Language Models with Meta-information

Yangyang Shi

Language Models with Meta-information

Proefschrift

ter verkrijging van de graad van doctor
aan de Technische Universiteit Delft,
op gezag van de Rector Magnificus prof. ir. K.C.A.M. Luyben,
voorzitter van het College voor Promoties,
in het openbaar te verdedigen op dinsdag 11 maart 2014 om 12:30 uur
door

Yangyang Shi

Master of Science in Mathematics Department, Southeast University, P. R. China
geboren te Yancheng, P. R. China.

Dit proefschrift is goedgekeurd door de promotor:
Prof.dr. C. M. Jonker

Copromotor: Dr. M. Larson

Samenstelling promotiecommissie:

Rector Magnificus	voorzitter
Prof.dr. C. M. Jonker	Technische Universiteit Delft, promotor
Dr. M. Larson	Technische Universiteit Delft, copromotor
Prof.drs.dr. L. J. M. Rothkrantz	Technische Universiteit Delft
Prof.dr.-ing. E. Nöth	Friedrich-Alexander Universität Erlangen-Nüremberg
Prof.dr. C. Witteveen	Technische Universiteit Delft
Prof.dr.ir. A. P. de Vries	Technische Universiteit Delft
Dr. M. Y. Hwang	Microsoft
Prof.dr. A. Hanjalic	Technische Universiteit Delft (reservelid)

SIKS Dissertation Series No. 2014-14.

The research reported in this thesis has been carried out under the auspices of SIKS, the Dutch Research School for Information and Knowledge Systems.

This thesis has been completed in partial fulfillment of the requirements of the Faculty of Electrical Engineering, Mathematics and Computer Science for graduate studies. The research described in this thesis was supported by China Scholarship Council.

Published and distributed by: Yangyang Shi
E-mail: yangyang.shi@tudelft.nl

ISBN: 978-94-6186-278-5

Keywords: Recurrent Neural Networks, Language models, Meta-information.

Copyright © 2013 by Yangyang Shi

All rights reserved. No part of the material protected by this copyright notice may be reproduced or utilized in any form or by any means, electronic or mechanical, including photocopying, recording or by any information storage and retrieval system, without written permission of the author.

Printed in The Netherlands

Preface

First of all I would like to express my deepest appreciation to Catholijn M. Jonker, my supervisor and promoter, for her great encouragement, generous support and thoughtful guidance. Research progress does not at all time progress smoothly. It were her words and guidance that kept on encouraging me. Martha Larson, who made a major contribution to my supervision in the final two years of the PhD program also provided valuable suggestions and helpful assistance. Pascal Wiggers provided the critical supervision that introduced me to the language modeling area. It was his insights that made it possible for me to orient myself at the beginning of the thesis and become established in this fascinating and productive topic.

I owe a lot of thanks to the excellent colleagues and support staff in the Interactive Intelligence Group for the great atmosphere and sincere friendship. In particular, I would like to thank Harold Nefs, who never hesitated in sharing his knowledge and helping me understand Dutch culture; Chao Qu and Yun Ling who always understood me and offered me a feeling of family; Changyun Wei, Tingting Zhang, Iulia Lefter, Tim Baarslag, Chang Wang, Ni Kang, Junchao Xu, Yi Zhu, Wenxin Wang, Hani Alers, Hantao Liu, Zhenke Yang and Nike Gunawan, who filled a lot of joys to my PhD life; Ruud de Jong, Bart Vastenhouw and Anita Hoogmoed, without whom behind the scene nothing would be possible.

It was a great experience for me to do internship in Microsoft. Cordial appreciation goes to the colleagues in Microsoft IPE, MSR and MSRA. In particular, I would like to thank Mei-Yuh Hwang, not only for giving me the opportunity to do an inspiring and productive internship in Microsoft, but also for close coaching and patient support. Despite a busy schedule, she took time to read my thesis and to give detailed and insightful comments. Sincere gratitude also goes to Kaisheng Yao for helpful instructions, inspiring discussion and enjoyable cooperation.

Many other people also contributed to this thesis. It has been a delight for me to collaborate with Joris Pelemans, Patrick Wambacq from Catholic University of Leuven and Kris Demuynck from University of Gent. Many thanks for kind help and insightful comments and suggestions.

The four years in Delft has been made so memorable and enjoyable due to my Chinese friends. I would like to thank you all for the delicious Chinese food, for the pleasurable trips around Europe and for the special moments and holidays we shared together.

Last but not least, I would like to give special thanks to my parents and wife. Without their faithful support and endless love, I even would not have survived during these years.

Yangyang Shi,
Suzhou, February, 2014.

Contents

Chapter 1

Introduction

In this chapter, we present the motivation and background for the research on language models that is presented in this thesis. In Section 1.1, we motivate the thesis. In Section 1.2, we undertake a survey of language modeling, with emphasis on the integration of linguistic, syntactic and socio-contextual features. Furthermore, we focus on various types of meta-information and discuss the potential role that meta-information can play to boost the performance of language models. Based on the findings in our literature survey, in Section 1.3, we formulate the main questions that we will answer in the thesis. The structure of the thesis is summarized in Section 1.4.

1.1 Motivation

In this thesis, we present studies in the area of language modeling, especially about how to improve language models using meta-information. Meta-information in the context of this thesis is information about language that goes beyond the identity of the individual word. We use meta-information to refer to all potential information that goes beyond text information.

The idea to combine language models with meta-information of language has been investigated by many researchers working in the area of language modeling. Cache-based language models [78] use a cache window to store statistical temporal information. The motivation for cache-based language models is that language is characterized by the fact that human tends to use language in a *bursty* way. In other words, a word that occurs in recent history has a higher chance of occurring again in the near future. Class-based language models [23] and topic-based language models [56] exploit the clustering of training data to improve language models. Structured language models [25] directly embed the syntactic structure of language into language models. In this thesis, we also exploit different attributes of language. In particular, we investigate different ways of integrating those attributes into state-of-the-art language models.

The dominant approach of current language modeling is the empiricist approach [93]. The empiricist approach believes that language learning starts with general operations for recognition, generalization and connection that are applied to learn more complicated structure. Most current probabilistic models only use the words themselves no other properties of languages. The models cannot represent the complexity of human language. State-of-the-art language models incorporate no knowledge reflecting the fact that what is being processed is a natural language expression; to them it is a sequence of symbols. In human language learning, word-symbols are associated with representations from different perspectives, e.g., the pronunciation of the word, the meaning, the word stem and the syntactic attributes. In fact, the lexicon can be considered to consist of structured units rather than simple word symbols.

In this thesis, we follow an empiricist approach that incorporates rich input information. We explore discourse level, sentence level and word level representations of each language unit in addition to a word. In order to integrate different levels of meta-information into language models, we build on new architectures that have been recently developed, such as recurrent neural network language models (RNNLMs). To integrate discourse level meta-information, we propose to use component models in RNNLMs that are trained by curriculum learning. To integrate sentence level meta-information, we use a forward-backward language modeling strategy. For word-level meta-information, we propose an extended RNNLM with more information in the input layer. Our motivation is not to train language models on large quantities of data, but to extract more information from the existing data, for example by moving the training of language models from simple to complex. In our thesis, empirical results are presented based on which different language models are compared and the usefulness of integrating meta-information into language models is shown. Finally, a new parallelization technique is described, which has been developed to speed up state-of-the-art language models.

1.2 Language Models

The objective of language models is to characterize, capture and exploit regularities in natural languages. Statistical language models attempt to tackle the task by assigning probabilities to sequences of words in a given language. Well-formed, syntactically and semantically plausible sentences receive high probabilities.

Statistical language models are used in practical applications such as automatic speech recognition, optical character recognition, handwriting recognition, spelling correction and statistical machine translation ([22, 35, 62, 73, 94, 148]).

The most prominent usage of statistical language models is in automatic speech recognition, which is the calculation of the most likely word sequence W with respect to a given speech signal observation O. The task of speech recognition can be formulated as:

$$\hat{W} = \arg\max_{W} P(W|O). \tag{1.1}$$

However, the conditional probability in (1.1) is difficult to calculate directly because, in practice, each signal observation is almost unique due to environment noise and variations in the speech characteristics of different speakers. Using Bayes' law ([11]), equation (1.1) can be transformed to:

$$\hat{W} = \arg\max_{W} \frac{P(O|W)P(W)}{P(O)}, \tag{1.2}$$

where $P(W)$ is the probability of the sequence of words. $P(O)$ is a normalization constant. Since we are interested in the W that maximizes equation (1.2), we can simplify the equation into:

$$\hat{W} = \arg\max_{W} P(O|W)P(W). \tag{1.3}$$

The probabilities of word sequences can be defined in multiple ways. The commonly adopted chain-rule presupposes that each word only depends on the previous words. The probability of a sequence of words W is formulated as a product of the conditional probability of current word w_i given the history of previous words $h(w_i)$:

$$P(W) = P(w_1)\prod_{i=2}^{n} P(w_i|h(w_i)), \tag{1.4}$$

where w_i is the i-th word in the sequence and $h(w_i) = w_1 w_2 ... w_{i-1}$. This formula was first used by Claude Shannon to calculate the entropy of printed English [136].

However, as can be seen from equation (1.4), the language model only makes use of words rather than other properties of languages. In this thesis we adapt the proposition that in order to achieve improvement, "We must put language back into language models" [126].

The assumption of statistical language modeling originates from the philosophy of statistics that historical information can be used to predict the future. The parameters of language models are learned from a preselected large number of word sequence samples. In order to obtain the most reliable statistical estimation possible, learning should be based on rich information. We can improve language models by adding information. In this thesis we exploit the fact this can be achieved not exclusively by increasing sample size, but also by extracting different types of information from samples.

In Section 1.2.1, we present the metrics that are commonly used for assessing the quality of language models. In the rest of Section 1.2, we give a survey of the most important language models, from simple n-gram to computationally driven and meta-information driven language models. Furthermore, we provide an introduction to meta-information and the language models that are based on that meta-information.

1.2.1 Measure

To evaluate the performance of language models, the following measures have been defined.

Cross-entropy

To measure the quality of a language model, one method is to estimate the logarithm likelihood $LP(W)$ of test data with n words, which are assumed to be drawn from the true data distribution.

$$LP(W) = \frac{1}{n}\sum_{i}^{n} \log_2(P(w_i)). \tag{1.5}$$

The negative value of this quantity, i.e., $-LP(W)$ is the cross-entropy. In information theory [91], the cross-entropy $H(p,q)$ of p and q measures how close a probability model q comes to the true model p of some random variable X, which is formulated as:

$$H(p,q) = -\sum_{x\in X} p(x)\log_2 q(x). \tag{1.6}$$

Perplexity

The most commonly used measure for language models is perplexity. The perplexity PL of a language model is calculated as the geometric average of the inverse probability of the words on the test data:

$$PL = (\prod_{i=1}^{t} P(w_i|h(w_i)))^{-\frac{1}{t}}, \tag{1.7}$$

where $h(w_i) = w_1w_2...w_{i-1}$. Perplexity is highly correlated with cross-entropy. It actually can be seen as exponential of entropy. Note that in most cases, the true model is unknown. Therefore perplexity can be viewed as an empirical estimate of the cross-entropy in (1.6).

Perplexity can be the measure for both the language and models. As the measure for the language, it estimates the complexity of a language [23]. When it is considered as the measure for models, it shows how close the model is to the "true" model represented by the test data. The lower the perplexity, the better the model is.

It is important to keep in mind that perplexity is not suitable for measuring language models using un-normalized probabilities. Also perplexity can not be used to compare language models that were constructed on different vocabularies. In these situations, other measures should be chosen.

Word prediction accuracy

Word prediction has applications in natural language processing, such as augmentative and alternative communication [175], spelling correction [34], word and sentence auto completion, etc. Typically word prediction provides one word or a list of words which fit the context best. This function can be realized by language models as a side product. Looking at this from the other side, word prediction accuracy provides a measure of the performance of language models [159]. Word prediction accuracy is calculated as follows:

$$WPA = \frac{C}{N}, \tag{1.8}$$

where C is the number of words that are correctly predicted. N is the total number of words in the testing.

Similar to WER, word prediction accuracy (WPA) is also correlated with perplexity. Intuitively, perplexity can be thought of as the average number of choices a language model has to make. The smaller the number of choices, the higher the word prediction accuracy is. Usually low perplexity co-occurs with a high WPA. However, there are also counterexamples in the literature [159].

Compared with perplexity, WPA has less constraints. It can be applied to measure unnormalized language models. It can also be applied to compare language models constructed from different vocabularies, which happens often in adaptive language models. Compared with the computation of WER, WPA is much easier to calculate. Where WER is speech recognizer dependent, WPA does not have extra dependencies, which makes it suitable to compare language models used in different speech recognizers, i.e. at different research sites.

Word error rate

In speech recognition, the performance of language models is also assessed by word error rate (WER), which is defined as

$$WER = \frac{S+D+I}{N}, \tag{1.9}$$

where S, D and I are the number of substitutions, deletions and insertions, respectively, when the prediction hypotheses are aligned with the ground truth according to a minimum edit distance.

WER is the measure that comes from speech recognition systems. In order to calculate a WER, a complete speech recognizer is needed. Compared with the calculation of perplexity, WER is more expensive. The WER results are noisy, because speech recognition performance also depends on the quality of acoustic models. Usually low perplexity implies low word error rate. However, this is not always true [29, 64]. Ultimately, the quality of language models must be measured by their effect on real applications. When comparing different language models on the same well constructed speech recognition systems, the WER is an informative metric.

1.2.2 N-gram language models

N-gram language models are the most well known type of language models. They approximate the probability of a word sequence as a product of conditional probabilities of the current word w_i given a history of the preceding $n-1$ words.

$$P(W = w_1, w_2 ..., w_n) = \prod_{i=1}^{n} P(w_i | w_{i-n+1}, ..., w_{i-1}), \tag{1.10}$$

where the n-gram probabilities are the relative frequencies of $w_{i-n+1}, ..., w_i$ to $w_{i-n+1}, .., w_{i-1}$ in the training data according to the maximum likelihood estimation:

$$P(w_i | w_{i-n+1}, ..., w_{i-1}) = \frac{C(w_{i-n+1}, ..., w_{i-1}, w_i)}{C(w_{i-n+1}, ..., w_{i-1})}, \tag{1.11}$$

where $C(w_{i-n+1}, ..., w_{i-1}, w_i)$ is the frequency count of the string $w_{i-n+1}, ..., w_{i-1}, w_i$ in the training data.

N-gram language models have dominated the speech recognition area for years due to their simplicity, efficiency and robustness. They are easy and efficient to train, and can be embedded into a speech recognizer. Because of the local n-gram independence assumption, they are robust to recognition distortions. However, n-gram language models are challenged by data sparseness, and by across-domain brittleness. The independence assumption they make does not completely capture word-dependencies in real world speech. We now discuss each of these issues in more detail.

Data sparseness

The number of possible different word sequences grows exponentially with the length of the n-gram chosen. For example, in a modest vocabulary with hundreds of thousands of words, there are 10^{15} different trigrams (3-grams). For this reason the n-grams seen by the language model in the test data can be mutually exclusive with the n-grams occurring in the training data. As is shown in [125], even observing all trigrams from 38 million words of newspaper articles, more than 30 percent of the trigrams in new articles from the same source are still novel. Additionally, maximum likelihood estimation suffers from the fact that only a small percentage of n-grams occur frequently and a vast majority of the observed n-grams in the training data have small counts.

The data sparseness problem is addressed by smoothing techniques [31] in n-gram language models, which include discounting, interpolation and back-off approaches. The basic idea is to reshape the probability distribution by redistributing probabilities over observed events to unseen events. Discounting techniques [68, 90] achieve this by removing the probability mass from observed events and redistributing it to unseen events. Linear interpolation techniques [67] do so by approaching high order n-grams using a combination of low order n-grams. In case that a high order n-gram is unseen, the lower order n-gram still can provide valuable information. Back-off methods use a set of component models to carry out smoothing. The most detailed model will be used, if enough data is available, otherwise a general model is used. These techniques include Katz backing-off [72], Kneser-Ney smoothing and its variants [31, 76]. Interpolated Kneser-Ney is widely acknowledged as the best technique across training data sizes, corpora types and n-gram order [59].

False independence assumption

A false assumption is made in conventional language models, which gives them only insufficient ability to model long-distance dependencies. This assumption is that the current word only depends on the previous $n-1$ words. The assumption simplifies the estimation of n-gram language models, however it also is its main disadvantage. The following simple example illustrates the problem this poses for n-gram language models.

THE DOG IN THE CORNER OF THE GARDEN BARKS.

According to the assumption of trigram language models, word 'BARKS' only depends on 'THE GARDEN'. Usually, 'THE GARDEN BARKS' does not make sense. The actual dependency we want to model is 'THE DOG BARKS'. Furthermore, the probability of the trigram 'THE GARDEN BARKS' is likely to be low in the training data. Even though this is a well formed sentence, the n-gram language model will assign it a low score.

Obviously, a trigram model is not able to model such long distance dependencies. Note that increasing the order of the n-grams, e.g., to 7-gram, does not solve the problem. The number of variants of the n-gram becomes so large that the amount of training data needed to capture such long-distant dependency is prohibitively large.

Cross-domain brittleness

As is pointed in [126], statistical language models are sensitive to the variations that occur in natural language in topic or genre of the text on which the models are trained [13]. Different domains of a language tend to evolve as relatively closed systems with different word sequence statistics. For example, the phrase ‘language models’ probably occurs more often in this thesis than in theses on politics. The brittleness effect is strong even for small variations of the data that comes from the same domain. In [125], the perplexity of a language model trained on Dow-Jones newswire is doubled when the language model is applied to the similar Associated Press newswire text from the same time period.

In the past thirty years, a great number of techniques have been proposed to address the challenges that we have just covered. In this introduction, we can not address all of them. Instead, we focus on the milestone works and the techniques that are suitable for the integration of meta-information.

1.2.3 Computation paradigm driven language models

A variety of other computation paradigms has inspired a range of other language models. This subsection provides a survey of these models with special emphasis on neural network language models, which are most relevant for this thesis.

Decision tree language models

Basically, a language modeling problem is a classification problem. Since decision trees are well-known classifiers, it is natural to apply decision trees in language modeling. For example, a decision tree was used in [10] to classify the preceding word history by asking questions about the history at every node. Actually n-gram language models can be considered as kind of decision-tree-based language model. Theoretically, an optimally constructed decision tree is at least is as good as an n-gram language model. However, in practice, a globally optimized decision tree is extremely difficult to construct. Usually a heuristic greedy approach is applied to generate a suboptimal decision tree. The practical performance of decision tree based language models in fact failed to outperform the n-gram language models [118]. Better performance can be achieved by smoothing and combining them with n-gram language models. A successful approach using an aggregation strategy is the random forest language model [170], which obtains a significant improvement over n-gram language models by linear interpolation of many randomly grown decision tree language models.

One appealing feature of decision-tree-based language models and their variants is that they provide more freedom to classify word histories than conventional n-gram language models. For example, decision-tree-based language models can be used to integrate the POS information into language models [60] by asking the question about the history at current word, “Is the last word a verb?” Morphological information, prosodic information, syntactic information and topic information have been integrated into random forest language models [153].

It should be noted that the improved performance of decision tree language models over conventional n-gram language models comes at the cost of high consumption of computer memory and computational time. The increased complexity is due not only to the decision tree growing, but also to the computation of the probability of test data.

Dynamic Bayesian networks for language models

Bayesian networks originate in artificial intelligence as a method for reasoning with uncertainty based on the formal rules of probability theory [113]. A Bayesian network represents the joint probability distribution over a set of random variables $X_1, X_2 \ldots X_N$. It consists of two parts:

1. A directed acyclic graph (DAG) G, i.e. a directed graph without any directed cycles. There exists a one to one mapping between the variables in the domain and the nodes of G, i.e. every node v_i in G represents exactly one variable X_i and every variable X_i is represented by exactly one node v_i. The directed arcs in the network represent the direct dependencies between variables. The absence of an arc between two nodes means that the variables corresponding to the nodes do not directly depend on each other.

2. A set of conditional probability distributions. A conditional probability distribution $P(X_i|Pa(X_i))$ is associated with each variable X_i. The distribution quantifies how X_i depends on $Pa(X_i)$, the set of variables represented by the parents of node v_i in G representing X_i.

The probabilities are obtained from domain experts, learned from data or a combination of both. Applying the chain rule of probability theory and the independence assumptions made by the network, we can write the joint probability distribution represented by the network in factored form as a product of the local probability distributions:

$$P(X_1, X_2,\ X_N) = \prod_{i=1}^{N} P(X_i|Pa(X_i)). \tag{1.12}$$

Inference in Bayesian networks is the process of calculating the probability of one or more random variables given some evidence, i.e., computing $P(X_Q|X_E = x_E)$ where X_Q is a set of query variables and X_E is a set of evidence variables. A number of efficient inference algorithms that exploit the independence of variables in a network exist.

Dynamic Bayesian Networks (DBNs) [39, 105] offer a concise way to model processes that evolve over time for which the number of steps is not known beforehand. A DBN can be defined by two Bayesian networks: an a priori model $P(X_1)$ and a transition model that defines how the variables at a particular time depend on the nodes at the previous time steps:

$$P(X_t|X_{t-1}) = \prod_{i=1}^{N} P(X_{i,t}|Pa(X_{i,t})), \tag{1.13}$$

were X_t is the set of variables at time t and $X_{i,t}$ is the ith variable in time step t. The parents of a node can either be in the current or in a previous time slice. Typically, first order Markov assumptions are made, i.e. the nodes in a time slice only depend on the nodes in the previous time slice.

The potential of DBNs in language modeling is that they provide an ideal framework for the construction of rich language models with additional information. As is shown in [137, 138, 165], the syntactic information, semantic relation and social background knowledge can be specified as a variable in the belief network with its network structure in a declarative way without the need for special-purpose inference routines.

However, DBNs are generalizations of n-gram models. Even though they make the construction and comparison of rich information language models easier, basically, they still suffer from the inherent problems that face n-gram language models, such as data sparseness.

Exponential language models

The exponential language model has the following form to model the conditional probability of word w_i given context h_i:

$$P(w_i|h_i) = \frac{1}{Z(h_i)} \exp([\sum_j \lambda_j f_j(h_i, w_i)]), \tag{1.14}$$

where λ_j are the parameters, $f_j(h_i, w_i)$ are arbitrary functions of the pair (h_i, w_i) and $Z(h_i)$ is a normalization factor. The $Z(h_i)$ can be calculated as follows:

$$Z(h_i) = \sum_{w_i \in V} \exp([\sum_j \lambda_j f_j(h_i, w_i)]). \tag{1.15}$$

The parameters are learned from the training data based on the Maximum Entropy principle [66]. This type of language models are also referred as Maximum entropy language models. It was first introduced into language modeling by [117]. Later, it was systematically investigated by Rosenfeld [125].

The strength of exponential language models is that they can incorporate an arbitrary knowledge source h_i. Using trigger and n-gram features, in [125] maximum entropy language models achieved significant improvement over n-gram language models in terms of perplexity and word error rate. Since then, maximum entropy language models have become one of the most promising research avenues of language modeling and have witnessed substantial success [3, 18, 30, 32, 127].

However, training a maximum entropy language model is computationally expensive, as for each word w_i, a normalization factor needs to be explicitly computed. Such a computational challenge has been addressed by [58, 127]. In [58], every word is assigned to an unique class. The prediction of a word given its history is decomposed to a prediction of the class given the history and a prediction of the word given its class and history. In [127], a whole sentence exponential language models are proposed. In these models, the normalization factor is fixed to a true constant. However, the whole sentence exponential model also brings its drawbacks. It is intractable for exact training. It has to take advantage of sampling techniques.

Neural-network-based language models

Since Y. Bengio *et al.* published the work [16], neural networks have been widely considered as the most promising technique for language modeling. Even with a small amount of data, neural network language models yield much better performance than the smoothed n-gram language models. The superior capability of neural network language models and their variants is their ability to map discrete words into a continuous space and express the joint probability of a word sequence in this continuous space.

In [16], feed-forward neural network language models were proposed, which are depicted in Figure 1.1. Each word in the vocabulary is mapped by a shared parameter matrix to a real-valued vector. The size of this real-valued vector is commonly chosen to be between 30 and 100, which is much smaller than the vocabulary size. The shared parameter matrix is referred to as the projection layer. Following the projection layer is the hidden layer, whose dimension is between 100 and 300. After the hidden layer, is the softmax output layer. The input of a feed-forward neural network language model is the previous n-1 words $w_{t-n+1},...,w_{t-1}$. The output is the conditional probabilities $p(w_t|w_{t-n+1},...,w_{t-1})$ of word w_t given its previous $n-1$ words.

An even bigger improvement boost has recently been achieved by Mikolov *et al.* [98]. In this work, it was proposed to use recurrent neural networks (RNN) [47, 122, 128, 129] in

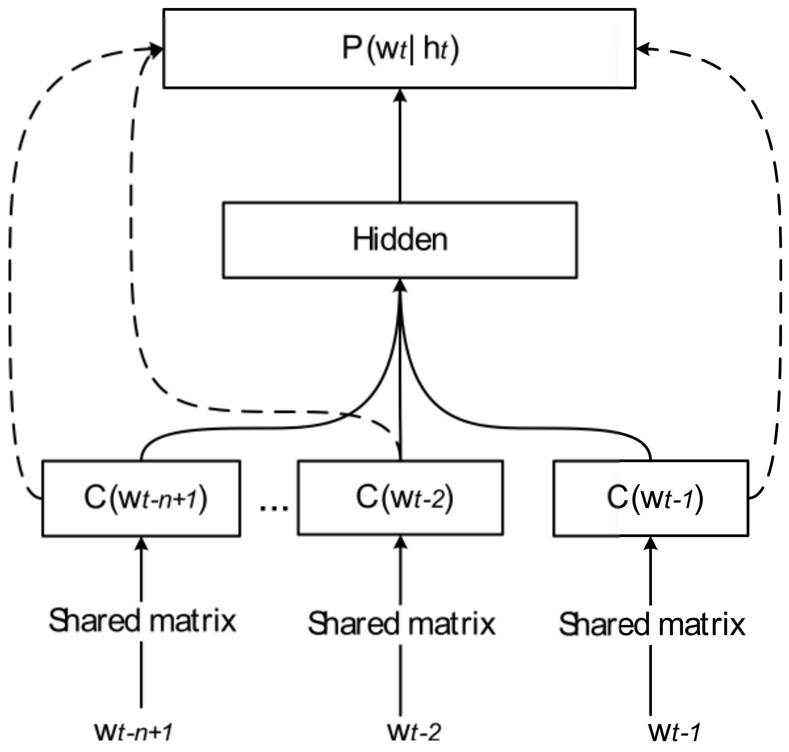


Figure 1.1: Feed-forward neural network language models. Each word depends on the previous n-gram. Each word in the n-gram is mapped to a real value vector by the shared parameter matrix. The real value vector can also be directly connected to the output layer, which is shown as the directed dashed line in the figure.

language models. In order to exploit long distance history information, the approach equips the network with a short memory. The input layer of the recurrent neural network language models (RNNLMs) is constituted by previous one word w_{t-1} and a copy of previous activated hidden layer h_{t-1}. The loop architecture in RNN theoretically can cycle an arbitrarily large amount of previous information up until the present. It also gives the RNN a deeper structure than neural networks without the loop. As is shown in [7], the performance of RNNLMs can be approached by neural network language models with more hidden layers.

In addition to their capability for generalization and for long-distance modeling, neural network language models are flexible, allowing the addition of arbitrary features. From machine learning perspective, neural networks can be seen as a set of logarithmic regressions. When additional features are integrated into neural network language models, the additional features are embedded into a continuous space, which allows the language models to generalize easily and makes them robust to noise from incorrect annotations. In [49] and [2], the contribution of syntactic or morphological information to neural network language models was investigated. Using Latent Dirichlet Allocation, topic information is also studied in [97]. The performance RNNLMs integrating syntactic features, morphological features, semantic features and social background features was studied in [140]. In this thesis, we collectively refer to all these features as meta-information. In the following chapters, we will focus on integrating these types of meta-information in language modeling.

However, the superior performance of neural-network-based language models is obtained at the cost of expensive training. This is one of the reasons why neural networks have only recently become popular in language modeling, despite the fact that they were introduced to describe language 20 years ago [47]. The high computational complexity severely constrained the early application of neural networks in language modeling. Basically, they were only applied to small amounts of data. In order to make neural networks capable of handling large amounts of data, most previous research focused on reducing the computation complexity. In [135], the output of an NNLM was constrained to a short list of most frequent words. Bengio *et al.* [15] used an adaptive importance sampling strategy to reduce the computation. Xu *et al.* [171] also used a subsampling strategy, but converted the multi-class prediction problem to a binary class prediction problem. In [104], it was proposed to use noise contrastive estimation to training NNLM. In [101], Mikolov proposed the class trick to factorize the output layer in RNNLM. The class trick was once used by [58] to reduce the computational complexity in Maximum Entropy language models. These methods already reduced the computation of the weight learning between hidden layer and output layer to less than 1%. In this thesis, we will also address the computational complexity problems by take advantage of a parallelization strategy, which will be covered in Chapter 6.

1.2.4 Meta-information

As we discussed before, state-of-the-art language models are in general based exclusively on the collection of statistics of words. These models somehow implicitly, yet blindly capture many of the phenomena of language. For example, n-grams indeed reflect many important syntactic and semantic collocations. The neural network hidden layer using abstract way to capture the similarities of different words [102].

However, current language modeling techniques still miss much potentially useful information that characterizes the language as it is. For example, language is a social product, which is reflected in the fact that in different social contexts, we probably would use different ways of organizing language to express the same idea. Language is also historical product, which is evidenced by the fact that some words appear, some words disappear, some words become shorter, and some words become longer, etc.

In this thesis, all the potential information is collectively called meta-information. We use “meta-information, since such a broad term reflects that there are still many types of information waiting to be exploited from language. The meta-information that we will discuss in this thesis is only the tip of the iceberg.

In previous research, Wiggers [164] provided a comprehensive explanation of the influence of context in automatic speech recognition. In this thesis, we will exploit some types of new meta-information as well as investigate methods to integrate meta-information into language models. The meta-information used in this thesis can be categorized according to different linguistic levels [55]:

Morphological features

In this thesis, we will exploit three types of morphological features. First, from the syntactic perspective, each word takes on a grammatical role when it occurs in a sentence, referred to as its part of speech (POS). It is the basic element of syntactic structure that constrains utterances to follow grammatical rules. Syntactic rules based words POS encode the grammatical relations among the words of a sentence, as well as their linear order and hierarchical organization. Language grammar allows humans to produce and understand sentences that they have never encountered before. Because of this impact, taking POS into account in language modeling can help to model long distance dependencies, as well as fight data sparseness. In previous studies, POS played an important role in improving the performance of language modeling and other natural language processing tasks [26, 93]

Second, from the semantic viewpoint, many different word entities bearing the same meaning, which can be derived from the same sub-word. These sub-words are called lemmas in this thesis. In previous studies, word content has been integrated into language models by taking the perspective of high level topic information [57]. Consider the fact that meaningful words must follow semantic rules to become meaningful phases and sentences. In addition to long-distance topic information, we believe that semantic rules can potentially be modeled by integrating the lemma into language models. Furthermore, the number of lemmas is smaller than vocabulary size. The usage of lemmas helps language models to overcome data sparseness problems.

Third, the lexicon level feature which we attempt to integrate into language models is word length. It initially appears to be a trivial feature, however, it actually reflects several aspects of a word. According to a law proposed by Zipf to account for natural language [116], the information can be conveyed as concisely as possible by giving the most frequently used meanings with the shortest word forms. In [8], it was confirmed that human memory span is highly related to word length across a wide range of materials. Word length also can reflect the number of syllables, number of phonemes and the POS to which the word belongs. A short word usually has fewer syllables and phonemes. A noun usually is longer than a determiner and a verb.

Sentence patterns of language

The sentence level information that we will exploit is sentence length as well as information about the words that succeed the present word in current sentence.

Language models prefer short sentences, since usually the longer a sentence is, the smaller its joint probability. For spontaneous spoken language this assumption is generally correct, but for more formal, written language it does not hold. As is shown in [164], the average sentence length varies according to different type of conversations.

People utter sentences with a plan. The present word in the sentence is not only dependent on previous word history information but also predicable by the succeeding word information in that sentence. In previous research on whole-sentence models [127], all the words in a sentence are used in language modeling as a bag-of-words. In this thesis, we use succeeding words as a source of information complementary to word-history information.

Information from the discourse level

Discourse level meta-information characterizes the relationships beyond the sentence level. Topic and situational information can be exploited on this level.

Topic information captures the semantic relationship among sentences. Within a discourse, each sentence is affected by the preceding sentences in various ways. For example, we often need to get the reference or meaning of pronouns according to the prior discourse. Prior discourse can also disambiguate words like “fox” in that the discussion may be about animal or crafty behavior.

Topic information has attracted a lot of attention from language modeling and natural language processing. Many methods such as latent semantic analysis [83, 84], Latent Dirichlet Allocation [20] etc., have been proposed to derive topic information from discourse. In language modeling, [57, 137, 166] treated topic information as a latent variable in language models. The number of topics is predetermined and the conditional probabilities of both topic given previous words and present word given present topic is trained by Expectation Maximization. In [65], a sentence level mixture model was proposed in which each component model contains the n-gram statistics of a specific topic. Taking advantage of recurrent neural networks, in this thesis (Chapter 3) we propose k-component incremental learning of topic information in language models.

Situational information is the non-linguistic environment in which a discourse happens. In this thesis, we give special consideration to information on the socio-situational setting in which speech is produced. The socio-situational setting reflects the social context of speech, which involves the communicative goals, number of speakers, number of listeners and the relationship among the speakers and the listeners. It is different from topic information, which is related to the content of the discourse. Socio-situational settings reflects the social restrictions on the discourse. An automatic classification method will be investigated in Chapter 2. The different ways of integrating the socio-situational setting into language models will be discussed in the Chapter 3 and Chapter 4.

1.2.5 Meta-information driven language models

In this subsection, we highlight several important advanced language models driven by the meta-information such as class, information from the cache window, the trigger pattern, the syntactic grammar and structure and topics. In this thesis, we will also propose discourse level meta-information driven language models, namely, *k*-component recurrent neural network language models.

Class-based language models

Class-based language models [23, 106] have the following format:

$$P(w_i|h(w_i)) = P(w_i|c_i)P(c_i|h(c_i)), \tag{1.16}$$

where $h(w_i)$ is the word history of w_i, c_i the class information of present word w_i and $h(c_i)$ the class history of c_i. The $h(c_i)$ can include the previous class information as well as the word history information.

In contrast to smoothing techniques, class-based language models battle data sparseness via mapping the words in the vocabulary onto a smaller number of classes. By exploiting similarities with sequences of words that have already see seen, successfully assigned classes can help the language model to make a reasonable prediction for a sequence of words that have not yet been seen.

The vocabulary clustering trick has also shown its potential in speeding up the training of advanced language models. The number of classes is smaller than the size of vocabulary. For this reason, class-based language models have fewer parameter than their word-based counterpart. As a result, the training of the model becomes faster and reliable. As previously mentioned, it can significantly speed up the maximum entropy model [58] and neural networks language models [101].

The quality of the class-based language models depends on the way in which the vocabulary is clustered. Much previous research has investigated the best way to cluster the vocabulary [14, 23, 106, 115, 158, 172]. Class-based language models usually benefit more from automatic generated classes than manually constructed classes [108]. It was shown in [60, 107] that class-based models can obtain decreased perplexity as well as word error rate, especially when there is only small amount of training data available.

However, from the equation 1.16, it is obvious that compared with their word-based counterparts, class-based language models actually lose information. Class-based language models simplify the dependence between words to the dependence between classes, which are less numerous than the number of words in the vocabulary. Better performance usually is achieved by combining the class-based models with word-based models. Furthermore, according to the empirical results in [59], with increased size of training data, the gain from the class-based model would vanish.

The socio-situational settings and topics from the perspective of data clustering can be viewed as specific classes. In Chapter 3, using the incremental learning, we propose an alternative way of using these types of discourse level meta-information.

Cache-based language models

In order to capture the phenomena that a word used in the recent past is much more likely to be used again sooner than predicted by its overall frequency in the vocabulary, [78] proposed a cache-based language model for speech recognition. The cache-based model is a dynamic model in which the probabilities are calculated as the relative frequency of the words within the cache. This dynamic model is further linearly interpolated with standard n-gram language models.

One advantage of cache-based model is that they can model longer term patterns, which are inadequately captured by n-gram language models. The dynamic fluctuation of the probabilities obtained from the static n-gram language model results in a significant reduction in perplexity [63, 78].

However, the perplexity reduction of the cache-based language models did not translate into the word error reduction in their application in speech recognition. As it is explained in [59], the probable reason is that the cache-based model is based on the assumption that previous words in the cache are known exactly, however, the real speech recognizer is not perfect. An incorrectly recognized word in the cache can increase the chance of the same error happening again.

Inspired by the cache-based model, there are a large number of variants that try to capture the long distance dependency. In [86] it was proposed to use the trigger-pair to capture the dependency within a sentence. $w_A \rightarrow w_B$ is a trigger pair, when a word w_A is significantly correlated with another word w_B. In other words, if word w_A occurs in the sentence, the probability of the word w_B will be increased. As the number of trigger-pairs are huge, the mutual information criterion was used. However, as it is reported in [124], almost 68% of the trigger-pairs selected according to this criterion are self-triggers, meaning that the word w_B in the trigger-pair is the same word as the word w_A. For this reason, the trigger-based models achieved only little improvement over the cache-based model.

Another important variant is to model the long-distance semantic relationships by integrating Latent Semantic Analysis in language models [12]. In latent semantic analysis, each word and document is represented by a modest size of vector, which to some degree reduces the data sparseness issue in language modeling. One important property of this vector representation is that semantically related words and documents are close in the vector space. Unlike conventional n-gram language models, which are suitable for capturing short-span patterns, language models using semantic analysis are good at modeling long-span patterns. The combination of these two methods results in multi-span language models. As it is shown in [12], multi-span language models achieve about 20% perplexity reductions, and 9% relative word error rate reductions when compared to a Katz trigram. Recently a similar approach has been proposed to combine Latent Dirichlet Allocation with language models [61], which also yielded promising results.

