion modeling procedure to be explained in Chapter 4. All N-best lists we use in our experiments include 50 hypotheses.

The Morfessor algorithm [16] is employed to determine the morphs. The N-best lists of m have around 46K unique morphs, and the N-best lists of a have around 55K unique morphs. Together, they make up about 58K unique morphs. The feature vector of the linear model, $\mathbf{\Phi}$, consists of morph unigram counts and therefore is high dimensional but sparse.

Two-class SVM tests are evaluated using the LIBLINEAR toolbox [24]. For the ranking SVM experiments, we use the SVM^{rank} tool [25] which provides a fast implementation of the algorithm.

The WFST-based confusion modeling system is implemented by the OpenFST library [26], and the MBR computations are done using the SRILM toolkit [22]. Significance analysis is done using the NIST Matched Pair Sentence Segment Word Error (MAPSSWE) test [27].

2.6. Training Scenarios

The main focus of this thesis is to investigate DLM training methods with respect to the availability of different data types. Table 2.3 shows four basic discriminative language modeling scenarios experimented in this study and the types of data involved.

Table 2.3. Training scenarios.

Scenario	$\mathcal{M}$	$\mathcal{A}$	$\mathcal{T}$
Supervised	DLM		
Semi-Supervised	CM		DLM
Unsupervised DLM		DLM	
Unsupervised CM		CM	DLM

In the first scenario in Table 2.3, the matched data $\mathcal{M}$ is used to train the DLM directly. This is the traditional way to train a DLM and is called the supervised setup, as an analogy to the term in pattern recognition which is used for cases where the class of the training examples are known. We explain the supervised training scenario in Chapter 3. In the second scenario, $\mathcal{M}$ is instead used to build a CM, which is then applied on $\mathcal{T}$ to generate artificial hypotheses. These hypotheses, together with $\mathcal{T}$ as their reference, are finally fed into DLM training. In the literature, this technique is known as the semi-supervised setup, as a small amount of training examples are used to generate ("label") the others [3, 28]. We investigate semi-supervised training in Chapter 4.

The third and fourth scenarios are analogous to the first and second, respectively, with a slight difference: Here the acoustic data $\mathcal{A}$ is not accompanied by matched reference text, making the training examples to be unlabeled and forcing the training procedure to be unsupervised. The solution we propose for these scenarios is to replace the reference with a target output sequence and continue the process in a similar way to the supervised and semi-supervised setups. Once the target ranks of the N-best hypotheses are determined, one can use them to train the DLM directly or via confusion modeling. We name the third scenario as the unsupervised DLM setup, and the fourth one as the unsupervised CM setup. We explore both setups in Chapter 5.

2.7. Previous Work

Discriminative estimation of language models has been studied in the ASR literature for over ten years. The techniques developed on the subject have been applied to automatic speech recognition [29], utterance classification [30], parsing [31], machine translation [32], call classification [30, 33], and automatic transcription and retrieval of broadcast news [17].

The linear model by [34], which we also adopt in this study, is one of the most studied discriminative modeling frameworks. Being a feature based approach, the linear model can integrate many different sources (syntactic, semantic, morphological, n-gram information) into a single mathematical structure [35, 36]). Other modeling frameworks include global conditional log-linear models (GCLM) [23] and exponential models [37].

The perceptron algorithm in a structured prediction setting is a popular method to estimate the parameters of the linear model [23, 32, 38]. Using the structured perceptron for correcting the errors of Turkish ASR, [36] achieve improvements of up to 0.8% over the baseline WER.

The perceptron algorithm has also been adapted for reranking purposes. In [39], a perceptron ranking (PRanking) algorithm that divides the space into regions bounded

by threshold levels is proposed. Each example is assigned a rank, and two neighboring ranks are set apart by a biased boundary. Note that this algorithm cannot be applied to discriminative language modeling, as it needs global ranks, i.e., the ranks of hypotheses in different utterances need to be comparable with each other. Another study includes a review of reranking and introduces two perceptron-based ranking algorithms to decrease data complexity and training time [31]. Such algorithms have been applied to parse reranking and machine translation tasks in [40] and [41]. To the best of our knowledge, the ranking perceptron variant algorithm we propose in this thesis has not been applied to ASR output reranking before.

[42] also provide an improvement over the structured perceptron by adding a word error rate sensitive distance measure into the update rule. This new measure is adapted to reranking in [3, 6].

Support vector machines (SVM) have also been used as an alternative discrimination method. Using an English word n-gram setup, [43] applies the SVM classifier over an equal number of positive and (artificially generated) negative examples. It is reported that the accuracy of the system increases if negative sentences are disrupted more, and if the total number of examples is increased. Several modified versions of the SVM algorithm are also used for other language modeling tasks such as lexical disambiguation [44], parsing and machine translation [45].

The SVM is adapted to ranking in [46]. In [47] the ranking SVM algorithm is compared with three other discriminative algorithms, namely perceptron, boosting, and minimum sample risk (an algorithm which applies line search to minimize the total error over a useful subset of features), in terms of accuracy and training time. Because of the long training time, the authors had to use 20-best lists instead of 100-best for their ranking SVM implementation. Despite this limitation, they show that this algorithm generalizes better, based on its performance on an unseen test set.

Weighted GCLM [48], Round-Robin Dual Discrimination (R2D2) [49], and margin-infused relaxed algorithm (MIRA) [50] are among the other methods that are used to

train a DLM. The MIRA has been applied to statistical machine translation in [51] and [52] and to parsing in [53].

Semi-supervised DLM training has recently been popular in the literature, and there are a number of approaches to construct an appropriate confusion model (CM) for this task. One of the approaches uses a weighted finite-state transducer (WFST) to represent the CM. In Kurata et al. [54–56], phoneme similarities estimated from an acoustic model are specified in the CM by a process called Pseudo-ASR. Jyothi et al. [38] follows a similar method by modeling the phonetic confusions with a WFST. Another approach makes use of a machine translation (MT) system to learn these confusions. For instance, Tan et al. [57] use a channel modeling approach under a phrase-based MT system to learn error models between parallel corpora of ASR output and reference text represented by phonemes, and show that using contextual information besides basic acoustic distances improves the system accuracy. Similarly, Li et al. [58] use translation alternatives of source phrase sequences to simulate confusions that could be made by an MT system. In a third approach, Xu et al. [37] make use of a separate text corpus and find the competing words (cohorts) which occur in the ASR outputs of untranscribed speech to train their CM. A comparison of these three approaches is given in Sagae et al. [28]. Although the authors use the same dataset (English n-gram features) for all experiments, the language unit they utilize for training different CMs is different (a phone-based WFST model and phrase-based MT and cohort models).

Although the confusion model generates a very large number of alternating hypotheses that are acoustically similar, not all of them are linguistically plausible. In order to obtain a sufficiently errorful subset which is at the same time sufficiently meaningful, these hypotheses are reweighted using a (generative) language model and sampled according to a scheme. Similar hypothesis selection schemes by sampling from an N-best list are investigated in [59] where it is argued that the selection of hypotheses based on WER should be preferred over the recognition score. Using the perceptron algorithm, the authors achieve an accurate and compact corrective model. In [41], a perceptron-like algorithm has been proposed that splits the training data into top r-ranked and bottom k-ranked translations. The same study also includes an ordinal

regression algorithm for reranking in an MT task and improvements in BLEU of up to 0.3% are reported.

Ways to build a DLM under the unsupervised setting have been investigated in the literature for the last couple of years. In Xu et al. [37] the confused words in ASR outputs are used in training, which is done by maximum likelihood optimization using an exponential model. A more recent study learns phrasal cohorts instead of word cohorts by pairwise alignments of the hypotheses, uses them to generate artificial N-best lists, and reports a decrease in word error rate (WER) of up to 40% of that of the supervised case with the perceptron algorithm [60]. In Jyothi et al. [61], a large amount of unlabeled data is reprocessed using a weak acoustic model, and small but statistically significant improvements in WER are reported. Finally in Kuo et al. [62], the authors employ the Minimum Bayes Risk (MBR) criterion to choose the reference hypothesis for training the DLM via the perceptron.

3. SUPERVISED DISCRIMINATIVE LANGUAGE MODELING

In this chapter we explore the fundamentals of discriminative language modeling by considering the simplest scenario, the *supervised* DLM training. In this scenario, we assume that the acoustic speech data and the manual transcriptions (references) are both available for training the DLM. The speech utterances are passed through the ASR system, which generates the hypotheses that are used as the training examples. By aligning each hypothesis to its corresponding reference, one can obtain the number of word errors (WE) for that hypothesis, which in turn defines its target class or rank, hence making the training process supervised.

Regardless of the training scenario, two fundamental choices to be made when training a DLM are the type of the model and the algorithm to train it. In this study we choose the linear model framework and utilize three popular algorithms, namely the perceptron, the margin-infused relaxed algorithm and the support vector machine to estimate the parameters of the linear model. Defined by the optimization criterion, two approaches in training are classification and reranking. We explore adaptations of these three algorithms for both approaches and compare their performances. We also propose several hypothesis sampling approaches to compensate for the high computational complexity of the training algorithms.

In the following sections we first give the theoretical information on the DLM training framework, and then present the experimental results for the supervised scenario. We finally give a detailed statistical analysis of results, followed by a discussion.

3.1. Linear Model

The type of the model sets the mathematical grounds for representing the discriminative language modeling problem. In this study we adopt a linear model similar

to that in [63]. The following is a list of the elements of the linear model, together with the symbols we use to denote them throughout this thesis:

- x is the spoken utterance that is input to the recognizer.
- y is the written counterpart of x. In a supervised scenario, y stands for the manual transcription of x, and is called the *reference*. In the unsupervised DLM scenario the manual transcription is not present, so y is a computed hypothesis that replaces the reference and is called the *target output*. For the scenarios where x does not take part in DLM training (i.e., semi-supervised and unsupervised CM), y stands for data from an unmatched written corpus and is called the *source text*.
- $\mathbf{GEN}(\cdot)$ is the function that is assumed to generate the hypotheses for training the DLM. Depending on the availability, this function either takes x as the input and represents the ASR system itself, or takes y as the input and represents the CM.
- $\tilde{y} \in \tilde{\mathcal{Y}}$ are the N-best hypotheses that are generated by $\mathbf{GEN}(\cdot)$, and that serve as training examples for discriminative modeling. Depending on the scenario, these may be real hypotheses output by the ASR system, or artificial hypotheses generated by the CM. The method we propose to generate artificial hypotheses will be explained in Chapter 4.
- $\mathbf{\Phi}(\cdot)$ is a mapping that represents a hypothesis in a d-dimensional modeling space. In our implementation, the vector $\mathbf{\Phi}(\tilde{y})$ contains the number of occurrences (frequencies) of each morph n-gram in $\tilde{y}$. Some experiments in this thesis also use the recognition score of the hypothesis as the first element (ϕ_0) of this vector, thus making $\mathbf{\Phi}$ also depend on x. To avoid any confusion, we will always use the symbol $\mathbf{\Phi}(\tilde{y})$ to denote the hypothesis vectors and will mention the usage case when we explain the experimental setup.
- $\mathbf{w}$ is the model vector that is estimated by discriminative training. Each element of $\mathbf{w}$ is the weight associated with the corresponding feature of $\mathbf{\Phi}$. The inner product $\langle \mathbf{w}, \mathbf{\Phi}(\tilde{y}) \rangle$ is the value that contributes to the modified score of $\tilde{y}$.

3.2. Training Algorithms

Training a DLM means estimating the parameters of the linear model, $\mathbf{w}$. With respect to the objective of the optimization criterion they use, discriminative model training algorithms can be grouped into two approaches: classification and reranking.

In the classification approach, discriminative modeling is defined as the separation of the most accurate hypothesis in an N-best list from the rest. Due to the fact that there is actually only a single class to be predicted, this approach is typically named *structured prediction*. Most of the studies on DLM training in the literature are based on the structured prediction approach. As a matter of fact, the hypotheses in an N-best list actually have different degrees of goodness (which can be measured by their WE), and their differences can also provide valuable information towards achieving a more robust discriminative model. Hence, it seems more natural to regard DLM training as the reranking of the N-best list where the WE provides the target ranks of the hypotheses. One of the objectives of this thesis is to investigate the reranking approach for DLM training, and to compare and contrast it with the structured prediction approach.

The reranking problem is similar to ordinal regression [64] in that all examples $\tilde{y}$ of the training set are given a rank $r_{\tilde{y}}$ instead of a class label. However, unlike ordinal regression, in reranking the ranks are defined only between the examples of the same utterance, i.e., the N-best list $\tilde{\mathcal{Y}}$. In our study, we define the rank of a hypothesis as:

$$r_{\tilde{y}} = 1 + \mathrm{WE}_{\tilde{y}} \tag{3.1}$$

In a reranking scenario, we would like to find $\mathbf{w}$ such that for any two hypotheses a and b from the same N-best list, if a has fewer word errors than b, it has a higher rank (is closer to the top of the list) than b. It must be noted that ranking ordering is the opposite of numeric ordering. For instance, if $r_a = 1$ and $r_b = 2$, then $r_a \succ r_b$.

In this study we utilize classification and reranking variants of three algorithms, namely the perceptron, the margin-infused relaxed algorithm and the support vector machine, which we explain in the following subsections.

3.2.1. Structured Perceptron

The perceptron algorithm used in DLM training is a multi-class perceptron variant used for structured prediction [23], which we will refer to as the structured perceptron. The objective of the structured perceptron is to pick the hypothesis in the N-best list ($\tilde{y} \in \tilde{\mathcal{Y}}$) which has the least WE with respect to y (either the reference, source text or target output).

input set of training examples $\{1 \leq i \leq I\}$,
number of iterations T
$\mathbf{w} = 0$, $\mathbf{w}_{sum} = 0$
for $t = 1 \ldots T$, $i = 1 \ldots I$ **do**
  $z_i = \text{argmax}_{z \in \tilde{\mathcal{Y}}} \langle \mathbf{w}, \mathbf{\Phi}(z) \rangle$
  if $r_{y_i} \neq r_{z_i}$ **then**
    $\mathbf{w} = \mathbf{w} + g(y_i, z_i)(\mathbf{\Phi}(y_i) - \mathbf{\Phi}(z_i))$
  $\mathbf{w}_{sum} = \mathbf{w}_{sum} + \mathbf{w}$
return $\mathbf{w}_{avg} = \mathbf{w}_{sum}/(IT)$

Figure 3.1. The generic structured perceptron algorithm.

The pseudocode of a generic structured perceptron algorithm is given in Figure 3.1. The structured perceptron takes into account only two hypotheses in $\tilde{\mathcal{Y}}$: y_i and z_i. y_i is the hypothesis which has the least WE with respect to y, and is called the *oracle*. Please note that in our implementation, if there is more than one hypothesis with the least WE, the one which has the highest recognition score is used as the oracle. z_i is the hypothesis which yields the highest inner product score under the current model, and is called the *current best*:

$$z_i = \underset{z \in \tilde{\mathcal{Y}}}{\text{argmax}} \langle \mathbf{w}, \mathbf{\Phi}(z) \rangle \tag{3.2}$$

Taking into account the fact that the oracle needs to have the highest inner product score in order to minimize the overall WER, the model weights are updated by favoring

the features which occur in y_i and penalizing the ones which occur in z_i:

$$\mathbf{w} = \mathbf{w} + g(y_i, z_i)(\mathbf{\Phi}(y_i) - \mathbf{\Phi}(z_i)) \tag{3.3}$$

The function $g(y_i, z_i)$ in Equation 3.3 is defined as the margin function and determines the strength of the update, therefore defining the behavior of the algorithm. Selection of the function $g(\cdot)$ is associated with the loss function that is being optimized. In this thesis we implement three different margin functions, which are summarized in Table 3.1.

Table 3.1. Naming of algorithms with respect to margin functions.

Margin Function	Naming	Abbreviation
$g(\cdot) = 1$	Perceptron	Per
$g(\cdot) = r_{z_i} - r_{y_i}$	WER-sensitive perceptron	WPer
$g(\cdot) = \frac{1}{r_{y_i}} - \frac{1}{r_{z_i}}$	Reciprocal perceptron	RPer

Our first margin function is $g(\cdot) = 1$, which corresponds to minimizing the number of misclassifications. This is the standard version used widely in the literature. We will refer to this version of the structured perceptron algorithm as *Per*.

In the second function, $g(\cdot)$ is defined in terms of the differences between the ranks of y_i and z_i. This type of perceptron was first proposed in [42] and is called the WER-sensitive perceptron. Unlike Per, the WER-sensitive perceptron algorithm minimizes a loss function which is defined using edit distances of hypotheses with the reference transcription, thus related to the total WE [65]. In this thesis we will abbreviate the WER-sensitive perceptron as *WPer*.

In the third alternative, the margin function is defined as $g(\cdot) = \frac{1}{r_{y_i}} - \frac{1}{r_{z_i}}$. This definition not only accentuates the update when the rank of the oracle and the current best are far apart, but also when they appear near the top of the ranking. Besides, it

is more conservative than Per and WPer in that it produces a multiplier less than 1. We name this algorithm the reciprocal perceptron and denote as *RPer*.

The structured perceptron makes several epochs (passes) over the training data and in the end, the model weights obtained at each update step are averaged over the number of utterances (I) and epochs (T) to increase model robustness:

$$\mathbf{w}_{avg} = \frac{1}{IT} \sum_{i,t} \mathbf{w}_i^t \tag{3.4}$$

3.2.2. Ranking Perceptron

As explained in Section 3.2, the motivation behind reranking is to make use of the relative differences between the hypotheses of the N-best list in DLM training. The ranking perceptron is based on this motivation and unlike the structured perceptron which tries to replace the top-most element of the N-best list by only considering the current best and oracle, it utilizes each and every hypothesis in the list for training.

input set of training examples $\{1 \leq i \leq I\}$,
number of iterations T, margin multiplier $\tau > 0$,
learning rate $\eta > 0$, decay rate $\gamma > 0$
$\mathbf{w} = 0$, $\mathbf{w}_{sum} = 0$
for $t = 1 \ldots T$ **do**
  for $i = 1 \ldots I$ **do**
    for $(a, b) \in \tilde{\mathcal{Y}}$ **do**
      if $r_a \succ r_b$ & $\langle \mathbf{w}, \mathbf{\Phi}(a) - \mathbf{\Phi}(b) \rangle < \tau g(a, b)$ **then**
        $\mathbf{w} = \mathbf{w} + \eta g(a, b)(\mathbf{\Phi}(a) - \mathbf{\Phi}(b))$
    $\mathbf{w}_{sum} = \mathbf{w}_{sum} + \mathbf{w}$
  $\eta = \eta \cdot \gamma$
return $\mathbf{w}_{avg} = \mathbf{w}_{sum}/(IT)$

Figure 3.2. The generic ranking perceptron algorithm.

Figure 3.2 shows the pseudocode of the generic ranking perceptron algorithm. Here a and b denote any hypothesis pair from the same N-best list. If a has fewer WE than b, then it must be ranked higher than b, which means that a's score (its inner product with the current model vector) must be significantly greater than that of b:

$$r_a \succ r_b \iff \langle \mathbf{w}, \mathbf{\Phi}(a) - \mathbf{\Phi}(b) \rangle \geq \tau g(a, b) \tag{3.5}$$

The score difference threshold is adjusted by a margin multiplier denoted by $\tau g(a, b)$ where τ is a positive constant. We use the same margin functions $g(\cdot)$ as in the structured perceptron. With regard to the selection of the $g(\cdot)$ function, the variants of the ranking perceptron algorithm are abbreviated as *PerRank*, *WPerRank* and *RPerRank*, respectively. Just like in the structured perceptron, the WER-sensitive and reciprocal rules update the model more when the difference between the WE of the two hypotheses are greater. Unlike WPerRank, the margin function of RPerRank also aims to achieve a greater separation between the hypotheses that are closer to the top of the list than those at the bottom. It further ensures that the following margin-rank relation is preserved:

$$r_a \succ r_b \succ r_c \iff \begin{cases} g(a, c) > g(a, b) \\ g(a, c) > g(b, c) \end{cases} \tag{3.6}$$

The model update is done just like the structured perceptron, this time for each (a, b) pair that violates the margin criterion stated above:

$$\mathbf{w} = \mathbf{w} + \eta g(a, b)(\mathbf{\Phi}(a) - \mathbf{\Phi}(b)) \tag{3.7}$$

Note that the model is not updated if $r_a \prec r_b$. Unlike the structured perceptron, this time a learning rate (η) multiplier is included to facilitate the convergence of the iterative optimization procedure. This learning rate is decreased by multiplying with a decay rate of $\gamma < 1$ at the end of each epoch and the weights are finally averaged as done in Equation 3.4.

3.2.3. Margin Infused Relaxed Algorithm (MIRA)

The generic MIRA [50] trains a prototype for each class such that the inner product of an instance with the *prototype* belonging to its class, $\langle \mathbf{w}_{c_n}, \mathbf{\Phi}(y_n) \rangle$, is higher than the inner product with any other class prototype, $\langle \mathbf{w}_{\bar{c_n}}, \mathbf{\Phi}(y_n) \rangle$. Here c_n is the class of $\mathbf{\Phi}(y_n), \mathbf{w}_{c_n}$ is its class prototype, and $\mathbf{w}_{\bar{c}_n}$ are the prototypes of other classes. The margin is defined as the minimum difference between the inner products and the aim is to train a classifier with a large margin.

For a two-class problem with $c_n \in \{\pm 1\}$, the binary MIRA iteratively updates a single prototype (model vector) $\mathbf{w}$, just like Per.

$$\mathbf{w} = \mathbf{w} + \tau_n c_n \mathbf{\Phi}(y_n) \tag{3.8}$$

However, here the learning rates τ_n are hypothesis-specific, and are found by solving the following optimization problem:

$$\begin{aligned} \min_{\tau_n} \parallel \mathbf{\Phi}(y_n) \parallel^2 \tau_n^2 + 2 c_n \tau_n \langle \mathbf{w}, \mathbf{\Phi}(y_n) \rangle \\ \text{s.t. } 0 \leq \tau_n \leq 1 \end{aligned} \tag{3.9}$$

which gives

$$\tau_n = G\left(-\frac{c_n \langle \mathbf{w}, \mathbf{\Phi}(y_n) \rangle}{\parallel \mathbf{\Phi}(y_n) \parallel^2} \right) \tag{3.10}$$

The function $G(\cdot)$ determines how much to update the prototype if it misclassifies the instance.

$$G(u) = \begin{cases} 0 & u < 0 \\ u & 0 \leq u \leq 1 \\ 1 & 1 < u \end{cases} \tag{3.11}$$

Note that binary MIRA cannot be directly applied to our problem since we do not have the true labels of the instances; we predict the best hypothesis among many (N-best list) candidates. We propose single and multiple update versions of MIRA for structured prediction, similar to those proposed in [51] and [52].

The single update version goes over the N-best lists $1 \leq i \leq I$ and updates only when the current best z_i is not the oracle y_i:

$$\mathbf{w} = \mathbf{w} + \tau_i^s(\mathbf{\Phi}(y_i) - \mathbf{\Phi}(z_i)) \tag{3.12}$$

$$\tau_i^s = G^s\left(-\frac{\langle \mathbf{w}, \mathbf{\Phi}(y_i) - \mathbf{\Phi}(z_i)\rangle}{\parallel \mathbf{\Phi}(y_i) - \mathbf{\Phi}(z_i) \parallel^2}\right) \tag{3.13}$$

$$G^s(u) = \begin{cases} 0 & u < 0 \\ u & \text{otherwise} \end{cases} \tag{3.14}$$

The multiple update version scans over pairs of the oracle y_i and all other hypotheses $y_k \in \tilde{\mathcal{Y}}$, and updates as:

$$\mathbf{w} = \mathbf{w} + \tau_k^m(\mathbf{\Phi}(y_i) - \mathbf{\Phi}(y_k)) \tag{3.15}$$

$$\tau_k^m = G^m\left(-\frac{\langle \mathbf{w}, \mathbf{\Phi}(y_i) - \mathbf{\Phi}(y_k)\rangle}{\parallel \mathbf{\Phi}(y_i) - \mathbf{\Phi}(y_k) \parallel^2}\right) \tag{3.16}$$

$$G^m(u) = \begin{cases} 0 & u < 0 \\ u & u \geq 0 \text{ and } y_k = z_i \\ u/(N-1) & u \geq 0 \text{ and } y_k \neq z_i \end{cases} \tag{3.17}$$

3.2.4. Ranking MIRA

We apply a modified ranking version of MIRA which updates the prototype if any pair of hypotheses with $r_a \succ r_b$ do not satisfy the margin requirement of $g(a, b)$.

$$\mathbf{w} = \mathbf{w} + \tau_{ab}(\mathbf{\Phi}(a) - \mathbf{\Phi}(b)) \tag{3.18}$$

$$\tau_{ab} = G^r \left(\frac{g(a,b) - \langle \mathbf{w}, \mathbf{\Phi}(a) - \mathbf{\Phi}(b) \rangle}{\| \mathbf{\Phi}(a) - \mathbf{\Phi}(b) \|^2} \right) \tag{3.19}$$

$$G^r(u) = \begin{cases} 0 & u < 0 \\ u & 0 \leq u \leq g(a,b) \\ g(a,b) & g(a,b) < u \end{cases} \tag{3.20}$$

3.2.5. Support Vector Machine (SVM)

The SVM is also a linear classifier and its aim is to find a separating hyperplane that maximizes the margin between the nearest samples of two classes. The constrained optimization problem is defined as:

$$\begin{aligned} &\min_{\mathbf{w}} \frac{1}{2}\langle \mathbf{w}, \mathbf{w} \rangle + C \sum_j \xi_j \\ &\text{s.t. } c_j \langle \mathbf{w}, \mathbf{\Phi}(y_j) \rangle \geq 1 - \xi_j \text{ and } \xi_j \geq 0 \end{aligned} \tag{3.21}$$

where $c_j \in \{\pm 1\}$ are the class labels and ξ_j are the slack variables for violations of the margin constraints for the linearly nonseparable case. Note that here, the index j is not constrained within an N-best list and covers the whole sample set. C is a user-defined trade-off parameter between violations and smoothness. It is possible to assign different C values to the positive and negative classes, especially when the classes are not balanced: $C_+ = \beta C_-$. The major advantage of SVM is that this is a convex

optimization problem that can be solved analytically unlike the perceptron that uses gradient-descent and risks getting stuck in local optima.

The labeling of training examples in an SVM setup is not straightforward. One can divide the hypotheses into positive and negative classes by setting a threshold either on the baseline recognition score, or the WE. In our implementation, we choose all hypotheses having the least WE of their N-best list as the positive examples, and the rest as the negative examples.

3.2.6. Ranking SVM

The ranking SVM algorithm is a modification of the classical SVM setup to handle the reranking problem, defined as

$$\begin{aligned} &\min_{\mathbf{w}} \frac{1}{2}\langle \mathbf{w}, \mathbf{w} \rangle + C \sum_{a,b} \xi_{ab} \\ &\text{s.t. } \langle \mathbf{w}, \mathbf{\Phi}(a) - \mathbf{\Phi}(b) \rangle \geq 1 - \xi_{ab} \\ &\forall (a,b) \in P,\ \xi_{ab} > 0. \end{aligned} \tag{3.22}$$

Here, C is again the trade-off value and P is the set of (a, b) pairs for which $r_a \succ r_b$. The constraint in Equation 3.22 implies that the ranking optimization can also be viewed as an SVM classification problem on pairwise difference vectors, $\mathbf{\Phi}(a) - \mathbf{\Phi}(b)$. In that sense, the algorithm tries to find a large margin linear function which minimizes the number of pairs of training examples that need to be swapped to achieve the desired ranking [46].

The relationship between the ranking perceptron and ranking SVM algorithms is given in the Appendix.

3.3. Testing

In the testing phase, the estimated model vector $\mathbf{w}$ is used to reweight the N-best hypotheses of an ASR output. The final output is the hypothesis which gives the highest inner product score with the estimated model:

$$y^* = \underset{\tilde{y} \in \tilde{\mathcal{Y}}}{\operatorname{argmax}} \Big\{ w_0 \log P(\tilde{y}|x) + \langle \mathbf{w}, \mathbf{\Phi}(\tilde{y}) \rangle \Big\}. \tag{3.23}$$

Here, $\log P(\tilde{y}|x)$ is the recognition score assigned to $\tilde{y}$ by the baseline recognizer for the given utterance x, and w_0 is a scaling factor which is optimized on a held-out set. The overall system performance is computed by considering all y^* and represented in word error rate (WER).

3.4. Hypothesis Sampling

The ranking algorithms use N-best hypotheses in a pairwise manner and complexity increases fast as the sample size and the number of unique ranks in the N-best list are increased. In this thesis we propose three hypothesis sampling schemes to relax some of these constraints and to decrease the computational complexity of the algorithm. All three schemes work on the N-best lists. The hypotheses in an N-best list must be sorted first with respect to their WE in ascending order, and then if WE are equal, with respect to their recognition scores in descending order.

- In *Uniform Sampling* (US), we select $n = \{2, 3, 5, 9\}$ instances in uniform intervals from the ordered N-best list. For instance in US-5 with a 50-best list, the hypotheses with the indices 1, 13, 25, 37 and 50 are selected. The best and the worst hypotheses are always in the shortlist. The rank assigned to the hypotheses is $r = 1 + \text{WE}$.
- *Rank Grouping* (RG) groups the hypotheses having the same unique WE and selects 1 or 2 examples as representatives from each group. In RG-1, this repre-

sentative is the hypothesis having the highest score. In RG-2, we also add the one with the lowest score. Again, the assigned rank is $r = 1 + \text{WE}$.

- In *Rank Clustering* (RC), we generate artificial rank clusters. We try RC-2×3 and RC-2×10, where we select 3 and 10 examples, respectively, from the top and bottom of the ordered list. Positive integers are assigned to these clusters as their ranks: $r = \{1, 2\}$.

A simplified example of these hypothesis sampling schemes on a 9-best list is presented in Table 3.2. The first column denotes the order of the hypotheses, and their WE are shown in the second column. In the columns that follow, the rank values associated with these hypotheses are given.

Table 3.2. An example of sampling schemes.

Order	WE	US-2	US-3	US-5	RG-1	RC-2×3
1	0	1	1	1	1	1
2	1				2	1
3	2			3	3	1
4	2					
5	2		3	3		
6	3				4	
7	3			4		2
8	4				5	2
9	4	5	5	5		2

The aim of US is to decrease directly the number of instances. It should be noted that each US scheme with increasing n also contains the instances from the previous ones. In RG, we would like to keep at least one instance from all available ranks, whereas in RC, we guarantee decreasing both the number of instances and ranks. Note that the sampling strategy here considers only single hypotheses but not hypothesis pairs.

3.5. Experimental Setup

The experiments in this chapter use the Bogazici University Turkish Broadcast News Database, explained in Section 2.4. We use all data in the set m for training, and the sets h and e for parameter validation and testing, respectively.

The N-best lists in all three sets have N=50 hypotheses, represented in morphs. The first element of the feature vector, $\mathbf{\Phi}$, is the recognition score, i.e., the log-probability of x in the lattice obtained from the baseline recognizer: $\phi_0(x, \tilde{y}) = P(\tilde{y}|x)$, This score includes the contribution of baseline acoustic and generative language models. The rest of the feature vector contains the frequencies (number of occurrences) of each morph n-gram in the corresponding hypothesis. We experiment with unigram, bigram and trigram morph models. There exist a total of 45,888 unique morphs, 2,226,825 morph pairs, and 6,969,412 morph triples in m. With such a large number of unique features, we have highly sparse feature vectors.

When we evaluate the performance of algorithms, we have two benchmarks to compare against. The first benchmark is the generative baseline, which is the WER of the 1-best (the hypothesis with the highest score). We need to achieve better WER than the baseline to prove the validity of our discriminative language modeling techniques. The second benchmark is the oracle rate, which is the best WER that can be obtained by selecting the oracle hypothesis in each N-best list. This rate gives us a lower bound as we are limited by the hypotheses in the N-best list.

The generative baseline WER is 22.9% on the held-out set (h) and 22.4% on the test set (e). The oracle rates on these sets are 14.2% and 13.9%, respectively.

3.6. Experimental Results

This section is divided into two parts. In the first part, we show the individual performances of the six training algorithms presented in Section 3.2. The experiments in this part are made using morph unigram features and 50-best list of hypotheses. In

the second part, we modify the number of hypotheses and the number of features to see how they effect discriminative language modeling performance.

3.6.1. Performance of Training Algorithms

In the following subsections we investigate the performances of the classification and reranking approaches of the perceptron, MIRA and SVM. We also explain the issues encountered in the implementation of the algorithms.

3.6.1.1. Structured Perceptron. The first of our supervised discriminative training experiments uses the canonical structured perceptron (Per) algorithm as outlined in Section 3.2.1 with the 50-best morph unigram setup. Earlier studies in the literature update $\mathbf{w}$ when $y_i \neq z_i$ [1,36]. Unlike them, we do model updates only when $r_{y_i} \neq r_{z_i}$. This means that we do not have a preference on the actual morphs that appear in the two hypotheses, as long as they have the same WE. This also ensures that the same set of candidate hypotheses are evaluated by the structured perceptron and its reranking variant.

The weight w_0 associated with ϕ_0 has a different range than the rest of the $\mathbf{w}$. The second implementation issue with the perceptron is the selection of w_0. We need to decide whether to include w_0 in training, and if yes, how to update this attribute. Empirical results have shown that it is better to optimize it over a fixed set of values during perceptron training. The possible values of w_0 to optimize over are chosen to be $w_0 = \{0, 1, 2, 4, 8, 16\}$.

Table 3.3 shows the WER on the held-out set (h) for different values of weight w_0, over at most three epochs over the data. Here, and in the tables that follow, the best result is shown with an asterisk.

For the optimal choice of w_0 and the number of epochs, the error rate on h gives 22.1% implying a reduction of 0.8% over the generative baseline of 22.9%. The same model yields 21.8% on the test set e.

Table 3.3. Per WER (%) on h.

w_0	0.0	1.0	2.0	4.0	8.0	16.0
WER	27.1	22.1*	22.3	22.4	22.5	22.6

3.6.1.2. Ranking Perceptron. There are several factors to optimize while training with the ranking perceptron. The weight updates can be made in an online mode at each data instance as exemplified in Figure 3.2, or in a batch mode, after all data instances (I) have been seen [31]. Furthermore, the weight w_0 can be fixed just like in the perceptron, or updated with the other weights. Finally, optimal values of the algorithm parameters, τ, η, γ must be searched for. In this experiment set we apply the ranking perceptron with the reciprocal update rule (RPerRank) to measure the performance of the algorithm and choose the optimal parameters through grid search over these values: $\tau = \{0, 1, 2, 4, 8, 16, 32, 64\}$, $\eta = \{0.1, 0.5, 1\}$, $\gamma = \{0.5, 0.9, 1\}$.

Table 3.4. RPerRank WER (%) on h.

Updates	w_0 fixed	w_0 free
Online	21.8*	21.8
Batch	21.8	21.8

Table 3.4 shows WERs on h with respect to different update strategies for optimal choices of algorithm parameters and (at most) 20 epochs. We see that the update strategy does not have a significant effect on the system accuracy. However, with two decimal digits' precision, the best result obtained is a WER of 21.75% by using online updates with the w_0 parameter fixed at $w_0 = 14.0, \eta = 1, \gamma = 0.9, \tau = 64$. We will use this update strategy in future experiments.

Comparing this WER with the best result in Table 3.3, we see that the ranking perceptron performs better than the perceptron by 0.3% absolute. The error rate on e for the same experiment is 21.5%, which is also better than that of Per.

In Figure 3.3, we present the change in WER with respect to the number of epochs for the best setup. It can be seen that for this setup 20 epochs are sufficient for the algorithm to converge.