In Chapter 3, we will also use Latent Dirichlet Allocation to obtain latent topic information for *k*-component recurrent neural network language models. Using Latent Dirichlet Allocation, each sentence is represented by a real vector. Using the real vectors, *k*-means clustering is applied to partition the data. The basic approach underlying Chapter 3 is an approach that models long-distance dependency by exploiting data clustering.

Mixture models

As previous discussed, a language corpus may contain different topics and different styles. Standard n-gram language models are very sensitive to topic or style changes. They are not capable of modeling long-distance information. In order to capture this type of information, mixture model strategies have been applied.

In [77], a mixture of models is constructed based on the word level for k different language models, which are trained on different component of the data set. These specific models are combined as follows:

$$P(w_i|h(w_i)) = \sum_k \lambda_k P_k(w_i|h(w_i)), \tag{1.17}$$

where $P_k(w_i|h(w_i))$ is the conditional probability of specific model k. The linear interpolation weights are tuned using held-out data.

In [63, 65], sentence mixture models are proposed in which the linear interpolation of different component language models are based on the joint probabilities of each sentence.

$$P(s) = \sum_k p_k(s) = \sum_k \lambda_k \prod_i P_k(w_i|h(w_i)), \tag{1.18}$$

where $h_(w_i)$ is the history of w_i in sentence s. If the component language models are n-gram models then $h(w_i) = w_{i-n+1}, ..., w_{i-1}$.

The first step of mixture modeling is to cluster the data set according to some criterion. For data belonging to several topics or styles at the same time, soft-clustering can be applied [63].

The performance of mixture models depends on the clustering of the data set. In [63], a two-stage clustering process was used. The first stage used an agglomerative clustering method, in which a similarity measure is combined with inverse document frequencies. The second stage used an Expectation-Maximization method based on n-grams to re-estimate the clustering [63]. To address the problem that a too aggressive partitioning of the data set may aggravate the data spareness problem for component language model training, mentioned in [63], the mixture probability is interpolated with an additional general model, as follows.

$$P(s) = \sum_k \lambda_k \prod_i [\alpha_i P_k(w_i|h(w_i)) + (1-\alpha_i) p_g(w_i|h(w_i))]. \tag{1.19}$$

Their experiments show that the sentence-mixture models can achieve a more than 20% perplexity reduction and almost 4% word error rate reduction with a small number of components (5 to 10). In [59], the sentence-mixture models achieved even better results when they were used on a large number of components (up to 128).

In Chapter 3, we address the k-component mixture models. In our proposed approach, the component language models are constructed according to incremental learning based on recurrent neural networks, which can effectively deal with data sparseness. Furthermore, not only the topic but also the socio-situational setting variation is considered in our proposed mixture models.

1.3 Research Questions

In this thesis, we address language modeling with meta-information motivated by the following three assumptions.

- To use meta-information in language modeling, different meta-information prediction methods need to be investigated.

- Meta-information should be integrated into state of the art language models. For each type of meta-information a suitable integration method has to be found.
- Meta-information integration will increase computational complexity. Methods of speeding up language modeling need to be investigated.

Based on these three assumptions, in this thesis, we formulate the following research questions and their motivations.

Research Question 1 How can we develop methods that classify socio-situational settings of transcripts and that perform more accurately than human? (Chapter 2)
In the previous sections, we presented an overview of statistical language models, meta-information and meta-information driven language models. From the overview, we find that many types of meta-information have been investigated before in language modeling, such as topics and part-of-speech tags. In this thesis, we not only integrate topics and part-of-speech tags into language models using a new computational framework, but also apply meta-information from the social perspective to language models. We propose to use socio-situational settings of languages in language modeling. In practice, only text information is available for language modeling. In order to use socio-situational settings in language modeling, we need to develop methods to infer this type of meta-information from the text. This point is addressed by our *Research Question 1*. In order to measure the performance of the proposed method, the performance of human on a socio-situational setting classification task can be treated as baseline.

Research Question 2 How to effectively integrate discourse level meta-information into language modeling? (Chapter 3)
When we know the socio-situational setting, we still have to find an effective way to integrate this kind of meta-information into language models. Socio-situational settings are a form of discourse level meta-information, which characterizes the style of sequences of words. An effective method needs to be developed to integrate such kinds of discourse level meta-information into language modeling, which is addressed by *Research Question 2*.

Research Question 3 How to effectively combine recurrent neural network language models with sentence level and word level meta-information? (Chapter 4, Chapter 5)
Recently, the paradigm of recurrent neural network language models has proven its worth in two ways. Recurrent neural network language models have much better generalization capabilities than other language models. Recurrent neural network

language models also have flexible structures to include other features. As discussed in the previous section, many types of meta-information have been applied in different kinds of language models, but not yet in recurrent neural network language models.

Research Question 4 What is the effect of integrating word level meta-information into recurrent neural network language models on the performance of these models? (Chapter 5)
Even though previous investigations have deployed different types of meta-information in language modeling, a systematic analysis and comparison of different meta-information is not available yet. In order to understand the contribution of different types of meta-information to improve recurrent neural network language models, we should make a systematic comparison of different recurrent neural network language model using different types and combinations of meta-information.

Research Question 5 How to speed up the training of recurrent neural network language models? (Chapter 6)
It is well-known that training of recurrent neural network language models is computationally expensive. Although integration of meta-information into recurrent neural network language modeling provides better-performing models, it increases the computational cost. To effectively use meta-information in recurrent neural network language models, the computational cost have to be affordable.

1.4 Structure of the thesis

This section explains which chapter addresses which research question. The references of the form [shi-x] refer to the publication list in Section 1.5.

Chapter 2, based on [shi-1], addresses the socio-situational setting of languages and its automatic classification methods and describes the set-up and results of a subjective experiment to obtain and analyze the cues mentioned by humans classifying the socio-situational setting of the selected transcripts. Based on these cues, we propose a static and dynamic classification method for socio-situational settings according to language use. Especially our dynamic classification method is motivated by the need for the socio-situational setting to be integrated into the language model on the fly, i.e., as the language model is used for practice.

Chapter 3, based on [shi-2], proposes *k*-component adaptive recurrent neural network language models using curriculum learning to incorporate language models with discourse level meta-information such as topics and socio-situational settings. Basically, we emphasize sub-domain patterns in component models by scheduling the order of the training data. We use a curriculum learning method to address two challenges of adaptive language modeling, namely within domain adaptation and limited-data domain adaptation.

Chapter 4, based on [shi-3], proposes recurrent neural network tandem language models to integrate meta-information in language modeling. The proposed model has two parts: one part for meta-information prediction, the other part for integrating the predicted meta-information into recurrent neural network language models. In Chapter 4, we also present the systematic comparison of the contribution to language modeling that is made by different types of meta-information in terms of perplexity, word prediction accuracy and word error rates.

Chapter 5, based on [shi-4], proposes forward-backward recurrent neural network language models to combine succeeding words information in language modeling. Several heuristic integration methods are investigated.

Chapter 6, based on [shi-5], proposes a subsampling stochastic gradient descent parallelization algorithm for speeding up the training of recurrent neural network language models.

Chapter 7 concludes the thesis and provides insights for future work.

1.5 Publication List

The work carried out by the author as a PhD student led to the following papers.

Journal papers

[shi-1] **Yangyang Shi**, Pascal Wiggers, Catholijn M. Jonker. Classifying the Socio-Situational Settings of Transcripts of Spoken Discourses. *Speech Communication*. 55(10):988-1002, 2013. (Chapter 2. It is an extension of [shi-11] and [shi-15])

[shi-2] **Yangyang Shi**, Martha Larson, Catholijn M. Jonker. Recurrent Neural Network Language Models Adaptation with Curriculum Learning. *Computer Speech and Language*. Under review. (Chapter 3. It is an extension of [shi-6])

[shi-3] **Yangyang Shi**, Martha Larson, Joris Pelemans, Catholijn M. Jonker, Patrick Wambacq, Pascal Wiggers, Kris Demuynck. Integrating Meta-Information into Recurrent Neural Network Language Models. *Speech Communication*. Under review. (Chapter 4. It is an extension of [shi-10])

Conference papers

[shi-4] **Yangyang Shi**, Martha Larson, Catholijn M. Jonker. Exploiting the succeeding words in Recurrent Neural Network Language Models. *14th Annual Conference of the International Speech Communication Association (INTERSPEECH)*. 2013. (Chapter 5)

[shi-5] **Yangyang Shi**, Mei-Yuh Hwang, Kaisheng Yao, Martha Larson. Speed Up of Recurrent Neural Network Language Models With Sentence Independent Subsampling Stochastic Gradient Descent. *14th Annual Conference of the International Speech Communication Association (INTERSPEECH)*. 2013. (Chapter 6)

[shi-6] **Yangyang Shi**, Martha Larson, Catholijn M. Jonker. K-component Recurrent Neural Network Language Models Using curriculum Learning. *IEEE Workshop on Automatic Speech Recognition and Understanding (ASRU)* . 2013.

[shi-7] Kaisheng Yao, Geffrey Zweig, Mei-Yuh Hwang, **Yangyang Shi**, Dong Yu. Recurrent Neural Networks for Language Understanding. *14th Annual Conference of the International Speech Communication Association (INTERSPEECH)*. 2013.

[shi-8] **Yangyang Shi**, Martha Larson, Pascal Wiggers, Catholijn M. Jonker. K-component Adaptive Recurrent Neural Network Language Models. *Text, Speech and Dialogue*. 8082:311-318, 2013.

[shi-9] **Yangyang Shi**, Pascal Wiggers, Catholijn M. Jonker. Adaptive Language Modeling with A set of Domain Dependent Models. *Text, Speech and Dialogue*. 7499:472-479, 2012.

[shi-10] **Yangyang Shi**, Pascal Wiggers, Catholijn M. Jonker. Towards Recurrent Neural Networks Language Models with Linguistic and Contextual Features. *13th Annual Conference of the International Speech Communication Association (INTERSPEECH)*. 1664-1667, 2012.

[shi-11] **Yangyang Shi**, Pascal Wiggers, Catholijn M. Jonker. Dynamic Bayesian Socio-situational Setting Classification. *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*. 5081-5084, 2012.

[shi-12] **Yangyang Shi**, Martha Larson, Pascal Wiggers, Catholijn M. Jonker. MediaEval 2012 Tagging Task: Prediction based on One Best List and Confusion Networks. *MediaEval 2012*. 2012.

[shi-13] Peng Xu, **Yangyang Shi**, Martha Larson. TUD at MediaEval 2012 genre tagging task: Multi-modality video categorization with one-vs-all classifiers. *MediaEval 2012*. 2012.

[shi-14] **Yangyang Shi**, Pascal Wiggers, Catholijn M. Jonker. Combining Topic Specific Language Models. *Text, Speech and Dialogue*. 6836:99-106, 2011.

[shi-15] **Yangyang Shi**, Pascal Wiggers, Catholijn M. Jonker. Socio-Situational Setting Classification based on Language Use. *IEEE Workshop on Automatic Speech Recognition and Understanding (ASRU)*. 455-460, 2011.

[shi-16] **Yangyang Shi**, Pascal Wiggers, Catholijn M. Jonker. Language Modelling with Dynamic Bayesian Networks using Conversation Types and Part of Speech Information. *The 22nd Benelux Conference on Artificial Intelligence*. 154-161, 2010.

Chapter 2

Classifying the Socio-Situational Settings of Transcripts of Spoken Discourses[1]

2.1 Abstract

In this paper, we investigate automatic classification of the socio-situational settings of transcripts of a spoken discourse. Knowledge of the socio-situational setting can be used to search for content recorded in a particular setting or to select context-dependent models for example in speech recognition. The subjective experiment we report on in this paper shows that people correctly classify 68% the socio-situational settings. Based on the cues that participants mentioned in the experiment, we developed two types of automatic socio-situational setting classification methods; a static socio-situational setting classification method using support vector machines (S3C-SVM), and a dynamic socio-situational classification method applying dynamic Bayesian networks (S3C-DBN). Using these two methods, we developed classifiers applying various features and combinations of features. The S3C-SVM method with sentence length, function word ratio, single occurrence word ratio, part of speech (POS) and words as features results in a classification accuracy of almost 90%. Using a bigram S3C-DBN with POS tag and word features results in a dynamic classifier which can obtain nearly 89% classification accuracy. The dynamic classifiers not only

[1]This chapter is an article published in Speech Communication. Y. Shi, P. Wiggers, C. M. Jonker. Classifying the Socio-Situational Settings of Transcripts of Spoken Discourses [145]. A few supplementary remarks are provided as footnotes.

can achieve similar results as the static classifiers, but also can track the socio-situational setting while processing a transcript or conversation. On discourses with a static social situational setting, the dynamic classifiers only need the initial 25% of data to achieve a classification accuracy close to the accuracy achieved when all data of a transcript is used.

2.2 Introduction

"You shall know a word by the company it keeps" [54]. We also shall know a conversation by the situation which it is used. Language is situated. Conversations take place in a particular social context and documents are written with, among other things, a particular purpose and audience in mind. Knowledge of this socio-situational setting can greatly benefit language processing applications. For example, a search engine may only return those documents or videos that match a particular speech style. In automatic speech processing, the socio-situational setting can be used to select dedicated language models and acoustic models for that context.

The socio-situational setting can be characterized by situational features such as: communicative goals, the number of speakers participating, and the relationship between speakers and listeners. It influences the way people speak. In different settings people use different speaking styles and different words. Socio-situational setting is a concept that is related to, but different from, the concepts of topic and genre that are well-known in the literature on natural language processing.

The socio-situational setting of a spoken discourse is independent of the topic of the discourse. For example, a professor lecturing on a particular topic may place emphasis on important terms by repeating them and pronouncing them clearly. In a spontaneous conversation with one of his students about the same topic, the professor may articulate less carefully and use more informal language and when explaining the topic to a family member the technical terms might be missing altogether. Different types of spoken discourses can relate to the same topics. For example, in web search one might be looking for a lecture on Western civilization, rather than a political debate which refers to Western civilization.

The socio-situational setting is related to but different from the genre. It can be seen as an aspect of genre. However, a genre often denotes a particular set of stylistic and rhetoric elements as well as some content related aspects to classify a text for example as fiction or mystery [75]. Depending on the setting people may display differences in the acoustic and prosodic aspects of their conversations as well as in the word use [6, 82]. The socio-situational setting as we define it here relates to broad categories of spoken language use such as spontaneous face-to-face conversations, debates or reading.

In this paper, we address socio-situational setting classification by automatic classifiers as well as humans. Two types of automatic classification methods are developed based on the features from the literature on automated document classification, see e.g., [75], and features that are based on the way humans do this classification task.

To obtain information about the performance of humans in this task and about the clues in the text they use to do the classification we set up and performed an experiment. In addition to the features, the subjective experiment of socio-situational setting classification provides a baseline for the automatic classification of socio-situational settings.

Two types of socio-situational setting classification methods are presented in this paper, which are a static socio-situational setting classification method using Support Vector Machines (SVM) (which we call S3C-SVM) and a dynamic socio-situational setting classification method using Dynamic Bayesian Networks (DBN) (which we call S3C-DBN). A set of static classifiers is constructed by the S3C-SVM method using different features as well as combinations of these features. In dynamic classifiers which are developed using the S3C-DBN method, we investigate the impact to the performance of classification not only from the perspective of different features but also from the perspective of dependencies among these features.

In the static classification method, we investigate the effect of sentence length, single occurrence word ratio, function word ratio, word, POS tags, POS trigrams and their combinations on the classification results. Static classifiers need to observe the complete discourse to do classification. When the context information is applied to language modeling [65, 137], the static classification method usually separates the usage of context information in language model testing into two sequential phases: one phase for context classification, the other phase for combining the context information into language modeling. Static classifiers are unsuitable for online classification as they need the complete text to make classifications. Therefore, in addition to the static classification methods, we propose a dynamic classification method of the socio-situational settings of a spoken discourse.

The dynamic classification method developed in this paper is an online classification method that sequentially processes text of a transcript. It reevaluates classification each time a word in the transcript is observed. We investigate how much of the text information is needed to achieve an acceptable classification accuracy. For example, guessing that the conversation beginning with “In this class, we will discuss something” is a lecture can be done with confidence, and the prediction that a conversation beginning with “Hello, this is Mike speaking.”, is a spontaneous conversation by phone can also be made with confidence. Therefore, classifying the socio-situational setting of a spoken discourse is a task for which dynamic classification is highly appropriate. In fact, the results of dynamic classifiers in our experiments, reach their final classification results having processed about 25% of the text.

The dynamic classification method can benefit context based adaptive language modeling. Knowing the socio-situational setting of a spoken discourse would benefits context based adaptive language modeling, as the socio-situational setting is a form of context information. In addition, the dynamic socio-situational setting classification method introduced in this paper can make classification of socio-situational settings on the fly. Therefore, the dynamic classification method can be directly integrated into context based adaptive language models in online prediction for next word.

The paper is organized as follows. In the next section, we give an overview of related work. In Section 3, we describe the Spoken Dutch Corpus which we used as the test data set. Section 4 discusses the possible differences of discourses in terms of their socio-situational settings. Our subjective experiment is described in Section 5. Section 6 discusses the features that we extracted from the discourse transcripts. Section 7 presents the S3C-SVM classification methods and their classification results. Section 8 discusses the S3C-DBN classification method, the structure of different models and the classification results. Finally, we compare the results from the human experiment with the results of our automatic socio-situational stetting classifiers and draw conclusions.

2.3 Related work

In this section, we discuss related work in socio-situation setting classification, genre classification and the features used in genre classification. We also present some related work on Support Vector Machines (SVMs) and probabilistic classifiers.

The socio-situational setting classification is related to traditional genre classification. The fundamental problem of automatic genre classification is how to define genre. As noted by [75] and used in some studies [88, 131, 149], the genre is the way a text is created, the way it is distributed, the register of language it uses and the kind of audience it is addressed to, such as Editorial, Reportage, Research articles etc. Some research [28, 132] focus on internet-based document genre classification, in which the genre includes different types of homepages, linklists, blogs etc. In [89, 121], they use the terminology activities rather than the genre. They suppose that the choice of individual discourse is restricted by different goals. They categorize the dialogues into story-telling, planning, discussion, etc. In this paper, we propose the term of socio-situational setting which defines the social restriction of a speaker's utterances.

The genre classification can benefit practical applications. It is pointed out by [75] that by taking genre into account, parsing accuracy, part-of-speech (POS) tagging accuracy and word-sense disambiguation can be enhanced. In automatic speech recognition, language models are sensitive to genre changes, even if the changes are subtle [126]. For example, the performance of a language model trained on Dow-Jones newswire text will be seriously degraded when it is applied to the Associated Press newswire [126].

In studies on automatic genre classification of discourse, various features have been proposed. Some structural cues (such as adverb count, character count, sentence count), lexical cues ("Me" count, "Therefore" count, etc) and token cues (chars per sentence average, character per word average, etc) have been used with discriminant analysis by [71]. Their work has achieved a classification accuracy of 65% on a data set with 15 genres. In the work done by [75], the cues have been classified in four categories: structural cues (passive, topicalized sentences and counts of part-of-speech tags, etc), lexical cues (words in expressing date, title, etc), character-level cues (punctuation, separators, delimiters, etc) and derivative cues (ratios and variation measures derived from measures of lexical and character level features). Using the same data set as [71], around 78% classification accuracy has been reported by [75]. Using the frequencies of occurrence of the common words and punctuation markers of an entire written language instead of a certain training corpus, an automatic text genre detection method for restricted text has been proposed by [149]. In the work by [149], more than 97% classification accuracy has been reported on the Wall Street Journal corpus of 1989 with 4 genres. The syntactic features in ten different genres in the British national corpus have been exploited by [6]. More recently, the use of POS histograms instead of POS *n*-grams in naive Bayes models has been proposed by [51]. However, all of this previous work is based on edited text rather than on spoken discourses.

In additional to words and POS-tags, we use some simple and low computational cost features such as: sentence length, single occurrence word ratio and function word ratio in socio-situational setting classification in this paper. These features are in part inspired by the work of [161], who analyzed the type-token ratio of a speaker's utterances from the socio-situational setting perspective. The type-token ratio is the ratio of the number of different words to the number of total words in a text or speech. They show that the type token ratio of texts is influenced by topic dependence as well as socio-situational effects. Conversations containing more informal, dialogic and/or spontaneous speech typically have lower type-token ratios than formal, monologic and/or prepared conversations.

Support Vector Machines (SVMs) [36] are well suited for text classification [70]. SVMs separate the data with a functional margin, which is not dependent on the number of features. In this paper, we apply the SVMs for the static classification of socio-situational settings.

Probabilistic classifiers offer alternative approaches to classification. One important probabilistic classifier in document classification is the naive Bayesian classifier described by [85]. The naive Bayesian classifier is extended by [114] to a chain augmented naive Bayesian classifier, which can be viewed as a combination of a naive Bayesian classifier and an *n*-gram language model. In this paper, we present a dynamic Bayesian (DB) approach to socio-situational setting classification, and compare it with the static approach.

The performance of humans in a genre classification task is investigated before. In the work done by [109], they investigated that whether participants use prosodic features in discourse genre identification. However, they did not propose to take advantage of this feature in an automatic classification methods. In this paper, we investigate people's performance in socio-situational setting classification as well as the performance of the automatic classifiers using the features mentioned by the participants in their classification task.

*Table 2.1: Overview of the Spoken Dutch Corpus (*CGN*)*

components	socio-situational setting	words	discourse
comp-SC	Spontaneous conversations ('face-to-face')	2,626,172	1537
comp-IT	Interviews with teachers of Dutch	565,433	160
comp-ST	Spontaneous telephone dialogues	2,062,004	1230
comp-BN	Simulated business negotiations	136,461	67
comp-DD	Interviews/ discussions/debates	790,269	642
comp-PD	(political) Discussions/debates/ meetings	360,328	248
comp-LR	Lessons recorded in the classroom	405,409	265
comp-LS	Live (eg sports) commentaries (broadcast)	208,399	325
comp-NR	Newsreports/reportages (broadcast)	186,072	506
comp-NB	News (broadcast)	368,153	5581
comp-CC	Commentaries/columns/reviews (broadcast)	145,553	364
comp-CS	Ceremonious speeches/sermons	18,075	16
comp-LE	Lectures/seminars	140,901	78
comp-RS	Read speech	903,043	1761

2.4 The Spoken Dutch Corpus

Previous genre classification studies focus on written text. Moreover, the corpora used are not designed according to genre categories. For example, the Brown corpus needs to be manually preprocessed to eliminate some texts that do not fall unequivocally into one of the predefined genre categories [75].

In contrast, in the overall design of the Spoken Dutch Corpus (Corpus Gesproken Nederlands, CGN) [110, 111] which we use in our experiments, the principal parameter is taken

Table 2.2: Overview of the experiment samples

comp	sample	socio-situational settings	sentences	words
SC	2	Spontaneous conversations ('face-to-face')	67	574
IT	2	Interviews with teachers of Dutch	89	812
ST	2	Spontaneous telephone dialogues	65	622
BN	1	Simulated business negotiations	36	398
DD	2	Interviews/ discussions/debates	90	765
PD	2	(political) Discussions/debates/ meetings	117	1271
LR	1	Lessons recorded in the classroom	43	485
LS	1	Live (eg sports) commentaries (broadcast)	5	49
NR	1	Newsreports/reportages (broadcast)	11	114
NB	2	News (broadcast)	35	324
CC	1	Commentaries/columns/reviews (broadcast)	15	171
CS	1	Ceremonious speeches/sermons	31	314
LE	1	Lectures/seminars	38	405
RS	1	Read speech	42	315

Table 2.3: Confusion matrix of human classification on socio-situational setting

comp	SC	IT	ST	BN	DD	PD	LR	LS	NR	NB	CC	CS	LE	RS	sum	ac(%)
SC	15		5												20	75
IT	2	15	1		2										20	75
ST	10		10												20	50
BN				8	2										10	80
DD					16	1			2		1				20	80
PD						20									20	100
LR		7			1		1						1		10	10
LS								5			5				10	50
NR									6	3	1				10	60
NB									7	12	1				20	60
CC					1				4		5	1			10	50
CS												10			10	100
LE						2						5	2	1	10	20
RS														10	10	100

to be the socio-situational setting. The recordings were collected along with the socio-situational settings. Details about the construction of the CGN can be found in [110].

The CGN contains audio recordings of standard Dutch spoken by adults in Netherlands and Flanders. As shown in Table 2.1, it contains nearly 9 million words divided into 14 components that correspond to different socio-situational settings. Components from comp-SC to comp-LR contain dialogues or multilogues and the components comp-LS to comp-RS contain monologues.

We performed all experiments and analyses described below on the correct transcripts of the recordings in the CGN. As these are transcripts of spoken language they do contain ungrammaticalities, incomplete sentences, hesitations and broken-off words. To make statistics reliable, we only selected words that appeared at least three times in the whole data set. This resulted in a vocabulary of 44368 words. All other words were replaced by an out-of-vocabulary token.

2.5 Differences among discourses from varied socio-situational settings

Socio-situational settings depict the social restrictions for spoken discourses. In this section, we analyze the differences among spoken discourses with different socio-situational settings from the following aspects: the social roles and the social goal of the participants in the discourses, and the social function of the discourses.

The social role of the participants in the discourse varies for the different socio-situational settings listed in Table 2.1. From “Spontaneous conversations (’face-to-face’)” to “Lessons recorded in the classroom”, the spoken discourses are dialogues or multilogues which need the participation from at least two speakers. In the rest of the socio-situational settings, usually there is only one speaker. In dialogue or mulitlogue situations, the participation of each speaker varies according to his or her social role. For example, in “Lessons recorded in the classroom” and “Interviews with teachers of Dutch”, there usually is one dominant speaker, who speaks most of time. The others respond to the dominant speaker. However, in “Simulated business negotiations” and “(political) Discussions/debates/meetings”, usually the dominant speaker is not easy to spot. In monologues, the differences in the social roles of the participants can be reflected by their different immersion and involvement. For example, in “News (broadcast)”, the speakers usually depict the News from third-person perspective, in which the speakers have less immersion than the speakers in “Ceremonious speeches/sermons”.

The social function of the discourse can also serve as a feature to characterize different socio-situational settings. Public formal discourse is different from the private informal discourse. For example, in “News (broadcast)”, the speakers hesitate less and there are less incomplete sentences than in “Spontaneous conversations (’face-to-face’)” and “Spontaneous telephone dialogues”. For some special social events, discourses even have their own distinguishable syntactic structures and terminologies. This is for example the case in “Ceremonious speeches/sermons”. The discourses bearing the function to disseminate knowledge usually have more repetitions than others, for example, “Lessons recorded in the classroom” and “Lectures/seminars”.

The social goal of the participants also distinguishes some socio-situational settings from others. For example, in “Interviews with teachers of Dutch”, the goal determines that in most cases, there is one speaker asking questions and the other one answering the questions. In “Spontaneous conversations (’face-to-face’)” and “Spontaneous telephone dialogues”, the social goal requires the involvement from participants. So there are many interruptions.

2.6 Socio-situational setting classification by humans

To get a feeling for the difficulty of the task and for possible features for classification, we set up a small experiment to answer the following questions:

1. What is the average accuracy people obtain in socio-situational setting classification?

2. How do humans do socio-situational setting classification and what kind of cues do people mention in socio-situational setting classification?

After reading a conversation thoroughly, a participant chose one of the 14 socio-situational settings listed in Table 2.1. In addition, the participant had to answer an open question on the kind of features which could help in socio-situational setting classification. Ten participants with a Master degree or higher, were invited to do the experiment. The age of the participants ranged from 27 to 44. Seven of them were male, the rest female.

As shown in Table 2.2, in total twenty samples were selected from the CGN. In comp-SC, comp-IT, comp-ST, comp-DD, comp-PD and comp-NB, two transcripts were randomly sampled. In the rest of the components, only one sample was randomly selected. Each participant was asked to label exactly the same twenty pieces of transcripts in different orders. All the selected samples are directly used without length normalization. In this way, the classification made by participants is based on the same information as the automatic classifiers.

Table 2.3 shows the confusion matrix of human performance in socio-situational setting classification. The human prediction accuracy ranges from 30% to 75%. The average

Table 2.4: Features human reported in classification. "-" means this feature is not mentioned by participant in that classification. √ indicates the feature is mentioned. The symble "m" in the first row means the number of the speakers in the conversation is bigger than 2. In the row of "SL", "S" and "L" stand for short and long average sentence length, respectively. In the row "IS", the "C" stands for complete structure; "I" for incomplete structure. "II" means informal and interruptive, "QA" refers to the question-answer style conversation like interview, "FF" means formal and fluent. "time" column list the times people mention these features in classification

F\C	SC	IT	ST	BN	DD	PD	LR	LS	NR	NB	CC	CS	LE	RS	time
SN	2	2	2	m	2	m	2	-	1	1	-	1	1	1	51
SL	S	-	S	-	L	-	-	-	-	-	-	-	-	L	11
IS	I	C	-	I	C	-	-	-	-	C	-	C	-	C	11
SW	√	√	√	√	√	√	-	-	-	√	-	-	√	-	20
SS	-	√	-	-	√	√	-	-	-	√	√	√	√	-	24
CT	√	√	√	√	√	√	√	√	√	√	√	√	√	√	47
FM	II	QA	II	II	QA	FF	II	-	FF	FF	-	FF	II	-	31

prediction accuracy of the participants is 68%. The standard deviation is 13.75%. People identified "(political) Discussions/debates/meetings", "Ceremonious speeches/sermons" and "Read speech" with 100% accuracy. People achieved low classification accuracy on "Spontaneous telephone dialogues", "Lessons recorded in the classroom", "Live (e.g. sports) commentaries (broadcast)", "Commentaries/columns/reviews" and "Lectures/seminars". Half of the participants misclassified spontaneous telephone dialogues as spontaneous face to face dialogue. Seventy percent of the participants classified the "Lessons recorded in classroom", as an "Interview with a teachers of Dutch". In Table 2.3, we find that people could tell news related broadcasting apart from other categories (eg. spontaneous conversation), but they made mistakes in telling apart "Live (e.g. sports) commentaries (broadcast)", "News reports/reportages (broadcast)", "News (broadcast)" and "Commentaries/columns/reviews (broadcast)".

Based on the answers of the participants to the second question, we compiled a list of cues that were repeatedly mentioned.

SN gives the number of speakers involved in the conversation. For example, the speaker number of a spontaneous conversation is two, while it is one in read speech. In 51 out of 200 answers, the number of speakers is mentioned as an important cue.

SL Stands for the average sentence length in a conversation. The average sentence length is shorter in spontaneous conversation than in formal lectures or read speech. A third of the answers related to spontaneous conversations mentioned this cue.

IS Depicts whether a spoken discourse has disfluency, hesitations and incomplete structures or not. For example, in News reports or ceremonies, the discourse is well prepared and contains less hesitation, disfluency and incomplete structure than spontaneous conversations.

SW Special words or lexicons are also reported by participants in their classification. For example, some participants identify a conversation as spontaneous because it contains many short words like “ja”, “nee” , “uh” and “mm”.

SS Special sentences clearly characterize some socio-situational setting. For example, all the participants correctly identify a discourse as an “Interview/discussion/debate (broadcast)”, because they noticed a special sentence: “welkom in de studio” (welcome to the studio). In fact, 25 out 200 answers mentioned the special sentences in classifying socio-situational settings.

CT Content and topic take 23.5% of all cues mentioned by participants in our experiments. For example, in spontaneous conversation, some content reflects that speakers have visual connection with each other. In a sermon, the content is religion related.

FM Formality characterizes the way a discourse is structured. It reflects the social status of each speaker in the conversations. For example, spontaneous conversations are informal and involve many interruptions. In interviews, the conversation generally could be in a question-answer style. According to the questionnaire results, we categorize “FM” into the following 3 types: informal and interruptive (II), question-answer style (QA), formal and fluent (FF).

2.7 Language socio-situational setting classification features

Based on the results described in the previous section and on the literature mentioned in section 2, we extracted features at both the discourse level and the word level. The discourse level features are sentence length, single occurrence word ratio and function word ratio. The word level features are POS tags and words.

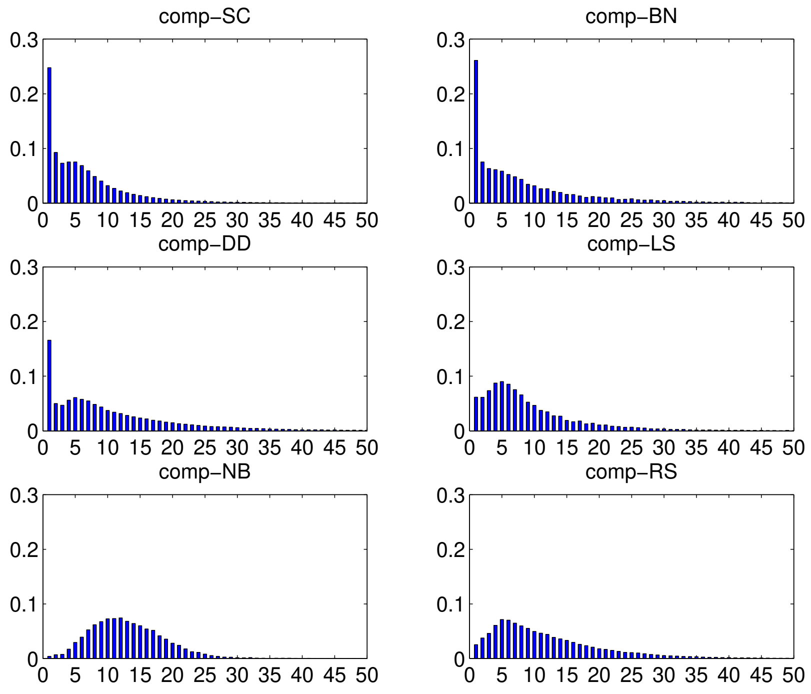


Figure 2.1: Sentence length distribution of components SC, BN, DD, LS, NB, and RS. The distribution varies among all components. Here we use components SC, BN, DD, LS, NB, and RS as examples. Horizontal direction stands for sentence length. Vertical direction stands for the probability of the sentence length in one component. Each bar represents the ratio of the number of sentences with the given length to the total number of sentences in that component.

2.7.1 Sentence length

[167] show that the sentence length (SL) distribution varies for different socio-situational settings. For example, in the CGN, for spontaneous speech (comp-SC, comp-ST) the average sentence length is below 7. In spontaneous face-to-face conversations almost 25% of the sentences contain only one word such as yes or no answers and interjections. In contrast, the means of sentence length in “(political) Discussion/debates/meetings” (comp-PD) and “Ceremonious speeches/sermons” (comp-CS), are 15 and 20, respectively. Fig 2.1 shows the sentence length distribution of 6 CGN components.

2.7.2 Single occurrence word ratio

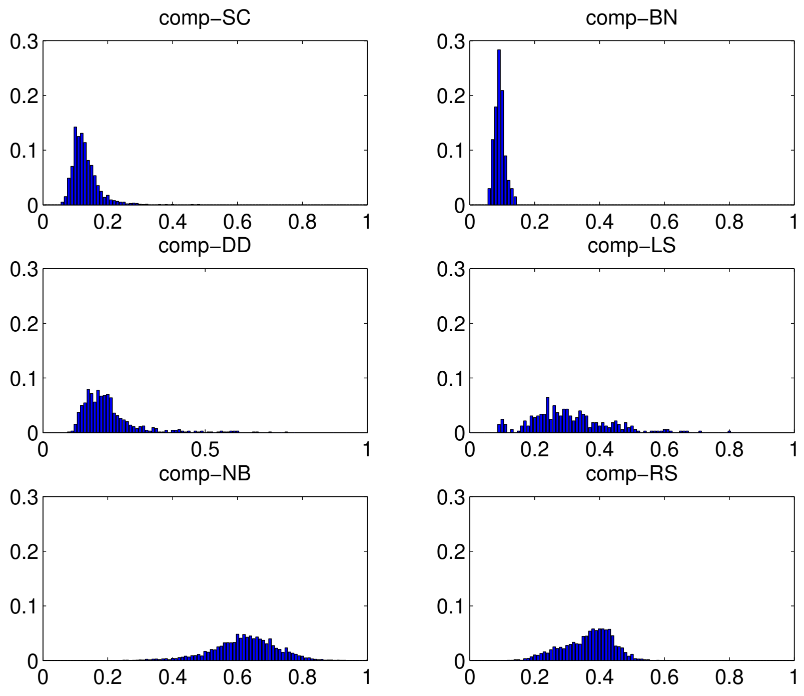


Figure 2.2: Single occurrence word ratio distribution of components SC, BN, DD, LS, NB, and RS. The distribution varies among all components. Here we use components SC, BN, DD, LS, NB, and RS as examples. Horizontal direction stands for single occurrence word ratio. Vertical direction stands for the probability of the single occurrence word ratio. Each bar represents the ratio of the number of transcripts that have the given single occurrence word ratio to the total number of transcripts in that component.

A word in the vocabulary that only appears once in a conversation is treated as a single occurrence word (SW). We calculate the single occurrence word ratio (SWR) of a discourse as the number of single occurrence words divided by the total number of words in the conversation. We find that the SWR distribution is different for different socio-situational settings. Fig. 2.2 shows some examples. In spontaneous speech (comp-SC, comp-BN), the SWR is less than for example broadcasted speech such as “(political) Discussion/debates/meetings” (comp-PD) and live commentaries and news report (comp-LS, comp-NB). Compared with other components, “News(broadcasts)” (comp-NB) uses most single occurrence words. The average SWR for news broadcasts is 0.627, while for example the SWR in business negotiations is below 0.1. Based on this analysis, we believe that the single occurrence word feature plays an important role in socio-situational setting classification.