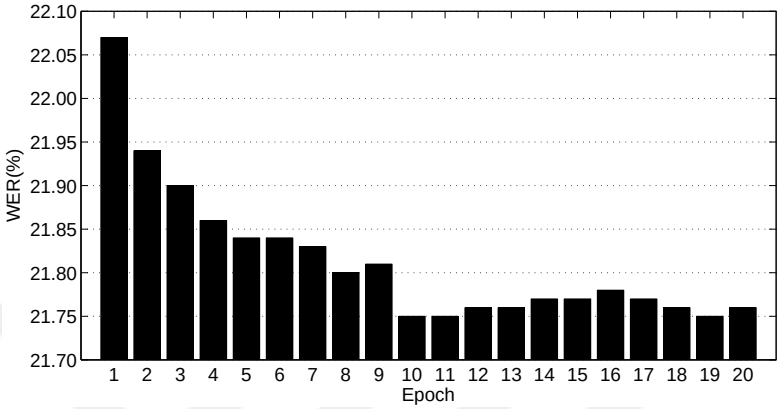

Figure 3.3. RPerRank WER (%) on h with respect to the number of epochs.

3.6.1.3. Margin Infused Relaxed Algorithm (MIRA). We measure the performance of MIRA with both single and multiple update rules. We use ϕ_0 in evaluating the inner product score of the hypothesis with the models but we do not use it in the norm calculations. Table 3.5 presents the best WERs on h for different values of fixed w_0.

Table 3.5. MIRA WER (%) on h.

w_0	0.0	1.0	2.0	4.0	8.0	16.0
Single Upd.	22.9	22.3	22.3	22.3	22.3	22.3
Multiple Upd.	22.9	22.3	22.3	22.3	22.3	22.3

We see that the value of w_0 has no effect on the MIRA performance. We also see no significant difference between the accuracies of single and multiple MIRA and from now on, we report only single update results because it is simpler. The test set WER for this case is 21.8%.

3.6.1.4. Ranking MIRA. The ranking MIRA (MIRArank) performance on h, shown in Table 3.6, is close to that of the perceptron. Just like MIRA, the value of w_0 (unless

zero) does not have any drastic influence. The WER of 22.2% is 0.1% better than MIRA but equally worse than Per. The test set error rate for the best case is 21.7%, which is slightly better than Per.

Table 3.6. MIRArank WER (%) on h.

w_0	0.0	1.0	2.0	4.0	8.0	16.0
WER	28.1	22.2*	22.2	22.2	22.2	22.2

3.6.1.5. Support Vector Machine (SVM). The parameters to optimize in SVM implementation are the trade-off value (C) and minority (positive) class weight (β). The sample labeling method explained in Section 3.2.5 assigns approximately 10% of the examples to the positive class. Table 3.7 shows the held-out set WERs with respect to some combinations of the parameters.

Table 3.7. SVM WER (%) on h.

	$C = 1$	$C = 10$	$C = 100$
$\beta = 0.01$	22.8	22.9	22.9
$\beta = 0.1$	22.8	22.8	22.8
$\beta = 1$	22.5*	22.8	22.5
$\beta = 10$	24.8	24.9	24.9
$\beta = 100$	29.3	29.6	29.3

With the optimal selection of parameters, the lowest WER that could be obtained on h is 22.5%, which is better than the baseline by 0.4% but worse than Per and MIRA. On the test set, the WER is calculated as 22.1%.

3.6.1.6. Ranking SVM. Table 3.8 shows the WERs obtained on h for different trade-off (C) values. The results suggest that ranking SVM leads to better results than the two-class SVM and is able to yield comparable results to those of Per. Furthermore,

on the test set it gives 21.6%, an additional improvement of 0.2%, which indicates that the model learned from SVMrank generalizes better.

Table 3.8. SVMrank WER (%) on h.

C	1	100	1000	10000	20000
WER	22.3	22.3	22.1*	22.1	22.1

3.6.2. Performance with Different Number of Hypotheses and Features

Having seen the individual performances of the training algorithms, let us now modify the training setup and observe how they react to the changes in the number of hypotheses and features.

3.6.2.1. Sampling from the N-Best List. In Table 3.9, we present held-out WERs of our three hypothesis sampling schemes explained in Section 3.4 for the six algorithms, with an optimal parameter selection that yields the highest accuracy on the same set. Results of the 50-best unigram setup are also repeated for comparison.

The first notable result here is the 22.1% WER by Per, which suggests that using only two examples (the best and worst in terms of WE) is as good as using all the examples in the list. This finding corroborates the result presented in [59]. Adding more examples to the training set does not decrease the error rate further with this method.

SVM, being the weakest algorithm in 50-best, prefers a sampled training set but cannot compete with Per. MIRA performs better with more hypotheses. Though it is similar to Per, it is not as accurate, neither in classification nor ranking.

Unlike Per, SVMrank benefits from increasing number of examples in the US and RC cases, but not in RG. The best result obtained here is 21.9% with the US-5 sampling scheme. This value is better than using 5-best lists (22.1%) or choosing 5

Table 3.9. Sampling schemes WER (%) on h.

Sampling	Per	MIRA	SVM	RPerRank	MIRArank	SVMrank
US-2	22.1*	22.7	22.4	22.0	22.6	22.7
US-3	22.1	22.5	22.5	21.9	22.3	22.2
US-5	22.2	22.6	22.6	21.8*	22.1	21.9*
US-9	22.2	22.4	22.6	21.8	22.0*	21.9
RG-1	22.3	22.4	22.3*	22.0	22.1	22.1
RG-2	22.3	22.5	22.5	21.9	22.0	23.0
RC-2×3	22.3	22.6	22.4	21.9	22.3	22.4
RC-2×10	22.2	22.4	22.5	21.8	22.1	22.1
50-best	22.1	22.3*	22.5	21.8	22.2	22.1

examples randomly (22.2%). The RG and RC schemes also provide results that are comparable to the baseline.

The superiority of the ranking perceptron is once more evident in Table 3.9. The algorithm outperforms others in all schemes and responds positively to increasing the sample size and the number of hypotheses. The performance obtained with US-5, US-9 and RC-2x10 are as good as using all of the examples. Considering the fact that the number of iterations in RPerRank is in the order of N^2, decreasing N to its one tenth (from 50-best to US-5) provides an enhancement in the order of 100.

3.6.2.2. Increasing the Number of Features. Up to now we did experiments using a morph unigram (1g) setup, and obtained a lowest WER of 21.8% with RPerRank. In this section we try to see how the algorithms behave if we extend the feature space by adding higher order n-grams. Figure 3.4 shows the held-out WERs of these experiments, all trained on 50-best lists and optimized within their parameter space.

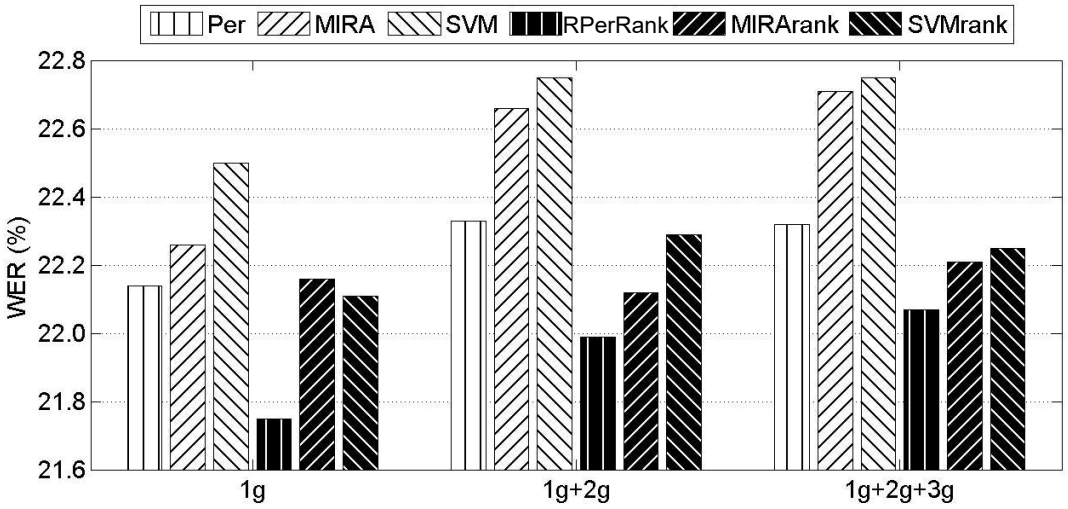

Figure 3.4. Higher-order n-grams WER (%) on h.

As we can see from the figure, for all algorithms, adding bigram (2g) and trigram (3g) features does not decrease but increases the error rates, most possibly due to the lack of sufficient training data which leads to overfitting. This finding is coherent with the observations of [36]. Note that even in this case, the ranking perceptron algorithm shows superior performance to the other five.

3.6.2.3. Dimensionality Reduction. We know that reducing the number of dimensions eases the curse of dimensionality and drastically decreases space and time complexity. It also provides a better generalization by eliminating the effects of noise in redundant features. In our problem, where the feature vector is very high dimensional, applying a dimensionality reduction technique can provide a better characterization for the linear classification of ASR output hypotheses.

In this subsection, we go the other way around and apply feature selection to reduce the dimensionality. We count the number of times each specific feature (morph n-gram) occurs in the training set, and define a threshold, below which that feature will be discarded. We call this approach Count-Based Thresholding (CBT).

Figure 3.5 shows the 50-best morph unigram held-out set results with Per and RPerRank for different values of the threshold, along with the number of dimensions in the reduced space. We see that the performance of Per is not degraded drastically even if we reduce the number of dimensions by one fifth, with the threshold of 500. On the other hand, a slight increase in WER is observed in RPerRank, which can be explained by the number of features utilized. This finding will be explained in Section 3.7.2. Note that CBT results also follow a similar trend for the bigram setup.

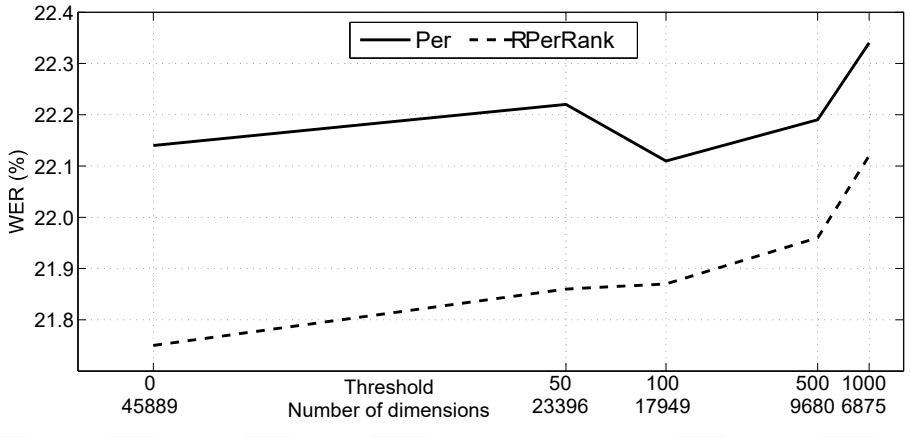

Figure 3.5. Count-based thresholding WER (%) on h.

To decrease the dimensionality, we also tried applying online PCA, but did not obtain any considerable gain due most probably to the restriction to a linear feature extractor [1].

3.7. Analysis of Results

So far we have only compared the models by looking at their accuracies. However the learning process reveals some other important side information about the performance and working behavior of these methods. In this section, we compare the algorithms with respect to their CPU times, model complexities, test set accuracies, and we check for differences that are statistically significant. In the following sections, we use the 50-best unigram setup in the experiments.

3.7.1. Comparison of CPU Times

We denoted earlier that ranking algorithms have higher complexity than classification algorithms, and that they need more computational resources as the number of instances and unique ranks are increased. Training the models by processing all the hypotheses takes <2 mins with Per, <3 mins with MIRA and <5 mins with SVM, whereas it takes <30 mins with RPerRank and MIRArank.

In terms of running time, SVMrank is much more costly than the other models. Even though SVM^{rank} toolkit provides a fast and efficient implementation, training is much slower. In Table 3.10, the total number of training instances and the elapsed CPU times for the SVMrank setup are shown with respect to different hypothesis sampling schemes. This time a fixed trade-off value of $C = 1000$ is used for fair comparison. We see that although SVMrank training takes time in the ranges of hours, by an intelligent sampling technique from the N-best list, it is possible to decrease the number of training instances and thus the CPU times considerably, while keeping the WERs still in a tolerable region.

Table 3.10. SVMrank training CPU times for fixed C=1000.

Sampling	Number of instances	CPU hours	WER(%) on h
All	4,939,368	25:00	22.3
US-2	209,675	0:14	22.8
US-3	313,120	0:51	22.5
US-5	518,234	1:42	22.1
US-9	923,770	3:34	22.2
RG-1	466,277	1:49	22.0
RG-2	867,045	4:26	23.1
RC-2×3	620,160	3:16	22.6
RC-2×10	2,020,450	16:45	22.2

3.7.2. Comparison of Models

We also compare the performances of six models in terms of their complexity and accuracy. Since all of the models are linear, simpler models are the ones which use fewer features. The weights of unused features are zero at the end of training and complexities of the models are compared in terms of the features they use and share. We consider weights not exceeding the threshold 10^{-4} as zero. In Table 3.11(a)-(d), we compare models with respect to the number of zero and nonzero features they use.

Although Per and MIRA use fewer than half of the nearly 46K features, the other models use almost all of them, due to the fact that the latter consider most of the hypotheses in their updates as opposed to only two (the oracle and the current best). It should also be noted that though two models might use the same features, their weights can have different signs.

Feature comparisons of perceptron and MIRA variants for bigram and trigram datasets are given in Table 3.12(a) and (b) (SVM and SVMrank results cannot be obtained here due to their infeasible training times). Similar to the unigram case, ranking algorithms use many more features.

3.7.3. Comparison of Test Set Accuracies

The models are also statistically compared in terms of their accuracy on e. WERs shown in Table 3.13 reveal that the ranking perceptron generalizes better than all other methods, by providing an improvement of 0.9% over the test set baseline of 22.4%.

Table 3.14 presents the significance values (p values) of the WER improvements of the algorithms on e, as measured by the NIST MAPSSWE test [66]. The improvement of RPerRank over Per is significant at $p = 0.003$. The test also shows that there is no statistically significant difference between RPerRank and SVMrank, nor between SVMrank and Per. But SVM has significantly higher WER than both RPerRank ($p < 0.001$) and SVMrank ($p = 0.004$). MIRA does not differ from the other methods except RPerRank ($p = 0.002$). MIRArank differs from SVM and RPerRank.

Table 3.11. Pairwise comparison of models in terms of the number of Zero and NonZero features they use on unigrams.

(a) Perceptron and SVM

		Per		SVMrank	
		NZ	Z	NZ	Z
SVM	NZ	19,946	25,223	44,390	779
	Z	175	545	710	10
RPerRank	NZ	20,119	24,896	44,291	724
	Z	2	872	809	65

(b) Perceptron and MIRA

		MIRA		RPerRank	
		NZ	Z	NZ	Z
MIRArank	NZ	20,540	23,290	43,767	63
	Z	7	2,052	1,248	811
Per	NZ	18,728	1,393		
	Z	1,819	23,949		

(c) SVM and MIRA

		MIRA	
		NZ	Z
SVM	NZ	20,481	24,688
	Z	66	654

(d) SVMrank and MIRArank

		MIRArank	
		NZ	Z
SVMrank	NZ	43,790	1,310
	Z	40	749

Table 3.12. Feature comparison for Perceptron and MIRA.

(a) Bigrams

		RPerRank		MIRA	
		NZ	Z	NZ	Z
Per	NZ	257,458	44	203,719	53,783
	Z	1,924,116	91,096	44,954	1,970,258
MIRArank	NZ	2,079,702	7,039	248,541	1,838,200
	Z	101,872	84,101	132	185,841

(b) Trigrams

		RPerRank		MIRA	
		NZ	Z	NZ	Z
Per	NZ	819,233	520	703,725	116,028
	Z	7,891,409	430,964	127,667	8,194,706
MIRArank	NZ	8,023,671	45,944	830,101	7,239,514
	Z	686,971	385,540	1,291	1,071,220

Table 3.13. WER (%) on *e*.

Per	MIRA	SVM	RPerRank	MIRArank	SVMrank
21.8	21.8	22.1	21.5	21.7	21.6

Based on these findings, the ordering of the training algorithms with respect to their accuracies is:

RPerRank SVMrank MIRArank MIRA Per SVM

Table 3.14. p values of MAPSSWE test.

	MIRArank	RPerRank	Per	SVMrank	SVM
MIRA	0.638	0.002	0.795	0.121	0.156
SVM	0.014	< 0.001	0.085	0.004	
SVMrank	0.168	0.226	0.131		
Per	0.764	0.003			
RPerRank	0.007				

Table 3.15. 10-fold cross-validated WER (%) on e.

Per	MIRA	SVM	RPerRank	SVMrank	MIRArank
21.87 ± 0.06	21.93 ± 0.05	22.19 ± 0.12	21.47 ± 0.05	21.63 ± 0.09	21.81 ± 0.03

3.7.4. Comparison of Statistical Significance

The results presented in the previous section are over a single run. To average over randomness, we also apply 10-fold cross validation by splitting the training dataset into 10-partitions, and results on the test set are given in Table 3.15, this time with two digits of precision. For each model, we use the parameters optimized over the held-out set. We apply 10-fold cross-validation paired t test on the test set WERs and Table 3.16 shows the p-values obtained by pairwise t tests. This time all the differences are significant and we have a clear ordering:

$$\text{RPerRank} < \text{SVMrank} < \text{MIRArank} < \text{Per} < \text{MIRA} < \text{SVM}$$

3.8. Discussion

In this chapter we investigated the supervised discriminative language modeling scenario, where the DLM is trained using matched data, i.e., the speech utterances and their reference transcriptions. We introduced classification and reranking vari-

Table 3.16. p values of 10-fold cross-validation t test on e.

	MIRArank	RPerRank	Per	SVMrank	SVM
MIRA	2.2×10^{-5}	6.0×10^{-9}	4.2×10^{-2}	2.4×10^{-5}	3.2×10^{-4}
SVM	9.3×10^{-6}	1.8×10^{-8}	2.0×10^{-5}	4.0×10^{-7}	
SVMrank	7.8×10^{-4}	2.1×10^{-3}	1.0×10^{-4}		
Per	0.9×10^{-4}	3.0×10^{-8}			
RPerRank	1.9×10^{-9}				

ants of three algorithms to train the DLM and measured their performances under the supervised scenario. We proposed hypothesis sampling schemes to decrease the computational complexity of the algorithms. We compared the performance of the algorithms with respect to changing training conditions.

We see first that reranking leads to lower WER than structured prediction, this is true for all three algorithms. The best WER is obtained by the ranking perceptron with an improvement of 1.1% over the generative baseline on the held-out set.

The disadvantage of ranking algorithms is that DLM training takes more time due to the increased complexity. The hypothesis sampling approaches we proposed in this chapter have been proven to be an efficient way to compensate for this effect.

Another generalized linear classifier which has been proven useful in binary classification tasks is the Support Vector Machine (SVM). However, the use of SVM leads to no significant decrease in error in our experiments and may even worsen performance. SVM-based techniques can also be computationally demanding when the number of training examples is large and the feature dimension is high. On the other hand, the complexity of SVM training can be curbed significantly (from a day to minutes) by intelligent sampling from the N-best list. We note however that these are SVM results using the linear kernel, and with better, application-specific kernels, SVM may provide more interesting results.

MIRA variants do not show significant improvements compared to their perceptron correspondents. A possible reason might be that the normalizing effect of the norms causes smaller updates, diminishing their correcting effects. Another point is that as long as ϕ_0 is used (w_0 is not zero), the other coefficients can adapt themselves accordingly and the same accuracy can be attained.

Ranking algorithms, though more accurate, use more features than classification algorithms. The reason is that they do more updates while the classification algorithms make only one update for each hypothesis set.

Using higher order n-gram data does not improve the performance but it increases the complexity and the training time. It is possible to do feature selection and use a subset of the features, thereby decreasing complexity without losing from accuracy.

4. SEMI-SUPERVISED DISCRIMINATIVE LANGUAGE MODELING

The effectiveness of a DLM increases as it sees more and more training data. However, sometimes the existing matched data are not sufficient to train the DLM using the supervised way. Or the baseline ASR system might have been trained on another domain, so that its outputs turn to be unsuitable for the context upon which the DLM needs to be based. In such cases, in order to increase the amount of training data, it is possible to incorporate an external text corpus (which does not have to be accompanied by any acoustic recording) into DLM training, via a process called *confusion modeling*.

A confusion model (CM) is a model which contains the acoustic confusions (misrecognitions) that could be made by the ASR system. In that sense, the CM represents the inherent variability in the ASR output hypotheses. The CM is trained using the available matched data. Once the CM is built, it can be applied on any word sequence from the text corpus (which we call the *source text*) to generate some other word sequences that resemble the source text acoustically. These sequences look like the hypotheses of a real ASR system, but are artificially generated. Therefore, we call these outputs the *artificial* hypotheses.

The source text and the artificial hypotheses can then be fed into DLM training as training examples, just like the supervised scenario. This whole process is called *semi-supervised* training, with respect to the fact that the matched (supervised) data is used not directly to train a DLM, but to train a CM which will generate the necessary examples for DLM training.

In this chapter we explore the methods for semi-supervised discriminative language modeling. We generate artificial hypotheses through a pipeline which consists of three stages: confusion modeling, language model reweighting and hypothesis sam-

pling. In the following sections we explain these three components in detail. Followed by the experimental setup and results, we present an analysis of the outcomes and a discussion of the methods.

4.1. Generating Artificial Hypotheses via Confusion Modeling

In this section we explore the methods to train a CM and to generate artificial hypotheses using the CM. We propose two approaches for confusion modeling. The first is based on weighted finite-state transducers (WFST) whereas the second is based on statistical machine translation (MT). We present our artificial hypothesis generation pipelines for the WFST and MT based confusion modeling approaches in the following subsections.

Even though it can be used for any language, our confusion modeling approaches are particularly arranged for Turkish. Since Turkish is an agglutinative language with a highly productive morphology, the number of distinct as well as long words is considerably high compared to other languages. Considering that this structure of Turkish can also cause too specific confusion possibilities, we experiment with confusion models at various granularities, namely, word, morph, syllable and character language units.

4.1.1. Weighted Finite-State Transducer (WFST) Based CM

The first approach we use for semi-supervised discriminative language modeling is a WFST-based confusion modeling setup. We apply five different CMs based on the language unit: Phone, character, syllable, morph, and word.

Our phone CM is trained using the acoustic similarities of phones available in the spoken language. To measure the similarity, we calculate the Bhattacharrya distance between the representative Gaussians of each phone in the baseline acoustic model of the ASR, as proposed in [54]. The distances are then converted into estimated probabilities of confusion between phone pairs.

The other four CMs are trained using the real ASR hypotheses and the reference transcriptions. CM training begins by rewriting the ASR hypotheses as a sequence of the chosen language unit. Then, each hypothesis is aligned to its reference using the Levenshtein (edit) distance. Unlike what was done in [38], we do not use any special cost function to get the best alignment. This alignment yields a list of matching language unit pairs that are confused by the recognizer, and the frequency of their match-ups gives the probability of their confusion. In implementation, to reduce computational costs, arcs having probability less than 0.01 are discarded. The CM is finally represented by a single-node WFST having these language unit pairs as input-output values and the confusion probabilities as weights. A simplified example of a word CM for the input "vize" is shown in Figure 4.1.

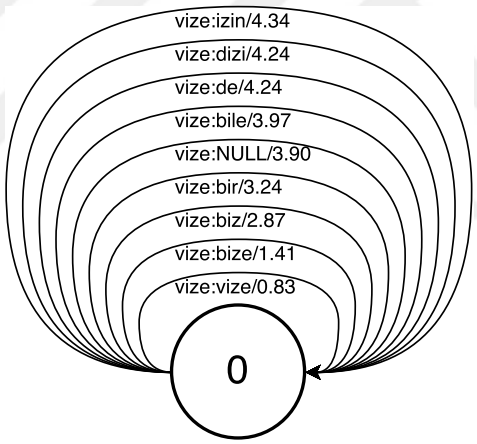

Figure 4.1. An example CM for the word "vize".

Once the CM is learned, generating artificial N-best lists given an input word sequence can be summarized with the following composition procedure:

$$\tilde{\mathcal{Y}} = \text{sample}(\text{N-best}(\text{prune}(W \circ \mathcal{L}_W \circ CM) \circ \mathcal{L}_{\mathcal{M}}{}^{-1} \circ \mathcal{G}_{\mathcal{M}})) \tag{4.1}$$

In Equation 4.1, W represents a word sequence from the source text y. This sequence is first converted to the granularity of the chosen language unit by composing it with the lexicon $\mathcal{L}_W$. The result is further composed with the confusion model (CM)

of the same granularity. This yields a graph which includes all possible confusions for that word sequence, together with their probabilities of occurrence. This graph resembles the lattice output of an ASR system that processed a spoken version of that source text. Depending on the length of the word sequence and the number of possible confusions available in CM, the resulting confusion graph may get so large to prevent efficient processing, and some of the confusions may even be unfeasible or unmeaningful. In order to circumvent this we prune this graph to the most probable 1000-best paths.

Since we would like our DLM to be based on morphs, we would like the artificial hypotheses represented in morphs. To obtain morph outputs, we compose the graph with the inverse morph lexicon $\mathcal{L}_M^{-1}$ and then reweight its arcs. $\mathcal{G}_\mathcal{M}$ scores the sequences in the lattice based on their likelihood seen in a typical sentence in the language being modeled. In the end, 1000 hypotheses having the highest score are extracted to an N-best list, which is then sampled using hypothesis sampling schemes to pick 50 of its examples. These artificial hypotheses, along with their source text, are fed into the DLM training algorithm as training examples, just like the supervised setting.

4.1.2. Machine Translation (MT) Based CM

The second approach we use to generate artificial hypotheses for semi-supervised DLM training is a statistical phrase-based machine translation (MT) framework. An MT system typically tries to match the words or phrases of a source language to those of the target language, and requires a bilingual parallel corpus. In our implementation, this parallel corpus consists of ASR N-best hypotheses and their references. We treat the references as the source language text and the hypotheses as their translations in the target language. This way, the translation alternatives learned by the MT system will yield a CM which is similar in principle to that obtained in the WFST-based approach. However, this time the CM is context dependent, since the MT system is phrase-based.

To establish the MT-based confusion modeling system, we use the Moses toolkit [67]. The steps of the MT-based system are as follows: First, the hypotheses and the references are represented in the chosen language unit, just like the WFST-based setup, and language unit alignment is performed between the parallel text. Unlike traditional MT, we use the Levenshtein algorithm as in the WFST setup rather than a more complicated word alignment package such as Giza++ [68], since we do not need to take into account reordering of language units during translation. Using these alignments, the system computes the maximum likelihood of the lexical translations, extracts phrases and tunes the weights of the feature functions for the phrase translation rules. Finally, the source text is decoded into artificial hypotheses using these translation probabilities. In order to preserve the alignment structure, no phrase reordering model is built and no distortions are allowed during decoding.

4.2. Language Model Reweighting

Regardless of the type of CM, the confusion graph may include many implausible word sequences. Therefore, it is reweighted with a language model to favor the meaningful sequences. In our setup, we apply three different LM reweighting approaches: GEN-LM, ASR-LM and NO-LM.

GEN-LM is estimated from Turkish newswire data collected from the Internet and represented by 5-grams with a vocabulary of 76K morphs. This is the same LM used in the baseline ASR system. ASR-LM is derived from the ASR's real N-best lists, represented by 4-grams out of 40K morphs. Since our ultimate goal is to simulate the ASR output, using the ASR-based LM is more intuitive as it is supposed to give higher scores to those alternatives that resemble the ASR output most. As a third approach, we choose not to apply any language model, which means that only the scores of the CM are used to pick the N-best out of the lattice at the end. This will be denoted by NO-LM.

4.3. Hypothesis Sampling

In Chapter 3 we proposed some hypothesis sampling schemes for the supervised case and showed that sampling from the N-best list decreases CPU times for DLM training drastically, while keeping the WER in a tolerable range. Hypothesis sampling can also be beneficial in semi-supervised training, because it enables us to customize the N-best lists by picking hypotheses from a larger set of broader variety [2].

In our setup we apply different sampling methods to pick 50 hypotheses out of 1000 that were generated in the resulting lattice. The first method is to simply select the highest scoring 50 hypotheses, therefore generating 50-best lists. We refer to this method as Top-50. We also utilize the Uniform Sampling (US-50) and the Rank Clustering (RC-5x10) schemes that were explained earlier in Section 3.4. To remind, US-50 selects instances from the WER-ordered list in uniform intervals, and RC-5x10 forms 5 clusters separated uniformly, each containing 10 hypotheses.

In addition to these methods, in this section we propose a new sampling scheme called ASR-distribution sampling, *ASRdist*. The aim of ASRdist is to achieve the same WE distribution of the real ASR outputs in the sampled N-best list. To do that, we first compute how frequently each number of word errors occurs in the ASR N-best lists and then simulate this distribution by picking samples from the artificially generated N-best in proportional numbers.

4.4. Experimental Setup

In our semi-supervised discriminative language modeling experiments, we use the Bogazici University Turkish Broadcast News Database as in the previous chapter. However, this time we assume that only one half of the training set (m) is transcribed. Therefore, this part constitutes the only matched data we have, which is used to train the CMs. As the source text upon which artificial hypotheses will be generated, we use the reference transcriptions of the remaining half of m. In the experimental results, we will denote real N-best lists of the first half with $m_{1/2}$, and the artificial N-best lists built from the references of the second half with $m^{\mathcal{T}}_{2/2}$.

We represent the hypotheses in 50-best lists with morphs. The feature vector $\mathbf{\Phi}$ consists of morph unigram counts. In the real ASR N-best lists of $m_{1/2}$ there are about 38K unique morphs. This value sets the upper limit on the number of morphs that can be represented in the CM.

Note that the artificial hypotheses do not possess a recognition score as the real ASR hypotheses, because they are generated by the confusion modeling process instead of an actual speech recognizer. Therefore, unlike supervised training, in semi-supervised training we cannot use a baseline score as the first element of the feature vector (ϕ_0). As a consequence, we do not include the weight w_0 in the training phase but only make use of it in testing, where real ASR hypotheses are involved.

The WFST-based confusion system is implemented by the OpenFST library [26] while the MT-based system is implemented by using the Moses SMT tool [67]. The SRILM toolkit [22] is used for building the language models for reweighting.

4.5. Experimental Results

The experiments of this chapter are divided into three parts. In the first part, we evaluate the performance of WFST-based confusion modeling with respect to different components of the semi-supervised DLM training pipeline. In the second part, we measure the performance of training algorithms in a similar setup. Finally in the third part, we compare the performance of WFST- and MT-based approaches.

4.5.1. Evaluation of WFST-based Confusion Modeling

In the following subsections we investigate WFST-based confusion modeling with respect to the type of CM, the hypothesis selection scheme and the type of data involved in semi-supervised training.

4.5.1.1. Effect of the CM Type. In the first of our experiments in this chapter, we measure the performance of different WFST-based confusion modeling techniques for semi-supervised DLM training. We use the first half of the training set ($m_{1/2}$) to train the confusion models, and the reference transcriptions of the second half ($m^{\mathcal{T}}_{2/2}$) as the source text to generate artificial 50-best lists, which are then used to train the DLM. The first element of the feature vector $\mathbf{\Phi}$ is the baseline score when available, and the rest consist of morph unigram counts. The parameters τ, η and γ of the ranking perceptron algorithm, and the weight w_0 are optimized over the held-out set h.

We apply the five different CMs presented in Section 4.1.1 with the three different LM approaches presented in Section 4.2. We utilize the ASRdist-50 sampling scheme mentioned in Section 4.3 to construct 50-best lists out of 1000 hypotheses. DLMs are trained using the WPer algorithm. Table 4.1 shows the WERs on the held-out (h) set for these 15 different configurations.

Table 4.1. WPer WER (%) on h for different CMs and LMs with ASRdist-50.

CMs	GEN-LM	ASR-LM	NO-LM
Phone	22.8	22.7	N/A[7]
Character	22.6	22.7	N/A[7]
Syllable	22.5*	22.4*	22.6
Morph	22.6	22.4*	22.5*
Word	22.6	22.5*	22.7

When compared with the baseline on h (22.9%), the configurations in Table 4.1 with an asterisk are significantly better at $p < 0.005$, based on the NIST MAPSSWE test. Phone and character CMs yield significantly smaller WER improvements over the baseline as compared to syllable, morph and word models. However, the difference between these three models is not significant. For the comparison of LMs, even though there is no significant difference between them, the configurations with ASR-LM have

[7]We did not run these configurations because not using an LM with these types of CMs takes too much computational time.

better WER than the ones with GEN-LM and NO-LM. The best configuration uses the morph CM and the ASR-LM and is significantly better than the baseline at $p < 0.001$.

If DLM training were done with the same half of the dataset ($m_{2/2}$), but this time in a supervised manner using matched data (real ASR N-best lists with their references), the WER obtained on h would be 22.1%. The WER of 22.4% that we obtain with the best semi-supervised training condition is higher than the supervised scenario as might be expected, but is still capable of achieving an improvement of 0.5% over the baseline, which is more than half of the gain obtained by supervised training.

4.5.1.2. Effect of the Hypothesis Selection Scheme. The results presented in the previous subsection were obtained with a fixed hypothesis sampling scheme, ASRdist-50. In this subsection, we compare different sampling schemes mentioned in Section 4.3 using the best configuration obtained in the earlier experiment set.

Table 4.2. Sampling from the 1000-best list with morph CM and ASR-LM.

Sampling	WER (%) on e	KL divergence
NoSampling	22.1	0.38
Top-50	22.1	0.43
US-50	22.0	0.27
ASRdist-50	21.8	0.23

The performance of WPer over the test set e is shown in Table 4.2. While the performances of Top-50 and US-50 sampling schemes are not significantly different than using no sampling at all, the improvement by ASRdist-50 is significant at $p = 0.006$. This supports our assumption that the more artificial N-best lists resemble the real ASR N-best lists in terms of WE distribution, the better WER improvement we get. Figure 4.2 shows how the WE distribution of the ASRdist-50 is more similar to that of the ASR's, compared to Top-50's. We also measure this similarity in terms of Kullback-Leibler (KL) divergence given in Table 4.2 for each sampling strategy, assuming that

the drop in KL divergence would correlate with the drop in WER. We calculate the correlation between the KL divergence and the WER over all experiments and obtain a value of 0.84, which further supports our assumption.

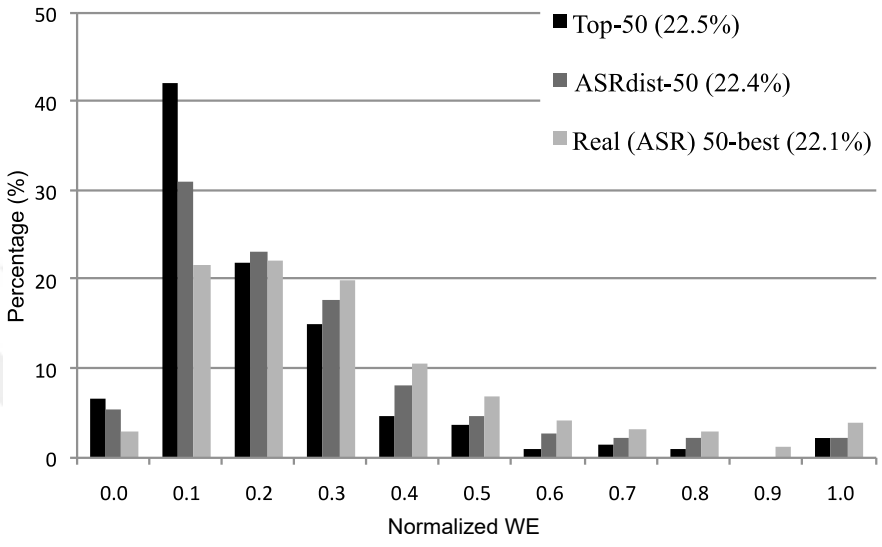

Figure 4.2. Word error distribution on h

Note that the best WER of 21.8% obtained in Table 4.2 is 0.6% better than the test set baseline of 22.4%. On the contrary to the held-out set, the performance on the test set is on par with that of the supervised scenario ($m_{2/2}$), which also yields 21.8%.