2.7.3 Function words

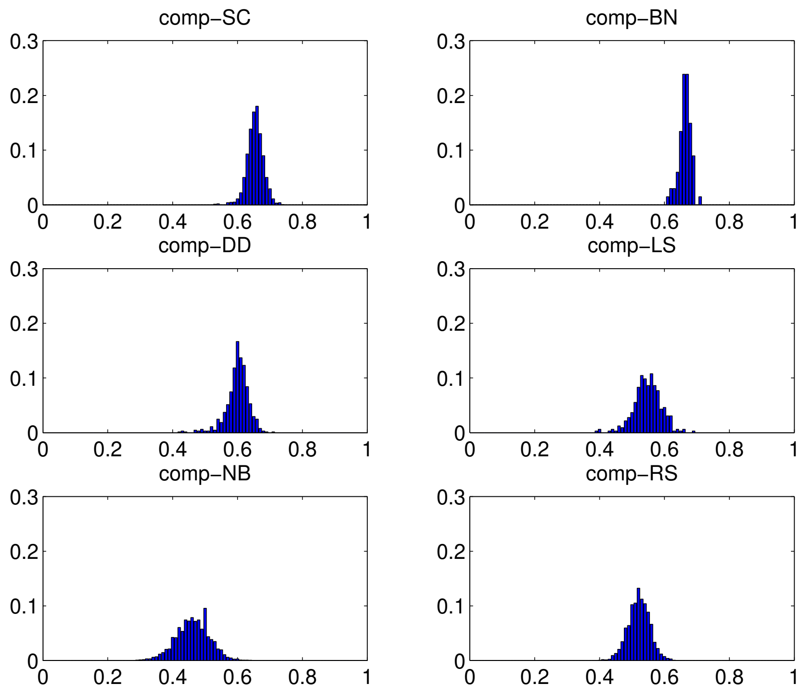


Figure 2.3: Function word ratio distribution of components SC, BN, DD, LS, NB, and RS. The distribution varies among all components. Here we use components SC, BN, DD, LS, NB, and RS as examples. Horizontal direction stands for function word ratio. Vertical direction stands for the probability of function word ratio. Each bar represents the ratio of the transcripts that have the given function word ratio to total number of transcripts in that component.

While for topic classification function words are usually removed, function words can serve as important cues in socio-situational setting classification.

Typically, the relative number of function words is higher in spontaneous speech than in more formal speech [167]. For every discourse we calculate the function word ratio as the number of function words divided by the total number of words in that discourse. Fig 2.3 shows that the CGN news broadcast component (comp-NB) has the smallest function word ratio, while business negotiations (comp-BN) have the highest average function word ratio.

Not only does the function word ratio vary over socio-situational settings, the distributions of specific function words also differ for different socio-situational settings. Fig 2.4 depicts the frequency distribution of six common function words over all components.

2.7.4 Words and POS-tags

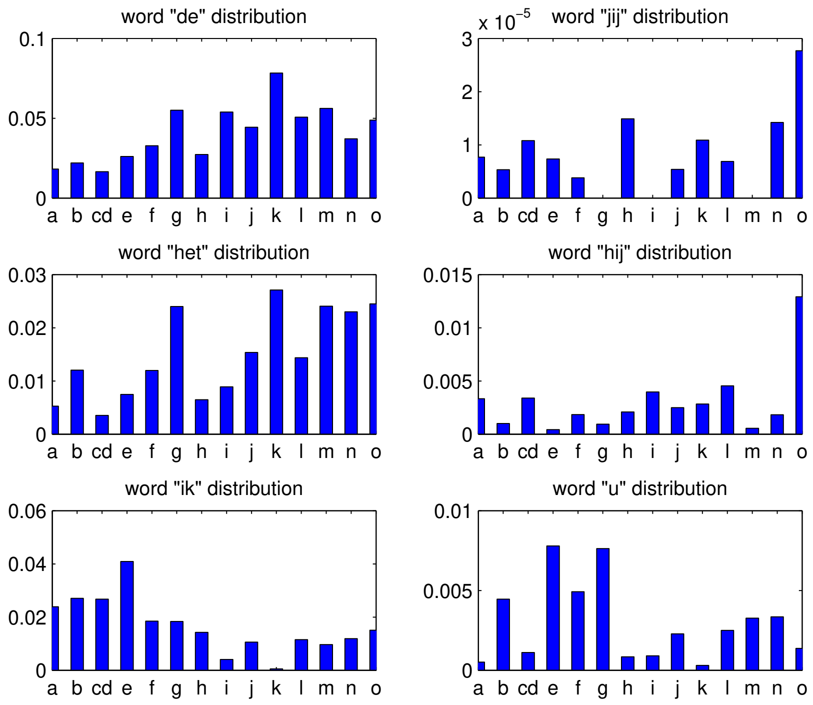


Figure 2.4: The distribution of function words "de" (the), "het" (the), "ik" (I), "jij" (you), "u" (you, formal), "hij" (he). Horizontal direction stands for components. Vertical direction stands for the frequency of the special function word in each component. In order to illustrate the distribution shape clearly, different vertical direction scales were chosen.

The choice of words is context dependent. We can capture this by using the word frequencies of all words in the vocabulary as features as is done for many text classification tasks [51, 75, 88, 131, 149].

In addition to words, part-of-speech tag frequencies also give useful information. For example, in spontaneous speech more adjectives are used on average than in formal speech, while in more formal speech more nouns are used on average [167].

Rather than using the direct frequency counts we apply a modified version of the term frequency inverse document frequency (tf-idf) metric, which is widely used in information retrieval [9], to calculate the weights of POS-tag and word features. The (tf-idf) helps to reduce the weight of common POS-tag [2] and word features which have little discriminative power and to increase the weight of rare features which have much discriminative power. The term frequency $\mathrm{tf}_{i,j}$ is the number of times term i appears in document j. The document frequency df_i is the number of documents that contain term i. Inverse document frequency $\mathrm{idf}(i)$ can be calculated by:

$$\mathrm{idf}_i = \log(\frac{N}{\mathrm{df}_i}), \tag{2.1}$$

where N is the total number of documents. The tf-idf weight is the combination of $\mathrm{tf}_{i,j}$ and idf_i.

$$\mathrm{weight}(i,j) = \begin{cases} (1+\log(\mathrm{tf}_{i,j}))\mathrm{idf}_i & \mathrm{tf}_{i,j} > 0, \\ 0 & \mathrm{tf}_{i,j} = 0. \end{cases} \tag{2.2}$$

$\mathrm{weight}(i,j)$ indicates the importance of term i in discriminating document j from other documents. To emphasize terms that are discriminative for socio-situational settings, we heuristically modify the inverse document frequency as

$$\mathrm{idf}_i = \log(\sqrt{\frac{N}{\mathrm{df}_i}\frac{S}{\mathrm{sf}_i}}), \tag{2.3}$$

where S is the total number of socio-situational settings in the CGN, sf_i represents the number of socio-situational settings that contain term i. In fact, this modification is intend to average the inverse document frequency with inverse socio-situational setting frequency in terms of logarithm value.

Based on the extracted features such as sentence length, single occurrence word ratio, function words, words and POS-tags discussed in this section, we will show two socio-situational setting classification methods in the following sections.

2.8 Static Socio-situational Setting Classification

For static socio-situational classification, we chose Support Vector Machines (SVMs) as these have shown good performance for high dimensional features spaces [155] and have successfully been applied in several text classification tasks [70, 156].

[2] POS-tags are provided with the CGN data set.

We represented each spoken discourse as a feature vector. The dimension of the vector is determined by the features used to represent the data. We experimented with several subsets of the seven features discussed above: sentence length (SL), function word ratio (FWR), function word (FW), single occurrence word ratio (SWR), POS tags, POS-trigrams and words [3] . Table 2.5 shows each of the subsets and the dimensions of the corresponding feature vectors.

Depending on the size of document vectors, different kernel functions are used in our experiment. For small feature vectors, such as feature set 1, feature set 4 and feature set 8 [4], we adopted the radial basis function (RBF)(2.4) as our kernel function:

$$K(x_i,x_j) = exp(-\gamma\|x_i - x_j\|^2), \gamma > 0. \tag{2.4}$$

For large size document vectors, we don't need to map data to a higher dimensional space, so the linear function (2.5) is applied as our kernel function:

$$K(x_i,x_j) = x_i^T x_j. \tag{2.5}$$

The classifiers using feature set 1, 3, 5 were trained with Libsvm [24] using C-SVM, the others were trained by Liblinear [50] using the L2-regularized L2-loss SVM. For small data sets such as set 1, set 5, and set 9, the grid parameter search algorithm [24] is directly applied to calculate the scale parameters and regularization weights. When dealing with large data sets, a small subset is randomly selected to calculate the parameters by the grid parameter search algorithm.

The results of these 18 SVM classifiers [5] using different feature sets, are shown in Table 2.5. The lowest prediction accuracy was obtained by only using SL, FWR and SWR features; however, these features have the lowest computational cost. The highest prediction accuracy of 89.55% is achieved by combining SL, FWR, SWR, POS and word features. This classifier also gets the best accuracy of 88.62%, when 10 fold cross validation is used.

Table 2.6 shows the confusion matrix of the best classifier in our experiments. Each column except the last one represents the label obtained from the automatic classification, each row stands for the correct label. The last column depicts the classification accuracy of the classifier on every component.

The third row in Table 2.6 shows that 32 of the conversations in comp-ST are incorrectly classified as comp-SC (while all others are classified correctly). The misclassification between comp-SC and comp-ST is not surprising, as both contain spontaneous conversations.

[3] All of the features are represented by continuous values. The dimension of POS tags and words is determined by original CGN data.

[4] Erratum: "feature set 8" should be "feature set 9".

[5] In this experiment, we randomly selected 80% of the CGN data for training, 10% for cross validation, the rest 10% for testing. All the classifiers are trained and tested on the this setting.

Table 2.5: Selected feature sets and their related classifiers prediction accuracy. 'C/γ' refers to the penalty parameter C and the kernel parameter γ. Both C and γ can be represented as exponentiation 2^n. In the table, we show the power n of C and γ. The 'dim' stands for dimension and 'ac' column for the prediction accuracy of the SVM *classifiers on the test data. The 'cv ac ' gives the 10 folders cross validation accuracy of the* SVM *classifiers.*

set	features	dim	C/γ	ac(%)	cv ac(%)
1	POS	326	5/-5	87.20	86.74
2	words	44,368	-3/0	82.45	81.08
3	FW	2,026	-2/0	83.65	85.25
4	POS-trigrams	8,466	-3/0	80.80	83.74
5	SL, FWR, SWR	4	7/3	74.05	74.48
6	SL, FWR, SWR and FW	2,030	-4/0	87.15	86.83
7	POS and FW	2,352	-1/0	86.15	87.58
8	POS and words	44,694	-1/0	88.85	88.56
9	SL, FWR, SWR and POS	330	1/-3	87.85	87.11
10	SL, FWR, SWR, FW and POS	2,356	-3/0	85.00	84.91
11	SL, FWR, SWR and word	44,372	-3/0	87.40	88.02
12	SL, FWR, SWR, FW and POS-trigrams	10,496	-5/0	85.40	86.72
13	SL, FWR, SWR, and POS-trigrams	8,470	-3/0	84.45	85.35
14	SL, FWR, SWR, POS-trigrams and words	52,838	-4/0	86.15	87.04
15	FW and POS-trigram	10,492	-1/0	83.10	86.87
16	POS-trigrams and words	52,834	-2/0	86.25	85.82
17	POS and POS-trigrams	8,792	-3/0	82.70	85.83
18	SL, FWR, SWR, POS and words	44,700	-3/0	89.55	88.62

The only difference is that comp-SC is face-to-face, while comp-ST is by telephone. As is discussed earlier, we found the same confusion for human classification.

We can also see in Table 2.6 that comp-IT and comp-BN are 100% correctly classified by our classifier. Component comp-CC has the lowest accuracy. It is confused most often with comp-NR and comp-NB – which are also confused with each other several times. All these three components contain news related broadcasts. The low accuracy of comp-CS most likely indicates that this component contains too little data to train a reliable classifier.

Fundamentally the performance of the automatic classification is jointly determined by the training data size as well as the distinguishability of corresponding components. In general, the classifier can get better accuracy with more data to train on. A Large training

Table 2.6: Confusion matrix of type 11 classifier [6].

comp	SC	IT	ST	BN	DD	PD	LR	LS	NR	NB	CC	CS	LE	RS	ac(%)
SC	209		6		5		1		2						93.72
IT		32													100.00
ST	32		166												83.84
BN				16											100.00
DD	3				78	1	1		10	2				2	80.41
PD	1				1	35				1				1	89.74
LR	2		1		8		37		1					1	74.00
LS					1			46	3	4	1				83.64
NR	1				12	1		2	43	23	9			2	46.24
NB					1				6	874	1			2	98.87
CC					4	2		4	15	15	6			8	11.11
CS						1					1	1			33.33
LE	1				1								8		80.00
RS										4	2			240	97.56

data set can improve the classification accuracy. For the distinguishable components, our results seem to show that even a small data set is sufficient for training a good classifier. For example, on the relatively small components such as comp-IT and comp-BN, the classifier actually obtains 100% accuracy. However, when the training data becomes too small, the distinguishability of the specific component is easy to be ignored by the automatic classification. For example, all the automatic classification methods get low classification accuracy on comp-CS, even though this component is obviously different from other components from a human's perspective.

2.9 Dynamic Bayesian document classification

The static classification method treats each document as a whole. For applications such as adaptive language modeling, this is not desirable. Therefore, we also investigate a dynamic classification method of socio-situational setting using dynamic Bayesian Networks (S3C-DBN). This method updates the classification result for each word that is observed. Before introducing the classifier, we briefly discuss DBNs.

[6] Erratum: "type 11 classifier" should be "type 18 classifier that using SL, FWR, SWR, POS and words".

2.9.1 Dynamic Bayesian networks

Bayesian networks are methods for reasoning with uncertainty based on Bayes rule of probability theory [113]. A Bayesian network represents the joint probability distribution over a set of random variables $\mathbf{X} = X^1, X^2 \ldots X^N$. It consists of two parts:

1. A directed acyclic graph (DAG) G. The variables X^i in the domain X are mapped one-to-one to the nodes v^i of G. The directed arcs in the network represent the direct dependencies between variables. The absence of an arc between two nodes means that the variables corresponding to the nodes do not directly depend on each other.

2. A set of conditional probability distributions (CPD). With each variable X^i a conditional probability distribution $P(X^i|Pa(X^i))$ is associated, which quantifies how X^i depends on $Pa(X^i)$, the set of variables represented by the parents of the node v^i in G representing X^i.

Dynamic Bayesian networks (DBNs) [39, 105] are an extension of Bayesian networks. They can model probability distributions of semi-infinite sequences of variables that evolve over time. A DBN can be represented by two Bayesian networks: an a priori model $P(X_1)$ and a two slice temporal Bayesian network which defines the dependence between the variables at a particular step and the variables at the previous time step:

$$P(X_t|X_{t-1}) = \prod_{i=1}^{N} P(X_t^i|Pa(X_t^i)) \tag{2.6}$$

where $\mathbf{X}_t$ is the set of random variables at time t and X_t^i is the ith variable at time step t. $Pa(X_t^i)$ are the parents of X_t^i.

2.9.2 Dynamic Bayesian document classifier

As before, we classify discourses based on their lexical transcripts. This can be seen as document classification, which maps a document d to one of a set of predefined classes $\mathbf{C} = \{c^1, c^2, ..., c^n\}$. In this paper, 18 different DB classification models are implemented. Words, POS-tags and sentence length and their combinations are used as features without calculating TF-IDF scores of these features.

Unigram DB classification

Fig 2.5 shows the graphical structure of the interpolated unigram DB classification model [7]. The interpolated conditional probability of words in the unigram DB classification method

[7]Fig 2.5 represent the interpolated unigram DB classification method using word, POS-tag and SL. Other models using a subset of features are subgraphs of Fig 2.5.

is:

$$P_{int}(w_t|c_t) = \lambda_1 P(w_t) + \lambda_2 P(w_t|c_t). \tag{2.7}$$

In case of using the combination of words and POS-tags, the interpolated probability is:

$$\begin{aligned} P_{int}(w_t|pos_t, c_t) &= \lambda_1 P(w_t) + \lambda_2 P(w_t|c_t) \\ &+ \lambda_3 P(w_t|pos_t) + \lambda_4 P(w_t|, pos_t, c_t). \end{aligned} \tag{2.8}$$

L in Fig 2.5 represents the current word position in a sentence. Together with the end of sentence node E, it reflects the sentence length. The relation between C, L and E indicates that different socio-situational settings have different sentence length distributions. The interpolation method is also applied in computation of the conditional probabilities for L_t and E_t:

$$\begin{aligned} P_{int}(l_t|l_{t-1}, e_{t-1}) &= \alpha_1 P(l_t|l_{t-1}) \\ &+ \alpha_2 P(l_t|e_{t-1}) + \alpha_3 P(l_t), \end{aligned} \tag{2.9}$$

$$\begin{aligned} P_{int}(e_t|l_t, p_t, c_t, w_t) &= \beta_1 P(e_t|l_t) + \beta_2 P(e_t|p_t) \\ &+ \beta_3 P(e_t|c_t) + \beta_4 P(e_t|w_t) + \beta_5 P(e_t). \end{aligned} \tag{2.10}$$

Bigram DB classification

The bigram DB classification using the combination of word, POS and sentence length, is depicted in Fig 2.6. These models assume a 1-order Markov chain. The models which only use some of these features are sub-graphs of Fig 2.6.

For bigram DB classification only using words or POS tags, the features at a particular time step t only depend on the features at $t-1$ and the current hidden variable c_t. For example, the following equation (2.11) gives the interpolated conditional probability of w in the bigram DB classification model which only considers the word feature:

$$\begin{aligned} P_{int}(w_t|w_{t-1}, c_t) &= \lambda_1 P(w_t|w_{t-1}, c_t) \\ &+ \lambda_2 P(w_t|w_{t-1}, c_t) + \lambda_3 P(w_t). \end{aligned} \tag{2.11}$$

For bigram DB classification models using both word and POS features, the current word w_t depends on the previous word w_{t-1} as well as the current p_t and socio-situational setting c_t. The interpolated conditional probability of current word w_t is calculated by:

$$\begin{aligned} P_{int}(w_t|w_{t-1}, p_t, c_t) &= \lambda_1 P(w_t) + \lambda_2 P(w_t|c_t) \\ &+ \lambda_3 P(w_t|w_{t-1}, c_t) + \lambda_4 P(w_t|p_t) \\ &+ \lambda_5 P(w_t|w_{t-1}, p_t) + \lambda_6 P(w_t|w_{t-1}, p_t, c_t). \end{aligned} \tag{2.12}$$

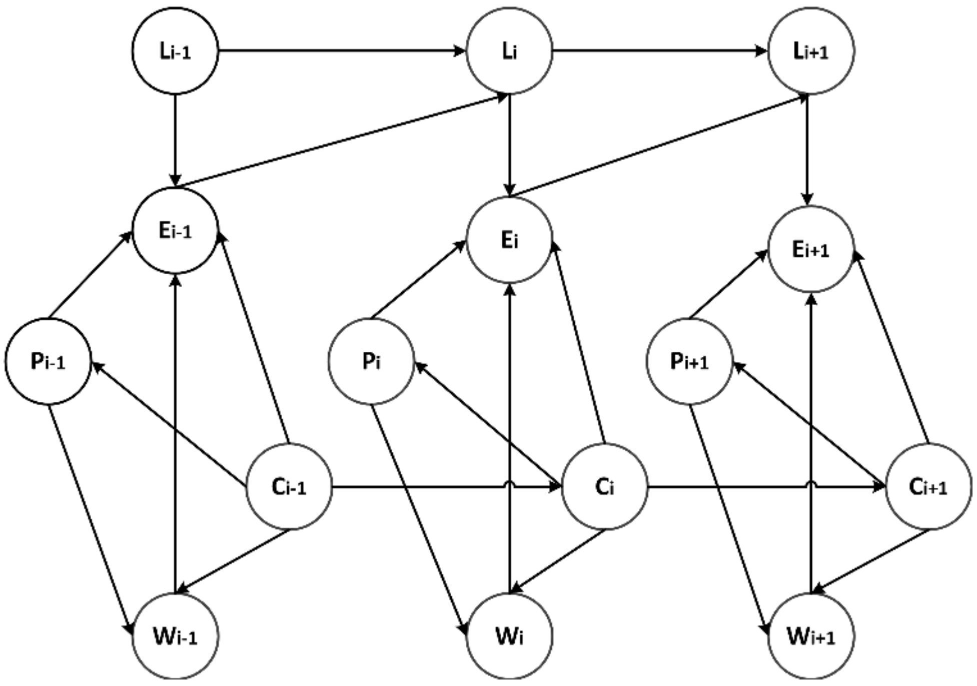


Figure 2.5: Unigram DB *classification model with words,* POS*-tags and sentence length. W_i, P_i, C_i and L_i stand for current word,* POS*-tags, classification label and sentence length, respectively. E_i indicates that whether current word is the end of a sentence* [8].

Trigram DB classification

A 2nd order Markov chain is applied in these models. Fig 2.7 shows the trigram DB classification model using word, POS and sentence length features. The word and POS features in this case depend on the features of the previous two time slices.

The conditional probability of the current word w_t given w_{t-1}, w_{t-2} and c_t is given by (2.13).

$$P_{int}(w_t|w_{t-1},w_{t-2},c_t) = \lambda_1 P(w_t|w_{t-1},w_{t-2},c_t) \\ + \lambda_2 P(w_t|w_{t-1},c_t) + \lambda_3 P(w_t|c_t) + \lambda_4 P(w_t). \tag{2.13}$$

In the trigram DB model combining word and POS features, the POS-tags are conditioned on the previous two POS-tags and the current socio-situational setting. The conditional

[8]Sentence length actually is current word position in a sentence.

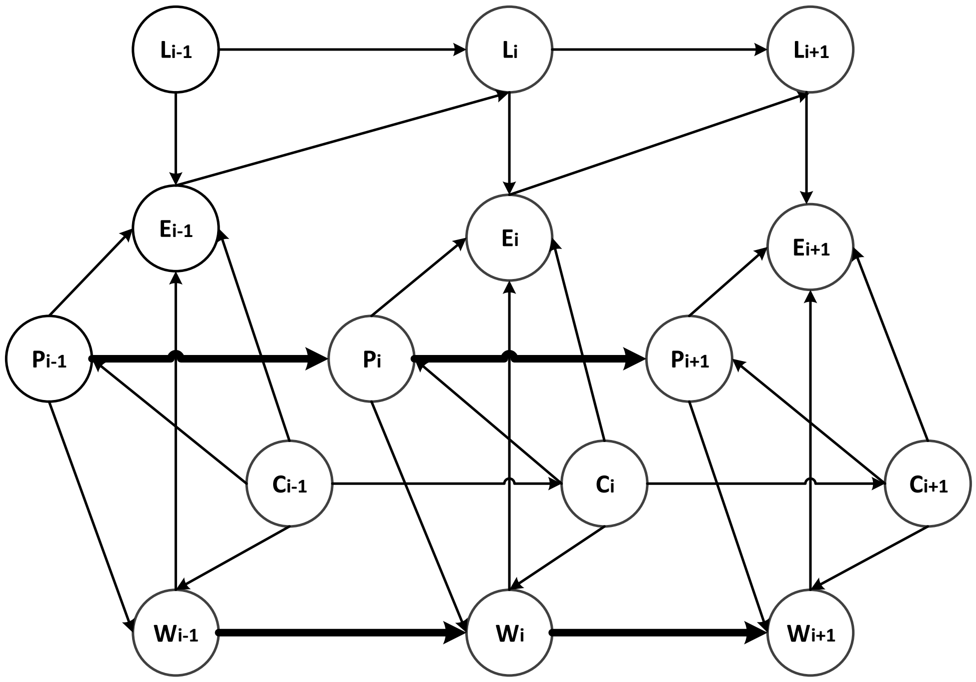


Figure 2.6: Bigram DB *classification model with words,* POS*-tags and sentence length.* W_i*,* P_i*,* C_i *and* L_i *stand for current word,* POS*-tags, classification label and sentence length, respectively.* E_i *indicates that whether current word is the end of a sentence* [9]*.*

probability of POS can be calculated using equation (2.13). The w_t in this case, depends on w_{t-1}, w_{t-2}, as well as on the current pos_t and socio-situational setting class label c_t. The following equation (2.14) gives the interpolation of the conditional probability of w_t:

$$\begin{aligned} & P_{int}(w_t|w_{t-1},w_{t-2},pos_t,c_t) = \lambda_1 P(w_t) \\ & + \lambda_2 P(w_t|c_t) + \lambda_3 P(w_t|w_{t-1},c_t) + \lambda_4 P(w_t|pos_t) \\ & + \lambda_5 P(w_t|w_{t-1},w_{t-2},c_t) + \lambda_6 P(w_t|w_{t-1},pos_t) \\ & + \lambda_7 P(w_t|w_{t-1},w_{t-2},pos_t) \\ & + \lambda_8 P(w_t|w_{t-1},w_{t-2},pos_t,c_t). \end{aligned} \tag{2.14}$$

In the equations (2.9)-(2.14), all interpolated parameters are treated as hidden variables in DB models. These parameters are trained on the held-out development set[10].

[9]Extra arcs are highlighted in the figure.

[10]The held-out development data is described in Section 2.9.3.

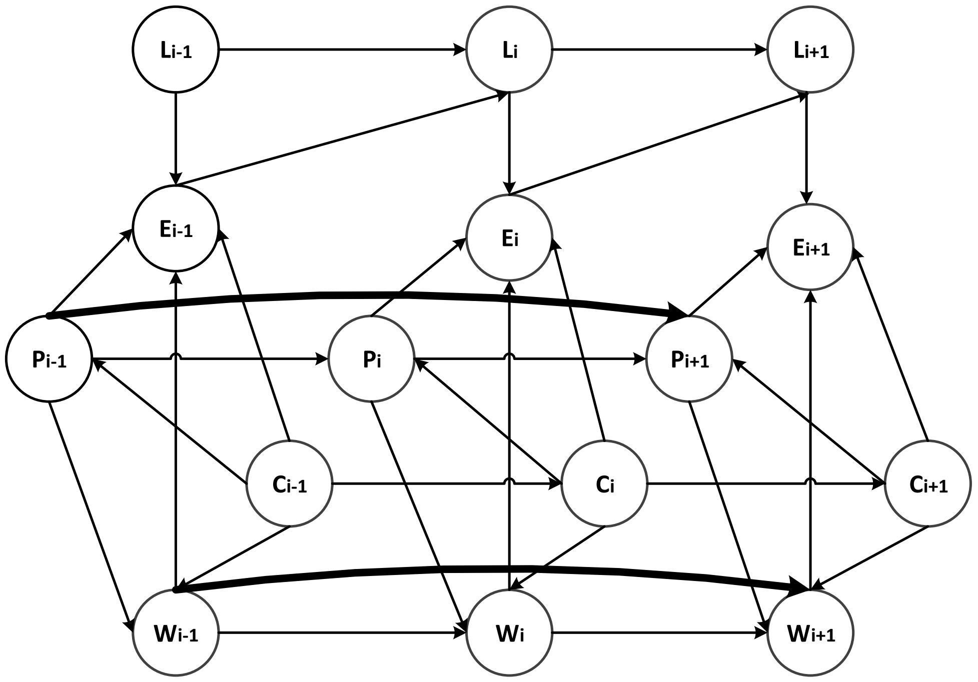


Figure 2.7: Trigram DB *classification model with words,* POS*-tags and sentence length.* W_i*,* P_i*,* C_i *and* L_i *stand for current word,* POS*-tags, classification label and sentence length, respectively.* E_i *indicates that whether current word is the end of a sentence* [11]*.*

2.9.3 Experiment

To test the DB classifiers we once again used the CGN data set. Of the data set 80% was randomly selected as training data, 10% was selected as developing data, and the remaining 10% was treated as testing data.

Fig 2.8 and 2.9 show the prediction accuracy of the 18 classifiers as a function of the percentage of test data observed. The exact classification accuracies of the 18 classifiers with 25, 50 and 100 percent data are listed in Table 2.7. In terms of overall performance, the DB classifier using POS-tag and word bigrams, which achieves a classification accuracy of 88.71%, performs best among the 18 classifiers. Its confusion matrix is shown in Table 2.8. As is indicated in Figure 2.8 and Figure 2.9, the classification accuracy increases rapidly for the first 20% of the data, then flattens. The DB classifiers using only words are more stable and precise than the classifiers that use only POS-tags. Based on 1% of the information, both trigram and bigram DB classifiers using words can correctly classify 70% the discourses,

[11] Extra arcs are highlighted in the figure.

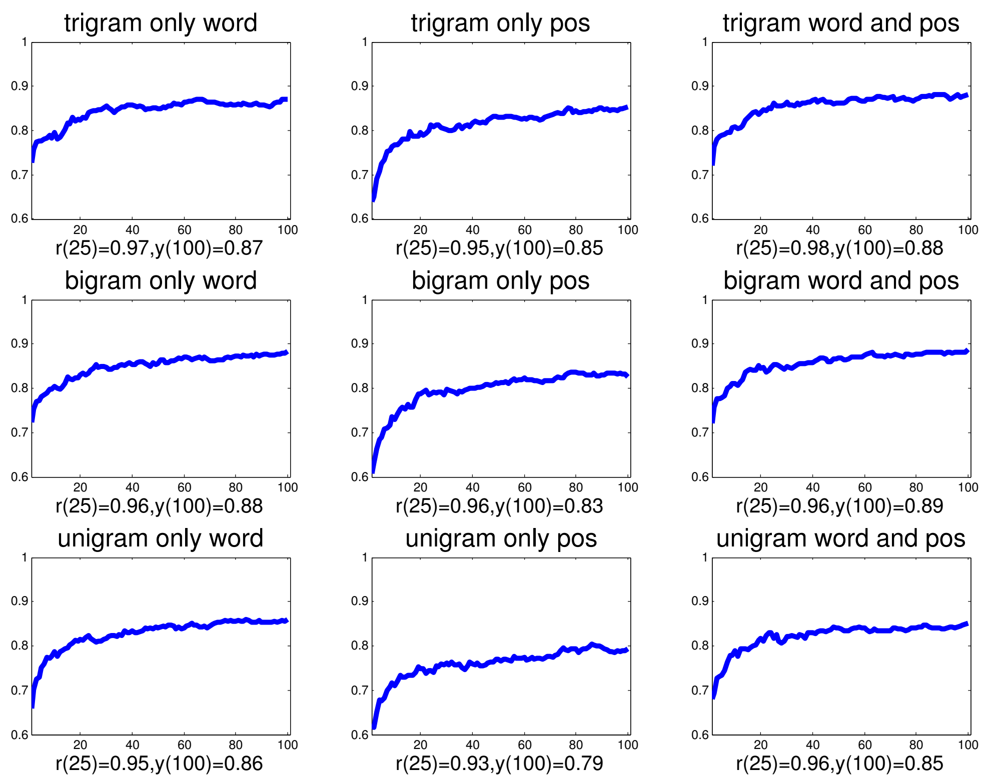


Figure 2.8: Classification accuracy trend over percent of each conversation, x,y axis represent the percentage of a conversation and prediction accuracy, respectively. $y(100)$ *represent prediction accuracy using* 100% *information,* $r(25) = y(25)/y(100)$

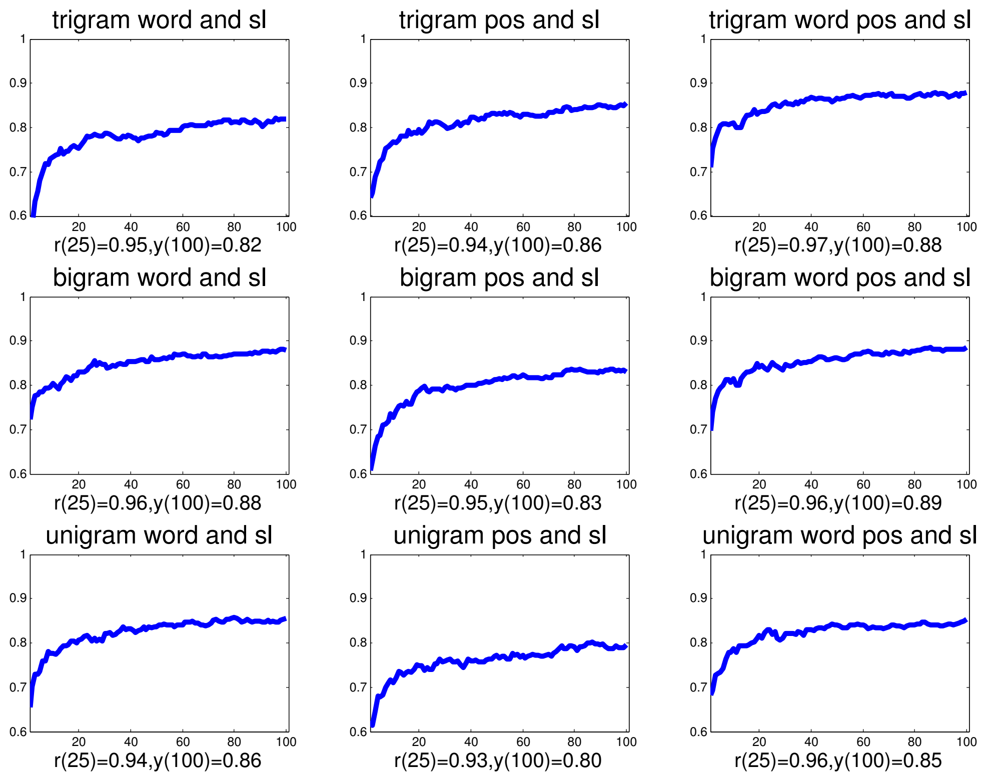


Figure 2.9: Classification accuracy trend over percent of each conversation, x,y axis represent the percentage of a conversation and prediction accuracy, respectively. $y(100)$ *represent prediction accuracy using* 100% *information,* $r(25) = y(25)/y(100)$

Table 2.7: The prediction accuracy of 18 dynamic classifiers

models	information	prediction accuracy		
		25%data	50%data	100%data
Trigram	word	84.33%	84.80%	86.83%
	POS	80.72%	82.92%	85.11%
	word, POS	86.05%	86.05%	87.93%
	word, sl	78.06%	78.84%	81.82%
	POS, sl	80.72%	82.76%	85.11%
	word, POS, sl	84.95%	86.52%	87.77%
Bigram	word	84.33%	85.42%	88.24%
	POS	79.15%	80.88%	82.45%
	word, POS	85.27%	86.68%	88.71%
	word, sl	84.33%	85.74%	87.77%
	POS, sl	79.15%	81.19%	82.76%
	word, POS, sl	84.95%	86.05%	88.40%
Unigram	word	81.19%	84.01%	85.89%
	POS	73.82%	76.80%	79.31%
	word, POS	81.66%	83.86%	84.95%
	word, sl	80.41%	83.54%	85.42%
	POS, sl	73.82%	77.12%	79.47%
	word, POS, sl	81.82%	83.86%	85.11%

while systems that use only POS-tags achieve less than 65% accuracy.

In this section, we show and compare the 18 dynamic socio-situational setting classifiers. In the following section, we discuss the relationship among the static classification, dynamic classification and human classification.

2.10 Discussion

Comparing the confusion matrices of the static, dynamic, and human classification we find three similarities and three differences. The similarities are:

1. The confusion between spontaneous face to face dialogue (comp-SC) and spontaneous telephone dialogue (comp-ST) is the main cause of misclassification in both components. In both components, the spoken discourses have many short ungrammatical sentences, repetitions and repairs. People usually use fewer determiners in spontaneous conversations than in read speech.

Table 2.8: Confusion matrix of bigram DB *classifier with word and* POS

comp	SC	IT	ST	BN	DD	PD	LR	LS	NR	NB	CC	CS	LE	RS	ac (%)
SC	72	2	11												84.71
IT		8													100.00
ST	8		55		1									0	85.94
BN				1										0	100.00
DD	3		1		29	1									85.29
PD	1					13									92.86
LR	1						11								91.67
LS	1							19							95.00
NR	5				6	1			5	4				1	22.73
NB					1			1		264				3	98.14
CC	1		2		7	1				2	5			5	21.74
CS						1									0.00
LE					1								4	2	57.14
RS														80	100.00

2. In all experiments, “Read speech” (comp-RS) is classified with high accuracy. Both human and dynamic classifiers can correctly classify all the “Read speech” (comp-RS). The static classification method can correctly classify more than 97% “Read speech” (comp-RS).

3. The sub-matrix of comp-LS, comp-NR, comp-NB and comp-CC of each confusion matrix has relative high density. There are non-zero values on non-diagonal elements. In particular, in the human based experiment and static classification method, the misclassification of these four news related components is caused by the confusion with each other.

The differences are:

1. In classifying lectures in the classroom (comp-LR), humans performed much worse than the static classifiers and the dynamic classifiers. In the confusion matrix in Table 2.3, seven out of ten people mistook the lectures in the classroom to be the interview with a Dutch teacher. Even though participants knew that the content of the conversations was about teaching, most of them were still misled by the question/answer style between teacher and one student.

2. In classifying interviews with teachers (comp-IT), both static and dynamic classification methods got 100% classification accuracy. But in the subjective experiment,

participants only achieve a 75% classification accuracy. Table 2.3 shows that some participants categorized the interview as a spontaneous conversation.

3. The participants got 100% accuracy in classifying ceremonious speech/sermons, but both the static and dynamic classification method do not perform well in these cases. The reason probably is that there is limited training data in these components of the CGN[12].

2.11 Conclusion

This paper studies the classification of socio-situational setting of a spoken discourse based on word level transcripts by humans and by automatic classification methods. The differences among socio-situational settings of discourses are discussed from the following perspectives: the social role and the social goals of the participants in the discourse, and the social function of the discourse.

In order to get a baseline for socio-situational setting classification, a subjective experiment was performed in which participants were asked to classify the socio-situational settings of discourses. The experimental results show that people can correctly classify 68% of the socio-situational settings. Inspired by the features mentioned by the participants, we extracted the average sentence length, the single occurrence word ratio and the function word ratio as features on the discourse level and TF-IDF counts of words, POS tags, POS-trigrams and function words as features on the word level.

A static S3C-SVM classification method was constructed with these features. The experiments on the static classifiers show that a combination of discourse level features and word level features performed best with a classification accuracy of almost 90%.

In addition to the static S3C-SVM classification method, a S3C-DBN method was proposed, which can achieve similar classification accuracy as the S3C-SVM method, but also can make socio-situational setting classification on a word-by-word basis. We experimented with 18 different S3C-DBN classifiers. In particular, the best S3C-DBN classifier we developed was the bigram DB classifier using word and POS tags which obtained a classification accuracy of almost 89%.

Both the static and the dynamic classifiers can be applied to provide the context information for language models. When the static classification methods are applied, the usage of the socio-situational setting information in the language models needs two phases, one phase for obtaining the socio-situational setting information by classifying the discourses,

[12]One difference is that humans actually use other prior knowledge they learned in their life. However, for machine, the data is constrained to the training data.

the other phase for applying the information in language modeling. The advantage of the dynamic classifiers is that they can provide online classification results to word level language models. The experimental results show that all the S3C-DBN classifiers using the initial 25% of the text in the transcripts can get at least 93% of the accuracy which they achieved on the complete transcripts.

In comparison, both the static and the dynamic classifiers outperform the human participants. Our experiments show that some socio-situational settings, such as “read speech”, are easy to identify, as both humans and all automated classifiers we developed scored 100% accuracy on these discourses.

2.12 Acknowledgement

The authors would like to thank the anonymous reviewers for their excellent comments.

Chapter 3

Recurrent Neural Network Language Model Adaptation with Curriculum Learning[1]

3.1 Abstract

This paper addresses the issue of language model adaptation for Recurrent Neural Network Language Models (RNNLMs), which have recently emerged as a state-of-the-art method for language modeling in the area of speech recognition. Curriculum learning is an established machine learning approach that achieves better models by applying a curriculum, i.e., a well-planned ordering of the training data, during the learning process. Our contribution is to demonstrate the importance of curriculum learning methods for adapting RNNLMs and to provide key insights on how it should be applied. RNNLMs model language in a continuous space and can theoretically exploit word-dependency information over arbitrarily long distances. These characteristics give RNNLMs the ability to learn patterns robustly with relatively little training data, implying that they are well suited for adaptation. In this paper, we focus on two related challenges facing language models: *within-domain adaptation* and *limited-data within-domain adaptation*. We propose three types of curricula that start with general data, i.e., characterizing the domain as a whole, and move towards specific data, i.e., characterizing the sub-domain targeted for adaptation. Effectively, these curricula result in a model that can be considered to represent an implicit interpolation between general data

[1]This chapter has been submited to Computer Speech and Language. Y. Shi, M. Larson, C. M. Jonker. Recurrent Neural Network Language Model Adaptation with Curriculum Learning

and sub-domain-specific data. We carry out an extensive set of experiments that investigates how adapting RNNLMs using curriculum learning can improve their performance.

Our first set of experiments addresses the within-domain adaptation challenge, i.e., creating models which are adapted to specific sub-domains that are part of a larger, heterogeneous domain of speech data. Under this challenge, all training data is available to the system at the time when the language model is trained. First, we demonstrate that curriculum learning can be used to create effective sub-domain-adapted RNNLMs. Second, we show that a combination of sub-domain-adapted RNNLMs can be used if the sub-domain of the target data is unknown at test time. Third, we explore the potential of applying combinations of sub-domain-adapted RNNLMs to data for which sub-domain information is unknown at training time and must be inferred.

Our second set of experiments addresses limited-data within-domain adaptation, i.e., adapting an existing model trained on a large set of data using a smaller amount of data from the target sub-domain. Under this challenge, data from the target sub-domain is not available at the time when the language model is trained, but rather becomes available little by little over time. We demonstrate that the implicit interpolation carried out by applying curriculum learning methods to RNNLMs outperforms conventional interpolation and has the potential to make more of less adaptation data.

3.2 Introduction

The task of statistical language models is to judge whether a sequence of words is well-formed or not. Conventional n-gram language models factorize the joint probabilities of all the words in a sequence into a product of probabilities of each word given information about its history, i.e., the preceding $n-1$ words. By using word histories, n-gram language models capture local regularities of languages. However, n-gram language models can only exploit an n-gram if the exact string of n words is present in the training data. As n grows large, the chance that an n-gram seen in the target data was also present in the training data falls off sharply. For this reason, conventional n-gram language models easily suffer from data sparseness. In practice, the history length $n-1$ that can be effectively exploited is quite limited. For this reason, n-gram language models lack adequate means to model long-distance dependencies.

These known shortcomings are addressed by Recurrent Neural Network Language Models (RNNLMs). Recently, RNNLMs have been demonstrated to outperform n-gram language models for speech recognition [98]. Their superior capabilities rely on two mechanisms. First, RNNLMs map the discrete word-based vocabulary into a continuous space. As a result, the model can exploit word sequences which are similar, without requiring them to be exactly identical. This mechanism helps to reduce the effect of data sparseness. Second, RNNLMs are explicitly equipped to handle long-distance dependencies. The recurrent loop in their architecture constitutes a memory that allows them to model arbitrarily long word histories theoretically.

In this paper, we investigate language model adaptation for RNNLMs, and specifically address two central challenges for language model adaptation, *within-domain adaptation* and *limited-data within-domain adaptation*, originally identified by Rosenfeld [124] and explained later in depth. The main contribution of this paper is to demonstrate that curriculum learning is an important technique for carrying out the adaptation of RNNLMs and to provide insights on how it must be applied in order to be effective for improving speech recognition.

Curriculum learning applies a specific, well-planned ordering of the training data, referred to as a 'curriculum', during the learning process and is an established approach in the machine learning community. When conventional *n*-gram language models are trained, the order in which the training data is processed has no impact on the outcome of the training process. In contrast, Neural Networks are indeed sensitive to the differences in the order in which the training data is presented to them. The work of Bengio et al. [17] attributes the benefits of curriculum learning in Neural Network training to an ability of the curriculum to guide the learner, in particular, directing it away from inappropriate local minima and towards more suitable ones.

The advantages that curriculum learning offers to RNNLMs for speech recognition has been previously established in the literature [98, 100]. The previous work has focused on dynamically updating language models during the recognition process [98] and in optimal reduction and sorting of the training data [100]. The existing work points out that training data presented later in the training process has more influence on the final form of the model than the initial part of the training data. As such, curriculum learning can be used to accomplish an implicit interpolation of the training data, where certain parts of the data is given more importance than others.

The specific issue of adaptation is particularly important for RNNLMs, and as such deserves dedicated attention. A key reason for its importance is the relatively high cost of training RNNLMs, which can be attributed to a range of factors. Here, we mention in particular the fact that the whole training set is usually presented to the model multiple times (referred to as 'training epochs'). The relatively high cost of the training phase of RNNLMs means that retraining the language model whenever new training data becomes available is prohibitively costly.

Curriculum learning has several distinction advantages to offer for the adaptation of RNNLM to specific sub-domains. First, curriculum learning provides a method to effectively carry out implicit interpolation that does not require the parameter optimization needed for conventional interpolation. Second, as has been pointed out by Iyer et al. [65], Iyer and Ostendorf [63], and, more recently, by Mikolov and Zweig [97], one danger of adaptation models that build individual component models on data sub-sets is fragmentation. Fragmentation refers to the fact that as more and more component models are built, relatively less data is available to train each. Using curriculum learning to train sub-domain adapted RNNLMs neatly circumvents the fragmentation issue. All training data can be used to train each adapted model; it is the order in which the data is presented to the model during training that makes the difference. In short, although the potential of curriculum learning for RNNLMs has been established and offers clear advantages, the challenges of curriculum learning to language model adaptation for RNNLMs remains nearly entirely unaddressed. The purpose of this paper is to fill that gap.

The key insight of this paper is that in order to carry out adaptation, curriculum learning should be used to train language models from general patterns, characteristic of the training data as a whole, to specific patterns characteristic of one specific sub-domain of the overall data. The move from general patterns to specific patterns can be viewed as a special case of a move from simple patterns to complex patterns, discussed in the literature, e.g., by Elman [48].

We propose three types of curricula created with three different strategies: Start-from-Vocabulary (SV), Data Sorting (DS) and All-then-Specific (AS). Our experiments are designed to give insight into which types of curriculum learning are best in which situations and which other factors influence the performance of curriculum learning for RNNLM adaptation. When RNNLMs are applied in practice to improve speech recognition, they are usually applied to the task of rescoring the N-best output produced by a speech recognizer that has carried out an initial pass over the spoken data. For this reason, all our experiments either produce word prediction results or rescoring results.

Our first set of experiments addresses the challenge of *within-domain adaptation*. As characterized by Rosenfeld [124], within-domain adaptation can be used to deal with heterogeneous data sets that contain different sub-domains. Different sub-domains are characterized by different word-usage patterns, which are, in turn, reflected by different *n*-gram distributions. For example, sub-domains in the Spoken Dutch Corpus, one of the corpora used for experiments in this paper, which is described in detail later, different speech styles (e.g., spontaneous conversation, lecture and read speech) form different sub-domains. Note that we focus our investigation on within-domain adaptation rather than cross-domain adaptation, since, as mentioned by Rosenfeld [124], it is relatively rare that a speech recognizer would be trained on one domain and used in a different one, i.e., for recognizing speech with radically different characteristics than the input speech. Under the within-domain adaptation challenge, the totality of the training data becomes available at the same time, and the goal is to create models that are adapted to specific sub-domains.

Our first within-domain adaptation experiment investigates the oracle situation in which sub-domain information is known during both training and testing. This experiment demonstrates that sub-domain-adapted RNNLMs indeed outperform both general RNNLMs as well as RNNLMs that have been adapted to the sub-domain using linear interpolation between the general model and a sub-domain specific model. The second experiment investigates the situation in which sub-domain information is known during training, but unknown during testing. Here, we investigate two methods of combining sub-domain-adapted RNNLMs. The third experiment investigates the situation in which no sub-domain information is known for either training or testing. We use Latent Dirichlet Allocation with *k*-means clustering [119] on the sentence level to automatically build sub-domains in the training data. Curriculum learning is applied to create sub-domain-adapted RNNLMs for each sub-domain, which are combined using the combination methods in the previous experiment.

Our second set of experiments addresses the challenge of *limited-data within-domain adaptation*. This challenge corresponds to the situation often faced by speech recognizers: a relatively large amount of data is available to train an initial model, and, with time, more and more data becomes available that can be used to update the model. The new data is from the same domain, but can be considered to be from a different sub-domain because of shifts in data characteristics over time. Here, we test a sub-domain adapted RNNLM, but focus particularly on the fact that the amount of adaptation data is limited and also on the fact that the vocabulary of the adaptation data is unknown at the time of training of the initial model.

The rest of the paper is organized as follows. Section 3.3 discusses related work in the areas of adaptive language modeling, curriculum learning and other advanced language models such as RNNLMs, mixture models and class based language models. In Section 3.4, we present the three types of curriculum learning that we use to train the sub-domain-adapted RNNLMs. We then present the experiments addressing the challenge of *within-domain adaptation* (Section 3.5) and the challenge of *limited-data within-domain adaptation* (Section 3.6). The final section concludes and presents an outlook.

3.3 Related work

The approaches presented in this paper related to previous work that has been carried out in areas of adaptive language modeling, curriculum learning for recurrent neural networks, and recurrent neural network language modeling, are covered in this section in turn.

3.3.1 Adaptive Language Modeling

Approaches to the adaptation of statistical language models have been characterized by [13] as belonging to three categories: model interpolation, constraint specification, and topic information. We discuss each category and mention its relationship to the approach proposed in this paper.

Cache-based statistical language models [69, 79] carry out dynamic adaptation during the process of recognition and fall under the category of model interpolation. These models represent one of the initial attempts to model long-distance dependency in language models. Basically, they involve the linear interpolation of an *n*-gram model and a dynamic cache component model. These models are based on the assumption that a word used recently has bigger probability than its overall probability. Cache-based statistical language models face the challenge of uncertainty of the adaptation data that is introduced by speech recognition errors occurring in the window of recent words that is used as the cache. This challenge was mentioned by Mikolov et al. [98] in their investigation of dynamic RNNLMs, which continue to update their weights during the test process. More recently, Mikolov and Zweig [97] proposed a dynamically updating RNNLM that uses an LDA representation of recently recognized words in order to increase the amount of topic-related context information exploited by the language model. We limit our treatment of cache-based language models to this relatively brief mention, since our focus here is set on language model adaptation methods that are applied during training rather than during testing.

Constraint specification methods are mainly associated with maximum entropy language models. The process by which the constraint features are integrated into the model during training is forced to respect the maximum entropy criterion. [125] showed that maximum entropy language models, using trigger and *n*-gram features, achieve a significant improvement over *n*-gram language models in terms of perplexity and word error rate. Since their introduction, maximum entropy language models have enjoyed substantial success in adaptive statistical language modeling [3, 18, 30, 127]. Actually, from the neural network language modeling perspective, the maximum entropy model can be viewed as a neural network language model with no hidden layer. In this paper, some of the RNNLMs are built with maximum entropy extensions [100].

Topic information methods for language model adaptation include topic-based language models and mixture models. Many of these models employ a model interpolation strategy, but are discussed here as topic information models rather than interpolation models because they attempt to incorporate explicit representation of topics.

The topic-based language models of Gildea and Hofmann [56] can be viewed as variants of class-based language models [23]. These models map the words in the vocabulary to a smaller number of classes. Topic-based language models use the same mathematical format as class-based models, except that in topic-based models the topic is treated as a hidden variable that is calculated according to the expectation maximization algorithm. Exploiting the same basic insight, topics have been treated as hidden nodes in Dynamic Bayesian Networks [137, 165]. In Tam and Schultz [154], Latent Dirichlet Allocation (LDA) language models were proposed. These models integrate topic information by interpolating the n-gram model with LDA unigram. Specifically, the LDA unigram is a combination of the topic probabilities with a conditional probability of present word given the latent topics which were estimated by LDA [20].

In the work of Iyer et al. [65] and Iyer and Ostendorf [63] on topic mixture models, the joint probability of each sentence is calculated as a linear interpolation of the sentence probabilities from k component language models, P_k, each modeling a separate topic,

$$P(s) = \sum_k \lambda_k P_k(s) = \sum_k \lambda_k \prod_i P_k(w_i|h(w_i)). \tag{3.1}$$

Here, $h(w_i)$ is the history of w_i in sentence s. The component models are trained on sub-sets of the training data, each containing the data that corresponds to an individual topic. The sub-sets are obtained by clustering the training data. In [63], a two-stage clustering process is used to partition the data. The quality of the clustering, i.e., the suitability of the resulting partitions, is essential for the performance of the model.

Our approach exploits many of the same mechanisms as this work. We use a mixture of component models, as in Eq. (3.1), as a *soft decision method* to combine individual sub-domain adapted RNNLMs. Because the sub-domain-adapted RNNLM is able to make a highly accurate prediction of the sub-domain, we compare the topic mixture model with a *hard decision method*. Instead of using interpolation, this method makes a definitive decision about the sub-domain to which a given sentence belongs, and uses the corresponding sub-domain-adapted RNNLM to score that sentence.

We also use a clustering method in order to create sub-domains in cases in which sub-domain information is not available in the training data. However, instead of using a two-stage clustering method, we make use of LDA. Our choice is motivated by the status of LDA as the state of the art in topic representation [20]. We form sub-domains by applying *k*-means clustering to LDA-based topic representations of the sentences in our data.

Finally, we would like to point out that our sub-domain adaptation method confronts the same challenge of fragmentation facing other topic-based adaptation approaches. Iyer et al. [65] and Iyer and Ostendorf [63] point out that a too aggressive partitioning of the training data may aggravate data spareness issues for the component language models, which are trained on the individual partitions. In order to address this issue, they interpolate the topic-specific models with a general model trained on the entire data set,

$$P(s) = \sum_k \lambda_k \prod_i [\alpha_i P_k(w_i|h(w_i)) + (1-\alpha_i)P_g(w_i|h(w_i))], \tag{3.2}$$

where P_k is a topic model and P_g a general model.

The key insight of our paper is that explicit interpolation as in Eq. (3.2) is not necessary for RNNLMs. Rather, we can make use of a well-planned curriculum that exploits the fact that training data presented later in the training process contribute more to the final state of the RNNLM.

We plan this curriculum so that it starts with general data, corresponding to the overall collection, and moves to specific data, corresponding to the individual sub-domains. Sparse data issues are alleviated by allowing all available training data to contribute to the component model corresponding to each individual sub-domain. The arrangement of the ordering of the training data implicitly adjust the weights between general model and component models. Because curriculum learning provides this possibility for implicit interpolation, explicit interpolation, as in Eq. (3.2), is not needed in order to train sub-domain-adapted RNNLMs. As previously mentioned, because the curricula make use of all available training data, only changing the order of presentation, our sub-domain-adapted RNNLMs are able to avoid the dangers of fragmentation. In the next section, we go on to discuss curriculum learning in more detail.

3.3.2 Curriculum Learning for Neural Networks

An investigation at the intersection of cognitive science and machine learning carried out by [48] is credited with laying the groundwork for curriculum learning. The goal of this work was to understand the contributions of restrictions existing early in the learning process to the final success of learning.

The topic was taken up again in the machine learning community by Bengio et al. [17] under the label ‘curriculum learning’. Here, the interest was not generally on restrictions during the learning process, but rather on restrictions on the input data. Specifically, Bengio et al. [17] shows that final ability improves if the learner is first presented with simple input and then moved to more complex input. Curriculum learning is connected to many other techniques used for learning, such as boosting and transfer learning. Bengio et al. [17] characterizes the special emphasis of curriculum learning as guiding the optimization process, especially in a way that steers it towards better local minima. The discussion in Bengio et al. [17] opens some intriguing vistas of inquiry. First, the notion that curriculum learning 'guides the optimization process' contributes a valuable insight into why curriculum learning works, but a more concrete understanding could be useful for applying curriculum learning in practice. Second, the notions of 'simple' and 'complex' are very general, and have multiple useful interpretations. Bengio et al. [17] provides a statistical formalization of a curriculum that moves from 'simple' to 'complex' based on the idea that the entropy of the data should increase so that the diversity of the training data increases. However, the experiments also show that naively creating two classes of 'simple' and 'complex' samples already provides improvement.

Our work is based on the idea that in a language model adaptation scenarios, 'simple' can be productively equated with 'general' and 'complex' can be equated with 'specific'. We take the difference between 'general' and 'specific' to be related to differences of both word choice and word usage, which are in term reflected in different *n*-gram distributions. However, in application scenarios it will not be practical to predict what this differences would be, or, expect to be able to calculate them precisely. For this reason, we keep our definition of 'general' vs. 'specific' naive, and assume nothing more than that the difference reflects the variations within a particular domain that lead to that domain being considered heterogeneous.

This assumption is consistent with the discussion in [48] on the reasons for which a language model is able to benefit from a particular ordering of training data to improve its ability for predicting words. According to [48], in the first learning stage the network learns a functional representation scheme that encodes an underlying structure of the data. Then, in the second learning stage, the learner moves to learn the surface reflexes of this underlying structure. Based on the initial 'simple' examples, the learner is able to create an approximation of the solution space that constrains the learner from making 'mistakes' when it is presented with more complex data. In our case we assume that by presenting the RNNLM first with 'general' input, it will learn overall patterns in the language, and that these will allow it to derive maximum benefit from the 'specific' input for the sub-domain, since it will be be protected from over-fitting as it adapts to the sub-domain.

The application of curriculum learning to RNNLMs proposed in [100] can be considered a progenitor of our work. [100] interprets 'simple' to mean 'closest to the target data'. Results are reported on two experiments in which the data presented to the RNNLM ordered by increasing perplexity with respect to the development data. The first experiment demonstrates that data ordering can lower the convergence time. The second experiment demonstrates that a combination of data ordering and filtering of high perplexity data leads to a reduction of the perplexity of the resulting RNNLM measured on the evaluation data. Considering the second experiment as a proof-of-concept, our work explores in depth the application of curriculum learning to RNNLM adaptation. Specifically, we establish that the benefits of curriculum learning go beyond an optimization of the training set to be useful to address two further challenges facing language models for speech recognition: *within-domain adaptation* and *limited-data within-domain adaptation.*

This paper builds on and extends our previous work [143], in which we first introduced the idea of applying curriculum learning to adapt RNNLMs. This work provided an initial exploration of curriculum learning for within-domain adaptation. Here, we examine the issue of within-domain adaptation in greater depth and detail and also extend the idea of curriculum learning for RNNLMs to limited-data within-domain adaptation.

3.3.3 Recurrent Neural Network Language Models

The pre-cursor of the today's RNNLMs can be considered to be [16], which proposed feed-forward neural network language models. In these models, each word in the vocabulary is mapped by a shared parameter matrix to a real vector. [98, 101] further extended the feed-forward neural network language model to a recurrent neural network by incorporating history information into the input layer. In his RNNLM, the input layer consists of the input word and also the activated hidden layer associated with the previous word.

As is shown in [99], RNNLMs outperform other advanced language models currently in common use. As previously mentioned, the superior performance of RNNLMs can be attributed to two properties. First, RNNLM projects each word from a large discrete vocabulary space to a small continuous vector space. The mapping from the discrete to the continuous space enables the model to learn generalization. The continuous representation of words can be effectively used to capture the similarities between different words and reduce the effect from data sparseness [102]. Second, the recurrent structure of RNNLMs equips them with a compact memory. Theoretically, recurrent neural networks can store relevant information from previous time steps for an arbitrarily long period of time, making it possible to learn long-term dependencies.

Curriculum learning can be applied to different types of neural network language models, however, in this paper, we only focus on applying curriculum learning to RNNLMs. As is stated in [96], state of the art results based on several benchmark data sets are achieved by RNNLMs.

Recently, Mikolov and Zweig [97] proposed a context dependent RNNLM, in which the context information is a latent topic vector obtained from the preceding text using LDA. In this paper, we also use LDA to create topical representations. However, instead of integrating latent topic information into one general model, we use it in order to produce a clustering of the data that serves as the basis for producing a series of sub-domain-adapted RNNLMs, which are then applied in combination.

3.4 Curriculum Learning For RNNLMs

In this section, we introduce three different methods of applying curriculum learning in constructing sub-domain-adapted models. At the beginning of the section, we describe the basic structure of a RNNLM. Then we give the details about the curriculum learning methods used in this paper.

3.4.1 Recurrent Neural Network Language Models

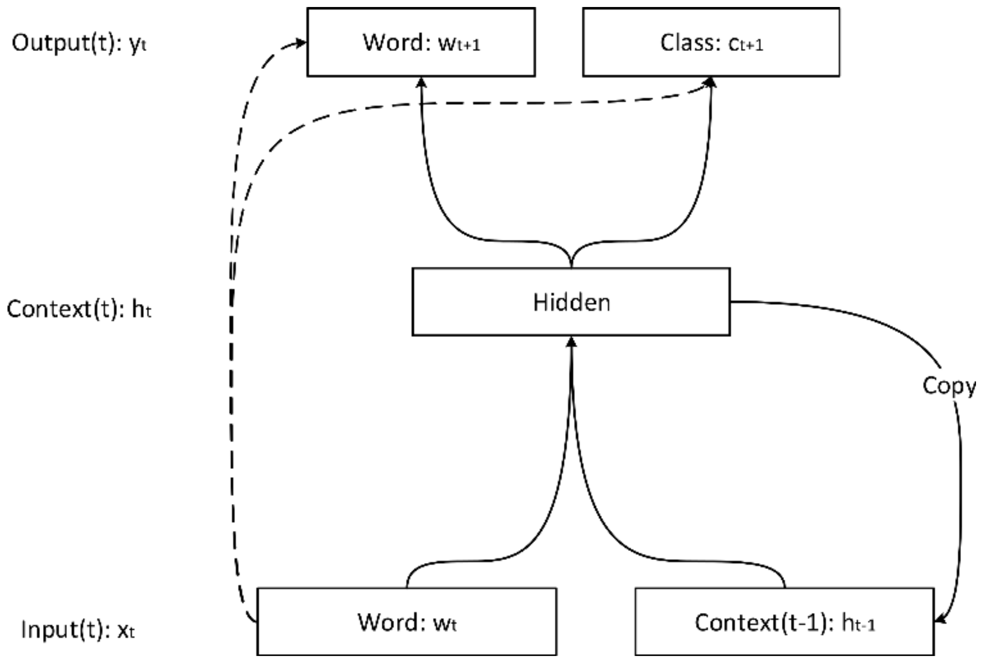
The recurrent neural network language models adopted in our work originated from [98]. As shown in Fig. 3.1 (solid line parts represent conventional RNNLM), it has three layers: an input layer x, a hidden layer h and an output layer y. It is characterized by the loop between the input layer and hidden layer, which plays the role as a short abstract memory to store the previous information. At each time t, the input vector x_t is constituted by the current word one-hot vector w_t as well as a copy h_{t-1} from the previous hidden neurons. The output layer y_t is factorized to two parts: one is word part w_{t+1}, the other one is class part c_{t+1}. The sigmoid function and softmax function are used as the activation functions in the hidden layer and output layer, respectively. The weight matrix between the input layer and hidden layer is estimated by backpropagation-through-time (BPTT)[101], which actually unfolds the loop as the deep neural network.

3.4.2 Three Curriculum Learning Methods

In this paper, we propose three different curriculum learning strategies for the training of sub-domain-adapted RNNLMs. The commonality of the three strategies is that they move from presenting the RNNLM with general data to presenting it with specific data. The strategies progressively give general patterns in the data more influence during model training.

Start-from-Vocabulary (SV) This curriculum uses the entire training data set to construct the vocabulary of the sub-domain-adapted RNNLM, but then uses only the training data from the corresponding sub-domain for training.

Figure 3.1: Recurrent Neural Network Language Models with Maximum Entropy Extension. Dash line parts represent the Maximum Entropy Extension.



Data sorting (DS) This curriculum uses the entire data set to train the sub-domain-adapted RNNLM. The data is sorted such that the final data presented during training is the data from the target sub-domain. The validation data is also selected from the corresponding sub-domain.

All-then-Specific (AS) Each sub-domain-adapted model starts with a number of epochs training on the whole data and then learns from its specific sub-domain data. In the general training period, we choose the validation data that covers all the sub-domains. In the specific training period, the validation data is selected from its corresponding sub-domain.

We now briefly discuss the rationale for these models and the difference between the three strategies. Of the three, the SV method uses the least general pattern information. The resulting sub-domain-adapted RNNLM model has only been exposed to patterns from its target sub-domain during the training process. Both the DS and the AS strategies use the whole training data including the sub-domain data. The strategies differ with respect to the point at which they move from general patterns to specific patterns. DS makes the transition inside each epoch, but AS makes the transition outside of the epochs. Under the DS strategy, the model is shaped to approach a domain-specific optimum within each training epoch. Under the AS strategy, the model is first fully optimized with respect to general patterns, and then pulled from this point to a domain-specific optimum.

3.4.3 Experimental Set-Up

In this section, we explain the design of our experiments and provide the details of the implementation of the experimental framework that we used.

3.4.4 Evaluation

Our experiments compare our proposed curriculum learning methods with two baselines. The first, is the unadapted baseline ('base'). This baseline is an sRNNLM that has been trained on the entire data set in its natural order. The second is conventional linear interpolation ('int'). The 'int' models are sentence level mixtures of individual sub-domain-adapted models, cf. Eq (3.2). Each sub-domain-adapted model is a linear interpolation of a general model with a specific model trained only using the data in the training set belonging to its specific sub-domain.

We use several metrics to report the results of our experiments. The perplexity (PPL) is a commonly used metric for measuring language models. It is calculated as the geometric average of the inverse probability of the words on the test data.

$$PPL = (\prod_{i=1}^{t} P(w_i|h(w_i)))^{-\frac{1}{t}}, \tag{3.3}$$

where $h(w_i) = w_1 w_2 ... w_{i-1}$. Word prediction accuracy WPA [159] is a measure of the performance of language models in practice.

$$WPA = \frac{C}{N}, \tag{3.4}$$

where C is the number of correctly predicted words and N the number of total words. Many applications in the area of natural language process, such as spell checking and sentence completion, use WPA. Word Error Rate (WER) is used for those experiments for which we carry out N-best re-scoring, described in more detail below. Based on a speech recognition system, using minimum edit distance, the prediction sentence is compared with reference sentence to get the number of substitutions S, the number of deletions D and the number of insertions I.

$$WER = \frac{S+D+I}{N}, \tag{3.5}$$

where N is the number of total words.

3.4.5 RNNLM Framework

In the experiments, the language models are applied to two closely related, but different tasks, word prediction and N-best rescoring. In word prediction, a word that gets maximum probability in the output layer is chosen as the predicted word. In N-best rescoring, the weighted combination of acoustic model score, language model score and word insertion penalty is used to select one best hypothesis sentence. The weight of the combination is determined by a held-out development N-best list data set through grid search.

Because of the differences between the different strategies for curriculum learning, different stopping criteria for training are applied. Start-from-Vocabulary (SV) and Data sorting (DS) stop when the sub-domain-adapted model can not achieve additional PPL improvement on the validation data. In each case, the validation data set is selected from the sub-domain, but mutually is exclusive with the training and test sets. AS training consists of two phases, one for whole data training, the other for sub-domain data training. In all data training, we specify the number of training epochs. The sub-domain training starts when the all data training finishes the specified number of epochs. The sub-domain training stops when the model can not achieve additional improvement on the sub-domain validation data.

The experiments are carried out using the RNNLM toolkit [2]. The detail of setting basic parameters of this toolkit such as class size and block size can refer to the website of this

[2]http://www.fit.vutbr.cz/ imikolov/rnnlm/

toolkit. In some experiments, we apply the Maximum Entropy Extension to RNNLMs, which uses a weight matrix that directly connects input features to the output layer. The structure of the Maximum Entropy Extension of RNNLMs are shown in Fig 3.1. The dash line in the figure represents the Maximum Entropy Extension part of RNNLMs. When the extension is applied to a condition it is applied to all experiments carried out for that condition and also indicated clearly in the text. By applying this extension, we are able to report the state of the art optimal values that are reachable by RNNLMs on our data.

3.5 Within-domain Language Model Adaptation

This section presents our investigation of the use of curriculum learning to address the challenge of *within-domain adaptation* for RNNLMs. Table 3.1 presents an overview of the experiments. As is shown in table, the experiments start from the situation that the sub-domain information is known during both training and testing, move to the situation that the sub-domain information is only known during training, and end with the situation that sub-domain information is unknown during both training and testing. Condition 1, in which sub-domain information is known during testing, is an oracle condition that lets us test the sub-domain-adapted RNNLMs individually on their domains, and gives us insight into their power to predict sub-domains. Condition 2 and 3 address the real-world case in which the sub-domain labels of the test data are unknown. In these cases, we use combinations the individual sub-domain-adapted RNNLMs. We experiment with a *hard decision combination* and a *soft decision combination*, described in more detail below.

Table 3.1: Within-domain adaptation experiments (Section 3.5): Overview of experimental conditions.

	Condition 1	Condition 2	Condition 3
Training data sub-domain	known	known	unknown
Test data sub-domain	known	unknown	unknown
Language Model	sub-dom.-adapted RNNLM	combinations of sub-dom-adapted RNNLMs	combinations of sub-dom-adapted RNNLMs

3.5.1 Sub-domain Information Known for Training Set

In this section, we investigate the case in which sub-domain information is known for the training data. Such situations occur naturally in cases that the sub-domain information can be obtained at the time that the data is collected. For example, if a sub-domain is associated with a certain phone, microphone or known-location. In this situation, sub-domain-adapted RNNLMs are trained on sub-domain-labeled data. We first present the data set and then go on to present the results of our experiments on Conditions 1 and 2.

Sub-domain Known Data Set: Spoken Dutch Corpus (Conditions 1 and 2)

For our investigation of cases involving known sub-domains, we choose the Spoken Dutch Corpus (Corpus Gesproken Nederlands, CGN) [110, 111]. This corpus is representative of a heterogenous data set consisting of data that has been recorded in multiple social-situational settings. In previous work we investigated the characteristics of the CGN corpus [139]. This work revealed that different socio-situational settings are characterized by different word distributions and different syntactic constructions.

Table 3.2 presents a detailed description of the CGN data set. The data set contains audio recordings of standard Dutch spoken by adults in Netherlands and Flanders. It comprises nearly 9 million words divided into 14 sub-domains that correspond to different socio-situational settings. The data sets contain recordings ranging from read speech to lectures and including spontaneous conversations. *comp-i* to *comp-o* involve a single speaker and *comp-a* to *comp-h* contain dialogues or multilogues. In the table, the column labeled “Tokens” gives the number of running words in each sub-domain.

From the CGN data, we randomly selected 80% for training, 10% for validation and 10% for testing. Note that comp-m is too small, this sub-domain is not randomly selected for the test data. In the test data, those words that are not in the language model vocabulary are replaced by an out-of-vocabulary token (the OOV rate is 3.8%). All RNNLMs trained on CGN data have a hidden layer of 300 neurons, and use 100 classes and 4 times BPTT with a block size of 10.

Condition 1: Sub-domain Known for Training and Test Set

Our investigation of the oracle condition, i.e., sub-domain information is known at training and testing time, allows us to understand the improvements that sub-domain-adapted RNNLMs achieve over our baseline, an unadapted RNNLM (‘base’) and over adapted models created using conventional linear interpolation (‘int’)(Eq (3.2)). The experiments reveal the relative improvements that sub-domain-adapted RNNLMs are able to achieve as well as their

Table 3.2: Overview of the Spoken Dutch Corpus (CGN).

Label	Socio-situational setting	Tokens
comp-a	Spontaneous conversations ('face-to-face')	2,626,172
comp-b	Interviews with teachers of Dutch	565,433
comp-c&d	Spontaneous telephone dialogues	2,062,004
comp-e	Simulated business negotiations	136,461
comp-f	Interviews/ discussions/debates	790,269
comp-g	(political) Discussions/debates/ meetings	360,328
comp-h	Lessons recorded in the classroom	405,409
comp-i	Live (eg sports) commentaries (broadcast)	208,399
comp-j	Newsreports/reportages (broadcast)	186,072
comp-k	News (broadcast)	368,153
comp-l	Commentaries/columns/reviews (broadcast)	145,553
comp-m	Ceremonious speeches/sermons	18,075
comp-n	Lectures/seminars	140,901
comp-o	Read speech	903,043

predictive power for sub-domains.

The results of the experiments are reported Table 3.3 (perplexity) and 3.4 (WPA). These results were produced by testing each sub-domain-adapted RNNLM (corresponding to each of the three curriculum learning strategies SV, DS, and AS) on its corresponding sub-domain. SV and DS were trained for until the convergence criteria were met. AS was trained with general data for 10 epochs and further with sub-domain specific data until the convergence criterion was met.

The results in Table 3.3 and 3.4 allow us to make several important observations. First, consulting the total WPA at the bottom of Table 3.4, we can see that baseline model (base), which is a RNNLM that has been trained on the whole data set without using curriculum learning, is outperformed by both conventional linear interpolation (int) and also the sub-domain-adapted RNNLMs trained with the AS or DS strategies for curriculum learning. This result confirms that language models are very sensitive to differences in sub-domain, and illustrates the importance of taking sub-domain information into consideration. In some cases the benefits of adapting to sub-domains are particularly dramatic. For example, for "News" (comp.-k), all the adapted models achieve an over 50% reduction in terms of perplexity and more than 30% of improvement in terms of WPA.

Second, the results in Table 3.3 and 3.4 demonstrate that overall the AS and DS curricu-

Table 3.3: The perplexity (PPL) comparison of conventional general RNNLM (base), sub-domain-adapted models in sentence level mixture models using linear interpolation (int) and sub-domain-adapted models using different strategies of curriculum learning on CGN data set under the oracle situation. SV is Start-from-Vocabulary, DS is Data-Sorting, and AS is All-then-Specific. The sub-domains indicated in bold are those for which the sub-domain-adapted models achieve substantial improvement.

comp	base	base-int	SV	DS	AS
a	82.6	80.4	90.6	81.0	80.4
b	104.2	100.2	120.2	91.9	89.6
c&d	73.1	69.7	81.2	69.3	67.8
e	80.1	52.3	62.2	47.6	47.8
f	157.4	142.3	180.1	133.6	129.2
g	283.4	190.6	245.2	180.0	179.4
h	141.9	133.4	177.0	120.4	117.6
i	341.3	141.5	189.8	146.9	145.8
j	222.8	202.3	338.8	176.0	174.3
k	553.1	222.4	292.9	221.6	230.3
l	293.3	347.6	486.8	236.0	235.5
n	289.1	23.2	411.8	228.2	228.3
o	480.2	274.1	328.4	269.2	261.3

*Table 3.4: The percent word prediction accuracy (*WPA*) comparison of conventional general* RNNLM *(base), sub-domain-adapted models in sentence level mixture models using linear interpolation (int), and sub-domain-adapted models using different strategies of curriculum training on the* CGN *data set under the oracle situation.* SV *is Start-from-Vocabulary,* DS *is Data-Sorting, and* AS *is All-then-Specific. The sub-domains indicated in bold are those for which the sub-domain-adapted models achieve substantial improvement.*

comp	base	int	SV	DS	AS
a	24.0	24.2	23.5	24.3	24.3
b	20.2	19.4	19.0	21.0	21.0
c&d	25.5	25.4	24.5	25.8	25.9
e	24.5	25.0	23.7	26.6	25.9
f	18.6	18.3	17.4	19.1	19.3
g	15.9	16.5	16.0	17.6	17.7
h	20.9	20.5	18.7	21.6	21.7
i	16.5	19.9	18.8	19.7	19.8
j	17.3	16.6	13.4	18.3	18.5
k	14.5	20.0	19.7	19.9	19.8
l	15.2	13.7	12.8	17.2	17.0
n	14.8	13.0	12.4	16.5	16.3
o	14.2	15.6	15.2	16.3	16.4
total	20.6	21.3	20.0	23.0	23.3

lum learning strategies outperform conventional linear interpolation. As previously mentioned, a priori, the implicit interpolation carried out by curriculum learning is superior to conventional linear interpolation since it achieves a balance between general and specific data without requiring weights to be tuned on the validation data, as is the case for conventional linear interpolation. The results provide evidence that in addition to being easier to apply in practice, curriculum learning also delivers a performance improvement over conventional linear interpolation.