4.5.1.3. Combining Real Hypotheses with Artificial Hypotheses. So far, we used only the artificial N-best lists of the set $m^{\mathcal{T}}_{2/2}$ for training the DLM. However, in the semi-supervised setting, we already have some matched data ($m_{1/2}$), which we used to build the CM. This data can actually be reused in DLM training to increase the number of examples, which would potentially lead to lower WERs.

In this subsection, we experiment with combining the artificial hypotheses with the real ASR hypotheses for training the DLM. One other case which we would like to observe is how much the ASR performance would differ, had we used real ASR N-best lists instead of the artificial N-best lists for DLM training. In the first experiment shown in Table 4.3, the DLM is trained in a supervised setting using the whole dataset, i.e.,

$m = m_{1/2} + m_{2/2}$. In the second experiment, we use $m_{1/2}$ for CM training, and add this data to the artificial N-best lists obtained from $m_{2/2}^{\mathcal{T}}$ for DLM training. In these experiments the ASRdist-50 hypothesis sampling scheme is used.

Table 4.3. WPer WER (%) on e for combining real and artificial N-best lists.

$m_{1/2}$	$m_{2/2}$	WER (%)
Real (ASR)	Real (ASR)	21.5
Real (ASR)	Artificial (CM)	21.6

Table 4.3 shows that when we combine the real N-best lists with the artificial N-best lists, we get an additional WER improvement of 0.2%. Furthermore, the achieved performance is almost as good as the performance obtained by using the real N-best lists of the whole dataset.

4.5.2. Performance Comparison of Training Algorithms for WFST-based Confusion Modeling

In the previous section, DLM training was made using the WPer algorithm. In this section we investigate the performance of the canonical (-), WER-sensitive (W) and reciprocal (R) variants of the structured perceptron and ranking perceptron algorithms.

4.5.2.1. Comparison of Training Algorithms Under Real Data. We first present the outcomes of algorithms, when trained on real 50-best lists from $m_{2/2}$. Table 4.4 shows the accuracies in terms of WER on the held-out (h) set. Note that here, since the hypotheses are real ASR outputs, we can include the baseline score to training. For the semi-supervised case these scores are not available, therefore ϕ_0 is necessarily zero. For a fair comparison with future experiments we provide the results for both cases. We see that Per and RPerRank achieve a significant performance improvement when the baseline score is used in training, while WPerRank yields the lowest WER.

Table 4.4. Training: $m_{2/2}$ real, WER(%) on h.

	Per			PerRank		
ϕ_0 in train	-	W	R	-	W	R
No	22.3	22.1	22.2	22.0	21.9	22.1
Yes	22.2	22.1	22.2	21.9	21.8	21.9

4.5.2.2. Comparison of Training Algorithms Under Artificial Data. We now test semi-supervised discriminative language modeling performance for all combinations of the four language units[8], three language models and four sampling methods presented in this chapter. Table 4.5 gives an overall comparison of algorithms in terms of means, standard deviations and minima of their WERs, when trained with artificial N-best lists from $m^{\mathcal{T}}_{2/2}$.

Table 4.5. Training: $m^{\mathcal{T}}_{2/2}$ artificial, WER(%) on h.

	Per			PerRank		
	-	W	R	-	W	R
mean	22.7	22.6	22.6	22.6	22.6	22.6
std	0.11	0.11	0.14	0.09	0.08	0.11
min	22.4	22.4	22.3	22.4	22.4	22.3

The superiority of the ranking algorithms is not noteworthy when it comes to artificial data. All algorithms have a similar mean WER value except Per. However, in terms of the minimum WER that could be obtained, RPer and RPerRank take the lead.

4.5.2.3. Overall Comparison With Respect to Setup Conditions. We also investigate whether the algorithms depict a clear ordering, regardless of their WER, under certain training conditions. We learn this by ranking them with respect to their WERs between 1 and 6, and by fixing one of the training factors and calculating an average rank within

[8]Phone CMs are discarded because of their inferior performance.

all experiments which include that factor. These results are given in Table 4.6 with the best marked with an asterisk.

Table 4.6. Training: $m^{\mathcal{T}}_{2/2}$ artificial, Rank averages on h.

	Per			PerRank		
	-	W	R	-	W	R
Confusion Model						
Character	5.88	4.00	5.13	1.38*	1.38*	2.88
Syllable	5.67	2.42*	3.33	2.75	3.67	2.67
Morph	5.67	2.92	3.25	3.25	3.25	2.00*
Word	3.75	3.58	3.58	3.00	3.42	1.83*
Language Model						
ASR-LM	5.25	2.63	3.88	2.75	2.94	2.38*
GEN-LM	5.19	3.19	3.38	2.63	2.81*	2.81*
NO-LM	5.08	3.83	3.92	2.75	3.58	1.50*
Sampling Method						
Top-50	5.45	3.73	3.09	2.36*	2.73	2.55
US-50	4.64	3.64	4.45	2.64	3.00	1.73*
RC-5x10	5.45	3.18	3.82	2.82	3.27	1.82*
ASRdist-50	5.18	2.09*	3.45	3.00	3.27	3.09

We observe that RPerRank has the highest average rank in most of the cases. It is beaten under four situations: when the syllable CM or the ASRdist-50 sampling scheme is used by WPer, and when the character CM or Top-50 sampling is used by PerRank.

4.5.2.4. Test Set Performance. Finally, we compare the algorithms with respect to the kind of data they use for training. Table 4.7 shows test set WERs of models optimized over parameter and training conditions on the held-out set. Similar to Table 4.3, we observe that in general, although artificial data alone cannot compete with real data, combining real and artificial data yields competitive results[9].

[9]The WERs shown on the WPer column of this table are slightly different than those of the similar training conditions of Table 4.2 and Table 4.3 due to a change in the range of algorithmic parameters.

Table 4.7. WER (%) on e.

$m_{1/2}$	$m_{2/2}$	Per			PerRank		
		-	W	R	-	W	R
None	Artificial	22.1	21.9	22.1	22.1	22.1	22.0
None	Real	21.8	21.8	22.0	21.6	21.6	21.6
Real	Artificial	21.7	21.6	21.5	21.6	21.7	21.8
Real	Real	21.8	21.6	21.8	21.5	21.4	21.5

4.5.3. Comparison of WFST- and MT-based CM

Our third experimental direction in this chapter is the comparison of WFST- and MT-based confusion modeling approaches presented in Sections 4.1.1 and 4.1.2, for semi-supervised discriminative language modeling.

For both approaches, the CM is based on morphs and is trained by aligning the ASR N-best outputs from $m_{1/2}$ with their reference transcriptions using the Levenshtein distance, and language model reweighting is applied using ASR-LM and GEN-LM. 100 sentences from the training corpus are selected as the development set for the MT-based model. Finally, the CMs are applied on the reference transcriptions of $m_{2/2}$ to generate 50-best lists.

We train the discriminative models with the WPer and WPerRank algorithms. The algorithms make 20 and 10 passes over the training data, respectively. The parameter w_0 is optimized on the held-out set.

4.5.3.1. Effectiveness of Artificial Data. In our first set of experiments, we investigate the effectiveness of the artificial data generated by the WFST- and MT-based confusion models. Table 4.8 reports the system performances in terms of WER on the held-out set, with respect to the confusion modeling technique and the language model employed.

Table 4.8. WER (%) on h.

Confusion Model	Language Model	WPer	WPerRank
WFST	ASR-LM	22.8	22.7
	GEN-LM	22.7	22.7
MT	ASR-LM	22.5	22.3
	GEN-LM	22.4	22.3

The interpretation of Table 4.8 is threefold: First of all, regardless of the language model or the algorithm, the WERs of the MT-based CM technique are lower than those of the WFST, which suggests that the artificial examples generated by the MT model are more appropriate for semi-supervised training. The level of improvement is about 0.3% for WPer and even more for WPerRank (the latter being statistically significant at $p < 0.05$). Second, WPerRank provides remarkably lower WER than WPer with MT, on the contrary to WFST where the difference is insignificant. Finally, for both choices of the confusion model or the training algorithm, GEN-LM seems to yield slightly better WER with respect to ASR-LM.

4.5.3.2. Combination of WFST and MT Hypotheses. As a second experiment, we consider whether combining the WFST and MT training examples will provide any further gains. Table 4.9 shows the error rates of individual and combined results with the GEN-LM language model, this time also including the test set performance. The supervised case in which real ASR outputs of $m_{2/2}$ are used for training the DLM is also given for comparison.

We see from Table 4.9 that combining artificial training data of two CMs does not result in a significant decrease in WER on the held-out set h. On the other hand, there is a 0.3% decrease with WPer on the test set e, which suggests that the learned discriminative model is more generalizable to unseen data. Furthermore, the WER is as low as the one achieved using the real ASR hypotheses for training.

Table 4.9. WER(%) on h and e with GEN-LM.

Confusion Model	WPer		WPerRank	
	h	e	h	e
WFST	22.7	22.1	22.7	22.3
MT	22.4	22.3	22.3	21.8
WFST + MT	22.3	22.0	22.2	21.9
Real ASR	22.2	22.0	21.9	21.6

Please note that the combination scheme used in this experiment was to simply concatenate the training examples of both sources. Other complicated methods like fusion strategies based on the model (constructing an intermediate model by averaging the weights of two models), score (choosing the hypothesis which is more confidently selected by any of the two models), or outputs (doubling the N-best lists) have also been tried, and were observed to yield very similar test set WER as the one reported.

4.6. Analysis of Results

In order to understand why the examples generated by the MT-based confusions are a better match for semi-supervised DLM training and why WPerRank provides lower WER than WPer in general, we look at the variability of artificially generated examples and the number of utilized features.

It was noted earlier that there are about 38K unique morphs in the N-best lists of $m_{1/2}$, which is used for training the confusion model. The N-best lists generated by the MT confusions contain more than 28K morphs whereas the ones of the WFST technique contain only about 22K. The number of unique morphs in real ASR outputs of $m_{2/2}$ is also 38K (not all of the features are the same as $m_{1/2}$). Considering occurrence frequencies of these features, the cosine similarity between the N-best sets can be seen in Table 4.10.

Table 4.10. Cosine similarities between real and artificial hypotheses.

	WFST	Real
MT	0.994	0.998
Real	0.996	

More than 20K features are shared by the WFST and MT systems. There are about 7K unique morphs in the MT outputs which do not exist in WFST's, as opposed to only about 1K for vice versa. Based on these evaluations we understand that the MT-based artificial examples have more variability than the WFST-based, and are much closer to what the real ASR outputs would look like for the same reference text.

We now look at the number of utilized features after training with both of the algorithms, which is summarized in Table 4.11. The results suggest that there is a positive correlation between the system performance and the number of features utilized. More features are utilized by WPerRank than by WPer since the former considers each and every hypothesis of the N-best list, rather than only two.

Table 4.11. Number of utilized features (GEN-LM).

CM	WPer	WPerRank
WFST	10,922	14,227
MT	15,320	24,597
WFST + MT	14,914	24,887
Real ASR	20,469	37,373

4.7. Discussion

In this chapter we applied semi-supervised discriminative language modeling techniques to improve ASR performance, and compared two artificial hypothesis generation methods, one based on the WFSTs and the other on MT. Using the artificial data as training examples, we trained our models with the several variants of the structured perceptron and ranking perceptron.

The results suggest that, unlike the supervised case, the ranking perceptron does not show a significant superiority over the structured perceptron with WFST-based confusion modeling. The performance differences between the algorithmic variants become visible only by ordering them for a specific setup condition, and this ordering also changes with respect to different setups.

More significant gains in semi-supervised modeling over the baseline appear when the artificial N-best lists are combined with the real N-best lists to train the model jointly. Using this approach the improvements in WER can reach the level that could be obtained by doubling the matched data.

Our artificial hypothesis generation pipeline also employs pre-trained generative language models, which are shown to aid in obtaining linguistically plausible word sequences.

Anecdotal evidence has shown that the top 50 artificial hypotheses from the confusion graph have a very different WE distribution than that of the ASR N-best lists: it has a much narrower WE range with smaller WEs. Hence we specifically sample 50 instances from the top 1000 in a way to match to the ASR WE distribution in the ASRdist sampling technique. Experiments have shown that by this technique we not only increase the diversity of the hypotheses in the artificial N-best lists, but also obtain better WERs.

We see that the MT-based artificial hypotheses provide a better basis for training the DLM, and that a significant WER reduction can be obtained using both of the algorithms, the ranking version performing slightly better. Fusing the WFST and MT confusions under different strategies yields a small improvement, closer to what the system would give if real ASR data were used.

5. UNSUPERVISED DISCRIMINATIVE LANGUAGE MODELING

In Chapter 3, we investigated the case where the DLM is trained using matched data, which is a combination of speech utterances and their reference transcriptions. In Chapter 4, we investigated the case where the available matched data is not sufficient to train a DLM, therefore used to train a CM instead. In this chapter we explore the case where we cannot find any matched data at all. In other words, the acoustic data (speech utterances) and their corresponding N-best lists are available, but they are not transcribed, so the references do not exist. This leads us to train the DLMs without supervision, i.e., without knowing the ground truths. Therefore, this way of training is called the *unsupervised* training.

Our approach in unsupervised discriminative language modeling is to determine a word sequence which can serve as the missing reference text, and apply this information to (i) determine the ranks of the ASR outputs in order to train the discriminative model directly as in the supervised case, or (ii) build a confusion model in order to generate artificial training examples as in the semi-supervised case.

5.1. Choosing the Target Output in the Absence of the Reference

The reference of an utterance is an essential element in discriminative language modeling as it provides the supervision for training. For training the DLM, it is needed to determine the accuracy of the hypotheses, which in turn defines their target ranks in the N-best list. For building the CM, the reference acts as the ground truth from which the probabilities of confusions are derived.

When the reference is not present, one possible method to continue DLM training is to find another word sequence that replaces it. Such a word sequence is called the target output. In this section, we explore three ways to generate or choose the target output, by investigating the variability in the N-best list.

5.1.1. 1-best

The *1-best* is the hypothesis in the N-best list which has the highest recognition score. With the expectation that more accurate hypotheses will also have higher recognition scores, a natural action would be to select the 1-best as the target output. In this case, the 1-best will have zero word errors, and the WE (thus, the rank) of other hypotheses will be computed by aligning each to the 1-best.

Although this gives a practical and easy replacement of the reference, choosing the 1-best as the target output has a downside for the structured perceptron algorithm setup: As shown in Section 3.2.1, the structured perceptron operates by comparing the oracle hypothesis to the current best. With 1-best as the target output, since the oracle and the current best hypotheses will always be the same, the structured perceptron cannot make any model updates. On the other hand, the ranking perceptron algorithm can still train the model by using the relationships between other hypotheses of the N-best list.

5.1.2. Minimum Bayes Risk

The target output can also be derived by minimizing an error function, or maximizing an objective function over the N-best list. A method used in the literature for this purpose is the Minimum Bayes Risk (MBR) [62, 69, 70]. The MBR value for a target output candidate $\hat{y}$ is defined as:

$$\begin{aligned} MBR(\hat{y}|x) =& E_{\tilde{y}|x}[\Delta(\tilde{y},\hat{y})] \\ =& \sum_{\tilde{y}\in\tilde{\mathcal{Y}}} \Delta(\tilde{y},\hat{y}) p(\tilde{y}|x) \end{aligned} \tag{5.1}$$

where $\Delta(\tilde{y},\hat{y})$ denotes the edit distance of the other hypotheses $\tilde{y}$ aligned to the candidate, and $P(\tilde{y}|x)$ denotes the posterior probability (recognition score) assigned to the hypotheses by the ASR system. The MBR target output is the hypothesis which yields

the lowest MBR score:

$$y = \underset{\hat{y} \in \tilde{\mathcal{Y}}}{\operatorname{argmin}} \, MBR(\hat{y}|x) \tag{5.2}$$

5.1.3. Segmental MBR

The MBR technique stated above operates on the sentence level. It is possible to adapt the behavior of MBR for the language unit level. In this technique, the hypotheses are aligned unit-wise using the Levenshtein algorithm, and each aligned piece is processed separately with the MBR formulation. This requires creating a token confusion network (also known as the *sausage*), and finding the best path along this graph from the first node to the last. This method is called Segmental MBR (SegMBR, in short) [71]. Combining the outputs for each piece yields the final SegMBR target output. Note that in this case, the target output may be different from any of the hypotheses in the N-best list.

5.2. Experimental Setup

The experiments in this chapter all utilize Bogazici University Turkish Broadcast News Database as before, but are varied in terms of the amount of data they use. These will be explicitly stated in the experiments.

The ASR N-best lists include 50 hypotheses. In order to obtain equivalent results, we limit the number of artificial hypotheses to 50 with the Top-50 hypothesis selection scheme. For experiments which involve confusion modeling, we use the morph CM with reweighting via the GEN-LM. The feature vector $\mathbf{\Phi}$ consists of morph unigram frequencies.

For all experiments, we train the discriminative models with the WER-sensitive structured perceptron (WPer) and ranking perceptron (WPerRank) algorithms. Some experiments in this chapter share the same data and algorithmic setup with experi-

ments presented in earlier chapters, but provide different (better) WERs. This is due to an increase in the number of epochs (a maximum of 50) and the range of algorithmic parameters. As before, all algorithmic parameters, together with the number of iterations over the data, are optimized on the held-out set.

Please note that some experiments of this section include an acoustic corpus that was collected at a later time than the data we have been using so far. Therefore we need a second held-out and test set for experiments which involve this dataset. We will call these sets h_2 and e_2, respectively. Table 5.1 shows the generative baselines and the oracle rates of the new held-out and test sets. We also provide the rates of the older sets for comparison.

Table 5.1. Baseline and oracle WER (%).

Subset	h	e	h_2	e_2
Baseline	22.9	22.4	24.1	23.9
Oracle	14.2	13.9	14.6	14.4

5.3. Experimental Results

In this section we investigate the two possible scenarios that can occur in unsupervised discriminative language modeling. The first of these is the unsupervised DLM scenario, where the real ASR outputs are analysed to determine the target output and DLM training is done using this target output as the reference, just like the supervised case. The second scenario is the unsupervised CM, where the target output is utilized in confusion modeling and the DLM is trained using artificial hypotheses generated from this confusion model, just like the semi-supervised case. In a third direction, we try to combine the data obtained in unsupervised DLM and unsupervised CM setups to check if the performance can be improved. We finally compare the performances of MT-based CMs and WFST-based CMs for the unsupervised setting.

5.3.1. Unsupervised DLM Training

In the first experimental set we create a case that is similar to supervised training where the DLM is trained using real ASR hypotheses. This time, however, we do not have the manual reference transcriptions. We apply the three approaches presented in Section 5.1 to choose the target output and use this as a reference to determine the target ranks of the hypotheses.

These experiments use half of the dataset m as the unsupervised data. We assume that we only have the acoustic speech utterances of this dataset, but not the transcriptions. Due to its acoustic-only nature ($\mathcal{A}$), we will denote it as $m^{\mathcal{A}}_{1/2}$. Please note that the fraction $1/2$ does not necessarily correspond to specifically the first half of $m^{\mathcal{A}}$. We apply two-fold cross validation by using the two halves interchangeably and average the results when reporting.

The first three rows of Table 5.2 show unsupervised DLM training performance in terms of WER with respect to the three target output selection approaches and two DLM training algorithms on the held-out (h) and test (e) sets. The last row contains the supervised case where the reference is used instead of the target output, and is included for comparison.

Table 5.2. Unsupervised DLM training WER (%)(Baseline: h 22.9%, e 22.4%).

Target output	WPer		WPerRank	
	h	e	h	e
1-best	22.9	22.4	22.3	22.1
MBR	22.7	22.3	22.3	22.1
SegMBR	22.7	22.3	22.3	22.1
Reference	22.2	22.0	21.8	21.6

We see from Table 5.2 that WPerRank outperforms WPer regardless of what word sequence is used as the target output. The choice of target output has no sig-

nificant effect on DLM performance. Note that WPer with the 1-best setup yields the same WER as the baseline, as expected. This observation is also consistent with [62]. The WER improvement with WPerRank over WPer on h is an absolute 0.4% with a significance value of $p < 0.001$, which corresponds to 50% of the gain that could be obtained under the supervised setup. This improvement is also reflected on the test set with a WER of 22.1%.

5.3.2. Unsupervised CM Training

In the second set of experiments we investigate the case where the in-domain data (the data we would like the DLM to be based on) consists only of text ($\mathcal{T}$), but we also have some other acoustic data ($\mathcal{A}$) available. We first build a CM using $\mathcal{A}$. Unlike semi-supervised training, the hypotheses in the N-best list are aligned to the chosen target output instead of the manual reference to determine the confusion probabilities. The learned CM can then be applied on $\mathcal{T}$ to generate artificial hypotheses for training the DLM. We use the manual transcriptions of the remaining half of m as the text data, to be able to obtain comparable results with the previous experiments. Table 5.3 shows WERs of the training algorithms with this setup.

Table 5.3. Unsupervised CM training WER (%)(Baseline: h 22.9%, e 22.4%).

Target output	WPer		WPerRank	
	h	e	h	e
1-best	22.7	22.3	22.5	22.3
MBR	22.9	22.4	22.5	22.2
SegMBR	22.8	22.4	22.5	22.2
Reference	22.6	22.2	22.4	22.1

The superiority of WPerRank over WPer is once again visible in Table 5.3, although the WER has slightly increased with respect to the unsupervised DLM results in Table 5.2. However, we now see that the gap between the semi-supervised case (shown in the last row) and the unsupervised CM case has decreased, yielding only an

absolute 0.1% change (which is not statistically significant) on both sets.

5.3.3. Combination of Data

In the experiments in Section 5.3.2, the (artificial) hypotheses which were used to train the DLM were generated on a different dataset than the (real) hypotheses used to train the CM. In this section we investigate whether there is any room for further improvement on WER by using a combination of these two hypothesis sets for DLM training. Table 5.4 shows the WER obtained with WPerRank for two experiments using this idea.

Table 5.4. WPerRank data combination WER (%)(Baseline: h 22.9%, e 22.4%).

Method	Training Data	h	e
Sup. + Semi-Sup.	$m_{1/2} + m^{\mathcal{T}}_{2/2}$	21.9	21.6
Unsup. DLM + Unsup. CM	$m^{\mathcal{A}}_{1/2} + m^{\mathcal{T}}_{2/2}$	22.2	22.0

The first row of Table 5.4 represents the case where the artificial hypotheses generated by the semi-supervised setup are combined with the real ASR hypotheses used for confusion modeling. Compared to the last row of Table 5.2 (the supervised case), h WER is slightly increased but e WER is unchanged. This means that adding artificial hypotheses to the training set brings no additional gain over supervised training. A similar comparison for the unsupervised setting is given in the second row of Table 5.4. With 1-best chosen as the target output, the same operation offers a slight but not significant improvement of 0.1% over the first row of Table 5.2.

5.3.4. MT-based Confusion Modeling for Unsupervised CM

Up to this section, confusion models were built using the WFST-based approach. In this section we evaluate the MT approach for unsupervised confusion modeling.

In the examples that follow, we assume that we have 60-hours of acoustic data, which corresponds to $m^{\mathcal{A}}_{1/3}$. The real ASR hypotheses obtained out of this dataset are used either to train the DLM directly, or to build the CM. The second piece consists of a text corpus of 34K sentences, which are the manual transcriptions of $m^{\mathcal{T}}_{2/3}$. We employ the second piece as the source text upon which artificial hypotheses are generated. As in our earlier experiments, we intentionally select these transcriptions but not some other text so that we can compare the artificial hypotheses with the real hypotheses of the same source. There are about 34K unique morphs in $m^{\mathcal{A}}_{1/3}$ and 19K unique morphs in $m^{\mathcal{T}}_{2/3}$. For simplicity, we will refer to the first set as $\mathcal{A}$ and the second set as $\mathcal{T}$.

Table 5.5 presents the WERs obtained by the WPerRank algorithm on the held-out and test sets. The first row represents the unsupervised-DLM scenario, where the DLM is trained directly using the real ASR hypotheses ($\mathcal{A}$). In the second row, $\mathcal{A}$ is used to build a WFST-based CM, which in turn generates the artificial hypotheses $\mathcal{T}_{WFST}$ through unsupervised-CM. The third row is similar to the second, this time using the MT-based approach to generate $\mathcal{T}_{MT}$.

Table 5.5. WPerRank WER (%) for different training data types.

Training Data	h	e
$\mathcal{A}$	22.5	22.0
$\mathcal{T}_{WFST}$	22.5	22.3
$\mathcal{T}_{MT}$	22.4	22.0

We see from Table 5.5 that all three experiments provide lower WER than the held-out baseline of 22.9%. Training the DLM using $\mathcal{T}_{MT}$ improves the held-out accuracy by 0.5%, which is statistically significant at $p < 0.001$. The test set performance is also better than using $\mathcal{T}_{WFST}$, which shows that the model obtained by the MT-based approach is more generalizable.

The unsupervised-DLM experiment which uses real hypotheses $\mathcal{A}$ to train the DLM shares the best test set WER with $\mathcal{T}_{MT}$. Note that in this experiment, the

manual transcriptions are not known, and the MBR target outputs are used instead. For comparison, if the manual reference transcriptions of $\mathcal{A}$ were available, the rate on the same set would be 21.6% (not shown on the table). This suggests that unsupervised training is able to provide half of the gains that could be obtained with the supervised technique, without altering the test set accuracy.

Table 5.6. N-best combination WPerRank WER (%).

Training Data	h	e
$\mathcal{A} + \mathcal{T}_{WFST}$	22.2	22.0
$\mathcal{A} + \mathcal{T}_{MT}$	22.3	22.1
$\mathcal{A} + \mathcal{T}_{WFST} + \mathcal{T}_{MT}$	22.2	22.0

It is also possible to combine the unsupervised-DLM and unsupervised-CM approaches by combining the real hypotheses of set $\mathcal{A}$ with the artificial hypotheses of set $\mathcal{T}$. Table 5.6 shows the performances for possible combinations of different data types. We see that combining all three sources decreases the held-out WER by an additional 0.2%, down to 22.2%.

5.4. Data Dependency of DLM Training Scenarios

In the previous section we examined the performance of unsupervised DLM and unsupervised CM techniques. With these experiments, we now have a complete set of tools for handling all four discriminative language modeling scenarios that were presented in Section 2.6.

One common observation with all the experiments that were done until now was the dependency of discriminative language modeling performance with respect to the amount of data. However, the experiments conducted in different sections were not comparable to each other because they used different data pieces and algorithmic setups. In this section we prepare a common setup, and compute the performance of all four discriminative language modeling scenarios and their combinations, with a special focus on the type and size of data involved in training.

5.4.1. Supervised Training

We begin by investigating supervised modeling performance with respect to the number of training examples. The set m of 188 hours is divided into three equal parts, shown as $m_{1/3}$,$m_{2/3}$,$m_{3/3}$. Table 5.7 presents the WERs obtained on h and e with respect to gradually decreasing number of training examples. Please note that for all experiments using part(s) of the dataset, the results are averaged through cross-validation.

Table 5.7. Supervised training WER (%).

DLM set	h		e	
	WPer	WPerRank	WPer	WPerRank
$m_{1/3} + m_{2/3} + m_{3/3}$	21.9	21.8	21.6	21.3
$m_{1/3} + m_{2/3}$	22.0	21.8	21.7	21.5
$m_{1/3}$	22.3	22.1	21.9	21.7

The implication of Table 5.7 is twofold: First, discriminative performance degrades with decreasing amount of training data. For instance, the improvement with WPer over the held-out baseline (22.9% for h) drops from 1.0% to 0.6% absolute as the training data is decreased to its one-third. Second, WPerRank outperforms WPer in all cases. It provides an additional improvement of around 0.2% over WPer, which is statistically significant at $p < 0.001$. The same improvement is also reflected on the test set, which implies that WPerRank provides a more generalizable model due to its broader utilization of the N-best list.

For an N-best list, the computational complexity of the WPerRank algorithm is quadratic in N whereas the complexity of WPer is linear. As a comparison, for the $m_{1/3} + m_{2/3} + m_{3/3}$ case the CPU times for a single run (for a certain parameter set with 50 epochs over the training data) of the WPer algorithm is around 21 minutes whereas a single run of a WPerRank algorithm takes about 51 minutes.

5.4.2. Semi-supervised Training

In the second experiment set, we assume that we have only some part of matched data ($\mathcal{M}$), and investigate the ways to do better than the second and third rows of Table 5.7 by adding other text data ($\mathcal{T}$) into training.

Table 5.8. Semi-supervised training WER (%) on h.

CM set	DLM set	WPer	WPerRank
$m_{1/3}$	$m^{\mathcal{T}}_{2/3}$	22.7	22.5
$m_{1/3}$	$m^{\mathcal{T}}_{2/3} + m^{\mathcal{T}}_{3/3}$	22.6	22.5
$m_{1/3} + m_{2/3}$	$m^{\mathcal{T}}_{3/3}$	22.6	22.4

The first row of Table 5.8 represents the case where $m_{1/3}$ is the available matched data ($\mathcal{M}$), and $m^{\mathcal{T}}_{2/3}$ is some source text ($\mathcal{T}$). $m_{1/3}$ is used to build a CM, which is then used to generate artificial hypotheses out of $m^{\mathcal{T}}_{2/3}$. In this experiment, we again intentionally choose $\mathcal{T}$ to be the reference transcriptions of $m_{2/3}$ but not some other text from t, so that we are able to compare the artificial hypotheses to the real ASR hypotheses of the supervised case. These hypotheses are in turn fed into DLM training. The result is a 0.4% decrease in held-out WER by the WPerRank, which is half of the improvement obtained in the supervised scenario (Table 5.7, row 3).

The second and third rows of Table 5.8 also show that doubling the number of artificial hypotheses by adding more source text ($m^{\mathcal{T}}_{2/3}+m^{\mathcal{T}}_{3/3}$), and doubling the amount of matched data to build a larger CM ($m_{1/3}+m_{2/3}$) may have a slightly positive effect on the system performance. The test set WER for all three cases follow a similar behavior (22.2% for WPer, 22.1% for WPerRank).

But there is still a way to decrease the WER, by reusing the CM data in DLM training. For instance, the first row of Table 5.9 shows that combining the real ASR hypotheses of $m_{1/3}$ (shown in parentheses) with the artificial hypotheses of $m^{\mathcal{T}}_{2/3}$ results

Table 5.9. Combining supervised and semi-supervised setups, WER (%) on h.

CM set	DLM set	WPer	WPerRank
$m_{1/3}$	$(m_{1/3})+m^{\mathcal{T}}_{2/3}$	22.3	22.0
$m_{1/3}$	$(m_{1/3})+m^{\mathcal{T}}_{2/3}+m^{\mathcal{T}}_{3/3}$	22.3	22.1
$m_{1/3}+m_{2/3}$	$(m_{1/3}+m_{2/3})+m^{\mathcal{T}}_{3/3}$	22.1	21.7

in an additional 0.5% improvement over Table 5.8. The held-out WER of 22.0% by WPerRank is slightly better than its supervised counterpart ($m_{1/3}$, 22.1%) shown on the third row of Table 5.7. This difference is significant at $p < 0.001$. The performance can be further improved by doubling the amount of matched data ($m_{1/3} + m_{2/3}$). Analysis has shown that this improvement is due to the availability of more real hypotheses for training the DLM, rather than an effect of a larger CM. With WPerRank, the test set WER of this last setup is 21.4%, which is better than the second row but worse than the first row of Table 5.7.

5.4.3. Unsupervised DLM Training

As a third experiment set, we assume that we have only acoustic data with no manual transcriptions ($\mathcal{A}$), and we investigate the performance of target output selection approaches. Table 5.10 shows held-out WERs over the real ASR hypotheses of $m^{\mathcal{A}}_{1/3}$.

Table 5.10. Unsupervised DLM training with $m^{\mathcal{A}}_{1/3}$, WER (%) on h.

Target output	WPer	WPerRank
1-best	22.9	22.5
MBR	22.7	22.5
SegMBR	22.7	22.5

The first observation is that with WPer and 1-best selected as the target output, no gains could be obtained, as expected and explained in Section 5.1.1. Choosing the MBR or SegMBR hypotheses as the target output, it is possible to decrease the WER by 0.2%. Regardless of the target output choice, improvements up to 0.4% over the baseline (significant at $p < 0.001$) could be achieved with WPerRank. WPerRank also outperforms WPer on the test set by 0.3% (22.0%). Further experiments have shown that the WER does not change significantly when we add more acoustic data.

In the last experiment, the DLM was trained using the acoustic component of the set m. One might argue that this might create a unfair biasing effect for the improvement in system performance, since m was already used in training of the baseline system. Therefore, in this new set of experiments we investigate the case where the acoustic data used to train the DLM is completely unknown to the baseline system. Table 5.11 compares the performance of set a to set $m^{\mathcal{A}} = m^{\mathcal{A}}_{1/3} + m^{\mathcal{A}}_{2/3} + m^{\mathcal{A}}_{3/3}$ on the second held-out (h_2) and test (e_2) sets. We see that using the set a for 1-best unsupervised DLM training yields the same held-out WER (23.6%), but performs better by 0.1% on the test set. This result also suggests that using matched data which was previously utilized in training of the baseline system does not yield any positive biasing effect on DLM performance.

Table 5.11. WPerRank 1-best unsupervised DLM training WER (%).

Unsup. DLM set	h_2	e_2
$m^{\mathcal{A}}$	23.6	23.4
a	23.6	23.3

5.4.4. Unsupervised CM Training

In the fourth experiment set, we use the acoustic data $\mathcal{A}$ to train the CM in an unsupervised way, via aligning the real hypotheses to the chosen target output (the 1-best in this case). The CM is used to generate artificial hypotheses from the source text $\mathcal{T}$. Optionally, we can include CM training data into DLM training, as done in the semi-supervised setup.

Let us now assume that some part of the set m is untranscribed (e.g. $m^{\mathcal{A}}_{1/3}+m^{\mathcal{A}}_{2/3}$), and the rest has reference transcriptions but not the corresponding audio (e.g. $m^{\mathcal{T}}_{3/3}$). In this subsection we train the unsupervised CM with part(s) of $m^{\mathcal{A}}$ and generate artificial hypotheses from $m^{\mathcal{T}}$. We choose the 1-best target selection scheme for all experiments in this section.

Table 5.12. Unsupervised CM training WER (%) on h.

Unsup. CM set	DLM set	WPer	WPerRank
$m^{\mathcal{A}}_{1/3}$	$m^{\mathcal{T}}_{2/3}$	22.8	22.5
$m^{\mathcal{A}}_{1/3}$	$(m^{\mathcal{A}}_{1/3})+m^{\mathcal{T}}_{2/3}$	22.7	22.3
$m^{\mathcal{A}}_{1/3}+m^{\mathcal{A}}_{2/3}$	$m^{\mathcal{A}}_{3/3}$	22.8	22.4
$m^{\mathcal{A}}_{1/3}+m^{\mathcal{A}}_{2/3}$	$(m^{\mathcal{A}}_{1/3}+m^{\mathcal{A}}_{2/3})+m^{\mathcal{T}}_{3/3}$	22.7	22.2

Table 5.12 shows unsupervised CM training performance on the h set. The CM set can also be added to DLM training, just like the semi-supervised case. In that sense, the first and second rows of Table 5.12 are analogous to the first rows of Table 5.8 and Table 5.9, respectively.