Third, these experimental results reveal that among the three curriculum learning strategies, SV performs the worst consistently over sub-domains. Recall that SV adopts only the vocabulary of the full training data set, and otherwise trains each sub-domain-adapted RNNLM only on sub-domain specific data. Apparently, SV does not offer enough exposure to general data during the training process and sub-domain data are insufficient to allow the RNNLM to generalize robustly. The low performance of SV can be attributed to data fragmentation, i.e., as mentioned above, a single topic-specific domain can easily fail to contain enough data to train a topic-specific model. Fourth, in Table 3.3 and 3.4 it can be seen that the performance of sub-domain-adapted RNNLMS using the DS curriculum learning is similar to the performance of those trained with the AS strategy. Recall that in each epoch of DS training, the data is sorted, meaning that within each epoch the RNNLM is first trained by the general data and then further trained by the specific sub-domain data. In contrast, AS is trained using a certain number of epochs of the general data set followed by an additional number of epochs of specific sub-domain until the convergence criterion is met. Table 3.5 presents results showing the impact of the number of general-data epochs used before beginning specific-data epochs. It can be seen that the maximum performance is reached with 10 epochs, i.e., the largest number of general training epochs. It should be noted that 10 epochs represents nearly a completely trained general model, which would otherwise reach its convergence criterion after 12 training epochs. This result confirms the importance of the contribution of the general data to training the sub-domain adapted RNNLMS.

To summarize, this section has shown the importance of adaptation in general, and also illustrated the superiority of curriculum learning for language model adaptation with respect to conventional interpolation. In the next section, we move on to investigate whether the performance improvements offered by curriculum learning can be extended to a condition in which the sub-domain of the test data is not known.

*Table 3.5: The perplexity (*PPL*) and word prediction accuracy (*WPA*) comparison using* AS *curriculum learning with increased number of epochs of general data training (gen.).*

comp	2 gen. epochs		4 gen. epochs		6 gen. epochs		8 gen. epochs		10 gen. epochs	
	PPL	WPA	PPL	WPA	PPL	WPA	PPL	WPA	PPL	WPA
a	83.8	24.0	83.4	24.0	81.6	24.1	81.2	24.2	80.4	24.3
b	96.6	20.3	94.7	20.8	90.6	21.0	90.3	21.1	89.6	21.0
c&d	73.2	25.1	71.5	25.4	69.3	25.6	68.6	25.8	67.8	25.9
e	49.0	26.3	48.1	26.6	46.4	26.0	46.4	26.2	47.8	25.9
f	142.6	18.4	139.2	18.7	133.1	18.9	132.1	19.1	129.2	19.3
g	196.6	17.1	186.5	17.3	178.5	18.0	177.9	17.9	179.4	17.7
h	131.3	20.7	125.1	21.0	120.0	21.3	118.9	21.5	117.6	21.7
i	151.2	19.7	144.3	20.1	140.4	20.4	141.4	20.0	145.8	19.8
j	212.1	16.8	196.9	17.2	182.5	17.9	178.1	17.9	174.3	18.5
k	238.5	20.2	228.7	20.3	221.7	20.3	223.8	20.1	230.3	19.8
l	287.7	15.7	271.8	16.2	247.8	16.6	242.3	17.0	235.5	17.0
n	273.5	15.3	261.1	15.1	241.5	16.1	233.9	16.3	228.3	16.3
o	278.1	15.9	269.9	16.2	260.0	16.4	260.2	16.5	261.3	16.4

3.5.2 Experiment 2: Sub-domain Known for Training Set and Unknown for Test Set

We now turn to the case in which sub-domain information is known for the training data, but not for the test set. In this case, we do not know in advance which sub-domain-adapted language model should be applied to a given segment of speech, which in the case of the CGN data set, is a sentence. For this reason, instead of applying a single sub-domain-adapted language model, we make use of approaches that combine sub-domain-adapted language models.

The first approach makes a *soft decision* about the correct sub-domain of the input sentence using a linear combination of the scores generated for a given sentence by all of the sub-domain-adapted language models. We use the conventional form for a sentence-level mixture, given above in Eq (3.1). In our implementation, the interpolation weights λ_k are dynamically determined according to the following heuristic method:

$$\lambda_k(s) = \frac{p_k(s,h_s)}{\sum_i p_i(s,h_s)}, \tag{3.6}$$

where $p_k(s,h_s)$ is the joint probability from the kth sub-domain-adapted for the present sentence s and its history h_s. Basically, the sub-domain-adapted model that assigns higher probability for current sentence receives more weight to determine the next word.

The second approach makes a *hard decision* for each input sentence. Our previous work has examined the relative advantages of the hard decision vs. the soft decision for other types of language models, and revealed it to be important [138, 141]. Here, we investigate both with respect to the combination of RNNLMs. In order to motivate the use of the hard decision to combine sub-domain adapted RNNLMs, we carried out an analysis of the ability of RNNLMs to predict the identity of sub-domains. Sub-domain-adapted RNNLMs can be used to predict the sub-domain of a sentence s $C(s)$ as follows:

$$C(s) = \arg\max_k p_k(s,h_s), \tag{3.7}$$

where h_s is the history of sentence s. $p_k(s,h_s)$ is the joint probability of sentence s with its *history*, which is assigned by the kth sub-domain-adapted model. Our experiments with sub-domain prediction yielded very good results. For nine of the thirteen sub-domains in the CGN data set, the prediction accuracy was above 95%. This result suggests that the hard decision method for combining sub-domain-adapted RNNLMs could possibly approach the neighborhood of the performance observed in Section 3.5.1 under the oracle condition where the identity of the sub-domain is known for the test data.

Table 3.6 shows perplexity and word prediction accuracy results of soft and hard decision on CGN data, in which sub-domain context is not available to the models during the

Table 3.6: Perplexity and word prediction accuracy result of the Spoken Dutch Corpus (CGN). The sub-domain of the test data is unknown.

models	PPL	WPA (%)
base	114.8	20.6
int	98.1	20.7
SV-soft	118.9	19.8
DS-soft	103.4	21.8
AS-soft	104.1	22.7
int-hard	-	21.3
SV-hard	-	20.1
DS-hard	-	22.1
AS-hard	-	23.2

testing. The soft decision perplexity results show that the soft decision combination of the sub-domain-adapted models using interpolation achieves the lowest perplexity. However, this performance does not transfer to the word prediction accuracy. Using soft decision combination method, AS with 10 epochs of general training obtains the best WPA. Based on sentence level WPA, "AS-hard" achieves significant improvement over baseline model. Although "DS-soft" also achieves substantial improvement over baseline model, it did not achieve statistical significant improvement.

Comparing the soft and hard combinations, we find that the hard combination performs better than soft combination method in terms of WPA. In addition, the hard combination method is also computationally less complex than soft combination one in word prediction task. Using soft decision method for word prediction, the system needs to put the hidden layer and output layer of each sub-domain-adapted RNNLM into memory. In addition, with the soft decision method, all the output layers of sub-domain-adapted RNNLMs have to be linear interpolated together to do prediction. However, the main disadvantage of the hard decision method is that it does not produce a normalized probability.

3.5.3 Experiment 3: Sub-domain Unknown in both Training and Test Sets

We now turn to the case in which sub-domain information is known for neither the training nor the test set. We are interested in exploring the abilities of curriculum learning in cases in which sub-domain information is not collected at the time that the data is captured, but instead must be inferred. In this section, we present an additional experiment about using the proposed RNNLM adaptation method to a benchmark testing data set which is Wall Street Journal (WSJ). WSJ data set is a well-known data set that has been used in previous work by [97, 98]. In previous sections, it shows that RNNLMs adaptation using curriculum learning can outperform both the conventional RNNLM and the sentence level mixture models on CGN data. However, we notice that the CGN covers many speech sub-domains. Each of these sub-domains actually is dramatically different with other sub-domains in style. WSJ is about newspaper articles, which we can assume belong to different topics, making it a reasonable choice to investigate topical structure. However, in WSJ the style is same, providing an interesting contrast with CGN. In this section, we will investigate the speech recognition performance of the proposed method for WSJ data set using latent topic clustering for the training data.

WSJ Data

In WSJ, we use 100-best speech recognition list from the DARPA WSJ’92 and WSJ’93 test data sets, as used by [98, 162]. In the 100-best list set, 333 sentences are used as development data for tuning the interpolation of language models score and acoustic model score (DEV). The rest, 465 sentences, are used for evaluation (EVAL). The oracle WER of 100-best lists for development data and evaluation data are 6.1% and 9.5%, respectively. The training corpus, referred to as the NYT data set, contains 37M words of running text from the New York Times section of English Gigaword. The validation data set contains 186K words. A held-out set of 230K words is used for testing (TEST). The vocabulary size is 194K.

Experiments

We use Latent Dirichlet Allocation (LDA) [20] to generate the latent topics for the training data. LDA is a probabilistic generative model, which use bag-of-words strategy to assign each document with latent topic representation. In this paper, each document is a sentence. Basically, we generate the latent topic representation for each sentence in the training data. In the training data, each sentence is treated as a document. The topic distribution can be viewed as a reduced latent topic representation for each sentence. Based on this normalized continuous representation of each sentence, we then use the *k*-means clustering method to partition the training data.

In the N-best list rescoring experiment, all RNNLMs have a hidden layer of 200 neurons, and use 100 classes and 4 times BPTT with a block size of 10. Additionally, we make use of the Maximum Entropy extension in the form of 1 billion elements that directly connect the input features to output layers using maximum entropy extension.

Table 3.7 shows the word error rate performance of the proposed RNNLM adaptation method in N-best rescoring. The models using DS curriculum learning achieves 0.2% improvement over the conventional RNNLM in terms of word error rate. Although this improvement is too small to be significant (based on the evaluation N-best list with 465 sentences, a paired t-test shows that $mean = 0.025\%$, $t = 0.146$, $p = 0.886$), we note that the gains we obtained here are comparable in magnitude to those reported on another type of RNNLM extension, namely context based RNNLM [97].

Compared with the experiment results on the CGN data, the RNNLM adaptation using curriculum learning can only gain small improvement over the conventional RNNLM on the WSJ data set. One reason is that, in the WSJ data, both the training data (NYT section of English Gigaword) and N-best list are news related. We suspect these data sets represent only very limited diversity. In addition, the training data is NYT section of English Gigaword, but the N-best list is WSJ data. The latent topic we constructed from training data is not applicable to the N-best data.

Table 3.7: The word error rate (WER) comparison on WSJ data set. 'base' is the conventional RNNLM. 'kn-5' is a Keneser-Ney 5-gram language model. 'int' is a conventional linear interpolation of sentence level mixture models (Eq. (3.1,3.2)). "SV", "DS" and "AS" are RNNLM adaptation using different types of curriculum learning. "T" column represent the number of topics. "hard" and "soft" represent hard decision (Eq. (3.7)) and soft decision (Eq. (3.1) and (3.6)), respectively.

models	T	dev (%)	eval (%)	T	dev (%)	eval (%)
kn-5		12.1	17.3			
base		10.3	14.9			
base-int-soft	5	11.2	15.3	10	11.3	15.9
base-int-hard	5	11.5	15.8	10	11.2	16.2
SV-soft	5	11.6	16.1	10	11.5	16.3
SV-hard	5	11.3	16.3	10	11.5	16.5
DS-soft	5	10.2	14.7	10	10.3	15.2
DS-hard	5	10.4	14.7	10	10.2	15.2
AS-soft	5	10.3	15.1	10	10.4	15.0
AS-hard	5	10.1	15.1	10	10.3	14.9

3.6 Limited-data Within-Domain Adaptation

Next we turn to our second set of experiments, in which we investigate the ability of curriculum learning to adapt language models in the face of the *limited-data within-domain adaptation* challenge. Limited-data domain adaptation is necessary in many real-world scenarios, since language models that are used in practical applications generally need frequent updating. In many cases, updates are carried out by retraining language models from scratch, adding a small amount of new, yet-unseen training data to the original large existing training data set.

3.6.1 Experimental data set

Our choice of data sets is intended to emulate the real-world situation, in which the system has already been trained on a relatively large amount of existing data and a smaller amount of yet-unseen data becomes available. In this experiment, the large existing data we choose is the NYT section of English Gigaword (see Section 3.5.3), which we used in previous section. The yet-unseen data set is the Penn Treebank (PTB) text data set. We use section 00-20 for training (972K word tokens), section 21-22 for validation (77K word tokens) and section 23-24 for testing (84K word tokens).

3.6.2 Experiments

In the experiment, we use the AS curriculum learning method for limited-data domain adaptation. We make this choice because neither the SV nor DS methods is fit to the limited-data domain adaptation. SV is not suited because it requires the vocabulary to be calculated on the entirety of the training data, and the adaptation data is not available to the system at the moment in which the original model is trained. DS is not suited because it requires the entirety of the training data be presented to the RNNLM in every training epoch. Using the AS curriculum learning method, we basically continue to train the existing language model on yet-unseen data. In the experiment, we compare this approach with a baseline that uses conventional linear interpolation to combine the existing language model with a new language model trained on the yet-unseen data. The models produced by conventional linear interpolation and by AS curriculum learning both maintain the same vocabulary as the existing language model.

In our experiment, the vocabulary of the existing language model, determined by the NYT, contains 194k words. All the words in the yet-unseen data that are not in the vocabulary are treated as OOV words. In the yet-unseen data set, the OOV rate for training data is 9.3%, for validation data 7.1% and for testing data 7.9%. During testing, the probability of OOV words is set to 10^{-8}. All the models in this subsection are trained with the same settings as models from subsection 3.5.3.

Table 3.8: Comparison of RNNLM*s trained by only yet-unseen data (*PTB *base),* RNNLM*s trained on existing data (*NYT *base), linear interpolation of* RNNLM*s trained on existing and yet-unseen data (int* NYT +PTB*), and implicit interpolation of existing and yet-unseen data using the* AS *curriculum learning method (*AS CL NYT +PTB*). Results (*PPL *and* WPA*) are reported on yet-unseen test data. The percentage of yet-unseen data used in training is given in the column marked 'ratio'.*

models	ratio	PPL	WPA
PTB base	1.0	125.9	24.6
NYT base	0.0	180.7	21.7
int NYT + PTB	0.2	121.5	23.7
int NYT + PTB	0.4	113.5	24.1
int NYT + PTB	0.6	107.8	24.7
int NYT + PTB	0.8	107.4	24.8
int NYT + PTB	1.0	104.5	25.0
AS CL NYT +PTB	0.2	113.3	24.2
AS CL NYT +PTB	0.4	108.4	24.6
AS CL NYT +PTB	0.6	105.5	24.8
AS CL NYT +PTB	0.8	104.6	25.0
AS CL NYT +PTB	1.0	103.5	25.0

Table 3.8 shows the existing language model can achieve substantial improvement on PTB testing data in terms of perplexity and WPA by specific training on the PTB training data. If only 20% of the PTB training data is used, using AS curriculum learning the existing language models can achieve 37.7% reduction in terms of PPL and 11.5% relative improvement in terms of WPA. The performance of the language models can be improved by using more PTB data in the training process. As is shown in the table, curriculum learning applied for limited-data domain adaptation can produce RNNLMs that are better, in terms of WPA, than conventional RNNLMs trained only on PTB data.

Table 3.8 also shows that the model trained by AS curriculum learning using 80% of yet-unseen data achieves similar performance as that achieved by a model that uses the interpolation of the existing language model and a language model trained on yet-unseen data based on the same vocabulary as existing model. This result suggests that curriculum learning can be used as an effective method to implicitly weight the patterns in the existing training and the pattern in the yet-unseen training data. The results of only using 20% yet-unseen training data suggests that curriculum learning is better in taking advantage of taking limited training data than linear interpolation. By using 80% yet-unseen training data, the adapted model trained using curriculum learning achieves the same performance as the adapted model trained using interpolation methods. In addition, using curriculum learning, only one final model is returned rather than two models as the interpolation method is based on. In other words, using curriculum learning method to deal with limited-data adaptation, we always only need to focus on one model. However, using interpolation method, we probably need to deal with the risk to handle more models when more unseen data available.

The results confirm the potential of curriculum learning to improve performance for limited-domain data adaptation of RNNLMs. In practice, the update of RNNLMs can be accomplished by the proposed AS curriculum learning method only on the yet-unseen training data rather than retrain the RNNLMs using the combination of the existing training data and the yet-unseen training data.

3.7 Conclusions

In this paper, we investigated the use of curriculum learning for the adaptation of RNNLMs. We focus on two situations for language model adaptation, namely, within-domain adaptation and limited-data within-domain adaptation.

To address within-domain adaptation, we use a component model method. Each component model is a sub-domain-adapted RNNLM trained by curricula that were scheduled from general patterns to specific patterns. For within-domain adaptation, three different experiments has been used to investigate three different situations, namely the oracle situation (sub-domain information is known during training and testing), the situation that sub-domain information is known only in training and the situation that the sub-domain information is unknown both in training and testing.

Three different curriculum learning methods were proposed and analyzed, namely starting from the same vocabulary (SV), data sorting (DS) and all then specific training (AS). We compared the sub-domain-adapted models that are trained by these methods with conventional RNNLMs using a heterogeneous spoken Dutch data set. The results under the oracle condition under which sub-domain information is known at test time show that sub-domain-adapted models that were trained using DS and AS outperform the conventional RNNLM. Especially on the “News” sub-domain, the sub-domain-adapted models achieved an over 50% reduction in terms of perplexity and a more than 30% improvement in terms of word prediction accuracy. The results reveal that curriculum learning can be used to shape the final RNNLMs to emphasize the specific patterns in the sub-domains.

When the sub-domain information is not available in testing, the sub-domain-adapted models need to be combined to make final prediction. On the sentence level, two combination methods were used. Using hard decision method, for each sentence, one sub-domain-adapted model that gave the maximum probability was selected. Using soft decision method, for each sentence, a heuristic dynamic linear interpolation was used to combine the different sub-domain-adapted models. Experimental results show that the proposed model using these two methods outperform conventional RNNLMs.

Under the situation that the sub-domain information is unknown both in training and testing, the proposed models were tested using WSJ data set in N-best rescoring. During the training, sub-domain information was obtained by using Latent Dirichlet Allocation in conjunction with *k*-means clustering. The experimental results show that RNNLM adaptation using DS curriculum learning can achieve a limited, but not entirely unpromising, reduction in terms of word error rate.

To sum up, the set of experiments on within-domain adaptation shows that curriculum learning method can be used to train sub-domain-adapted models that emphasize the patterns characterizing specific sub-domains. Sub-domain-adapted models trained using curriculum learning outperform conventional RNNLMs on the corresponding sub-domains. When sub-domain information is unknown during testing, the combinations of the sub-domain-adapted models using a soft combination or a hard combination outperforms conventional RNNLMs and sentence level mixture RNNLMs.

To address limited-data within-domain adaptation, we use curriculum learning as an implicit interpolation method to combine patterns characteristic of existing training data with patterns characteristic of yet-unseen data. The results from our experiment on limited-data domain adaptation reveal that curriculum learning methods are more effective than conventional interpolation methods. The experiment also shows that updating of the existing RNNLMs with curriculum learning requires training only on the yet-unseen data without retraining models from scratch by adding the yet-unseen data to the existing data.

The comparison of these three types of curriculum learning proposed in this paper reveals that good performance is achieved by the curricula that are designed to be appropriate for specific data and specific challenges. In this paper, results on the WSJ data set and on the CGN data set did not achieve comparable improvement. For this reason, our future work will focus on the relationship between the design of the currculum and the characteristic of the target data.

In this paper, we have chosen RNNLMs to be representative of neural network language models, under the assumption that the application of curriculum learning to other neural network language models would yield similar behavior. Our future work will explore this assumption in greater detail, allowing further insight onto how the method put forth in this paper should best be operationalized.

Chapter 4

Integrating Meta-Information into Recurrent Neural Network Language Models[1]

4.1 Abstract

Due to their advantages over conventional *n*-gram language models, recurrent neural network language models (RNNLMs) recently have attracted a fair amount of research attention in the speech recognition community. These advantages include their ability to learn generalizations in the face of sparse data conditions and their capacity to capture long-distance word dependencies. In this paper, we explore another advantage of RNNLMs, namely, the ease with which they allow the integration of additional knowledge sources. We concentrate on features that provide complementary information w.r.t. the lexical identities of the words. We refer to such information as *meta-information*. We single out three cases and investigate their merits by means of N-best list re-scoring experiments on a challenging corpus of spoken Dutch (referred to as CGN). First, we look at Parts of Speech (POS) tags and lemmas, two sources of *word-level linguistic information* that are known to make a contribution to the performance of conventional language models. We confirm that RNNLMs can profit from these sources as well. Second, we investigate Socio-situational Settings (SSSs) and topics, two sources of *discourse-level information* that are also known to benefit lan-

[1]This chapter has been submited to Speech Communication. Yangyang Shi, Martha Larson, Joris Pelemans, Catholijn, M. Jonker, Patrick Wambacq, Pascal Wiggers, Kris Demuynck. Integrating Meta-Information into Recurrent Neural Network Language Models

guage models. SSSs can be seen as a proxy for the language register. For the purposes of our investigation, we assume that information on the SSS can be captured at the moment at which speech is recorded. Topics refer to the subject matter of speech and are inferred automatically. In order to predict POS, lemmas, SSS and topic, a second RNNLM is coupled to the main RNNLM. We refer to this architecture as a Recurrent Neural Network Tandem Language Model (RNNTLM). Our experimental findings show that if high-quality meta-information labels are available, both word-level and discourse-level information improve performance. However, deriving high-quality labels automatically from unlabelled data proved to be challenging. Third, we investigate sentence length and word length (i.e., token size), two sources of *intrinsic information* that are readily available for exploitation because they are known at the time of re-scoring. Intrinsic information has been largely overlooked by language modeling research. Since sentence length and word length are known to correlate with Socio-Situational Setting, they hold promise for improving the performance of language models as well. RNNLMs allow these features to be incorporated with ease, and obtain improved performance.

4.2 Introduction

Language models capture the extent to which a sequence of words can be considered well formed. Most state-of-the-art language models treat language as a sequence of symbols, and ignore the fact that this sequence also reflects the underlying structure of the language. Language structure is evident on multiple levels. In this work, we focus particularly on its manifestations at the word level and at the discourse level. We refer to language-related information that goes beyond the lexical identities of the spoken words as *meta-information*. Examples that are relevant to our investigation include word-level meta-information such as Part of Speech (POS) or lemmas and discourse-level information such as the setting in which the speech is delivered (referred to as the Social-Situational Setting) and topic.

Past efforts in language modeling have demonstrated that incorporating additional language-related information at different levels can improve the performance of language models. Conventional *n*-gram language models, however, offer relatively limited possibilities for incorporating meta-information. In order to predict the next word of a word sequence, a conventional *n*-gram language model relies solely on the $n-1$ words that precede it. This strategy is simple and robust, but is limitted in its ability to capture long distance dependencies between words and doesn't generalize well from sparse data.

Recently, recurrent neural network language models (RNNLMs) [98, 101] have demonstrated potential to address these shortcomings. The success of RNNLMs can be attributed to two factors. First, RNNLMs map the discrete, word-based vocabulary to a continuous space. This mapping makes it possible to learn generalizations over word sequences that are not completely identical, thus reducing the effect of data sparsity. Second, the recurrent loop in the RNNLM architecture, which feeds the hidden layer back into the input layer at every time step, constitutes a memory that serves to capture long-distance dependencies. In this paper, we focus on a third advantage of RNNLMs that has received relatively little attention in the literature. Incorporating meta-information into *n*-gram language models is cumbersome. Generally, it is necessary to design specialized architectures, to create hand-crafted models, or to train weighting parameters. In contrast, integrating meta-information into RNNLMs just requires adding the extra features to the input layer. Viewing recurrent neural networks as a set of logarithmic regressions helps to make clear that adding extra information can be accomplished elegantly: no special changes to the architecture of the model must be made in order to accommodate the new information.

In practice, RNNLMs are applied in the last pass of a multi-pass speech recognition system. In our experiments this was implemented as a N-best list re-scoring task. We choose to work with data drawn from a large and challenging corpus of spoken Dutch. This corpus contains, by design, very diverse material. In particular, the data has been captured in different Social-Situational Settings (SSSs), i.e. different settings that affect language register and correlate with different topics. This allows us to extend our investigation of discourse-level meta-information beyond the level of automatically derived topics.

Our investigations cover three cases of meta-information. First, we investigate *word-level linguistic information*, represented by Part of Speech (POS) tags and lemmas. Previous work has established the contribution that these sources make to enhancing the performance of conventional language models. We confirm that RNNLMs also benefit from these sources. Second, we look at *discourse-level information*, more specifically SSSs and topics. These sources of meta-information are also known to benefit conventional language models. Here again, we demonstrate their ability to improve the performance of RNNLMs. Finally, a third case concerns meta-information that can be considered *intrinsic*. In other words, the information is inherent in words and word-sequences and does not need to be inferred. Specifically, we investigate sentence length and token size, two features that are readily available for exploitation, but which have been largely overlooked in previous work on conventional language models. It is difficult to identify a single factor responsible for the lack of attention to intrinsic features in the literature. Most likely, the oversight is due to the combination of the relatively large overhead required to integrate meta-information into conventional language models, already mentioned above, and the a priori impression that intrinsic information is trivial. When RNNLMs are used, the incorporation of extra information is straightforward and elegant, and our experiments demonstrate that trivial information can be exploited to achieve performance gains that are themselves non-trivial.

In our investigation, the information on the SSS was captured at the moment at which speech is recorded. As a contrastive condition, we also investigate a setup where no such labels are available, and hence a logic labelling system must be learned unsupervised. For this work, we investigate the integrating SSS and topics. The SSS is available for training but not for testing. Topics in this paper are automatically detected word usage patterns, which are unavailable in training and teasing. Therefore, we first use an unsupervised method to obtain the topics for the training data. The topic information obtained by the unsupervised method is further used to train a meta-information predictor that is used to predict topics for testing data.

In general, meta-information to be exploited by language models is not known in advance, but rather must be predicted on the fly. Recurrent Neural Networks have shown good results compared to conventional methods on natural language processing tasks such as named entity recognition and syntactic analysis [95, 173]. We therefore opted to infer the required meta-information by training an additional recurrent neural network. The RNN that extracts the meta-information feeds into the RNN that models the word sequences (the RNNLMs), resulting in an architecture that we refer to as a Recurrent Neural Network Tandem Language Model (RNNTLM). The first network takes the entries in the N-best list as input and outputs meta-information for each word. The output of the first network in combination with the N-best word sequence feeds into the main network, which outputs a prediction of the probability of the next word, given the history. We demonstrate how a second RNNLM which predicts POS, lemma, SSS and topic can be coupled to the main RNNLM.

Our experimental findings indicate that both word-level and discourse-level information can improve performance. However, in order to obtain a tangible performance improvement the meta-information must be accurate. In our challenging task, information obtained via unsupervised training did not attain a high enough accuracy and hence incorporating this information showed little to no improvement. For this reason, we turn to the *intrinsic information* such as sentence length and token size.

The rest of the paper is organized as follows. Section 4.3 discusses related work on inferring meta-information and on previous methods that have exploited the integration of meta-information into language models. In Section 4.4, we describe our approach for incorporating meta-information into RNNLMs, including the RNNTLM architecture. Section 4.5 and 4.6 describe the experimental setup and present the experimental results on the spoken Dutch data that we used for our investigation. The final section provides conclusions and an outlook.

4.3 Related work

In this section, we present work related to two key aspects of our approach. First, we survey various forms of meta-information. Next, we discuss previous work that has integrated meta-information into language models and explain how our work builds on and extends these approaches.

4.3.1 Meta-Information

We use the term *meta-information* to refer to information that goes beyond the identity of the word itself. In this section, we briefly survey the types of meta-information that we focus on in this paper.

Word-Level Meta-Information

The word-level meta-information we consider includes Part-of-Speech (POS) tags, lemmas and token size (i.e., word length). As is discussed in further detail in the next section, POS information benefits language modeling. POS tag sequences provide a limited amount of syntactic information to language models. They, for example, allow the language model to capture regularities such as the fact that adjectives are often followed by nouns.

POS information is not an intrinsic property of a word, and for this reason, if it is to be used in language modeling it must be predicted. Both the task of labeling words with POS tags [5, 37] and methods to integrate POS with language models [25, 103] have received considerable research attention. In this paper, the prediction of POS tags as well as the integration of POS tags into language modeling is achieved by using the proposed RNNTLMs.

A lemma is the set of all word-forms that share the same meaning. The citation form of a word that is used in the dictionary, represents a lemma. Lemmas provide the language model with morphological information about each word. The number of lemmas is much larger than the number of POS.

Word length, referred to here as token-size (TS), is the size of the word. Here, we measure token size by counting the number of letters in the written form of the word. Token size reflects information about other properties of words. For example, the average token size of content words is bigger than the average token size of function words. Another important characteristic was pointed out by Zipf [178], namely that token size reflects the frequency with which a word is used in a language. For these reasons token size is an interesting quantity to divide words into classes. Surprisingly, token size has not been exploited extensively in previous work on language modeling.

Adding word-level information to a language model can be seen as a form of smoothing, especially in conventional *n*-gram language modeling. For word sequences for which there is little or no evidence in the training data, the model can fall back on information concerning the classes to which words belong.

Sentence Length (SL)

Sentence length is defined as the number of words in a sentence. It is a indicator of discourse style and genre. This relationship was established even before the advent of the Digital Age in the field of authorship attribution [157]. Recent work observing the relationship between sentence length and genre includes [146] and [167] for the spoken Dutch data used in this work. In particular, sentence length distribution varies for different conversation styles. For example, for spontaneous speech the average sentence length is below 7. In spontaneous face-to-face conversations almost 25% of the sentences contain only one word such as yes or no answers and interjections. In contrast, the mean length of sentences in political discussion/debates/meetings is 15, and in ceremonious speeches/sermons, it is 20. Language models have yet to exploit information on sentence length fully. An isolated exception may be [21], who demonstrated that combining separate language models, each created for sentences of different lengths, improves recognition performance in the domain of voice search. In our paper, we aim to exploit the benefits of sentence length in a more general domain.

Topic and Socio-Situational Settings

Both the topic being spoken about and the situation in which language is used, referred to here as Socio-situational Setting (SSS), impact word distributions. The topic is related to the subject under discussion by the speaker or speakers. In contrast, the SSS is more of a proxy for the language register (style of speech), which is influenced by the goal of the conversation, the relationship between speakers and listeners, and the number of speakers and listeners involved. Certain topics may be more typical for some SSSs than others, so in general it is not useful to assume that the two are independent. The main distinction in the context of this paper is that the SSS can be captured at the time of recording whereas regularities that are discovered automatically in the data are considered to be topic related. In research where topic is expected to mainly reflect the subject matter under discussion, the topic models almost invariably differentiate between topics based on the distribution of content words only, ignoring function words. In this work, we are interested in modeling underlying clusters in general, and are agnostic if they are related to style or subject matter. Hence, all words are allowed to contribute to the topic model.

Table 4.1 shows the 14 different SSSs used in this paper. Our previous research [145] investigated the dynamic classification of SSSs using Dynamic Bayesian Networks. In this paper, a recurrent neural network is used to predict the SSS and topic for each sentence of the input data. This information is then fed into the RNNLM for the purpose of word prediction and N-Best re-scoring.

Table 4.1: Overview of the Spoken Dutch Corpus (CGN)

components	socio-situational setting	words
a	Spontaneous conversations (‘face-to-face’)	2,626,172
b	Interviews with teachers of Dutch	565,433
c&d	Spontaneous telephone dialogues	2,062,004
e	Simulated business negotiations	136,461
f	Interviews/ discussions/debates	790,269
g	(political) Discussions/debates/ meetings	360,328
h	Lessons recorded in the classroom	405,409
i	Live (e.g., sports) commentaries (broadcast)	208,399
j	News reports/reportages (broadcast)	186,072
k	News (broadcast)	368,153
l	Commentaries/columns/reviews (broadcast)	145,553
m	Ceremonious speeches/sermons	18,075
n	Lectures/seminars	140,901
o	Read speech	903,043

4.3.2 Language Models Integrating Information beyond Word Identity

Previous research has established the usefulness of information that goes beyond the identity of words in improving language models. In this sub-section, we survey some of the most successful work exploiting this information and explain its relationship to our work.

Decision-tree-based language models [10] are one of the earlier language modeling methods that integrate meta-information with information about word identity. For example, part-of-speech (POS) information can be integrated into language models by asking questions about the word history such as, “Is the last word a verb?” [60]. In [153], the Random Forest Language models of [170] are extended with morphological, prosodic, syntactic, and topic information.

Class-based language models [23] can be viewed as language models that integrate meta-information. Since the quality of the class-based language models depends on how the vocabulary is grouped into clusters, much previous research has been devoted to understanding the best way to cluster the vocabulary [14, 23, 106–108, 115, 158, 172]. Language models that group words according to POS tag, allow easy integration of POS information with n-gram language models [106, 108]. In this work, we show that automatic determination of the categories yields improved performance over the original POS categories, presumably because it allows control over the category size and composition.

Structured language models [25, 27] represent another important technique to exploit information beyond the word level. These language models incorporate information concerning the syntactic structure of a language as well as the grammatical function of words in the form of their POS class.

Some language models have integrated meta-information in an effort to better encode information about long distance dependencies between words.

Language models incorporating latent topics [12, 61] are a key example. These models use a topical representation of the data created by a method such as latent semantic analysis [12] or latent Dirichlet allocation [61]. In this paper, we also employ Latent Dirichlet Allocation to construct a representation of documents that captures generalizations over topic. We then create topics by using k-means clustering.

Dynamic Bayesian Networks (DBNs) [39, 105] offer a concise method to integrate additional features into a language model. Syntactic information, semantic relationships and social background knowledge can be simply specified as a variable into the network structure of the belief network [137, 138, 165]. However, DBNs are generalizations of n-gram language models, and as such share some of their drawbacks. In particular, because they model exact sequences, they tend to suffer in the face of sparse data.

Maximum entropy language models [117, 125] are among the best existing methods for integrating additional information into a language model. These models exploit the maximum entropy principle [66] in order to incorporate additional knowledge sources, which can be completely arbitrary in nature. In [125] maximum entropy language models using trigger and n-gram features are shown to achieve significant improvement over n-gram language models in terms of perplexity and word error rate. Maximum entropy language models can be viewed as a variety of neural network language models which include no hidden layer. In this paper, we use the maximum entropy extension of the RNNLM (RNNME) proposed by [101], to incorporate meta-information into RNNLMs. The so called RNNME includes a direct connection between the input layer and the output layer effectively incorporating a maximum entropy language model into the RNNLM architecture.

Neural network based language models, which include feed-forward neural network language models [16] and recurrent neural network language models [98], are representative of the current state of the art in language modeling. As previously mentioned, neural network language models are acknowledged for their ability to generalize and their ability to capture long-distance dependencies. Here, we focus on a third advantage, namely their flexible structure, which allows the integration of arbitrary features. [49] and [2] investigated the incorporation of syntactic or morphological information into neural network language models.

Factored language models proposed by [19] treat each word as a vector of factors. In [168], the RNNLM is extended to a factored RNNLM. However, in [168], only word-level information is used. In this paper, we investigate not only word-level information, but also sentence-level and discourse-level information. Furthermore, we investigate the incorporation of intrinsic information such as word and sentence length, which initially gives the impression of being trivial, but actually has the ability to improve language models. The usefulness of integrating topic information, derived via Latent Dirichlet Allocation, into RNNLMs has been studied by [97]. Here, we significantly expand on both [97] and our own previous work on integrating linguistic and contextual information into RNNLMs [140]. A full range of different types of meta-information is investigated. Further, we go beyond [140] in that we evaluate our models applied not only to the task of word prediction, but also to the task of N-best re-scoring. Lastly, we propose a recurrent neural network tandem language model (RNNTLM) which employs RNNLMs both for inferring meta-information and for predicting the probability of the next word.

4.4 Recurrent Neural Network Language Models

The original RNNLMs proposed by [98], consist of three layers: an input layer x, a hidden layer h and an output layer y. RNNLMs are characterized by a loop that integrates a delayed copy of the previous hidden layer into the current input layer at each time step. This loop acts as a short abstract memory that stores previous information. In the hidden layer, the output of a neuron i is:

$$h_i(t) = \varphi(\sum_j u_{ij} x_j(t)), \tag{4.1}$$

where the activation function $\varphi(z)$ is a sigmoid function:

$$\varphi(z) = \frac{1}{1+e^{-z}}. \tag{4.2}$$

The activation function $\phi(z_m)$ in the output layer is a softmax function:

$$\phi(z_m) = \frac{e^{z_m}}{\sum_k e^{z_k}}, \tag{4.3}$$

with the index m corresponding to one of the words in the vocabulary. The weight $u_{i,j}$ between input layer context part and hidden layer is estimated by backpropagation-through-time (BPTT) [101], which actually unfolds the loop as a deep neural network.

In [100], a maximum entropy extension of RNNLMs (RNNMEs) is proposed. As is shown in Fig. 4.1, an additional weight matrix directly connects the n-gram features to the output layer,

$$p(w|hist) = \frac{\exp \sum_{i=1}^{N} \lambda_i f_i(hist, w)}{\sum_w \exp \sum_{i=1}^{N} \lambda_i f_i(hist, w)}, \tag{4.4}$$

where f_j is one feature, λ_i is the weight for feature i and $hist$ is the history of features. The feature f_j includes bigrams (w_{t-1}), trigrams (w_{t-2}, w_{t-1}) up to n-grams. The problem with such feature representation is that for high order n-gram, it has impractically large feature set. Most of the features in the feature set will never show in the data. So to reduce the complexity of the huge weight matrix connecting the input features to the output layer, a hash function is used to map every n-gram to a single value in a hash array.

$$f(w_{t-2}, w_{t-1}) = ((w_{t-2}) * P_1 * P_2 + w_{t-1} * P_1)\% SIZE. \tag{4.5}$$

Where P_1 and P_2 are large prime numbers. $SIZE$ is the size of the hash array. % is a module function.

In [101], a class-based RNNLM is proposed. The class-based RNNLM factorizes the output layer using classes. The classes are determined according to the word frequency in the training data. Using a class-based RNNLM, the probability of a word w_{t+1} at time $t+1$ given its history $hist_{t+1}$ is calculated in the following way:

$$p(w_{t+1}|hist_t) = p(w_{t+1}|c_{t+1}, hist_t)p(c_{t+1}|hist_t), \tag{4.6}$$

where c_{t+1} is the class to which word w_{t+1} belongs. Switching to a class-based RNNLM substantially reduces the computations for updating the weight matrix between hidden layer to output layer. Instead of updating a $H \times V$ weight matrix (H is the hidden layer size, V is the vocabulary size), the class-based RNNLM only updates a $H \times C$ weight matrix (C is the class size) connecting hidden layer with class part of output layer as well as a $H \times VC$ sub-matrix (VC is the number of words belongs to class c_{t+1}). As shown in [101], the class-based RNNLM achieves 15 times speedup at a cost of 1% accuracy degradation.

4.4.1 Recurrent Neural Network Tandem Language Models

Our approach to integrating meta-information into RNNLMs consists of models containing two parts, one part uses a recurrent neural network for predicting the meta-information, and

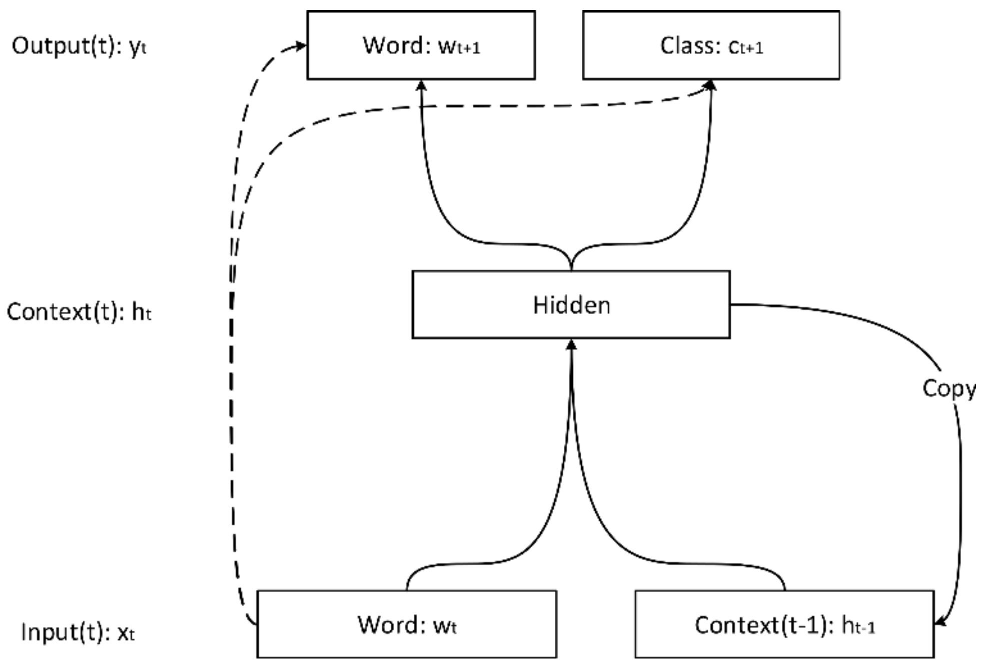


Figure 4.1: Class-based Maximum Entropy extension of RNNLMs*. The dashed arrows represent the direct connection of n-gram features in the input to the output.*

the other part integrates the predicted meta-information into RNNLMs.

Different types of meta-information predictors are needed to extract the various types of meta-information used in this research. Meta-information such as token size and sentence length, is 'intrinsic', meaning that it can be derived directly by inspecting the data. However, to obtain the word-level information (POS, lemma) and the discourse-level information (Socio-Situational Settings and topics) for the test data, we need the aid of a meta-information predictor. In the following subsections, we discuss the two cases in turn.

Integrating Word-Level Meta-Information

Word-level meta-information is predicted using the history of the current word. In order to incorporate word-level meta-information, we use the Recurrent Neural Network Tandem Language Model(RNNTLM) architecture that is illustrated in Fig 4.2. The meta-information prediction component is an RNNLM as well. In order to predict meta-information $m(t)$ for the current word $w(t)$, the previous meta-information $m(t-1)$, the current word $w(t)$ and the copied hidden layer $h_m(t-1)$ are fed into the network.

$$x(t) = [w(t)^T m_1(t-1)^T \dots m_p(t-1)^T h(t-1)^T]^T, \tag{4.7}$$

where p is the number of types of meta-information. The word vector $w(t)$ and all meta-information vectors $m_{1...p}(t-1)$ are represented using a 1-of-N encoding. Because the maximum entropy extension is used, as is shown in equation (4.4), the previous $n-1$ words with current word and the previous $n-1$ meta-information with current meta-information are directly connected to the meta-information output layer. Both the sequences of previous $n-1$ words with current word and previous $n-1$ meta-information with current meta-information are encoded as large hash based vectors using encoding 1-of-N. In this paper, both word sequence and meta-information sequence are represented by hash vectors with 1 billion elements. Note that the input to the maximum entropy extension part is different from the input to the RNNLMs part. Actually, the input to the maximum entropy part is much larger than the input to the RNNLMs. For convenience, in Fig 4.2, we indicate the maximum entropy extension by using dashed lines that directly connect the input layer of the RNNLMs to the output layer of the RNNLMs. The maximum entropy extension adds information to the RNNLMs about the ordering of the word sequence ending with the current word. Hence, the usage of the maximum entropy extension helps RNNLMs to capture local regularities. In the output layer of the meta-information predictor, the meta-information $m^*(t)$ that obtains the highest probability is selected and encoded in a 1-of-N representation, which is fed to the RNNLM:

$$m^*(t) = \arg\max_{m(t)} p(m(t)|w(t), m(t-1), hist(t-1)). \tag{4.8}$$

When the meta-information is unknown, the current predicted meta-information $m^*(t)$ is copied to the input of the meta-information predictor in order to predict next meta-information $m^*(t+1)$.

As is shown in Fig 4.2, the part above the horizontal dashed line is also a recurrent neural network. It uses the current word $w(t)$, and the predicted meta-information for the current word $m^*(t)$ and the copied hidden layer $h_w(t-1)$, to predict the next word $w(t+1)$. In the proposed RNNTLM, the structures of the two recurrent neural network are basically the same; they differ only in their input and output.

Integrating Discourse-Level Meta-Information

Discourse-level information is predicted using a recurrent neural network as well, when the information is not available in testing. However, since discourse-level meta-information is predicted for a full segment rather than individually for each word, the architecture from Fig 4.2 is no longer suitable.

Because the RNNLM is applied within an N-Best rescoring framework, segment information is available at the moment the language model is applied. This information has been generated by the speech recognition system that produced the N-Best lists. By looking at

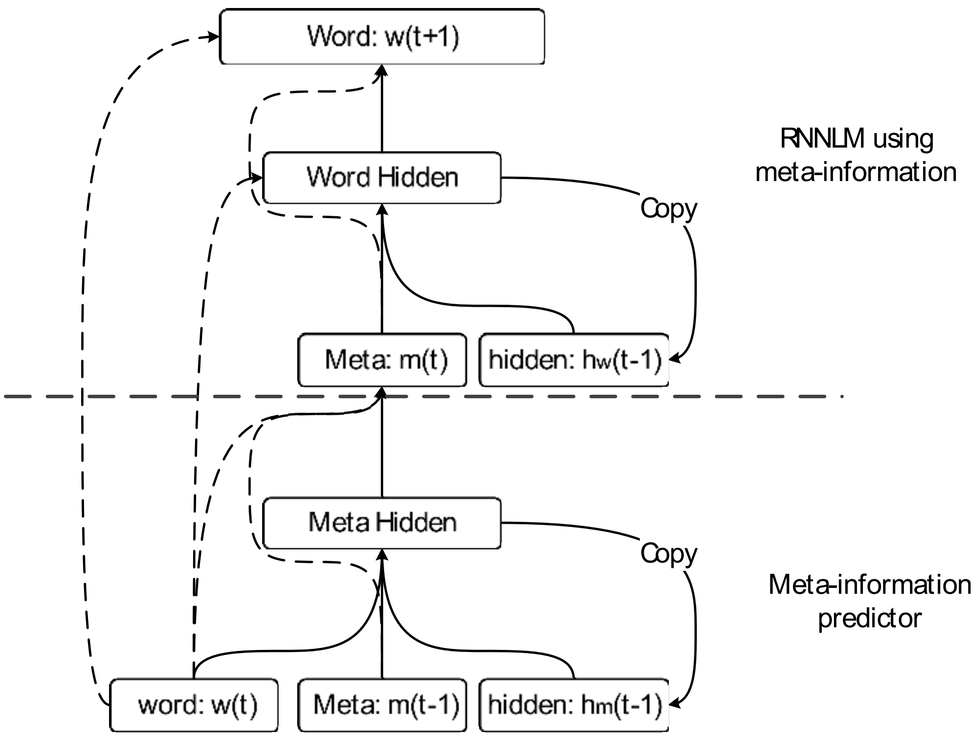


Figure 4.2: Recurrent neural network tandem language models integrating word-level information. The part under the dashed line is used for predicting meta-information on word level. The part above the dashed line is used for incorporating meta-information in the RNNLM. *The dashed arrows represent the direct connection of n-gram features in the input to the output.*

the full segment instead of only at the preceeding words (cf. Fig 4.1), a better prediction of the discourse-level information can be obtained.

We test under known and unknown conditions. In known conditions the information about the correct discourse-level meta-information category is available at test time. As such, this condition constitutes an oracle.

In unknown conditions the information must be predicted. In order to predict discourse-level meta-information, we train one sub-domain-specific RNNLM for each SSS or topic (also referred to as a component). This model is trained using the curriculum learning method for training domain-adapted RNNLMs. Our previous work has demonstrated the effectiveness of this method for creating RNNLMs for a heterogeneous domain that is composed of a number of sub-domains [143]. Curriculum learning [17, 48] makes use of the fact that neural networks are sensitive to the order in which data is presented to them during training.

By presenting the RNNLM first with general domain data and only later in the training phase with sub-domain data, we create models which emphasize the patterns in the sub-domain data. Curriculum learning can be regarded as a form of implicit interpolation between a domain model and a sub-domain model. It achieves the same goal as conventional linear interpolation, but does so with a single, continuously trained model that dispenses with the need to explicitly train weights of individual sub-models. Discourse-level labels, which are predicted at the segment level, are propagated to the word level in order to be integrated into the RNNLM.

The first type of discourse-level information that we consider is SSS. As previously mentioned, this information is captured at the time the speech is recorded and hence is available to train the language model.

The predicted discourse level meta-information $c(s)$ of a sentence s is derived from the probabilities returned by the different component models as follows:

$$c(s) = \arg\max_{k} p_k(s), \tag{4.9}$$

where $p_k(s)$ is the probability of segment s given by the kth component model. Each component model is an RNNLM, so the probability of segment s is calculated as follows:

$$p_k(s) = p_k(w_0)p_k(w_1|w_0)...p_k(w_t|w_0,...,w_{t-1}), \tag{4.10}$$

where $p_k(w_t|w_0,...,w_{t-1})$ is the output of the kth component RNNLM for word w_t.

For each word, the predicted discourse-level meta-information for the segment to which that word belongs is fed into the language modeling part of the RNNTLM and used to predict the next word.

The second type of discourse-level information considered in this work are topics. The topics are derived automatically from the training data using Latent Dirichlet Allocation (LDA) [20] in conjunction with k-means clustering. LDA is a probabilistic model that describes the generation process of documents. A document is considered to be a mixture of underlying topics that give rise to the words it contains. LDA applies a bag-of-words strategy, allowing each document to be represented as a latent topic vector whose components reflect the relative contributions of the individual latent topics. We choose to make use of LDA since it represents the state of the art in topic representations. We construct latent topic representations by considering each segment to be a document. We then apply k-means clustering in order to cluster the data. The result is a set of topic clusters. Each word in a segment bears the topic label of the cluster the segment belongs to.

4.5 Experimental Setup

In this section, we describe the setup used to carry out our experiments. We evaluate our language models using two types of experiments, N-best rescoring and Word Prediction.

4.5.1 Data

Our language model training and test data comes from the Spoken Dutch Corpus (Corpus Gesproken Nederlands, CGN) [111], which contains recordings of standard Dutch spoken by adults in the Netherlands and Flanders in a variety of language usage settings. As shown in Table 4.1, the entire corpus contains nearly 9 million words divided into 14 components. We used the component as a proxy for the socio-situational setting. Each component is further divided into segments that contain one or more sentences. Segments may be as large as 1,000 words.

Components *a* to *h* contain dialogues or multilogues and components *i* to *o* contain monologues. Our experiments are carried out on a test set that contains 10% of the data randomly selected from components *h*, *g*, *n* and *o*. The choice of these components was informed by practical considerations, which included the need to exclude the data used to train the acoustic models for the speech recognition system that generated the N-best list (further described in Section 4.5.4).

In total the test set contains 974*K* running words and 149 segments. For language model training, 80% of the CGN data, mutually exclusive from the test set, was used. Another mutually exclusive set of 10% of the data was used for validation.

4.5.2 Part-Of-Speech And Lemma Prediction

CGN provides (manually verified) Part of Speech (POS) tags and lemmas for each word [160]. There are 281 POS tags represented in the training data. The POS tags consist of a basic set (i.e., including ‘noun’, ‘adjective’, ‘verb’) enriched by further information. Examples of the further information include, for nouns, the type of noun (common noun or proper noun), the number (plural or singular), the degree (whether or not the noun is diminutive), and case (e.g., genitive or dative). The RNNLM trained to predict parts of speech achieves an accuracy of 93.5 when 180 hidden units are used. Changing the number of hidden units has negligible impact on the performance.

The process of lemmatization involves mapping the inflected forms of words, as they occur in text, to their basic forms, i.e., the way that the word would be cited in the dictionary. Several forms of a word map to the same basic form, for example, the singular and the plural of a word both map to the singular form. There are 84*k* lemmas (also pluralized 'lemmata') represented in our training data. Note that although many lemmas can be uniquely determined by inspecting the form of a word token, there exist some word tokens in the Dutch language that are ambiguous. In these cases, the context must be considered in order to determine the correct lemma. Because of the large number of lemmas that must be predicted, we use a class-based recurrent neural network [101] to speed up the training of the lemma predictor, the classes are determined according to the lemma frequency in the training data. Prediction proceeds in two steps. First, we predict the class using Eq. 4.11.

$$cl^* = \arg\max_{cl} p(cl|w), \tag{4.11}$$

Then, we predict the lemma according to the predicted class using Eq. 4.12.

$$lemma^* = \arg\max_{lemma} p(lemma|w, cl). \tag{4.12}$$

Using a hidden layer with 30 neurons, the RNN based lemma predictor achieves a 96.3% prediction accuracy. Increasing the number of hidden units causes the performance to decay slightly, attributable to the relatively smaller number of examples available per lemma in the training data. In the experiments, we test the 'known' condition (i.e., oracle condition) in which the POS or the lemma label for a word is known at test time, as well as the 'unknown' condition for which the labels are predicted at test time.

4.5.3 Socio-Situational Setting and Topic Prediction

Both SSS and topic we mentioned in this paper are simply ways of dividing the data set into subsets that can be helpful. The SSS was collected when the data was recorded (see Section 4.5.1). The topic information is derived by LDA and clustering was described at the end of Section 4.4.1. The topic we used in this paper may also contain other information (e.g. style). We refer the output from LDA and clustering algorithm as topic because these algorithms are generally used to find topics. Prediction of SSS and topic for the N-best lists returned by the recognizer is carried out using the RNNLM-based prediction method described in Figure 4.2. It is important to note that this method achieves good performance in predicting SSS. The average accuracy on the test data covers all of the components in the CGN corpus is 76.2%. Nine out of thirteen components achieve above 95% accuracy. The performance for the components that are used for N-best lists is: 98%, 96%, 93% and 100% for components *h*, *g*, *n* and *o*, respectively.

4.5.4 Generating The N-Best List

For the automatic speech recognition (ASR) experiments we used a Large Vocabulary Continuous Speech Recognition (LVCSR) system. The system, which is an updated version of [43], was built by ESAT using their state-of-the-art ASR toolkit SPRAAK [40, 42]. It was initially developed for the Dutch N-Best evaluation benchmark [74]. The system is a speaker independent speech recognizer that has the capability to select components and adjust parameter settings on the fly, based on observed conditions in the audio. The installation used here distinguishes between two conditions, studio quality broadband speech containing mainly prepared speech and telephone interviews which are expected to contain more spontaneous speech.

The acoustic models employ 49 three-state acoustic units (46 phones, silence, garbage and speaker noise) and one single-state phone (short schwa), which are modeled using SPRAAK's default tied Gaussian approach. Under this approach, the density function for each of the 4k cross-word context-dependent tied states is modeled as a mixture of an arbitrary subset of Gaussians drawn from a global pool of 50k Gaussians. The mixtures use on average 180 gaussians to model a 36 dimensional observation vector of MIDA features [40]. These were obtained by means of a mutual information based discriminant linear transform (MIDA) on vocal-tract length normalized (VTLN) and mean-normalized MEL-scale spectral features and their first and second order time derivatives. The acoustic models are trained on Broadcast News (components f, i, j, k, l in Table 4.1) and Conversational Telephone Speech (components c, d in Table 4.1).

Using a lexicon of 400k words, 5-gram language models (LMs) with modified Kneser-Ney discounting were trained on 4 main text components: 12 Southern Dutch newspapers, 10 Northern Dutch newspapers and transcriptions of broadcast news (component f, i, j, k, l in Table 4.1) and conversational telephone speech (component c, d in Table 4.1)(Northern Dutch refers to the Dutch spoken in the Netherlands; Southern Dutch refers to the Dutch spoken in Belgium). This training data set is exclusive with the testing data we used in all the experiments. The four LMs were interpolated linearly and perplexity minimization was done to find the optimal interpolation weights on the N-best development data. Lexicon creation was handled by an updated version of the system described in [41]. Dutch has a substantial amount of (regional) pronunciation variation, which was addressed by using phonological rules to generate the likely pronunciation variants. This resulted in a median of 3.8 pronunciations per word or 1.13 variants per phone in the canonical word transcriptions.

Since Dutch compounds are always written as a single word, the word recognition results are post-processed for compounding. Two subsequent words are replaced by their compound if the following criteria are met: 1) the words are longer than 3 letters, 2) the words are not very rare, 3) the unigram count of the compound is higher than the bigram count of the individual words. This approach effectively extends the 400k lexicon to a 6M lexicon.

The main parameters of the system control hypothesis pruning and combining the language model and the acoustic models. To combine the model scores, we employ our standard way of handling this problem [40], by having a LM scaling factor and a word startup cost. Beam search pruning was applied to control the amount of hypotheses in the search space [150]: a threshold indicates how much the score of a hypothesis can drop below the score of the most likely hypothesis; if most hypotheses have a similar score, a beam width parameter is applied to indicate how many hypotheses can be retained, keeping only the best ones.

Adopting the pruning parameters that yield recognition in real time, we create a lattice with the most likely word sequence hypotheses for each speaker turn in each component. Using SRI's lattice tool, each lattice is converted into an N-best list containing the 10000 best sentences, disregarding filler words and silences.

4.5.5 Re-scoring The N-Best List With The RNNLMs Integrating Meta-Information

The RNNLM models that we test in our experiments all use the maximum entropy extension (RNNMEs), as mentioned in Section 4.4. They use 300 hidden neurons and one weight matrix with 1 billion elements that directly connect input to output. All the models are trained using Backpropagation Through Time (BPTT) with 5 steps.

4.5.6 Evaluation Metrics

We evaluate our language models in terms of perplexity (PPL), word prediction accuracy (WPA), and word error rate (WER). Both the PPL and word prediction accuracy WPA are calculated using the language model directly. In other words, the speech recognition system, which is described in Section 4.5.4, is not involved in calculating these evaluation metrics. PPL is the geometric average of the inverse probability of the words on the test data. WPA [159] is a practical measure of language models. It is defined as the accuracy achieved when the language model is provided with information about preceding words and required to predict the word that would occur next. Word prediction is important for natural language processing tasks, such as spelling correction and auto completion. WER is evaluated by carrying out a rescoring experiment that takes as input the N-best list generated by the speech recognition system.

4.6 Experimental Results

In this section, we present the results obtained with our proposed approach for integrating meta-information into RNNLMs. We compare our models to two baselines, KN5GRAM, which is a conventional Kneser-Ney 5-gram language model and also RNNME, which is the same RNNLM that we use in our approach, except for the fact that it does not integrate any meta-information. A conventional Kneser-Ney 5-gram language model achieves a WER of 40.1 on our data set.

First, we discuss our experiments that integrate word-level information (POS and lemmas). The results are reported in the lines labeled 'POS' and 'lemma' in Table 4.2 for two test conditions: the 'known' condition, which is an oracle condition under which the information is known at test time, and the 'unknown' condition, under which the RNNTLM architecture in Fig 4.2 is used, to predict the information at test time.

Table 4.2: Perplexity (PPL), word prediction accuracy (WPA) and word error rate (WER) results with one feature under the condition that meta-information is unknown and known during testing.

Model	Known		Unknown		
	PPL	WPA	PPL	WPA	WER
KN5GRAM	140	-	140	-	40.1
RNNME	112	21.3	112	21.3	38.7
POS	97	22.8	104	22.0	38.9
lemma	109	21.8	114	21.7	38.3
SSS	105	22.2	107	22.1	37.8
T30	96	22.9	118	21.3	38.6
TS	110	21.7	110	21.7	38.2
SL	109	21.9	109	21.9	37.9

Looking at the WPA for the 'known' and the 'unknown' conditions, we see that both improve over the RNNLM baseline. The WPA deteriorates when the meta-information must be predicted at test time (i.e., under the 'Unknown' condition). However, the difference is relatively small, reflecting the strong performance of the underlying meta-information predictors in the RNNTLM. In the lemma case, the improvement translates into a small improvement in WER when rescoring, however, adding POS information deteriorates rather than improves WER performance. In conclusion, the contribution of word-level information is very modest and quite fragile. However, our results suggest that errors introduced by meta-information prediction do not have a large impact over the theoretical performance achievable under the oracle situation.

Next, we turn to the experiments that integrate discourse-level information (SSS and topic). For the model integrating information on SSS, we see that whether SSS labels are known at test time, or must be predicted has relatively little impact on the WPA. In both cases, the integration of SSS information achieves an improvement in WPA over the baseline RNNME. This improvement also translates into a reduction in WER in the N-best rescoring experiment. Comparing the case of SSS to the model integrating topic information (reported in the line labeled ‘T30’ in Table 4.2), we notice that topic information is more difficult to exploit. In the oracle case in which topics are known at test time, an improvement in WPA can be achieved. However, the improvement that the integration of topics offers in the ‘unknown’ conditions is very slim, noticeable in the WER, but not the WPA. Note that we know the topic labels for the oracle condition because the topics were created by clustering all the data simultaneously into 30 topics. Hence the topics of the oracle condition were created on more data (all data in one segement of the CGN database) than the topics of the ‘unknown’ condition (one ‘sentence’ as returned by the N-best list module). As shown in Table 4.3, the improvement offered by integrating topics varies somewhat with the number of topics chosen, reaching its maximum with 30 topics. This result suggests that the optimization of the number of topics is an aspect of meta-data information that must be taken into consideration when integrating topic as meta-information into an RNNLM.

*Table 4.3: Perplexity, word prediction accuracy (*WPA*) and word error rate (*WER*) results using different numbers of topics, conditioned on that meta-information is unknown and known during testing.*

Model	Known		Unknown		
	PPL	WPA	PPL	WPA	WER
T10	102	22.5	121	20.7	38.9
T20	99	22.7	126	19.7	39.7
T30	96	22.9	118	21.3	38.6
T40	98	22.7	121	21.0	38.7

In sum, the results suggest that SSS has potential as a source of meta-information to improve RNNLMs and if it has been captured at recording time, it can be used, either directly on the test data, or for training SSS predictors. In cases in which no information has been captured at recording time, topic discovery can be applied, but it seems that it is only worth the effort in cases in which enough data is available to create high quality clusters.

Table 4.2 shows that under the oracle situation, both topic and POS improve the RNNME the most when looking at WPA. However, neither topic nor POS help in speech recognition N-best rescoring. This contradiction reveals that the small errors in topic and POS prediction impact the performance of language models.

Finally, we turn to the topic of integrating 'intrinsic' meta-information into RNNLMs. These results are reported in the line labeled TS (token size) and SL (sentence length) in Table 4.2. Recall that intrinsic meta-information is particularly interesting since its use has been largely overlooked in the literature on conventional language models. It is 'free' information in the sense that it can be derived directly, without the need for prediction. Both with respect to WPA and with respect to WER, intrinsic meta-information is able to achieve performance improvement over the RNNME baseline. The performance is slightly more in the case of SL than in the case of TS. In conclusion, these results suggest that intrinsic information, although trivial to derive, should not be considered trivial when it comes to integrating meta-data into RNNLMs. Instead, this sort of 'free' information should be exploited. It is capable of yielding a performance improvement of the same magnitude of the one attainable by more costly methods that require the training of a meta-information predictor.

We finish this section by noting the potential of combining multiple sources of meta-information in order to achieve further improvement. Our experiments revealed, for example, that combining predicted SSS information with SL information yields a WER of 37.7%, a full 1% absolute improvement over the RNNME baseline.

4.7 Conclusions

In this paper, we investigated the integration of meta-information into RNNLMs. We looked at three cases, the integration of word-level information using a Recurrent Neural Network Tandem Language Model (RNNTLM) architecture, the integration of discourse level information, and the integration of 'intrinsic' information, which can be derived directly without prediction.

Our results yield interesting insights. First, we noted that performance improvement that can be attained by using word-level meta-information is limited. However, the source of these limitations appears to be a function of the information itself. Errors introduced by the prediction of word-level information do not appear to be the limiting factor.

Second, we found that information on SSSs improves the performance. Again here, prediction is not a limiting factor. Our results revealed that discourse-level information is very difficult to exploit effectively in cases in which discourse-level information is not recorded at the time the data was captured, automatically derived topics showed close to no

*Table 4.4: Perplexity, word prediction accuracy (*WPA*) and word error rate (*WER*) results with two or more features under the condition that meta-information is unknown and known during testing.*

Model	Known		Unknown		
	PPL	WPA	PPL	WPA	WER
POS +SSS	94	23.0	**102**	22.0	39.1
POS +SL	99	22.6	106	**22.1**	38.8
POS +lemma	96	22.7	107	**22.1**	38.7
POS +T30	89	23.6	115	21.8	39.2
POS +TS	97	22.7	108	21.9	38.3
SSS +T30	94	23.1	119	21.4	38.6
lemma+T30	97	23.0	113	21.6	38.8
TS +SL	110	21.8	110	21.8	38.2
SSS +SL	105	21.7	110	21.7	**37.7**
SSS +lemma+TS	104	22.3	106	22.3	**37.6**
POS +SSS +T30	85	**24.1**	109	22.0	38.3
POS +lemma+T30	88	23.7	115	21.6	38.4
POS +SSS +SL +lemma+TS +T30	**84**	23.9	105	21.8	38.4

improvement.

Finally, we have noted that the contribution that can be made by ‘intrinsic’ meta-information should not be overlooked. In fact, information sources such as word and sentence length, which are trivial to derive, can make a contribution to RNNLMs that rivals that of meta-information that must be predicted. The best results can be achieved by judicious combination of intrinsic and predicted meta-information sources, with the combination of SSS and sentence length achieving a full 1% absolute improvement over the RNNME baseline.

Chapter 5

Exploiting the succeeding words in Recurrent Neural Network Language Models [1]

5.1 Abstract

In automatic speech recognition, conventional language models recognize the current word using only information from preceding words. Recently, Recurrent Neural Network Language Models (RNNLMs) have drawn increased research attention because of their ability to outperform conventional n-gram language models. The superiority of RNNLMs is based in their ability to capture long-distance word dependencies. RNNLMs are, in practice, applied in an N-best rescoring framework, which offers new possibilities for information integration. In particular, it becomes interesting to extend the ability of RNNLMs to capture long distance information by also allowing them to exploit information from succeeding words during the rescoring process. This paper proposes three approaches for exploiting succeeding word information in RNNLMs. The first is a forward-backward model that combines RNNLMs exploiting preceding and succeeding words. The second is an extension of a Maximum Entropy RNNLM (RNNME) that incorporates succeeding word information. The third is an approach that combines language models using two-pass alternating rescoring. Experimental results demonstrate the ability of succeeding word information to improve RNNLM

[1]This chapter is a lightly modified version of the paper appearing in the Proceedings of Interspeech 2013. Y. Shi, M. Larson, P. Wiggers, C. M. Jonker. Exploiting the succeeding words in Recurrent Neural Network Language Models [144]. We would like to thank Xunying Liu and Geoffrey Zweig for their comments.

performance, both in terms of perplexity and Word Error Rate (WER). The best performance is achieved by a combined model that exploits the three words succeeding the current word.

5.2 Introduction

Most statistical language models decompose the probability of a word sequence into a product of conditional probabilities of each word given its history, i.e., the preceding words. The conventional n-gram language models use the previous $n-1$ words as history. State-of-the-art recurrent neural networks language models (RNNLM) [98, 101] theoretically can use word information from arbitrarily long distances. However, conventional language models use the assumption that the word w_i is dependent only on words preceding it. In fact, any given word within a sentence can be considered to be conditioned not only on preceding words, but on succeeding words as well. For example, in the sentence "the dog barks", the occurrence of the word "dog" is strongly correlated with the occurrence of the succeeding word "barks". The conditional probability of "dog" given "the" is much smaller than the conditional probability of "dog" given the succeeding word "barks". This example demonstrates the predictive potential of succeeding words, and motivates our use of both preceding and succeeding words in language models.

This study investigates methods for integrating the predictive ability of succeeding words of a sentence into RNNLM. Recent studies [98, 99, 101] have demonstrated that RNNLMs can outperform the conventional n-gram language models. RNNLMs offer a unique opportunity for exploiting succeeding words. Due to considerations of high computational complexity, in most applications for automatic speech recognition, RNNLMs must be applied during N-best rescoring. In other words, RNNLMs are, in practice, used to deal with complete sentences or utterances. For this reason, information from succeeding words is available to be exploited by RNNLMs.

This paper proposes new variety of RNNLMs that extend the ability of conventional RNNLMs to integrate long-distance information from preceding words to also include information about succeeding words. Three approaches are discussed in this paper. The first is a forward-backward model that combines RNNLMs exploiting preceding and succeeding words. The second is an extension of a Maximum Entropy RNNLM (RNNME) that incorporates succeeding word information. The third is an approach that combines language models using two-pass alternating rescoring.

The forward-backward model is the aggregation of a conventional forward RNNLM with a backward RNNLM at the sentence level. In the second method, the present word w_t and the succeeding words w_{t+k} ($k \geq 2$) are used as input to the RNNLM to predict the next word w_{t+1}.

In this case, the recurrent hidden layer in the RNNLM cannot be used to store information from the succeeding words, because the information in the hidden layer will be further used to *predict* succeeding words. For this reason, we propose a maximum entropy model that is applied to combine the succeeding words by directly connecting the input to the output.

The rest of the paper is organized as follows. The next section covers relevant related work. Section 3 introduces our approaches: forward-backward RNNLM, RNNME with succeeding words, and the two-pass alternating rescoring. Section 4 presents the results of the evaluation of these models in term of perplexity and word error rate. Based on the experiment results, Section 5 presents the conclusions.

5.3 Related work

In 2003, Bengio *et al.* [16] proposed a feed-forward neural networks language model, in which they projected the vocabulary to a continuous feature vector space and directly applied the previous $n-1$ word feature vectors as the input. Recurrent neural network language models [98, 101] avoid this explicit modeling of the word history, by copying the previous hidden layer into the current input layer. This equips the language model with a compact memory to store previous long term history information. Additionally the RNNLM maps the discrete vocabulary to a continuous space, which allows the language model to better handle sparse data. In this paper, taking advantage of the fact that information from the entire sentence is available when an RNNLM is applied for rescoring, we propose three ways of integrating succeeding words into the language model in order to improve the performance of the RNNLM.

One way to use the succeeding words in language models is to apply a forward-backward modeling strategy. The bidirectional strategy has been applied in phoneme classification [123, 133]. In language modeling, Duchateau *et al.* [46] showed that both the forward and backward smoothed n-gram language models yielded nearly the same perplexity. However, they did not use the backwards model to improve the word error rate (WER) performance of n-gram language models in an automatic speech recognition task. Instead, they illustrated the performance of backwards n-gram confidence measure using the metric described by [147]. When a confidence measure is combined with the score from a backward language model, a significant increase in terms of cross entropy was observed. The practical usage of the backward n-gram language models for machine translation was demonstrated by Finch and Sumita [53]. In this work, the scores from the forward decoder and the backward decoder are linearly interpolated with equal weights. Their bidirectional decoding got substantial improvement over 272 different language pairs from 17 languages. The improvement that can be achieved in statistical machine translation by using a backward *n*-gram model was demonstrated by Deyi Xiong *et al.* [169], who proposed to use a backward *n*-gram language model and a mutual information trigger models to enhance language models in phrase-based statistical machine translation. The novelty of our work is that we propose a forward and backward strategy for RNNLM to the task of automatic speech recognition, and specifically to rescoring N-best lists.

One of the methods that we propose to exploit succeeding words in RNNLMs makes use of maximum entropy language models. These models have been applied to model the whole sentence in speech recognition [127]. Each sentence is represented as a "bag of features". In fact, maximum entropy language models can be viewed as neural networks with no hidden layer. Instead, they use a weight matrix directly connecting the input to output layer [100]. Mikolov *et al.* [100] proposed to apply the maximum entropy model to *n*-gram feature histories and use the result to update output of the RNNLM. However, they do not take advantage of the succeeding-word information. In this paper, we extend the use of maximum entropy language models in the RNNLM with succeeding words.

As already mentioned, our proposed approaches for integrating information from succeeding words is applied in speech recognition, for N-best rescoring. As is discussed in [4], it becomes computationally challenging to incorporate language models that capture long distance dependencies into a speech recognizer. Two-pass N-best rescoring is currently the method most widely used to apply RNNLMs to the speech recognition. The standard rescoring approach was proposed in [112]: one system computed the N-best hypotheses, a second system rescored these hypotheses, and all the scores were combined to improve the overall performance. Subsequently, in [152] the standard approach was modified to optimize the combination weight with regards to average WER. In this paper, we propose a two-pass alternating rescoring method for rescoring the N-best list. The advantage of our method is that it helps to protect against the danger that combination weights for different models falling into a local optimum.

5.4 RNNLM with succeeding words information

Our approaches are based on a state-of-the-art language model–RNNLM. The RNNLM has an input layer x, a hidden layer h and an output layer y. Each time t, the input vector $x(t)$ consists of the current word vector $w(t)$ as well as a copy $h(t-1)$ of the previous hidden neurons. The activation function in the hidden layer is a sigmoid function. In the output layer, the activation function is a softmax function. The weight matrix between layers is estimated by back-propagation-through-time [100, 163].

In this section, we present our proposed approaches: a forward-backward RNNLM, an extension of RNNME with succeeding words, and an additional two-pass alternating rescoring method for decoding that integrates succeeding words into sentence hypotheses scoring.

5.4.1 Forward-Backward RNNLM

Given a word sequence $W = w_1, w_2, ..., w_n$, the RNNLM assigns the probability to W as follows.

$$p(W) = \prod_i p(w_i|h_i), \tag{5.1}$$

where h_i is the word history of word w_i. As it is shown in the Equation 5.1, the RNNLM predicts a word in the forward direction. We refer to this RNNLM as the forward RNNLM.

Different from the forward RNNLM, the backward RNNLM assigns the probability to W in the reverse direction.

$$p(W) = \prod_i p(w_i|s_i), \tag{5.2}$$

where s_i is the set [2] of succeeding words of w_i in the word sequence W. The training of the backward RNNLM uses the same algorithm as the forward RNNLM, but the order of the words in the sentence is reversed during training.

There are many possibilities to combine the forward RNNLM and backward RNNLM. This paper studies the following heuristic methods:

1. Sentence level linear interpolation (SI):

$$p(W) = \lambda_b \prod_i p(w_i|s_i) + \lambda_f \prod_i p(w_i|h_i). \tag{5.3}$$

2. Word level geometric interpolation (WG)[3]:

$$p(W) = \prod_i (p(w_i|s_i)^{\alpha_b} p(w_i|h_i)^{\alpha_f})^{\frac{1}{\alpha_b+\alpha_f}}, \tag{5.4}$$

where $\alpha_b \geq 0$ and $\alpha_f \geq 0$.

3. Sentence level Maximization (SM):

$$p(W) = \max\{\prod_i p(w_i|s_i), \prod_i p(w_i|h_i)\}. \tag{5.5}$$

[2]Not just the set (bag of words), but also the sequence of words are taken into consideration.

[3]It is actually also sentence level log linear combination.

Method 1 treats the sentence as the basic unit which is scored by the forward RNNLM as well as the backward RNNLM. Methods 2 and 3 may not generate the proper probabilities. However, in perspective of N-best rescoring, method 2 simply is the linear interpolation of language model logarithm scores. The interpolation weights in method 1 and 2 are tuned on the development data set. Exploratory experimentation revealed that linear interpolation deteriorated the performance, supporting the conclusion that the logarithmic scale is the most appropriate for interpolating the forward and backward probabilities. Method 3 actually is an extreme case of method 2 under which one weight is zero in the interpolation.