We see from the table that WPerRank is again more efficient than WPer, yielding significant improvements over the baseline setup even with this limited scenario. Combining real but unsupervised hypotheses ($m^{\mathcal{A}}_{1/3}$) with artificial examples generated by the unsupervised CM ($m^{\mathcal{T}}_{2/3}$) decreases the WER down to 22.3%. This value is again significantly ($p < 0.05$) better than the 22.5% WER obtained by training with only unsupervised DLM using the same set. Increasing the amount of the CM set ($m^{\mathcal{A}}_{1/3}+m^{\mathcal{A}}_{2/3}$) does not yield any change in WER for WPer but a 0.1% decrease for WPerRank. Further experiments (not shown here) have also shown that using MBR or SegMBR hypotheses as the target output, or increasing the amount of DLM sets do not alter the system performance.

5.4.5. Combination of Methods

Up to this section, the experiment sets included individual performances of the four basic discriminative modeling scenarios. Based on these results, in this section we explore the effect of combining different types of data ($\mathcal{M}$, $\mathcal{A}$ and $\mathcal{T}$) on the system accuracy. Figure 5.1 is an illustration of WERs obtained on the h set with WPerRank, for selected experiments that combine different types and amounts of training data. The relevant components of the set m (either m, $m^{\mathcal{A}}$ or $m^{\mathcal{T}}$) are used for training.

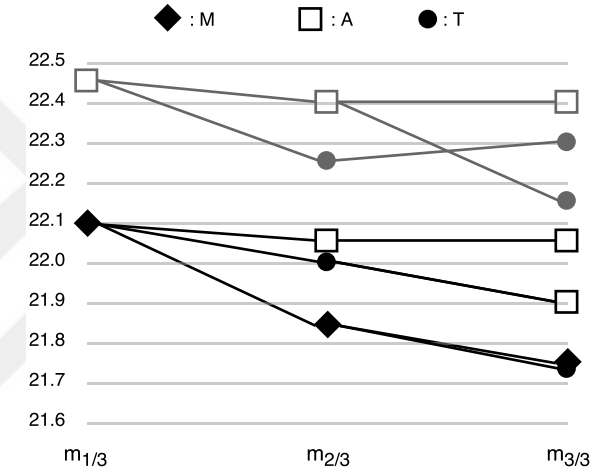

Figure 5.1. WPerRank WER (%) on h for different types and amounts of training data.

In Figure 5.1, the diamonds represent matched data ($\mathcal{M}$), the empty rectangles represent acoustic data ($\mathcal{A}$), and the full circles represent text data ($\mathcal{T}$). In terms of training scenarios, the diamonds stand for the supervised setup and the empty rectangles stand for the unsupervised DLM training setup. A circle tied to a diamond stands for the semi-supervised setup whereas a circle tied to a rectangle stands for the unsupervised CM setup. From left to right, we include an additional piece of data into training. The experiment set shown on the upper part of Figure 5.1 is composed of unmatched data. The lower set, on the other hand, has at least one piece of supervised (matched) training data.

Let us begin by investigating the lower experiment set that includes at least one piece of matched data. The all-supervised-training setup shown with the path at the very bottom of the graph is a graphical representation of Table 5.7. Here the WER is decreased with each additional $\mathcal{M}$ piece. In fact, we can achieve a similar WER when the third piece is $\mathcal{T}$ instead of $\mathcal{M}$. We also observe a significant decrease in WER by joining a single piece of the three data types, which is a combination of supervised, semi-supervised and unsupervised DLM settings. The least effective path in this experiment set seems to be adding two pieces of $\mathcal{A}$ onto $\mathcal{M}$.

Starting with one piece of $\mathcal{A}$ in the upper experiment set, we see that adding one piece of $\mathcal{T}$ decreases the WER on the contrary to adding one other piece of $\mathcal{A}$, which has no significant effect on the system accuracy. We also observe that further addition of the same data type (either $\mathcal{A}$ or $\mathcal{T}$) does not change the WER. The best case in this experiment set is the case where two pieces of $\mathcal{A}$ are combined with one piece of $\mathcal{T}$. It is interesting to see that with this approach the WER can be decreased by 0.3% to less than 22.2%, which is very close to what we would obtain if we had one piece of $\mathcal{M}$ (22.1%). The difference between the two results is not statistically significant.

5.5. Analysis of Results

In this section we elaborate on the experimental results presented in Section 5.4. We first give an analysis on the optimal combination of data types. We then discuss the effectiveness of artificial hypotheses. We finally consider the effect of using out-of-domain data instead of an in-domain text source.

5.5.1. Optimal Data Combination

The results in Section 5.4.5 hinted that combining three different types of data ($\mathcal{M}$, $\mathcal{A}$ and $\mathcal{T}$) for training the DLM can yield a cumulative improvement in ASR accuracy. However, that experiment set only allowed an equally-balanced combination of data, as each type was composed of one-third of the whole training set. An important question that arises in this regard is how to arrange the data balance in order to achieve the lowest WER. This is the first question that we would like to answer in this section.

In this new experiment we divide the training set m into twelve equal parts as opposed to three. We assume that one piece is fixed to be matched data ($\mathcal{M}$). This piece is included to DLM training as well as used to build a CM. The remaining eleven pieces will be split into acoustic ($\mathcal{A}$) and textual ($\mathcal{T}$) components, which will produce their own hypotheses through semi-supervised and unsupervised DLM settings as exemplified in earlier sections.

Figure 5.2 presents the held-out WERs of such an experiment set with respect to the number of pieces assigned to the $\mathcal{T}$ and $\mathcal{A}$ components. Please note that the number of both components sum up to eleven, so that the total amount of training data is the same for all instances.

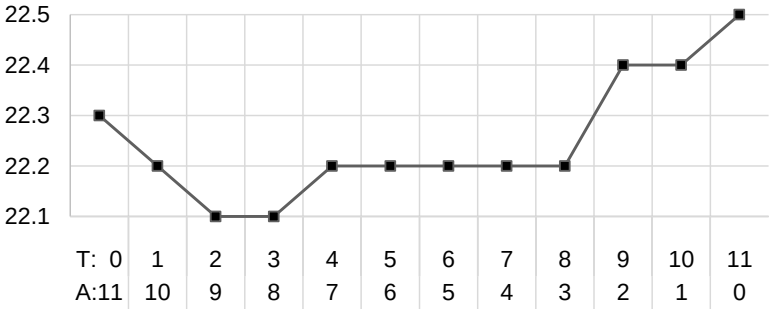

Figure 5.2. WPerRank WER (%) on h for different number of $\mathcal{T}$ and $\mathcal{A}$ data pieces for training.

The first instance on the graph is the result of combining eleven pieces of $\mathcal{A}$ with the base piece $\mathcal{M}$, which yields a WER of 22.3%. Then we start including one additional piece of $\mathcal{T}$ while discarding a comparable amount of $\mathcal{A}$, which at first causes a gradual decrease in WER, down to 22.1%. This $\mathcal{T}/\mathcal{A}$ balance of 2/9 (or, 3/8) turns out to be the ideal combination of different types of data in order to achieve the best performance under such setting. Including more $\mathcal{T}$ in place of $\mathcal{A}$ begins to increase the WER to the level of 22.5%. This level is in fact equal to training the DLM with only one piece of $\mathcal{M}$, without combining with any $\mathcal{A}$ or $\mathcal{T}$.

5.5.2. Effectiveness of Artificial Hypotheses

The second question that we would like to answer in this section is the effectiveness of the artificial hypotheses generated by the semi-supervised and unsupervised CM setups. We examine this by showing the similarity of the artificial hypotheses to the real ASR hypotheses of the supervised setup, measured by the number of unique morphs that are shared across these sets and their KL divergence.

Table 5.13. Similarity of artificial hypotheses to real hypotheses.

Setup	Number of morphs			WER (%)
	Total	New	Utilized	on e
Supervised	33.8K	-	33.0K	21.7
Semi-supervised	18.5K	1.0K	14.5K	22.1
Unsupervised CM	19.7K	1.2K	15.4K	22.2

Table 5.13 presents the number of unique morphs that are input to and utilized by the WPerRank algorithm along with their WER on e, for different training setups which use the $m_{3/3}$ piece of the dataset. We see that the real N-best lists of this piece contain a total of 33.8K unique morphs. Using $m_{1/3}$ to build the CM, we can artificially generate 18.5K unique morphs for the semi-supervised setup and 19.7K unique morphs for the unsupervised CM setup. Therefore, we see that the artificial hypotheses have a much narrower coverage of the morph feature space than the real hypotheses. Nevertheless, there exists around 1K morphs in each set that do not occur in the supervised set. This result shows the efficiency of the confusion modeling technique that we use in handling unseen data.

The fourth column of Table 5.13 shows the number of morphs that are actually used by the WPerRank algorithm, i.e., these are the morphs that have nonzero weights after training the DLM. We see that about 97% of the 33.8K morphs in real ASR hypotheses are utilized by the algorithm, whereas for the artificial sets, only up to 80% of the morphs can be utilized. Looking at the last column, we can deduce that the total

and utilized number of morphs have a direct influence on the test set performance.

Table 5.14. KL divergence of artificial examples.

	Supervised	Semi-supervised
Unsupervised CM	0.040	0.019
Semi-supervised	0.034	

Another type of comparison across these hypothesis sets is their KL divergences, shown in Table 5.14. These values are obtained by comparing the frequency histograms of one set to another. We observe that, although the morphs in the semi-supervised setup span a smaller portion of the feature space, their distribution is more similar to the supervised (real) morphs than the morphs in the unsupervised CM setup. This is considered to be one of the reasons for the lower WER.

5.5.3. Effect of Using Unmatched Audio and Out-of-Domain Source Text

The source text $\mathcal{T}$ we have used up to now, $m^{\mathcal{T}}$, is in-domain as it is composed of the reference transcriptions of spoken utterances in our dataset. It is also possible to use an out-of-domain text corpus instead of, or in addition to, the in-domain data.

The performance of discriminative language modeling under a semi-supervised setting that uses out-of-domain data as the source text has previously been investigated in the study by [72]. This study uses the same broadcast news dataset as ours as the in-domain data, together with a collection of sentences from newspaper articles as the out-of-domain data. The results show that under some conditions, similar or better performance can be achieved by using at least 10 times more out-of-domain data instead of in-domain data.

In order to extend the work in [72] to unsupervised CM setting, in this section we experiment with the effect of using out-of-domain data t, explained in Section 2.4. We generate the CM using the $m^{\mathcal{A}}$ and a sets and present the results in terms of WPer and WPerRank held-out WER in Tables 5.15 and 5.16.

Table 5.15. Unsupervised CM training with $m^{\mathcal{A}}$ and t, WER (%) on h.

Unsup. CM set	(Unsup.) DLM set	WPer	WPerRank
	$(m^{\mathcal{A}}_{1/3} + m^{\mathcal{A}}_{2/3})$	22.9	22.4
$m^{\mathcal{A}}_{1/3} + m^{\mathcal{A}}_{2/3}$	t	22.7	22.5
$m^{\mathcal{A}}_{1/3} + m^{\mathcal{A}}_{2/3}$	$(m^{\mathcal{A}}_{1/3} + m^{\mathcal{A}}_{2/3})+t$	22.7	22.4
	$(m^{\mathcal{A}})$	22.9	22.4
$m^{\mathcal{A}}$	t	22.7	22.5
$m^{\mathcal{A}}$	$(m^{\mathcal{A}})+t$	22.7	22.3

Table 5.15 shows the h set performance of experiments where CM is trained using whole or parts of $m^{\mathcal{A}}$. In the first row, we see that if $m^{\mathcal{A}}_{1/3} + m^{\mathcal{A}}_{2/3}$ is used to train a DLM the unsupervised way, WPerRank provides a WER of 22.4% on the held-out set, which is a 0.5% improvement over the baseline. In the second row, the DLM is instead built on artificial hypotheses generated from the out-of-domain source text t, via an unsupervised CM trained using $m^{\mathcal{A}}_{1/3} + m^{\mathcal{A}}_{2/3}$. Finally in the third row, these two hypothesis sets are combined to train a single model. We observe that although adding out-of-domain data into training has no positive effect on the performance of WPerRank, it yields a 0.2% decrease for WPer.

The setups shown on the second and third rows of Table 5.15 use the same CM with the last two rows of Table 5.12. The only difference is that the in-domain source text ($m^{\mathcal{T}}_{3/3}$) of approximately 35K utterances is replaced by the out-of-domain newspaper corpus t of 500K sentences. Comparing the WER of 22.4% shown in the third row with the corresponding WER of 22.2% shown in the last row of Table 5.12, we see that the system performances for both setups are not significantly different, and that out-of-domain data can be a viable alternative to using in-domain data for unsupervised training of DLMs.

In the second part of Table 5.15, the unsupervised CM is built upon all of $m^{\mathcal{A}}$. This time, combining the real hypotheses with the artificial hypotheses from t beats the unsupervised DLM WPerRank result (22.3% vs 22.4%). This result is also consistent with our previous observations in Figure 5.1 that the artificial hypotheses become more effective with a well-trained CM.

Table 5.16. Unsupervised CM training with a and t, WER (%) on h_2.

Unsup. CM set	(Unsup.) DLM set	WPer	WPerRank
	$(m^{\mathcal{A}})$	24.1	23.6
$m^{\mathcal{A}}$	t	24.0	23.8
$m^{\mathcal{A}}$	$(m^{\mathcal{A}})+t$	23.9	23.7
	(a)	24.1	23.6
a	t	24.0	23.8
a	$(a)+t$	23.8	23.5

In Table 5.16, we repeat the experiments using the set a instead of m to build the CM. We report the WERs on the set h_2 since we employ a as the training data. Here WPer achieves an improvement of 0.3% over the baseline on h_2 by combining the artificial N-best lists of t with the real N-best lists of a. With WPerRank the improvement over the baseline is 0.6%, however most of this gain comes from unsupervised DLM training, which was already reported in Table 5.11. Comparing the performance of similar experiments on the first and second parts of Table 5.16, we see that the WER values are not significantly different. This result suggests that for unsupervised CM training, there is no difference between using a dataset of unmatched audio a which is completely unknown to the baseline system and $m^{\mathcal{A}}$ that was used in training of the baseline acoustic model.

5.6. Discussion

In this chapter we focused on the unsupervised discriminative language modeling problem where the manual transcriptions of acoustic inputs are not available for training the DLM. We applied three different methods to choose the target output to replace the manual reference. We trained the discriminative models by (i) using the target output to determine the ranks of the real ASR hypotheses and (ii) building a confusion model to generate artificial examples on a text corpus.

The ranking perceptron algorithm is more suited to unsupervised DLM training problem in that it offers better system accuracies than the structured perceptron. By combining the two hypothesis sets for CM and DLM training, a slight decrease in WER can be obtained.

We also compare WFST- and MT-based artificial hypothesis generation approaches for unsupervised discriminative language modeling. These techniques allow us to make use of acoustic and textual data that are coming from different sources to train the discriminative language model, with no supervision at all. Experiments show that the MT-based approach is able to yield to slightly better WER than the WFST-based approach, although the superiority of MT-generated hypotheses are not as apparent as semi-supervised training.

6. CONCLUSION

In this thesis we apply discriminative language modeling techniques for Turkish ASR, with a special focus on approaches to improve training accuracy for cases when the amount of manually transcribed acoustic data is limited or not available at all. For the semi-supervised setting we build a WFST-based confusion model, and show that it is possible to decrease the WER significantly by combining real ASR hypotheses with generated artificial hypotheses. For the unsupervised setting, we compare three methods to choose a target output which replaces the missing reference text in order to define the ranks of the hypotheses for training the DLM. We adapt the confusion modeling technique for the unsupervised case, and show that even with acoustic data that is not manually transcribed and with text data not accompanied by any recording, it is possible to decrease baseline WERs significantly.

We use and compare the performance of three algorithms, namely, perceptron, MIRA and SVM, for both classification and reranking. We apply thresholding as a dimensionality reduction technique on the sparse feature set, and some hypothesis selection strategies to decrease the complexity of training.

Experiments have shown that the reranking variant of the algorithms outperforms the structured prediction variants for all scenarios. This superiority comes from considering each hypothesis in the N-best list instead of only two as in the former. The downside, however, is the increased training time due to algorithmic complexity.

The main advantage of unsupervised DLM training is that it shows improvements in ASR accuracy even when matched acoustic and text data are not present. The reranking approach can also be efficiently applied to unsupervised training. We believe that the techniques developed in this study will be beneficial especially to build ASR systems for under-resourced languages, where it is hard to find a large amount of transcribed acoustic data.

Although we investigate discriminative language modeling for ASR on Turkish, the techniques we developed are applicable to any language, thanks to the feature-based representation of the linear model. Any grammatical, syntactical or semantical information can also be easily integrated into the modeling scheme.

Some prospective topics on the area would be investigating the effect of using larger N-best and feature sets. The n-gram representation is very sparse and high-dimensional and an interesting future research direction is to represent such a long sparse vector using fewer features. The variability of the artificial hypotheses also seems to be an important factor in system performance, and it would be worthwhile to think more on the ways to improve variability within the artificially generated N-best lists.

The use of discriminative language modeling is not only limited to improving system performance for automatic speech recognition, and we believe that the techniques developed in this study can also be beneficial for other tasks such as machine translation and keyword search.

REFERENCES

1. Dikici, E., M. Semerci, M. Saraçlar and E. Alpaydın, "Data Sampling and Dimensionality Reduction Approaches for Reranking ASR Outputs Using Discriminative Language Models", *Proc. Interspeech*, pp. 1461–1464, Florence, Italy, 2011.

2. Çelebi, A., H. Sak, E. Dikici, M. Saraçlar, M. Lehr, E. Prud'hommeaux, P. Xu, N. Glenn, D. Karakos, S. Khudanpur, B. Roark, K. Sagae, I. Shafran, D. Bikel, C. Callison-Burch, Y. Cao, K. Hall, E. Hasler, P. Koehn, A. Lopez, M. Post and D. Riley, "Semi-Supervised Discriminative Language Modeling for Turkish ASR", *Proc. ICASSP*, pp. 5025–5028, Kyoto, Japan, March 2012.

3. Dikici, E., A. Çelebi and M. Saraçlar, "Performance Comparison of Training Algorithms for Semi-Supervised Discriminative Language Modeling", *Proc. Interspeech*, pp. 206–209, Portland, Oregon, Sept 2012.