5.4.2 Maximum Entropy Model Extension in RNNLM

In a maximum entropy model, the conditional probability of the current word given the history features works as follows:

$$p(w|h) = \frac{\exp \sum_{i=1}^{N} \lambda_i f_i(h,w)}{\sum_w \exp \sum_{i=1}^{N} \lambda_i f_i(h,w)}, \tag{5.6}$$

where f_i is one feature, λ_i is the weight for feature i and h is the history of features. In fact, it can be viewed as a neural network language model, with the difference that the features are continuous value and automatically learned from the history. Mikolov [100] implemented the RNNME as an extension of RNNLM integrating the maximum entropy model that takes advantage of the preceding n-gram features.

In this paper, we extend the RNNME with the succeeding words as it is depicted in Fig 5.1. The preceding word w_t is treated as the input to recurrent neural networks as well as the feature in the maximum entropy model. As we can see in Fig. 5.1, the succeeding words s are connected by a dashed line to the output, which indicates that the succeeding words are only treated as features to maximum entropy model. With the succeeding information, the maximum entropy model has the following form:

$$p(w|c) = \frac{\exp(\sum_i \lambda_i f_i(h,w) + \sum_j \lambda_j g_j(s,w))}{\sum_w \exp(\sum_i \lambda_i f_i(h,w) + \sum_j \lambda_j g_j(s,w))}, \tag{5.7}$$

where c contains the preceding and succeeding information. s are the succeeding word features.

5.4.3 Two-Pass Alternating Rescoring

In addition to extending the RNNLM with succeeding words, we propose a two-pass alternating rescoring method to combine the conventional RNNLM with the RNNLM with succeeding words in speech recognition. In the N-best rescoring paradigm [112], the combination weight is learned from held-out data, and then applied to the evaluation data. Finding the optimal combination weight for each model is an unconstrained nonlinear global optimization problem. The resulting optimum also determined using held-out data is typically a local optimum. In order to avoid this situation, the two-pass alternating rescoring strategy uses a filtering method to solve an optimization problem. As is discussed in [151], a global optimization problem is better treated as a filtering problem when the objective function cannot be exactly evaluated.

The two-pass alternating rescoring strategy works as follow. For two different language models m_1 and m_2, N-best hypotheses N and ratio $\alpha \in (0,1)$,

1. The language model m_1 is used to rescore the N-best hypotheses N, and the top α portion of the best hypotheses are selected. $N := \alpha * N$,

2. The selected hypotheses are rescored by m_2. The size of the N-best list is reduced again, yielding a new $N := \alpha * N$. Repeat begining with step (1) until only one hypothesis is left, the best one.

In fact, at each iteration, a language model provides an optimized search space for the next language model. An advantage is that this strategy can be directly applied to the target data, as it does not require predetermined weights, which would need to be learned from an additional source of data (i.e., held out data).

5.5 Experiments

The experiments are based on the Wall Street Journal (DARPA WSJ'92 and WSJ'93) data sets, which is the same data sets used by Mikolov in [98]. The training corpus contains 37M words from the New York Times section of the English Gigaword set with a vocabulary of 195K words. An independent set of 230K words is used for testing (i.e., measuring perplexity).

In the 100-best list set, 333 sentences are used as development data for tuning the interpolation weights language model scores and the acoustic model score. The rest, 465 sentences, is used for evaluation.

Table 5.1 shows the perplexity and WER results of the baseline models and the proposed models. 'WER rescore' designates the WER for the individual models. 'WER alternating' is the word error rate after applying the two-pass alternating rescoring strategy to combine the conventional RNNLM and the model indicated in the row.

There are four parts in Table 5.1. The top part is the results of the Kneser-Ney 5 order *n*-gram language model (KN5) and the forward (conventional) RNNLM. A 5-gram model

Table 5.1: Word error rate results on WSJ with 100 hidden neurons.

model	WER(%) rescore	WER(%) alternating
KN5	17.30	-
RNNLM	16.83	-
RNNLM-B	16.55	16.03
RNNLM-SI	16.38	16.18
RNNLM-WG	**15.69**	**15.44**
RNNLM-SM	16.56	16.27
RNNME-P3	15.23	14.98
RNNME-S1	16.60	16.21
RNNME-S2	16.55	16.18
RNNME-S3	16.47	16.18
RNNME-P3S3	**15.07**	**14.97**
RNNME-B-P3S3	15.53	15.30
RNNME-SI-P3S3	15.24	15.04
RNNME-WG-P3S3	**14.62**	**14.44**
RNNME-SM-P3S3	15.48	15.24

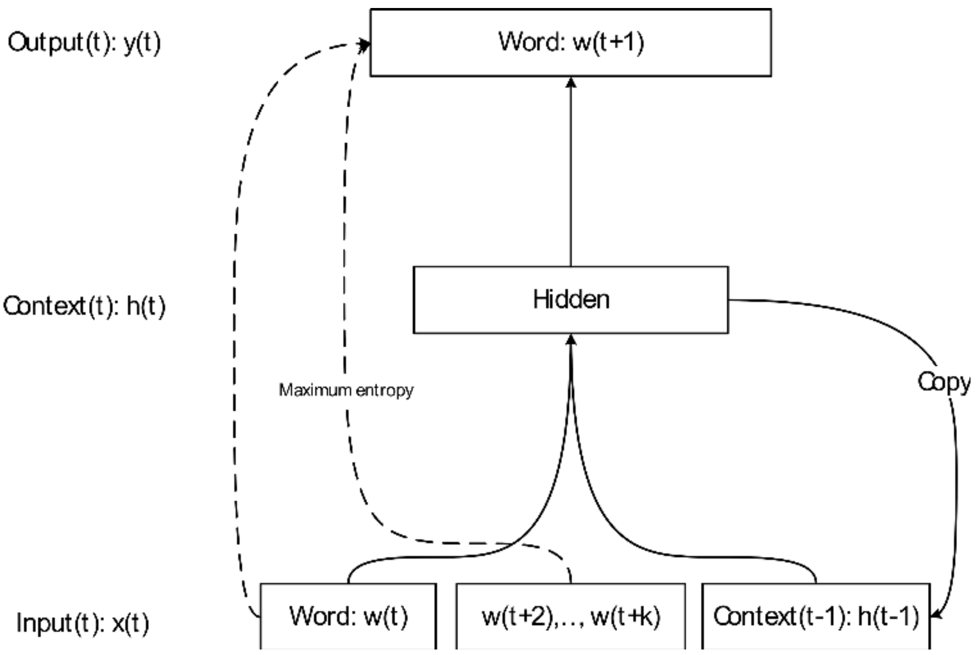


Figure 5.1: RNNME combine with succeeding words. w_t denotes the preceding word, $w_{t+2},...,w_{t+k}$ the succeeding word. Dash represent that the preceding word and the succeeding words are treated as features in maximum entropy langauge models.

has been shown to provide good n-gram model performance on this data [52] and is used in [98, 99, 101]. The second part gives the results from the backward RNNLM as well as three different combinations of the backward RNNLM with the forward RNNLM. The results from the RNNME and an extension of RNNME with succeeding words are given in the third part of the table. The bottom part of the table shows the performance of the forward and backward strategy in the RNNME with succeeding words. All the RNNLM models listed in the table use 100 classes and 100 neurons in hidden layer. The weights of these models are trained by 4 times backpropagation through time (BPTT) with block size 10. In two-pass alternating rescoring, the decay ratio α is 0.9. In RNNME, all the models use 1 billion parameters to represent preceding *n*-gram and succeeding words.

In the second part of Table 5.1, the results show that all the forward-backward RNNLM improve over the RNNLM in terms of WER. The individual backward RNNLM (RNNLM-B) achieves similar perplexity as the conventional RNNLM, but it reduces the WER by 0.28% absolutely. Among the three forward-backward combination strategies, the word level geometric interpolation performs best. Used alone, it achieves 1.14% absolute reduction in

WER. The additional two pass alternating rescoring further increases the reduction to 1.39%.

The third part of Table 5.1 gives the results from the RNNME and RNNME with succeeding words. The 'RNNME-P3' designates a conventional RNNME that uses the preceding three words. The 'RNNME-S3' designates an RNNME that uses the succeeding three words. The 'RNNME-P3S3' designates an RNNME that uses both the preceding and the succeeding three words. The results indicate that using the maximum entropy model to integrate succeeding words reduces the WER of the RNNLM. The RNNME with preceding and succeeding words performs best, achieving a 1.86% absolute reduction in WER using the additional two-pass alternating rescoring.

The bottom of Table 5.1 shows the results of combining the forward-backward strategy with the maximum entropy model in the RNNLM. The RNNME with the preceding and succeeding three words using word level geometric interpolation achieves the best result. It reduces the WER from 16.83% to 14.62%.

Table 5.2 shows the results based on the neural networks with 200 hidden neurons. In this table, we only focus on the combination of the forward-backward strategy and RNNME with succeeding words, which obtains better performance than conventional RNNLM. However, 'RNNME-WG-P3S3' only gets small improvement over itself with 100 hidden neurons. With an increased hidden layer size, both the forward-backward strategy and the maximum entropy method are more effectively in exploiting succeeding information. The advantage of the combination nearly vanishes.

To sum up, the three approaches proposed here to integrate succeeding words with RNNLM all achieve improvement. In language model training, in terms of WER, the maximum entropy modeling performs better than a forward-backward strategy. However, the forward-backward strategy in practice is more efficient than maximum entropy modeling: under the forward-backward strategy, both forward RNNLM and backward RNNLM can be trained in parallel, as they do not share parameters with each other. In RNNME, both the succeeding words and preceding words are treated as features of the whole sentence, which share the same huge size of hash vector. As we see from the two tables, in the decoding, the additional two-pass alternating rescoring also helps to reduce the WER.

5.6 Conclusion

In order to make use of information both preceding and succeeding the present word in recurrent neural network language models for automatic speech recognition, three different approaches were proposed in this paper. They were a forward-backward RNNLM, a RNNME with succeeding words in language model training, and the two-pass alternating rescoring

Table 5.2: Word error rate results on WSJ with 200 hidden neurons

model	WER(%) rescore	WER(%) alternating
RNNLM	15.48	-
RNNME-P3	14.91	14.89
RNNME-B-P3	15.02	14.91
RNNME-SI-P3	14.68	14.70
RNNME-WG-P3	**14.75**	**14.51**
RNNME-SM-P3	14.88	14.83
RNNME-P3S3	15.10	15.01
RNNME-B-P3S3	15.14	14.98
RNNME-SI-P3S3	14.75	14.54
RNNME-WG-P3S3	**14.78**	**14.40**
RNNME-SM-P3S3	15.02	14.80

in speech decoding. As variants of the forward-backward RNNLM, we implemented word level geometric interpolation, sentence level linear interpolation and sentence level maximum selection to combine the forward RNNLM with backward RNNLM. The word level geometric interpolation achieved the best results. Individually, with a hidden layer of size 100, it achieved an absolute WER reduction of 1.14%. It obtained an extra 0.25% of absolute reduction when it was combined with the RNNLM using the two-pass alternating rescoring approach. The integration of the RNNLM with succeeding words using maximum entropy (RNNME) achieved an even larger improvement than the forward-backward RNNLM. When the RNNME integrated the succeeding three words and used 100 hidden neurons the WER was reduced by 11% relative to that of RNNLM. The combination of the three approaches obtained the highest performance over RNNLM. It reduced the WER from 16.83% to 14.44%. With an increased hidden-layer size, the combination of the three approaches still yielded the best performance. However, it did not get noticeable improvement over itself probably because of the duplication in the exploiting of succeeding word information.

This work has also opened some interesting questions for future investigation. The comparison of Table 5.1 and Table 5.2 reveals that with a larger hidden-layer size, the improvement effect over the baseline model becomes smaller. We note that integrating information from succeeding words information in RNNME also increases memory consumption. In order to avoid hash table collision, a large hash vector has to be used. Our future work will investigate techniques to address this challenge. Furthermore, we will study the performance of our proposed method under conditions in which the amount of training data is severely limited. Under such conditions, we anticipate that the contribution of succeeding words could be especially useful.

5.7 Acknowledgement

Many thanks to Tomas Mikolov for making the RNNLM Toolkit publicly available and for helpful discussions.

Chapter 6

Speed Up of Recurrent Neural Network Language Models[1]

6.1 Abstract

Recurrent neural network based language models (RNNLM) have been demonstrated to outperform traditional *n*-gram language models in automatic speech recognition. However, the superior performance is obtained at the cost of expensive model training. In this paper, we propose a sentence-independent subsampling stochastic gradient decent [2] algorithm (SIS-SGD) to speed up the training of RNNLM using parallel processing techniques under the sentence independent condition. The approach maps the process of training the overall model into stochastic gradient descent training of submodels, of which the update directions are aggregated and used as the weight update for the whole model. In the experiments, synchronous and asynchronous SIS-SGD are implemented and compared. Using a multi-thread technique, the synchronous SIS-SGD can achieve a 3-fold speed up without losing performance in terms of word error rate (WER). When multi-processors are used, a nearly 11-fold speed up can be attained with a relative WER increase of only 3%.

[1]This chapter is published in Interspeech 2013. Y. Shi, M, Hwang, K. Yao, M. Larson. Speed Up of Recurrent Neural Network Language Models With Sentence Independent Subsampling Stochastic Gradient Descent [142]. A few supplementary remarks are provided as footnotes.

[2]Erratum: "decent" should be "descent"

6.2 Introduction

Statistical language models play a crucial role in applications such as automatic speech recognition, machine translation, spelling correction. They model probability distributions over all possible word sequences in a language. Conventional *n*-gram language models have dominated automatic speech recognition for years due to their simplicity, good performance and robustness. However, they suffer in face of sparse data and their ability to capture long-distance dependencies is insufficient.

It has been shown that both of these problems can be addressed by recurrent neural network language models (RNNLM) [98, 101]. Conventional *n*-gram language models are discrete in nature, and despite of smoothing techniques [31, 76], their ability to generalize remains limited. In contrast, better generalization can be achieved by RNNLMs, which map a discrete word to a point in a continuous space. In most practical applications, *n*-gram language models predict the next word by conditioning only on two or three previous words. However, in RNNLM, the recurrent connections between the input layer and hidden layer theoretically can allow the information to cycle for an arbitrary length of time [98].

However, RNNLMs achieve their performance at the cost of expensive training. It is also well known that it is difficult to speed up a RNNLM with parallel processing without degrading performance because the SGD algorithm needs to update the weight of RNNLM word by word. Because of the recurrent structure of an RNNLM, it is necessary to copy the activations of the hidden layer from the previous word to the current input layer. In order to support the copying process, RNNLMs are trained sequentially.

In order to make the parallel training of RNNLM possible, we propose a sentence-independent subsampling stochastic gradient descent method (SIS-SGD) in this paper. In SIS-SGD, we constrain RNNLM by imposing the condition of sentence independence. The novel contribution of our approach is that it imposes a constraint on the history of a word in a way that is shown to both enable parallelization and also retain the ability of RNNLM to benefit from information cycling in the network. A second contribution is the trick of subsampling to minimize perplexity increase.

This paper is organized as follows. In the next section, we present the relevant related work. In Section 6.4, we explain the details of the SIS-SGD approach. In Section 6.5, we use the Penn Treebank and the Wall Street Journal data sets to show its effectiveness. The final section gives the conclusion.

6.3 Related work

In this section, we discuss the NNLM, RNNLM and SGD algorithms and speed up strategeies that have been applied in the literature. Bengio *et al.* [16] proposed a neural probabilistic language model using feed forward neural networks, in which the input is the preceding $n-1$-words. Their experiments showed that a NNLM can yield lower perplexity than conventional smoothed n-gram language models. Even though NNLM is easier to speed up by parallel processing [80, 134], compared with RNNLM, NNLM has two drawbacks. One is that the length of preceding context information that NNLM uses to predict next word is fixed. It is usually constrained to five to ten words, which is less the context used for RNNLM. [3] NNLMs are shallower than RNNLMs in terms of contextual history. The recurrent hidden layer effectively equips RNNLM with multiple hidden layers, which can learn more abstract representation of the input. This effect is confirmed by the fact that an NNLM is able to approach the performance of an RNNLM, when additional hidden layers are added [7].

Most of the previous studies on speeding up NNLMs have focused on reducing the learning of the weight matrix from the hidden layer to output layer. In [135], the output of NNLM was constrained to a short list of most frequent words. Bengio *et al.* [15] used an adaptive importance sampling strategy to reduce the computation. Xu *et al.* [171] also used a subsampling strategy, but converted the multi-class prediction problem to a binary class prediction problem. However, the subsampling strategy is influenced by noisy samples, which make the model training unstable and difficult to adjust. Alternative to subsampling, [104] proposed to use noise contrastive estimation to training NNLM. In [101], Mikolov proposed the class-based RNNLM which used a simple and stable class trick to factorize the output layer in RNNLM. These methods already reduced the computation of the weight learning between the hidden layer and output layer to less than 1%. This class trick has been adopted in NNLM parallel training by [80]. However, the parallelization in RNNLM has not been addressed due to the constraints from recurrent neural networks and the SGD. In this paper, we focus on the speed up of class-based RNNLMs via parallelization.

The online stochastic gradient descent (SGD) algorithm [81] has been extensively applied in neural networks, especially for large scale problems. Recently, [33] applied the Map-Reduce for SGD parallelization. Under Map-Reduce, a master machine distributes the computation of gradients to multiple slave machines, and then aggregates all the gradients to perform a global update. However, this algorithm requires many passes of scanning through the training data and many synchronization sweeps for convergence. In [92], the

[3] Erratum: "which is less the context used for RNNLM" should be "which is less than the context used by RNNLM".

authors proposed to use a full batch SGD in each slave machine. Each slave machine thus holds the data necessary to compute one update direction. In the end, the master machine averages all the directions to get a final update direction. In this way, it significantly reduces the cost of communication among processors. However, in each processor full batch SGD was used and in practice the resulting perplexity performance was much worse than the single machine SGD[4]. In [44, 45, 174], a similar parallel strategy for the deep neural network learning is used. The parallel SGD proposed in [177] improved the algorithm in [92] by using online SGD in each slave processor rather than the full-batch SGD. In this paper, we also use a similar algorithm to [177], in which each slave uses SGD to get an update direction, the master aggregates all the directions to perform one iteration of parameter update. One major difference in our SIS-SGD is that it uses subsampling instead of partitioning of the training data. Our experimental results reflect the importance of subsampling.

In order to reduce the latency among processors, the paper [1, 38, 80, 120, 176] proposed a variant parallel SGD algorithm—the asynchronous SGD—in which each slave performs SGD to calculate each direction independently of the others and updates the shared parameters on the master asynchronously. Specifically, each slave updates a delayed parameter vector. In [176] it was shown that the large delay, due to the fact that each slave handles a significant subset of the training data, in fact degraded the model quality. In other words, it is better if the slave can send parameter update directions to the master regularly after a small number of SGD steps are performed. In many applications, such overhead is acceptable. However, in RNNLM, the model has a huge number of parameters. The frequent communication of parameters among processors will dilute the gain from parallel processing. The parallelization approach in this paper attempts to minimize the overhead. [5]

6.4 Sentence Independent Subsampling Stochastic Gradient Descent Algorithm

RNNLMs [98] generally have three layers: an input layer x, a hidden layer h and an output layer y. At each time t, the input vector $x(t)$ is constituted by the current word vector $w(t)$ as well as a copy $h(t-1)$ from the previous hidden neurons. The sigmoid function and softmax function are used as the activation functions in the hidden layer and output layer, respectively. In [101], the output layer consists of two groups of output units. The first group represents the vocabulary; hence there are V output units if V is the vocabulary size (UNK can be included). The second group represents classes; usually there are about 100-

[4] Here SGD means single machine sequential SGD.

[5] At the same time with no or minimum degradation on model quality.

200 class units used. For each input word $w(t)$, it is mapped to a unique class, which is connected to a constrained, fixed subset of output units in the first group that possibly can be activated and thus, computed. [6]

In the traditional RNNLM, the matrix weights between different layers are trained according to the stochastic gradient descent algorithm. To apply RNNLM to large data sets, we propose SIS-SGD.

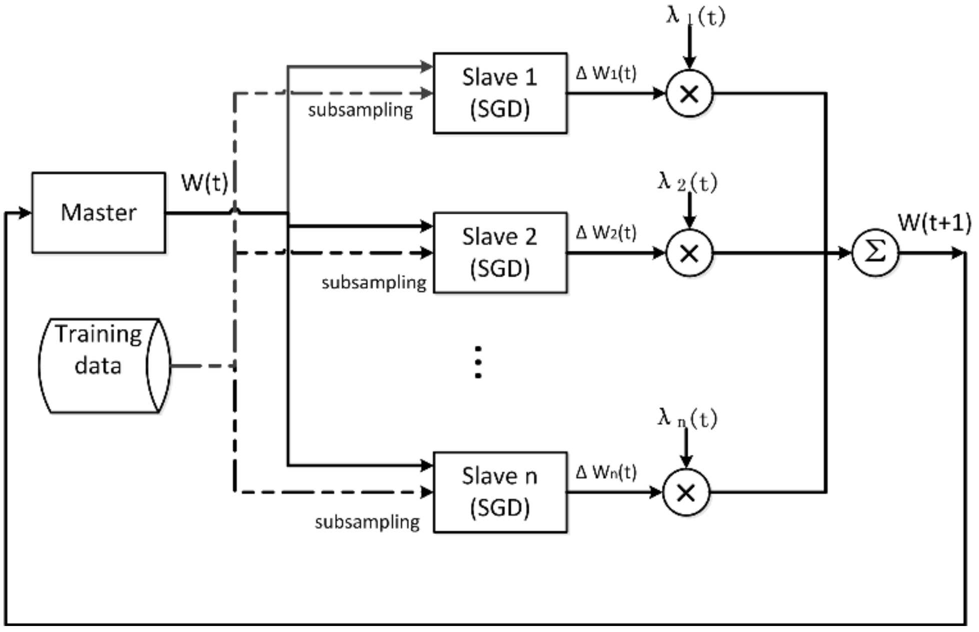


Figure 6.1: The SIS-SGD *algorithm*

The SIS-SGD algorithm uses a master/slave scheme to carry out message passing among different threads or processors. As shown in Figure 6.1, at each Map-Reduce iteration t, all slaves start from the same parameter, $W(t)$. Each slave randomly samples a subset of the training data as its own training set, and runs the regular online SGD. After finishing up its own samples, it computes the gradient direction from the initial $W(t)$ to the currently converged point at its own data set. These directions are then weighted and aggregated by the master to compute the new parameter $W(t+1)$:

$$W(t+1) = W(t) + \frac{\alpha}{n} * \sum_{k=1}^{n} \lambda_k(t) \Delta W_k(t) \tag{6.1}$$

[6]For each predicted word $w(t+1)$, it is mapped to a unique class, which in turn activates and updates a subset of connections from all hidden units to a subset of output units. The subset of output units correspond to all the words that belong to the same class as $w(t+1)$.

where n is the number of slaves. $\Delta W_k(t)$ is the overall update direction that slave k learned from its training subset. $\lambda_k(t)$ is the weight for the direction from slave k.

In neural network training, the model benefits more from new patterns in the training data. In this paper, we apply the following heuristic method to determine the weight λ_k.

$$\lambda_k(t) = \frac{ln(E_k(t))}{\sum_{l=1}^{n}(ln(E_l(t)))}, \tag{6.2}$$

where E_k is the entropy of the subset training data in slave k. Usually new patterns of training data are revealed by high entropy, and the directions calculated on the basis of these new patterns should get more weight in the final update direction.

6.4.1 Sentence Independence

SIS-SGD operates under the assumption of sentence independence. In a standard RNNLM, the prediction of the next word is based on the history information including previous sentences. In fact, however, an infinite history is unnecessary in many applications. For example, in voice search, there is not necessarily a strong relationship among sentences. At the same time, although theoretically information in an RNNLM can cycle for an arbitrary length of time, in practice, the span of the back-propagation-through-time algorithm that is used to learn the weights, is limited. The study in [101] shows that after 4 to 5 steps, back propagation through time has very little benefit.

At the same time, due to the random sampling at each slave, sentence dependency has no real relevance. Section 6.5.2 will verify that assuming sentence independence has a barely noticeable effect on the the perplexity of the trained model. Although in this work we use sentence-level sampling, we expect our results to be generalizable to different units of approximately the same size, e.g., utterances.

6.4.2 Running SGD inside each Slave

As we mentioned before in SIS-SGD, each slave updates its model replica by the standard online SGD algorithm. It does not communicate with the master until it finishes training on its own subset of samples, hence avoiding frequent messaging between slaves and master. The master waits for all slaves to finish one iteration of training. Within each slave, the direction is effectively computed via SGD, rather than mini-batch SGD or batch SGD.

6.4.3 Subsampling

An important principle of the subsampling in SIS-SGD is to make sure that the subsampled sentences from each slave overlap with each other and that the union of all subsamples cover the original entire training set. Instead of doing true data parallelism, allowing overlaps regularizes parallel SGD. After each iteration of parameter update at the master, we reshuffle the whole training data and continue to do subsampling, for additional regularization.

In SIS-SGD, we use subsampling instead of disjoint partitioning for two reasons: One factor is determined by the special structure of RNNLM. In RNNLM, both the weight matrix from the input word to the hidden layer and hidden layer to output word, are sparse and huge. Partitioning leads to a situation in which each slave confronts a data sparseness problem in producing an effective direction from its own subset. The other factor is the diversity of the directions from each slave. Each partition of the training data can be quite different from each other, which means that the directions from different partitions can conflict with each other. The aggregation of these diverse directions from disjoint partitions can result in a very small update for the overall parameter learning. The performance observed during our initial exploratory experimentation was so severely degraded in the case of disjoint partitions, that we did not consider the possibility further in the main experiments.

6.4.4 Practical Tricks for Robustness

In SIS-SGD, we use a smaller learning rate for the lower layer weight update [87]. Fundamentally, neural networks can be viewed as a combination of many different regressions. Theoretically weights from different layers should use different learning rates. However, in practice, only one learning rate is used for all the weights. However, the higher layer usually has a larger gradient than the lower layer. A smaller learning rate on the lower layer avoids divergence during training.

To accelerate the training of SIS-SGD, momentum [130] is applied in the master weight update in SIS-SGD. The basic approach of momentum is to interpolate current weight updates with the weight update history. The idea is that if the current updates are close to the historical updates, the momentum will increase the current updates. Otherwise, current updates will be reduced by the historical weight updates. This trick makes the training tolerant to noise and also speeds up training.

6.5 Experiments

6.5.1 Data Set

To evaluate the proposed SIS-SGD algorithm, we use two different data sets. The first one is Penn Treebank text data set (refered to as PTB). We use section 00-20 (972K word tokens) for training, section 21-22 (77K words) for validation during the training, section 23-24

Table 6.1: The perplexity and WER *results of the sentence dependent (*DEP*)* RNNLM *and sentence independent (*INDEP*)* RNNLM *reported on* PTB *and* WSJ. RAND *means randomized training sequences.*

model	PPL TEST	WER(%) DEV	WER(%) EVAL
PTB-RNN-DEP	138.7	-	-
PTB-RNN-INDEP	136.6	-	-
PTB-RNN-DEP-RAND	137.4	-	-
PTB-RNN-INDEP-RAND	135.3	-	-
WSJ-RNN-DEP	138.3	11.36	15.48
WSJ-RNN-INDEP	144.0	11.17	15.54
WSJ-RNN-DEP-RAND	139.4	10.88	15.66
WSJ-RNN-INDEP-RAND	140.6	11.15	15.69

(86K words) for testing. The vocabulary size for our experiment is 10K. We will measure only the perplexity on the Penn Treebank data set.

The second data set is WSJ, for which we use 100-best speech recognition list from the DARPA WSJ'92 and WSJ'93 data sets, as used by [98, 162]. In the 100-best list set, 333 sentences are used as development data for tuning the interpolation of language model score and acoustic model score (DEV). The rest, 465 sentences, are used for evaluation (EVAL). The oracle WER for development data and evaluation data are 6.1% and 9.5%, respectively. The training corpus contains 37M words of running text from the NYT section of English Gigaword. The validation data set contains 186K words. A held-out set of 230K words is used for testing (TEST). The vocabulary size is 194K. Our experiments rescore the N-best lists to compare various RNNLM models VS the baseline.

All the RNNLM models in our experiments have 200 hidden neurons and are trained using 4-step back propagation through time (BPTT). The baseline model is obtained using the most recent version of RNNLM toolkit, in which the class-based SGD [101] is implemented. We use 100 word-classes in all of our experiments. The other baseline model is a Kneser-Ney 5-gram model, which has been shown to provide good n-gram model performance on this data [52] and is used in [98, 99, 101]

Table 6.2: The perplexity and N-best rescored WER *results of* RNNLM *trained by parallel* SIS-SGD *via multi-thread architecture on* WSJ *data. The 'training data size' column stands for the relative size of the slave training subset, compared to the whole training data set. '*PPL*' shows the perplexity results on the test data.*

model	training data size	training time	PPL	WER	WER
	(%)	(hours)	TEST	DEV(%)	EVAL(%)
KN5-base	100	-	174.5	12.09	17.30
RNNLM-base	100	77.5	140.6	11.15	15.69
8 threads	100	78.4	140.5	11.00	16.10
8 threads	50	65.9	138.5	10.77	15.84
8 threads	40	46.1	140.4	11.01	15.71
8 threads	25	33.6	144.4	11.07	15.99
16 threads	12.5	26.1	151.0	11.14	15.66

6.5.2 Sentence Independence Verification

As we discussed in previous sections, SIS-SGD imposes the condition of sentence independence [7]. Table 6.1 verifies that disrupting the sequences of sentences barely degrades the performance of RNNLM. As a matter of fact, going from DEP to INDEP-RAND, there is a small improvement in perplexity on PTB and a small degradation on WSJ. These experiments provide evidence to support the strategies of our SIS-SGD: sentence independence with randomization. Equally importantly, the randomization seems to regularize the convergence, especially under sentence independent training.

6.5.3 Speed up with Multi Threads

On a single machine, we can apply our proposed parallel SIS-SGD to train RNNLM via multi-thread architecture. The last row in Table 6.2 shows that the RNNLM model trained by 16 parallel SIS-SGD threads achieves the best training speed without losing performance in terms of word error rate (WER). Each slave thread trains a model replica on a subset of 12.5% of the whole training data. That is, the 16 threads together train twice ($16 * 12.5\% = 2$) as much the original data, yet with a 3-fold speedup. The price is the increased memory requirement, as now 16 replicas of the models must be accommodated in memory, along

[7] This means that in the training on each sentence, we always start from a fresh sentence-begin symbol $< s >$, and re-initialize the copied hidden part at the input layer.

Table 6.3: The perplexity and WER *results of* RNNLM*s trained by the parallel* SIS-SGD *on* WSJ *data with different numbers of processors. '*' denotes that the model is trained with more iterations (a lower stopping criterion).*

model WSJ	Training data size (%)	Training time (hours)	PPL TEST	WER DEV(%)	WER EVAL(%)
KN5-base		-	174.5	12.09	17.30
RNNLM-base	100	77.5	140.6	11.15	15.69
8p	25	33.7	146.9	11.19	15.70
16p	12.5	23.8	150.3	11.03	15.73
24p	8.3	27.2	148.9	11.01	15.67
32p	7.8	15.4	159.5	11.21	16.03
40p	5	19.7	154.7	11.10	15.92
48p	4.2	18.1	156.4	11.17	16.09
100p	2	11.2	172.5	11.41	16.16
*100p	2	27.4	142.0	11.14	15.62

with an additional copy on the master. In our case, our machine is an HP Z600 workstation with 12GB of RAM and 4 cores.

6.5.4 Speed up with Multi-Processors

The single machine has limited resources in terms of both cores and memory. In order to fully take advantage of distributed hardware architecture, we implemented a multi-process version of the parallel SIS-SGD. The models listed in Table 6.3 are trained by an HPC cluster via MPI processes.

In Table 6.3, the SIS-SGD models actually subsample the whole training data twice, except for the one with 32 processes, where the original train data is processed 2.5 times. When we used 32 processors, we achieved a 5-fold speed up with a relative 2% degradation on the evaluation data in terms of WER. If we use 100 processors, we can speed up the training by almost 7 times with relative 3% WER increase on evaluation data. We note that the model can be further improved by additional 16 hours of training iterations. As it is shown on the bottom of the table, SIS-SGD uses almost one third of the standard SGD training time but results in a model with approximately the same quality performance.

6.5.5 Asynchronous SIS-SGD

Parallel SIS-SGD can also be implemented in an asynchronous manner. In [80, 176], the master immediately updates the model when it receives the direction from any slave. Afterwards, the master directly sends the updated model back to the corresponding slave. Basically, the master suffers extremely low inter-process latency as it does not require synchronization among slaves. However, asynchronous communication introduces model parameter delay among slaves during training. In [176] it is shown that a small delay to the master in fact improved the performance of the model as well as accelerated the training. Large delay from the slave to the master in fact degraded the quality of the model. That is, one would like to send updates to the master as frequently as possible[8].

Figure 6.2 shows the whole training data entropy of our proposed synchronous SIS-SGD, asynchronous SIS-SGD and baseline model against wall clock time. In both the synchronous and asynchronous SIS-SGD, we use 8 processors, of which each has 25% of the whole training data. Asynchronous SIS-SGD is implemented in the round-robin way in order to obtain a deterministic result across different training runs. Hence it could have been faster if no round-robin constraint were enforced.

As we can see in Figure 6.2, the synchronous SIS-SGD converges faster than the asynchronous one. The probable reason is that with asynchronous SIS-SGD, model delay is too large. The delay in this case is one model learned from the 25% of the whole training data. Although asynchronous SIS-SGD reduces the inter-processor latency, the large model delay actually slows the parameter convergence.

6.6 Conclusions

This paper has presented a sentence-independent subsampling stochastic gradient descent method (SIS-SGD), in which RNNLM is trained in parallel under the condition of sentence independence. In SIS-SGD, each slave trains a replica of the model on its own training data subset, which is generated by subsampling instead of disjoint partitioning. In order to minimize the data communication overhead, each slave sends its update to the master only after it finishes one pass of SGD learning on its own training subset. The master aggregates all the directions in a heuristic weighted way to perform the final model update, and then broadcast the updated model to all slaves to start the next iteration of training. The experimental results showed that using a multi-thread technique, SIS-SGD can achieve a 3-fold speed up without losing performance. Using the multi-process technique and taking advantage of 100

[8]However, this incurs high communication cost.

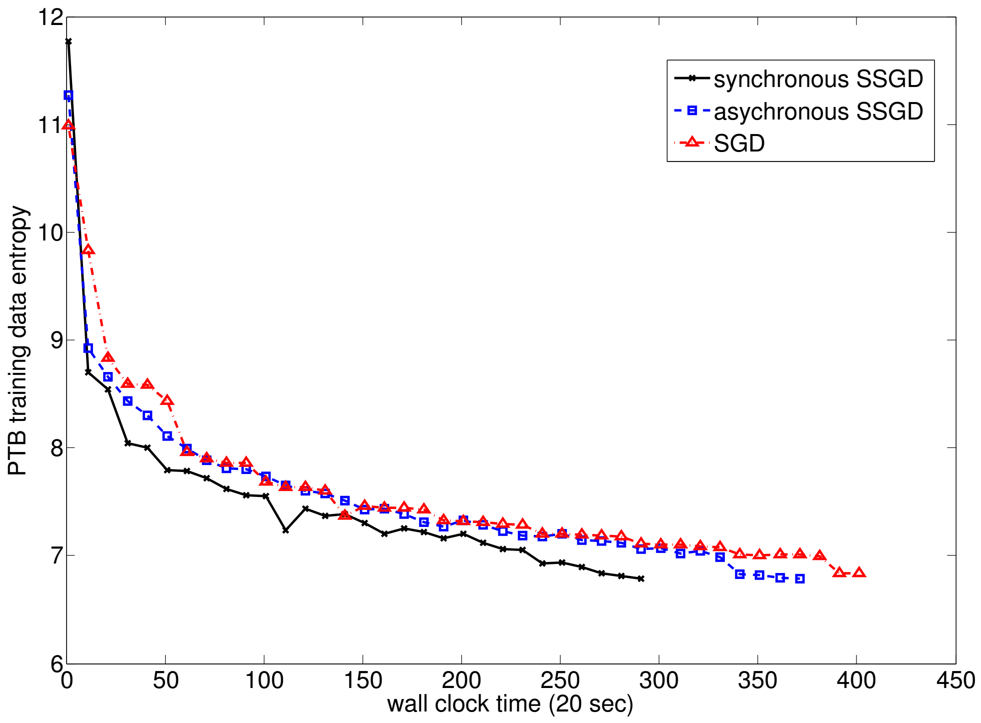


Figure 6.2: Whole training data entropy (Penn Treebank) of asynchronous parallel SIS-SGD*, synchronous parallel* SIS-SGD *and standard* SGD *against the wall clock time. The parallel* SIS-SGD *use 8 processors, each of which has* 25% *of training data.*

processors, it can obtain a 7-fold speed up with a 3 percent relative degradation in terms of WER by taking advantage of 100 processors.

6.7 Acknowledgements

The authors are grateful for the insights and discussions from Dong Yu, Frank Seide, Tomas Mikolov and Puyang Xu, in addition to the technical help from Yong Ni.

Chapter 7

Conclusions and future research

7.1 Conclusions

In this section, the research questions in Chapter 1 will be answered, which were inspired by the observation that "Language should be put back into language models" [126]. With recently proposed advances and flexible computational paradigms, it becomes possible to take into account additional information rather than just the word itself. This thesis proposes language models with meta-information.

This thesis discusses different types of meta-information along with their corresponding uses, such as discourse level information, sentence level information and word level information. To apply meta-information in language models, we propose *k*-component adaptive recurrent neural network language models (RNNLMs) using curriculum learning, recurrent neural network tandem language models and forward and backward RNNLMs, which are discussed in detail in Chapter 3, 4 and 5, respectively.

The discourse level meta-information discussed in this thesis includes topics and socio-situational settings. The topics define the content of a discourse. The socio-situational settings define the social restrictions of a discourse. The definition and classification of topic information has been extensively investigated in e.g., information retrieval, natural language understanding. However, only a few studies discuss the classification and application of socio-situational settings in language modeling.