4. Dikici, E., M. Semerci, M. Saraçlar and E. Alpaydın, "Classification and Ranking Approaches to Discriminative Language Modeling for ASR", *IEEE Transactions on Audio, Speech, and Language Processing*, Vol. 21, No. 2, pp. 291–300, 2013.

5. Dikici, E. and M. Saraçlar, "Müfredat Tabanlı Ayırıcı Dil Modeli Eğitimi", *Proc. SIU*, Girne, Kuzey Kıbrıs, April 2013.

6. Dikici, E., E. Prud'hommeaux, B. Roark and M. Saraçlar, "Investigation of MT-based ASR Confusion Models for Semi-Supervised Discriminative Language Modeling", *Proc. Interspeech*, pp. 1218–1222, Lyon, France, August 2013.

7. Dikici, E. and M. Saraçlar, "Gözetimsiz Ayırıcı Dil Modeli Eğitimi", *Proc. SIU*, pp. 1158–1161, Trabzon, Turkey, April 2014.

8. Dikici, E. and M. Saraçlar, "Unsupervised Training Methods for Discriminative Language Modeling", *Proc. Interspeech*, pp. 2857–2861, Singapore, Sept. 2014.

9. Dikici, E. and M. Saraçlar, "MT-based artificial hypothesis generation for unsupervised discriminative language modeling", *23rd European Signal Processing Conference (EUSIPCO)*, pp. 1401–1405, IEEE, Nice, France, 2015.

10. Saraclar, M., E. Dikici and E. Arisoy, "A Decade of Discriminative Language Modeling for Automatic Speech Recognition", *Speech and Computer*, pp. 11–22, Springer International Publishing, 2015.

11. Dikici, E. and M. Saraçlar, "Semi-supervised and unsupervised discriminative language model training for automatic speech recognition", *Speech Communication*, Vol. 83, pp. 54 – 63, 2016.

12. Dahl, G. E., D. Yu, L. Deng and A. Acero, "Context-dependent pre-trained deep neural networks for large-vocabulary speech recognition", *IEEE Transactions on Audio, Speech, and Language Processing*, Vol. 20, No. 1, pp. 30–42, 2012.

13. Graves, A., A.-r. Mohamed and G. Hinton, "Speech recognition with deep recurrent neural networks", *Proc. ICASSP*, pp. 6645–6649, IEEE, 2013.

14. Chen, S. F. and J. Goodman, "An Empirical Study of Smoothing Techniques for Language Modeling", *Proceedings of the 34th Annual Meeting on Association for Computational Linguistics*, ACL '96, pp. 310–318, Association for Computational Linguistics, Stroudsburg, PA, USA, 1996.

15. Forney, G. D., "The viterbi algorithm", *Proceedings of the IEEE*, Vol. 61, No. 3, pp. 268–278, March 1973.

16. Creutz, M. and K. Lagus, *Unsupervised Morpheme Segmentation and Morphology Induction from Text Corpora Using Morfessor 1.0*, Tech. rep., Helsinki University of Technology, Palo Alto, CA, March 2005, publications in Computer and Information Science Report A81, http://www.cis.hut.fi/projects/morpho, accessed at July 2016.

17. Arısoy, E., D. Can, S. Parlak, H. Sak and M. Saraçlar, "Turkish Broadcast News Transcription and Retrieval", *IEEE Transactions on Audio, Speech, and Language Processing*, Vol. 17, No. 5, pp. 874–883, 2009.

18. Arısoy, E., M. Saraçlar, B. Roark and I. Shafran, "Syntactic and sub-lexical features for Turkish discriminative language models", *Proc. ICASSP*, pp. 5538–5541, 2010.

19. Saraçlar, M., "Turkish Broadcast News Speech and Transcripts LDC2012S06", , 2012, Philadelphia: Linguistic Data Consortium. Web Download.

20. Mohri, M., "Finite-state transducers in language and speech processing", *Computational Linguistics*, Vol. 23, No. 2, pp. 269–311, 1997.

21. Goffin, V., C. Allauzen, E. Bocchieri, D. Hakkani-Tur, A. Ljolje, S. Parthasarathy, M. Rahim, G. Riccardi and M. Saraclar, "The AT&T WATSON Speech Recognizer", *Proc. ICASSP*, Vol. 1, pp. 1033–1036, March 2005.

22. Stolcke, A., "SRILM – an extensible language modeling toolkit", *Proceedings of ICSLP*, Vol. 2, pp. 901–904, Denver, 2002, http://www.speech.sri.com/projects/srilm, accessed at July 2016.

23. Roark, B., M. Saraclar and M. Collins, "Discriminative n-gram language modeling", *Computer Speech and Language*, Vol. 21, No. 2, pp. 373–392, April 2007.

24. Fan, R.-E., K.-W. Chang, C.-J. Hsieh, X.-R. Wang and C.-J. Lin, "LIBLINEAR: A Library for Large Linear Classification", *Journal of Machine Learning Research*, Vol. 9, pp. 1871–1874, 2008.

25. Joachims, T., "Training Linear SVMs in Linear Time", *Proceedings of the ACM Conference on Knowledge Discovery and Data Mining (KDD)*, pp. 217–226, 2006, http://svmlight.joachims.org, accessed at July 2016.

26. Allauzen, C., M. Riley, J. Schalkwyk, W. Skut and M. Mohri, "OpenFst: A General

and Efficient Weighted Finite-State Transducer Library", *CIAA 2007*, Vol. 4783 of *LNCS*, pp. 11–23, Springer, 2007, http://www.openfst.org, accessed at July 2016.

27. Pallett, D., W. M. Fisher and J. G. Fiscus, "Tools for the Analysis of Benchmark Speech Recognition Tests", *Proc. ICASSP*, Vol. 1, pp. 97 – 100, 1990.

28. Sagae, K., M. Lehr, E. T. Prud'hommeaux, P. Xu, N. Glenn, D. Karakos, S. Khudanpur, B. Roark, M. Saraçlar, I. Shafran, D. Bikel, C. Callison-Burch, Y. Cao, K. Hall, E. Hasler, P. Koehn, A. Lopez, M. Post and D. Riley, "Hallucinated N-best lists for discriminative language modeling", *Proc. ICASSP*, pp. 5001–5004, 2012.

29. Roark, B., M. Saraclar, M. Collins and M. Johnson, "Discriminative language modeling with conditional random fields and the perceptron algorithm", *Proceedings of the 42nd Annual Meeting on Association for Computational Linguistics*, ACL, pp. 47–54, Association for Computational Linguistics, Stroudsburg, PA, USA, 2004.

30. Saraclar, M. and B. Roark, "Joint Discriminative Language Modeling and Utterance Classification", *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, Vol. 1, pp. 561 – 564, 18-23, 2005.

31. Shen, L. and A. K. Joshi, "Ranking and Reranking with Perceptron", *Machine Learning*, Vol. 60, pp. 73–96, September 2005.

32. Li, Z. and S. Khudanpur, "Large-scale Discriminative n-gram Language Models for Statistical Machine Translation", *Proc. of the 8th AMTA Conference*, pp. 133–142, Hawaii, Oct 2008.

33. Saraçlar, M. and B. Roark, "Utterance classification with discriminative language modeling", *Speech Communication*, Vol. 48, No. 3-4, pp. 276 – 287, 2006, Spoken Language Understanding in Conversational Systems.

34. Collins, M., "Discriminative Training Methods for Hidden Markov Models: Theory and Experiments with Perceptron Algorithms", *Proc. EMNLP*, pp. 1–8, 2002.

35. Arısoy, E., B. Roark, I. Shafran and M. Saraçlar, "Discriminative n-gram language modeling for Turkish", *Proc. Interspeech*, pp. 825–828, 2008.

36. Arısoy, E., M. Saraçlar, B. Roark and I. Shafran, "Discriminative Language Modeling With Linguistic and Statistically Derived Features", *IEEE Transactions on Audio, Speech, and Language Processing*, Vol. 20, No. 2, pp. 540–550, Feb 2012.

37. Xu, P., D. Karakos and S. Khudanpur, "Self-supervised discriminative training of statistical language models", *Proc. ASRU*, pp. 317–322, 2009.

38. Jyothi, P. and E. Fosler-Lussier, "Discriminative language modeling using simulated ASR errors", *Proc. Interspeech*, pp. 1049–1052, 2010.

39. Crammer, K. and Y. Singer, "Pranking with Ranking", *Advances in Neural Information Processing Systems*, Vol. 14, pp. 641–647, MIT Press, 2001.

40. Shen, L. and A. K. Joshi, "Flexible margin selection for reranking with full pairwise samples", *Proc. of the 1st Int. Joint Conf. of Natural Language Processing (IJCNLP)*, pp. 446–455, 2004.

41. Shen, L., A. Sarkar and F. Och, "Discriminative Reranking for Machine Translation", S. Dumais, D. Marcu and S. Roukos (Editors), *Proc. HLT-NAACL*, pp. 177–184, Boston, MA, USA, 2004.

42. Sak, H., M. Saraçlar and T. Güngör, "Discriminative Reranking of ASR Hypotheses with Morpholexical and N-best-list Features", *Proc. ASRU*, pp. 202–207, 2011.

43. Vuorinen, M., *Discriminative Language Modeling*, Tech. rep., 2007.

44. Bergsma, S., D. Lin and D. Schuurmans, "Improved natural language learning via variance-regularization support vector machines", *Proc. CoNLL*, CoNLL, pp. 172–181, Association for Computational Linguistics, Stroudsburg, PA, USA, 2010.

45. Cherry, C. and C. Quirk, "Discriminative, syntactic language modeling through

latent SVMs", *Proc. of the 8th AMTA Conference*, pp. 65–74, Hawaii, Oct 2008.

46. Joachims, T., "Optimizing Search Engines Using Clickthrough Data", *ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD)*, pp. 133–142, 2002.

47. Zhou, Z., J. Gao, F. Soong and H. Meng, "A Comparative Study of Discriminative Methods for Reranking LVCSR N-Best Hypotheses in Domain Adaptation and Generalization", *Proc. ICASSP*, pp. 141–144, 2006.

48. Oba, T., T. Hori and A. Nakamura, "Efficient training of discriminative language models by sample selection", *Speech Communication*, Vol. 54, No. 6, pp. 791 – 800, 2012.

49. Oba, T., T. Hori, A. Nakamura and A. Ito, "Round-Robin Duel Discriminative Language Models", *IEEE Transactions on Audio, Speech, and Language Processing*, Vol. 20, No. 4, pp. 1244–1255, May 2012.

50. Crammer, K. and Y. Singer, "Ultraconservative Online Algorithms for Multiclass Problems", *Journal of Machine Learning Research*, Vol. 3, pp. 951 – 991, 2003.

51. Watanabe, T., J. Suzuki, H. Tsukada and H. Isozaki, "Online Large-Margin Training for Statistical Machine Translation", *Proc. EMNLP-CoNLL*, pp. 764–773, June 2007.

52. Chiang, D., Y. Marton and P. Resnik, "Online Large-Margin Training of Syntactic and Structural Translation Features", *Proc. EMNLP*, pp. 224–233, 2008.

53. McDonald, R., K. Crammer and F. Pereira, "Online large-margin training of dependency parsers", *Proc. ACL*, ACL, pp. 91–98, Association for Computational Linguistics, Stroudsburg, PA, USA, 2005.

54. Kurata, G., N. Itoh and M. Nishimura, "Acoustically discriminative training for

language models", *Proc. ICASSP*, pp. 4717–4720, 2009.

55. Kurata, G., N. Itoh and M. Nishimura, "Training of error-corrective model for ASR without using audio data", *Proc. ICASSP*, pp. 5576–5579, 2011.

56. Kurata, G., A. Sethy, B. Ramabhadran, A. Rastrow, N. Itoh and M. Nishimura, "Acoustically discriminative language model training with pseudo-hypothesis", *Speech Communication*, Vol. 54, No. 2, pp. 219 – 228, 2012.

57. Tan, Q., K. Audhkhasi, P. Georgiou, E. Ettelaie and S. Narayanan, "Automatic Speech Recognition System Channel Modeling", *Proc. Interspeech*, pp. 2442–2445, 2010.

58. Li, Z., Z. Wang, S. Khudanpur and J. Eisner, "Unsupervised Discriminative Language Model Training for Machine Translation using Simulated Confusion Sets", *Coling 2010: Posters*, pp. 656–664, Beijing, China, Aug. 2010.

59. Oba, T., T. Hori and A. Nakamura, "An approach to efficient generation of high-accuracy and compact error-corrective models for speech recognition", *Proc. Interspeech*, pp. 1753–1756, 2007.

60. Xu, P., B. Roark and S. Khudanpur, "Phrasal Cohort Based Unsupervised Discriminative Language Modeling", *Proc. Interspeech*, pp. 198–201, Portland, Oregon, Sept 2012.

61. Jyothi, P., L. Johnson, C. Chelba and B. Strope, "Distributed Discriminative Language Models for Google Voice Search", *Proc. ICASSP*, pp. 5017–5021, 2012.

62. Kuo, H.-K. J., E. Arısoy, L. Mangu and G. Saon, "Minimum Bayes Risk Discriminative Language Models for Arabic Speech Recognition", *Proc. ASRU*, pp. 208–213, 2011.

63. Collins, M. and N. Duffy, "New Ranking Algorithms for Parsing and Tagging:

Kernels over Discrete Structures, and the Voted Perceptron", *ACL*, pp. 263–270, 2002.

64. Herbrich, R., T. Graepel and K. Obermayer, *Large Margin Rank Boundaries for Ordinal Regression*, chap. 7, p. 115–132, MIT Press, January 2000.

65. Sak, H., M. Saraçlar and T. Gungor, "Morpholexical and Discriminative Language Models for Turkish Automatic Speech Recognition", *IEEE Transactions on Audio, Speech and Language Processing*, Vol. 20, No. 8, pp. 2341–2351, 2012.

66. Pallet, D. S., W. M. Fisher and J. G. Fiscus, "Tools for the analysis of benchmark speech recognition tests", *Proc. ICASSP*, Vol. 1, pp. 97–100, April 1990.

67. Koehn, P., H. Hoang, A. Birch, C. Callison-Burch, M. Federico, N. Bertoldi, B. Cowan, W. Shen, C. Moran, R. Zens, C. Dyer, O. Bojar, A. Constantin and E. Herbsts, "Moses: Open source toolkit for statistical machine translation", *Proceedings of the 45th Annual Meeting of the ACL Interactive Poster and Demonstration Sessions*, pp. 177–180, 2007.

68. Och, F. J. and H. Ney, "A comparison of alignment models for statistical machine translation", *Proceedings of the 18th Conference on Computational Linguistics*, pp. 1086–1090, 2000.

69. Goel, V. and W. Bryne, "Minimum Bayes-Risk Automatic Speech Recognition", *Computer Speech and Language*, Vol. 14, pp. 115–135, 2000.

70. Mangu, L., E. Brill and A. Stolcke, "Finding Consensus in Speech Recognition: Word Error Minimization and Other Applications of Confusion Networks", *Computer Speech and Language*, Vol. 14, pp. 373–400, 2000.

71. Goel, V., S. Kumar and W. Byrne, "Segmental Minimum Bayes-Risk ASR Voting Strategies", *Proc. Interspeech*, pp. 139–142, 2000.

72. Çelebi, A. and M. Saraçlar, "Semi-Supervised Discriminative Language Modeling with Out-of-Domain Text Data", *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 727–732, Association for Computational Linguistics, Atlanta, Georgia, June 2013.

APPENDIX A: RELATIONSHIP BETWEEN PerRank AND SVMrank

If $r_a \succ r_b$, we require $f(a) > f(b)$ where $f(u) = \langle \mathbf{w}, \mathbf{\Phi}_u \rangle$. With the ranking perceptron, the update rule is

$$\mathbf{w} = \mathbf{w} + g(a,b)(\mathbf{\Phi}_a - \mathbf{\Phi}_b) \tag{A.1}$$

This is applied when $f(a) < f(b)$. So the error function we should minimize is

$$E = \sum_{\substack{r_a \succ r_b \\ f(a)<f(b)}} [f(b) - f(a)] \tag{A.2}$$

If we use gradient-descent, we get

$$\Delta\mathbf{w} = -\eta[\nabla_{\mathbf{w}} f(b) - \nabla_{\mathbf{w}} f(a)] = \eta(\mathbf{\Phi}_a - \mathbf{\Phi}_b)$$

If we penalize differences with respect to their rank differences as given by a function such as $g(a,b)$:

$$E = \sum_{\substack{r_a \succ r_b \\ f(a)<f(b)}} [f(b) - f(a)]g(a,b) \tag{A.3}$$

when we use gradient-descent, we get

$$\Delta\mathbf{w} = \eta g(a,b)(\mathbf{\Phi}_a - \mathbf{\Phi}_b)$$

which is the update rule of Eq. A.1 with $\eta = 1$.

Let us now consider the case of SVM. We require $f(a) > f(b)$, so when this is not satisfied, we need slack variables ξ_{ab} to make up for the difference:

$$f(b) \leq f(a) + \xi_{ab}, \forall r_a \succ r_b$$

and the total error is

$$\begin{aligned} \min \quad & \sum_{r_a \succ r_b} \xi_{ab} \\ \text{subject to} \quad & f(a) \geq f(b) - \xi_{ab} \end{aligned} \tag{A.4}$$

If we require a difference of at least 1 unit as the margin, the constraints become

$$\text{subject to } f(a) \geq f(b) + 1 - \xi_{ab}$$

We can add a L_2 regularizer for smoothness and the error becomes

$$\min \frac{1}{2}\langle \mathbf{w}, \mathbf{w} \rangle + C \sum_{r_a \succ r_b} \xi_{ab}$$

where C denotes the relative weights of the first regularizer and the second data-misfit terms. With the perceptron too, if we like we can add a similar term—this is known as "weight decay" in neural network terminology.

So we see that, as would be expected, the ranking perceptron and ranking SVM minimize very similar error functions with some slight differences: (1) SVM enforces a minimum margin between differences, (2) In perceptron, error terms are weighted by the $g(r_a, r_b)$ term whereas for SVM, all have the same weight of C, and (3) SVM has an additional regularizer term—in perceptron, we have tried a version with weight decay but this did not a cause a significant difference.