As is stated in *Research Question 1*, having the socio-situational settings and having a method to classify the socio-situational setting, is a prerequisite for the usage of this information in language models. In Chapter 2, on the basis of an experiment, we analyze and extract the features that were used in human manual classification task of socio-situational settings. These features include word tokens, function word tokens, part-of-speech tags, n-grams of part-of-speech tags, sentence length, function word ratio and single occurrence word ratio. Based on these features, we propose a static automatic classification method for socio-situational settings using support vector machines. This static method achieves 89.55% classification accuracy. Furthermore, according to the attributes of socio-situational settings, we propose a dynamic socio-situational setting classification method using a Dynamic Bayesian Network that has the following advantages. One is that it can provide online classification results to language modeling. The other one is that using the initial 25% of the whole data, it obtains 95% of the accuracy achieved by classifiers that use the whole data. The results in Chapter 2 also show that the proposed static and dynamic classification methods achieve better classification accuracy than humans. Note that the dynamic classification method is specialized for socio-situational settings. More research is needed to determine whether this dynamic classification method also achieves good results for other types of meta-information. In order to use meta-information in language modeling, meta-information needs to be available. Therefore, in this chapter, we not only propose methods for socio-situational setting classification, but also propose a method to classify the socio-situational setting on the fly.

In order to address *Research Question 2* on how to effectively use discourse level meta-information in language modeling, we propose *k*-component adaptive recurrent neural network language models using a curriculum learning method. Using the curriculum learning method, each component model is trained following a curriculum that starts from general patterns, characteristic of the training data as a whole, to specific patterns, characteristic of one specific sub-domain of the overall data. In Chapter 3, we propose three curriculum learning methods, namely, Start-from-Vocabulary (SV), Data Sorting (DS) and All-then-Specific (AS). The results confirm that by using curriculum learning methods, discourse level meta-information adaptive RNNLMs outperform conventional RNNLMs. In Chapter 3, we also show that implicit interpolation carried out by curriculum learning methods outperforms traditional linear interpolation. As an implicit interpolation method, curriculum learning can be used to update existing language models using only the newly collected data, without retraining models from scratch by adding the yet-unseen data to the existing data. Additionally, the results in this chapter also indicate that the performance of the proposed methods is suitable for highly diverse data sets. This fact requires us to make sure that the design of curricula should be based on the characteristics of the target data sets.

Part of *Research Question 3* concerns combining sentence level meta-information into recurrent neural network language models. The sentence level meta-information which we address in this thesis is about the whole sentence word information. In Chapter 5, we focus on handling the whole sentence information in RNNLMs. Conventional RNNLMs predict the next word according to previous history information. In other words, the problem of dealing with whole-sentence information basically concerns how to integrate information on succeeding words into language models. In this thesis, three approaches for exploiting succeeding-word information in RNNLMs are proposed. The first is a forward-backward bidirectional recurrent neural network language model that exploits preceding and succeeding words. The second approach uses a Maximum Entropy method that incorporates succeeding word information in recurrent neural network language models. The third one is a two pass alternating rescoring method to combine the models exploiting both preceding and succeeding words. In Chapter 5, the results show that the combination of these approaches generates best performance. The proposed approaches in this chapter are based on the assumption that whole-sentence information is available in advance for language models. This assumption may not be satisfied for other language models. RNNLMs offer a unique opportunity to satisfy this assumption. Due to their high computational complexity, in most applications for automatic speech recognition, RNNLMs are applied during N-best rescoring. In N-best rescoring, the whole sentence information is available.

Research Question 3 also addresses how to integrate word level meta-information into recurrent neural network language models. We attempt to represent each word by a set of information that includes the part-of-speech of the word, the lemma of the word, the topic which the word belongs to and the socio-situational setting in which the word occurs. The application of meta-information from the word level in language modeling is addressed by the proposed recurrent neural network tandem language models in Chapter 4. The RNNTLMs have two parts: one part for meta-information prediction from words, the other part for integrating the predicted meta-information into RNNLMs. In the integration part of the RNNTLMs, the input layer of RNNLMs is expanded with predicted meta-information. Not only the current word, but also the current predicted meta-information is fed into RNNLMs.

Research Question 4 is addressed in Chapter 4. According to the results in this chapter, using an oracle to provide the correct word level meta-information, all the proposed types of meta-information help to improve RNNLMs. Especially, RNNLMs with POS and RNNLMs with topics perform better than RNNLMs with other types of meta-information in terms of perplexity and word prediction accuracy. Under the oracle situation, the experimental results also show that by combining more types of meta-information, RNNLMs can achieve better performance. However, if the meta-information is unknown during testing, a meta-information predictor has to be used. This holds for some types of meta-information such as topics, socio-situational settings, and POS. The experimental results show that using the predicted meta-information instead of the correct meta-information degrades the improvement. Therefore, improving the prediction accuracy of meta-information predictors is an important direction of research in the future.

Note that some meta-information such as token size and sentence length, can be directly obtained from the text without using a meta-information predictor. Although easy to extract, the experiments show that these trivial features contribute in a non-trivial and robust way in improving the performance of RNNLMs. In N-best rescoring, the best performance is achieved by incorporating socio-situational settings, lemma and token size into RNNLMs.

As is shown in previous chapters, using meta-information improves the effectiveness of RNNLMs, but it decreases the efficiency. In order to use RNNLMs in practice, the efficiency problem has to be solved, which is addressed by *Research Question 5*. The computational complexity of training RNNLMs is linear in the size of data set, however, the size of a practical data set is prohibitively large for training RNNLMs. For example, on the WSJ data set with 37M of words, the training of RNNLMs costs 77.5 hours. In practice, the size of training data sets can be tens of billions of words. Training on such big data sets can cost hundreds of days. Chapter 6 addresses this problem by using parallel processing techniques. We propose a sentence-independent subsampling stochastic gradient descent algorithm to speed up the training of RNNLMs. The approach maps the process of training the overall model into stochastic gradient descent training of submodels. The update directions of submodels are aggregated and used as the weight update for the whole model. Using a multi-thread technique, the proposed algorithm can achieve a 3-fold speed up without losing performance in terms of word error rate. When multi-processors are used, a nearly 11-fold speed up can be attained with a relative small degradation of word error rate. In addition, the proposed method has the potential to increase speed up in relation to the size of the data set, i.e., the bigger the data, the bigger the speed up.

7.2 Future research

During our investigation of language modeling with meta-information, we found a number of interesting problems worthy of further research work. In this section, we discuss a section of the challenges that set the scene for future work.

In this thesis, we basically use discourse level meta-information such as socio-situational settings and topics, sentence-level meta-information including succeeding words, and sentence length and token-level meta-information such as part-of-speech tags, lemmas and token size, to improve recurrent neural network language models. However, there is other meta-information that we could exploit to make language models better. One possible type of meta-information is word encoding information. For example, in language such as English the sequence of letters within words carries information that reflects meaning and syntactic function. It would be productive to further investigate how word encoding information can benefit language models. Another potential type of meta-information is the word embedding vector. The embedding vector for word *i* is defined as the vector of connections from the input unit that represent word *i* to all the hidden units. Mikolov [102] shows that semantically similar words have close embedding vectors.

Different model structures can be investigated to integrate meta-information. In this thesis, we use the forward-backward training strategy to put the succeeding word information into recurrent neural network language models. An alternative method is to directly use bi-directional recurrent neural networks [133]. In bi-directional recurrent neural networks, the combination of the forward model and the backward model is optimized by the neural network. This approach can possibly provide better optimization than our proposed models described in Chapter 5.

New model structures can be exploited to improve the language models. The recurrent neural network language model is a successful example of how a good model structure can benefit language modeling. The structure of recurrent neural networks can improve language models in terms of effectiveness, however it becomes time consuming to train such a structure. The effective loop structure in RNNLMs forces the training of language models in a sequential way, which makes it difficult to use parallel processing. New structures can be investigated to balance effectiveness and efficiency.

One of the most challenging problems for recurrent neural network language models is how to integrate them into real time speech recognition systems. Strategies and methods should be investigated from the perspective of using computationally complex language models for online speech recognition. The recurrent neural network language models show superior capability in language modeling. However, in practice they can only be applied into speech recognition by a two-pass rescoring strategy. The speech recognition performance can possibly achieve further improvement by embedding RNNLMs directly into speech recognition.

In Chapter 3, Curriculum Learning is also used as an implicit interpolation method to update the existing language models with newly collected data. This type of experimentation can be further extended to enable language models to learn new words that are not in the old vocabulary. When new words appear, using curriculum learning, recurrent neural network language models can be updated with new neurons and new connections related to these new neurons by only learning from small amount of yet-unseen data containing these new words.

Samenvatting

Taalmodellen spelen een cruciale rol in natuurlijke taalverwerking en -begrip. Beginnend met de algemene structuren, taalmodellen zijn in staat natuurlijke taal te leren met rijke invoergegevens. Maar de meest avanceerde taalmodellen maken alleen gebruik van de woorden zelf, welke niet voldoende zijn om een taal te karakteriseren. In dit proefschrift verbeteren we recurrent neural networks voor taalmodellen (RNNLM) door ze te trainen met additionele informatie. Verschillende methoden om de verschillende typen van additionele informatie te integreren, worden in dit proefschrift voorgesteld.

Alle potentiele informatie afgezien van het woord zelf, die kan worden gebruikt om taal te karakteriseren, wordt meta-informatie genoemd. In dit proefschrift stellen we voor verschillende soorten meta-informatie te gebruiken zoals gespreksniveau informatie, welke het gehele gesprek reflecteert, het zinsniveau, welke de structuur van de zin beschrijft en morfologische informatie die het woord vanuit verschillende standpunten beschrijft.

Bijvoorbeeld, we bekijken de volgende Nederlandse paragraaf. $< s >$ staat voor het begin van de zin, $< /s >$ staat voor het einde van de zin.
$< s >$ *kan allemaal nog natuurlijk* $< /s >$
$< s >$ *maar ze ontlopen dan de groepswinnaar in elk geval in de kwartfinale* $< /s >$
$< s >$ *en vooral Nederland wil graag in Rotterdam die kwartfinale spelen* $< /s >$
$< s >$ *en dan moet er groepswinst behaald worden* $< /s >$
$< s >$ *anders verhuizen ze naar Brugge en krijgt het Jan Breydelstadion Oranje dus op bezoek* $< /s >$
$< s >$ *we gaan er even uit* $< /s >$
$< s >$ *slotfase zit eraan te komen* $< /s >$
$< s >$ *twee minuten nog tot het einde plus de toegevoegde tijd* $< /s >$
$< s >$ *dat is uh toch nog ook wel een paar minuten denk ik* $< /s >$
$< s >$ *maar de wedstrijd is gespeeld* $< /s >$

Op gespreksniveau is deze paragraaf geclassificeerd als “Live commentaries (broadcast)” vanuit het sociale situatie “SSS” perspectief en als “sport” vanuit het onderwerps-perspectief. Op zinsniveau is elk woord behalve het begin woord $< s >$ en het eindwoord $< /s >$, geannoteerd met informatie over zijn voorafgaande woord en het volgende woord. Beschouw bijvoorbeeld het woord “slotfase” in de volgende zin:

$$< s > \textit{slotfase zit eraan te komen} < /s >$$

Dit woord heeft als voorgaande informatie “$< s >$ dus de wonderen” en als volgende informatie “zit eraan te komen $< /s >$”. Op het woordniveau, zoals in Tabel 7.1 is weergegeven, is het woord “slotfase” geannoteerd door een vector met sommige van de voorgestelde meta-informatie.

Tabel 7.1: woordniveau meta-informatie van het woord ”slotfase”

woord	slotfase
SSS	Live commentaries (broadcast)
onderwerp	sport
token grootte	8
zinslengte	6

Op het gespreksniveau onderzoeken we classificatiemethoden voor sociale situaties en onderwerp. Op het zinsniveau concentreren we ons op volgende-woord informatie en gehele-zin informatie. In dit proefschrift is elk woord geannoteerd door een vector van verzamelde meta-informatie.

Verschillende methoden worden voorgesteld om meta-informatie in taalmodellen op te nemen. Op het gespreksniveau is een methode om het curriculum te leren gebruikt om de sociale situatie en het onderwerp te combineren. Op het zinsniveau werden forward-backward recurrent neural networks voor taalmodellen voorgesteld om volgende-woord en gehele-zin informatie in taalmodellen op te nemen. Op het woordniveau werd elke woord bewerkt op basis van zowel voorgaande woorden als voorgaande meta-informatie.

De resultaten laten zien dat meta-informatie kan worden gebruikt om de effectiviteit van taalmodellen te verbeteren. Maar de resultaten laten ook zien dat het gebruik van meta-informatie de trainingsefficiëntie van de taalmodellen doet dalen. We pakken dit probleem aan door parallelle-verwerkingstechnieken toe te passen. Een subsampling stochastic gradient descent algorithm is voorgesteld om de training van recurrent neural networks voor taalmodellen te versnellen.

Yangyang Shi

Summary

Language modeling plays a critical role in natural language processing and understanding. Starting from a general structure, language models are able to learn natural language patterns from rich input data. However, the state-of-the-art language models only take advantage of words themselves, which are not sufficient to characterize the language. In this thesis, we improve recurrent neural network language models (RNNLM) by training them with additional information. Different methods of integrating the different types of additional information into RNNLMs are proposed in this thesis.

All the potential information beyond the word itself that can be used to characterize the language is called meta-information. In this thesis, we propose to use different types of meta-information to represent languages such as discourse level information, which is reflected from the whole discourse, sentence level information which characterize the patterns of sentences and morphological information which represents the word from different perspectives.

For example, we consider the following Dutch paragraph. $< s >$ represents sentence beginning. $< /s >$ stands for the sentence ending.
$< s >$ *kan allemaal nog natuurlijk* $< /s >$
$< s >$ *maar ze ontlopen dan de groepswinnaar in elk geval in de kwartfinale* $< /s >$
$< s >$ *en vooral Nederland wil graag in Rotterdam die kwartfinale spelen* $< /s >$
$< s >$ *en dan moet er groepswinst behaald worden* $< /s >$
$< s >$ *anders verhuizen ze naar Brugge en krijgt het Jan Breydelstadion Oranje dus op bezoek* $< /s >$
$< s >$ *we gaan er even uit* $< /s >$
$< s >$ *slotfase zit eraan te komen* $< /s >$
$< s >$ *twee minuten nog tot het einde plus de toegevoegde tijd* $< /s >$
$< s >$ *dat is uh toch nog ook wel een paar minuten denk ik* $< /s >$
$< s >$ *maar de wedstrijd is gespeeld* $< /s >$
On the discourse level, this paragraph is labeled as “Live commentaries (broadcast)” from

the socio-situational setting (SSS) perspective and “sport” from the topic perspective. On the sentence level, each word except for the beginning word $< s >$ and ending word $< /s >$, is annotated with its preceding word information and succeeding word information. For example, we consider word “slotfase” in the following sentence.

$$< s > \textit{slotfase zit eraan te komen} < /s >$$

This word has preceding information “$< s >$” and succeeding information “zit eraan te komen $< /s >$”. On the word level, as is shown in Table 7.2, the word “slotfase” is annotated by a vector containing some of the proposed meta-information.

Tabel 7.2: word level meta-information of word “slotfase”

word	slotfase
SSS	Live commentaries (broadcast)
topic	sport
token size	8
sentence length	6

On the discourse level, we investigate classification methods for socio-situational settings and topics. On the sentence level, in this thesis, we focus on information such as succeeding words information and whole sentence information. In this thesis, each word is annotated by a vector containing the meta-information collected.

Different methods are proposed in this thesis to integrate the meta-information into language models. On the discourse level, a curriculum learning method has been used to combine the socio-situational settings and topics. On the sentence level, forward-backward recurrent neural network language models have been proposed to integrate the succeeding word information and whole sentence information into language models. On the word level, each word has been conditioned on its preceding words as well as on preceding meta-information.

The results reported in this thesis show that meta-information can be used to improve the effectiveness of language models at the cost of increasing training time. In this thesis, we address this problem by applying parallel processing techniques. A subsampling stochastic gradient descent algorithm has been proposed to accelerate the training of recurrent neural network language models.

Yangyang Shi

Bibliography

[1] Alekh Agarwal and John C. Duchi. Distributed delayed stochastic optimization. In *Proceedings of the 51th IEEE Conference on Decision and Control*, pages 5451–5452, 2012.

[2] Andrei Alexandrescu and Katrin Kirchhoff. Factored neural language models. In *Proceedings of the Human Language Technology Conference of the NAACL*, pages 1–4, 2006.

[3] Fredy A. Amaya and José M. Benedí. Improvement of a whole sentence maximum entropy language model using grammatical features. In *Proceedings of 39th Annual Meeting of the Association for Computational Linguistics*, pages 10–17. Association for Computational Linguistics, July 2001.

[4] Tomas Mikolov Anoop Deoras and Kenneth Church. A fast re-scoring strategy to capture long-distance dependencies. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 1116–1127, 2011.

[5] Juan Antonio, Perez-Ortiz, and Mikel L. Forcada. Part-of-speech Tagging with Recurrent Neural Networks. In *Proceedings of International Joint Conference of Neural Networks*, pages 1588–1592, 2001.

[6] Shlomo Argamon, Moshe Koppel, and Galit Avneri. Routing documents according to style. In *Proceedings of First International Workshop on Innovative Information Systems*, 1998.

[7] Ebru Arisoy, Tara N. Sainath, Brian Kingsbury, and Bhuvana Ramabhadran. Deep neural network language models. In *Proceedings of the NAACL-HLT 2012 Workshop: Will We Ever Really Replace the N-gram Model? On the Future of Language Modeling for HLT*, pages 20–28, 2012.

[8] Alan D. Baddeley, Neil Thomson, and Mary Buchanan. Word length and the structure of short-term memory. *Journal of Verbal Learning and Verbal Behavior*, 14(6):575 – 589, 1975.

[9] Ricardo Baeza-Yates and Berthier Ribeiro-Neto. *Modern Information Retrieval*. Addison Wesley, May 1999. ISBN 020139829X.

[10] Lalit R. Bahl, Peter F. Brown, Peter V. de Souza, and Robert Mercer. A tree-based statistical language model for natural language speech recognition. *IEEE Transactions on Acoustics, Speech and Signal Processing*, 37(7):1001–1008, 1989.

[11] Tomas Bayes. An essay towards solving a problem in the doctrine of chances. *Phil. Trans. of the Royal Soc. of London*, 53:370–418, 1763.

[12] Jerome R. Bellegarda. A multispan language modeling framework for large vocabulary speech recognition. *IEEE Transactions on Speech and Audio Processing*, 6(5): 456–467, 1998.

[13] Jerome R. Bellegarda. Statistical language model adaptation: review and perspectives. *Speech Communication*, 42(1):93–108, 2004.

[14] Jerome R. Bellegarda, John W. Butzberger, Yen-Lu Chow, Noah B Coccaro, and Devang Naik. A novel word clustering algorithm based on latent semantic analysis. In *Proceedings of IEEE International Conference on Acoustics, Speech, and Signal Processing*, pages 172–175, 1996.

[15] Yoshua Bengio and Jean Sebastien Senecal. Adaptive importance sampling to accelerate training of a neural probabilistic language model. *IEEE Transactions on Neural Networks*, 19(4):713 –722, april 2008.

[16] Yoshua Bengio, Rejean Ducharme, Pascal Vincent, and Christian Janvin. A neural probabilistic language model. *Journal of Machine Learning Research*, 3:1137–1155, 2003.

[17] Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. Curriculum learning. In *Proceedings of International Conference on Machine Learning*, pages 41–48. ACM, 2009. ISBN 978-1-60558-516-1.

[18] Adam L. Berger, Stephen A. Della Pietra, and Vincent J. Della Pietra. A maximum entropy approach to natural language processing. *Computational Linguistics*, 22: 39–71, 1996.

[19] Jeff A. Bilmes and Katrin Kirchhoff. Factored language models and generalized parallel backoff. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology*, pages 4–6, 2003.

[20] David M. Blei, Andrew Y. Ng, and Michael I. Jordan. Latent Dirichlet allocation. *Journal of Machine Learning Research*, 3:993–1022, 2003.

[21] Enrico Bocchieri, Diamantino Caseiro, and Dimitrios Dimitriadis. Speech recognition modeling advances for mobile voice search. In *Proceedings of IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 4888–4891, 2011.

[22] Peter F. Brown, John Cocke, Stephen Della Pietra, Vincent J. Della Pietra, Frederick Jelinek, John D. Lafferty, Robert L. Mercer, and Paul S. Roossin. A statistical approach to machine translation. *Computational Linguistics*, 16(2):79–85, 1990.

[23] Peter F. Brown, Peter V. deSouza, Robert L. Mercer, Vincent J. Della Pietra, and Jenifer C. Lai. Class-based n-gram models of natural language. *Computational Linguistics*, 18:467–479, 1992.

[24] Chih-Chung Chang and Chih-Jen Lin. LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2:27:1–27:27, 2011.

[25] Ciprian Chelba. A structured language model. In *Association for Computational Linguistics*, pages 498–500, 1997.

[26] Ciprian Chelba and Frederick Jelinek. Exploiting syntactic structure for language modeling. In *Proceedings of the 17th international conference on Computational linguistics*, pages 225–231, 1998.

[27] Ciprian Chelba and Frederick Jelinek. Structured language modeling. *Computer Speech & Language*, 14(4):283–332, 2000.

[28] Guangyu Chen and Ben Choi. Web page genre classification. In *Proceedings of the 2008 ACM symposium on Applied computing*, pages 2353–2357, 2008.

[29] Stanley Chen, Douglas Beeferman, and Ronald Rosenfeld. Evaluation Metrics for Language Models. In *DARPA Broadcast News Transcription and Understanding Workshop (BNTUW)*, February 1998.

[30] Stanley F. Chen. Shrinking exponential language models. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 468–476. Association for Computational Linguistics, 2009.

[31] Stanley F. Chen and Joshua Goodman. An empirical study of smoothing techniques for language modeling. Technical report, 1998.

[32] Stanley F Chen, Kristie Seymore, and Ronald Rosenfeld. Topic adaptation for language modeling using unnormalized exponential models. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 681–684, 1998.

[33] Cheng T. Chu, Sang K. Kim, Yi A. Lin, Yuanyuan Yu, Gary R. Bradski, Andrew Y. Ng, and Kunle Olukotun. Map-Reduce for Machine Learning on Multicore. In *Advances in Neural Information Processing Systems*, pages 281–288, 2006.

[34] Kenneth W. Church and William A. Gale. Probability scoring for spelling correction. *Statistics and Computing*, 1(2):93–103, 1991.

[35] Kenneth Ward Church. A stochastic parts program and noun phrase parser for unrestricted text. In *Proceedings of the second conference on Applied natural language processing*, ANLC '88, pages 136–143, Stroudsburg, PA, USA, 1988. Association for Computational Linguistics.

[36] Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine Learning*, 20:273–297, 1995.

[37] Doug Cutting, Julian Kupiec, Jan Pedersen, and Penelope Sibun. A practical part-of-speech tagger. In *Proceedings of The Third Conference On Applied Natural Language Processing*, pages 133–140, 1992.

[38] Jeffrey Dean, Greg S. Corrado, Rajat Monga, Kai Chen, Matthieu Devin, Quoc V. Le, Mark Z. Mao, MarcAurelio Ranzato, Andrew Senior, Paul Tucker, Ke Yang, and Andrew Y. Ng. Large scale distributed deep networks. In *Advances in Neural Information Processing Systems*, 2012.

[39] Thomas Dean and Keiji Kanazawa. A model for reasoning about persistence and causation. *Computational Intelligence*, 5(3):142–150, 1989.

[40] Kris Demuynck. *Extracting, Modelling and Combining Information in Speech Recognition*. PhD thesis, K.U.Leuven ESAT, 2001.

[41] Kris Demuynck, Tom Laureys, and Steven Gillis. Automatic generation of phonetic transcriptions for large speech corpora. In *Proceedings of Interspeech*, volume I, pages 333–336, 2002.

[42] Kris Demuynck, Jan Roelens, Dirk Van Compernolle, and Patrick Wambacq. SPRAAK: An open source SPeech Recognition and Automatic Annotation Kit. In *Proceedings of Interspeech*, pages 495–498, 2008.

[43] Kris Demuynck, Antti Puurula, Dirk Van Compernolle, and Patrick Wambacq. The ESAT 2008 system for N-Best Dutch speech recognition benchmark. In *IEEE workshop on automatic speech recognition and understanding*, pages 339–343, 2009.

[44] Li Deng, Brian Hutchinson, and Dong Yu. Parallel training for deep stacking networks. In *Proceedings of Interspeech*, pages 2598–2601, 2012.

[45] Li Deng, Dong Yu, and John C. Platt. Scalable stacking and learning for building deep architectures. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 2133–2136, 2012.

[46] Jacques Duchateau, Kris Demuynck, and Patrick Wambacq. Confidence scoring based on backward language models. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 221–224, 2002.

[47] Jeffrey L. Elman. Finding structure in time. *Cognitive Science*, 14(2):179–211, 1990.

[48] Jeffrey L. Elman. Learning and development in neural networks: the importance of starting small. *Cognition*, 48(1):71 – 99, 1993.

[49] Ahmad Emami and Frederick Jelinek. A neural syntactic language model. *Machine Learning*, 60(1-3):195–227, 2005.

[50] Rong-En Fan, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-Rui Wang, and Chih-Jen Lin. Liblinear: A library for large linear classification. *Journal of Machine Learning Research*, 9:1871–1874, August 2008.

[51] Sergey Feldman, Marius A. Marin, Mari Ostendorf, and Maya R. Gupta. Part-of-speech histograms for genre classification of text. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 4781 –4784, april 2009.

[52] Denis Filimonov and Mary Harper. A joint language model with fine-grain syntactic tags. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*, pages 1114–1123, 2009.

[53] Andrew Finch and Eiichiro Sumita. Bidirectional phrase-based statistical machine translation. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*, pages 1124–1132, 2009.

[54] John R. Firth. A synopsis of linguistic theory, 1930-1955. *Studies in Linguistic Analysis*, pages 1–32, 1957.

[55] Victoria Fromkin and Robert Rodman. *An Introduction to language*. New York: Harcourt Brace Jovanovich, 1993.

[56] Daniel Gildea and Thomas Hofmann. Topic-based language models using EM. In *Proceedings of EUROSPEECH*, pages 2167–2170, 1999.

[57] Daniel Gildea and Thomas Hofmann. Topic-based language models using EM. In *Proceedings of EUROSPEECH*, pages 2167–2170, 1999.

[58] Joshua Goodman. Classes for fast maximum entropy training. In *Proceedings of IEEE International Conference onAcoustics, Speech, and Signal Processing*, volume 1, pages 561–564 vol.1, 2001.

[59] Joshua T. Goodman. A bit of progress in language modeling. *Computer Speech & Language*, pages 403–434, 2001.

[60] Peter A. Heeman. Pos tags and decision trees for language modeling. In *Proceedings of The Joint SIGDAT Conference on Empirical Methods in Natural Language Processing and Very Large Corpora*, pages 129–137, 1999.

[61] Aaron Heidel, Hung An Chang, and Lin Shan Lee. Language model adaptation using latent dirichlet allocation and an efficient topic inference algorithm. In *Proceedings of Interspeech*, pages 2361–2364, 2007.

[62] Jonathou Hull. Combining syntactic knowledge and visual text recognition: A hidden markov model for part of speech tagging in a word recognition algorithm. In *AAAI symposium: Probabilistic Approaches to Natural Language*, pages 77–83, 1992.

[63] Rukimini Iyer and Mari Ostendorf. Modeling long distance dependence in language: Topic mixtures vs. dynamic cache models. *IEEE Transactions on Speech and Audio Processing*, 7(1):236–239, 1999.

[64] Rukimini Iyer, Mari Ostendor, and Marie Meteer. Analyzing and predicting language model improvements. In *Proceedings of the IEEE Workshop on Automatic Speech Recognition and Understanding.*, pages 254 – 261, 1997.

[65] Rukmini Iyer, Mari Ostendorf, and Jan Robin Rohlicek. Language modeling with sentence-level mixtures. In *Proceedings of the workshop on Human Language Technology*, pages 82–87, 1994.

[66] Edwin Thompson Jaynes. Information Theory and Statistical Mechanics. *Physical Review Online Archive (Prola)*, 106(4):620–630, May 1957.

[67] Frederick Jelinek and Robert L. Mercer. Interpolated estimation of Markov source parameters from sparse data. pages 381–397, May 1980.

[68] Frederick Jelinek and Robert L. Mercer. Probability distribution estimation from sparse data. *IBM Technical Disclosure Bulletin*, 28:2591–2594, 1985.

[69] Frederick Jelinek, Bernard Mrialdo, Salim Roukos, and Martin J Strauss. A dynamic language model for speech recognition. In *Proceedings of the workshop on Speech and Natural Language*, pages 293–295. Association for Computational Linguistics, 1991.

[70] Thorsten Joachims. Text categorization with support vector machines: learning with many relevant features. In *Proceedings of 10th European Conference on Machine Learning*, pages 137–142, 1998.

[71] Jussi Karlgren and Douglass Cutting. Recognizing text genres with simple metrics using discriminant analysis. In *Proceedings of the 15th conference on Computational linguistics*, pages 1071–1075, 1994.

[72] Slava M. Katz. Estimation of probabilities from sparse data for the language model component of a speech recognizer. In *IEEE Transactions on Acoustics, Speech and Signal Processing*, pages 400–401, 1987.

[73] Mark D. Kernighan, Kenneth W. Church, and William A. Gale. A spelling correction program based on a noisy channel model. In *Proceedings of the 13th conference on Computational linguistics*, pages 205–210, 1990.

[74] Judith Kessens and David A. van Leeuwen. N-Best: the Northern- and Southern-Dutch benchmark evaluation of speech recognition technology. In *Proceedings of Interspeech*, pages 1354–1357, 2007.

[75] Brett Kessler, Geoffrey Numberg, and Hinrich Schütze. Automatic detection of text genre. In *Proceedings of the eighth conference on European chapter of the Association for Computational Linguistics*, pages 32–38, 1997.

[76] Reinhard Kneser and Hermann Ney. Improved backing-off for *m*-gram language modeling. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 181–184, 1995.

[77] Reinhard Kneser and Volker Steinbiss. On the dynamic adaptation of stochastic language models. In *Proceedings of Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing-93, Minnapolis(USA)*, volume II, pages 586–589, April 1993.

[78] Roland Kuhn and Renato de Mori. A cache-based natural language model for speech recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12 (6):570–583, 1990.

[79] Roland Kuhn and Renato de Mori. A cache-based natural language model for speech recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12 (6):570–583, 1990.

[80] Hong-Kwang Jeff Kuo, Ebru Arisoy, Ahmad Emami, and Paul Vozila. Large scale hierarchical neural network language models. In *Proceedings of Interspeech*, 2012.

[81] Harold Joseph Kushner and G. George Yin. *Stochastic Approximation Algorithm and Applications*. New York: Springer-Verlag, 1997.

[82] William Labov. *Sociolinguistic patterns*. University of Pennsylvania Press, 1972.

[83] Thomas K Landauer and Susan T. Dutnais. Solution to Plato's Problem: The Latent Semantic Analysis Theory of Acquisition, Induction and Representation of Knowledge. *Psychological Review*, (104), 1997.

[84] Thomas K. Landauer, Peter W. Foltz, and Darrell Laham. An Introduction to Latent Semantic Analysis. *Discourse Processes*, (25):259–284, 1998.

[85] Pat Langley, Wayne Iba, and Kevin Thompson. An analysis of Bayesian classifiers. In *proceedings of the tenth national conference on artificial intelligence*, pages 223–228, 1992.

[86] Raymond Lau, Ronald Rosenfeld, and Salim Roukos. Trigger-based language models: a maximum entropy approach. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing*, volume 2, pages 45 –48 vol.2, april 1993.

[87] Yann LeCun, Leon Bottou, Genevieve B Orr, and Klaus Robert Müller. Efficient BackProp. In G. Orr and K. Müller, editors, *Neural Networks—Tricks of the Trade*, volume 1524, pages 5–50. Springer Verlag, 1998.

[88] Yong-Bae Lee and Sung Hyon Myaeng. Text genre classification with genre-revealing and subject-revealing features. In *Proceedings of the 25th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 145–150, 2002.

[89] Stephen C. Levinson. Activity types and language. *Linguistics*, 17.5-6:365–400, 1979.

[90] George James Lidstone. Note on the general case of the Bayes–Laplace formula for inductive or a posteriori probabilities. *Transactions of the Faculty of Actuaries*, 8: 182–192, 1920.

[91] David J. C. Mackay. *Information Theory, Inference and Learning Algorithms*. Cambridge University Press, first edition edition, 2003.

[92] Gideon Mann, Ryan Mcdonald, Mehryar Mohri, Nathan Silberman, and Daniel D. Walker. Efficient large-scale distributed training of conditional maximum entropy models. In *Advances in Neural Information Processing Systems*, pages 1231–1239, 2009.

[93] Christopher D. Manning and Hinrich Schütze. *Foundations of statistical natural language processing*. MIT Press, Cambridge, MA, USA, 1999.

[94] Eric Mays, Fred J. Damerau, and Robert L. Mercer. Context based spelling correction. *Information Processing and Management*, 27(5):517–522, 1991.

[95] Grégoire Mesnil, Xiaodong He, Li Deng, and Yoshua Bengio. Investigation of recurrent-neural-network architectures and learning methods for spoken language understanding. In *Proceedings of Interspeech*, page to appear, 2013.

[96] Tomas Mikolov. *Statistical Language Models Based on Neural Networks*. PhD thesis, Brno University of Technology, 2012.

[97] Tomas Mikolov and Geoffrey Zweig. Context dependent recurrent neural network language model. In *IEEE Workshop on Spoken Language Technology*, pages 234–239, 2012.

[98] Tomas Mikolov, Martin Karafiát, Lukas Burget, Jan Cernocký, and Sanjeev Khudanpur. Recurrent neural network based language model. In *Proceedings of Interspeech*, pages 1045–1048, 2010.

[99] Tomas Mikolov, Anoop Deoras, Stefan Kombrink, Lukas Burget, and Jan ernock. Empirical evaluation and combination of advanced language modeling techniques. In *Proceedings of Interspeech*, pages 605–608, 2011.

[100] Tomas Mikolov, Anoop Deoras, Daniel Povey, Lukas Burget, and Jan Cernocký. Strategies for training large scale neural network language models. In *IEEE Workshop on Automatic Speech Recognition and Understanding*, pages 196–201, 2011.

[101] Tomas Mikolov, Stefan Kombrink, Lukas Burget, Jan Cernocky, and Sanjeev Khudanpur. Extensions of recurrent neural network language model. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 5528 –5531, 2011.

[102] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *CoRR*, abs/1301.3781, 2013.

[103] Piotr Mirowski, Sumit Chopra, Suhrid Balakrishnan, and Srinivas Bangalore. Feature-rich continuous language models for speech recognition. In *Proceeding of IEEE Spoken Language Technology Workshop*, pages 241–246, 2010.

[104] Andriy Mnih and Yee Whye Teh. A fast and simple algorithm for training neural probabilistic language models. In *Proceedings of the 29th International Conference on Machine Learning*, pages 1751–1758, 2012.

[105] Kevin Patrick Murphy. *Dynamic Bayesian Networks: Representation, Inference and Learning*. PhD thesis, University of California, Berkeley, 2002.

[106] Hermann Ney, Ute Essen, and Reinhard Kneser. On Structuring Probabilistic dependencies in stochastic language modelling. *Computer Speech and Language*, 8, 1994.

[107] Thomas Niesler and Philip C. Woodland. Combination of word-based and category-based language models. In *Proceedings of International Conference on Spoken Language*, volume 1, pages 220–223 vol.1, 1996.

[108] Thomas R. Niesler, Edward W. D. Whittaker, and Philip C. Woodland. Comparison of part-of-speech and automatically derived category-based language models for

speech recognition. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 177–180 vol.1, 1998.

[109] Nicolas Obin, Volker Dellwo, Anne Lacheret, and Xavier Rodet. Expectations for discourse genre identification: a prosodic study. In *Proceedings of Interspeech*, pages 3070–3073, 2010.

[110] Nelleke Oostdijk. Building a corpus of spoken Dutch, 1999. URL http://lands.let.kun.nl/cgn/.

[111] Nelleke Oostdijk, Wim Goedertier, Frank Van Eynde, Louis Boves, Jean Pierre Martens, Michael Moortgat, and Harald Baayen. Experiences from the Spoken Dutch Corpus project. In *Araujo (eds), Proceedings of the Third International Conference on Language Resources and Evaluation*, pages 340–347, 2002.

[112] Marl Ostendorf, Ashvin Kannan, Steve Austin, Owen Kimball, Rich Schwartz, and Jan Robin Rohlicek. Integration of diverse recognition methodologies through reevaluation of n-best sentence hypotheses. In *Proceedings of the workshop on Speech and Natural Language*, pages 83–87, 1991.

[113] Judea Pearl. *Probabilistic Reasoning in Intelligent Systems - Networks of Plausible Inference*. Morgan Kaufmann Publishers, Inc., 1988. ISBN 0-934613-73-7.

[114] Fuchun Peng and Dale Schuurmans. Combining naive Bayes and n-gram language models for text classification. In *25th European Conference on Information Retrieval Research*, pages 335–350, 2003.

[115] Fernando Pereira, Naftali Tishby, and Lillian Lee. Distributional clustering of English words. In *Association for Computational Linguistics*, pages 183–190, 1993.

[116] Steven T. Piantadosi, Harry Tily, and Edward Gibson. Word lengths are optimized for efficient communication. *Proceedings of the National Academy of Sciences*, 108 (9):3526–3529, January 2011.

[117] Stephen A. Della Pietra, Vincent J. Della Pietra, Robert L. Mercer, and Salim Roukos. Adaptive language modeling using minimum discriminant estimation. In *Proceedings of the workshop on Speech and Natural Language*, pages 103–106, 1992.

[118] Gerasimos Potamianos and Frederick Jelinek. A study of n-gram and decision tree letter language modeling methods. 24(3):171–192, 1998.

[119] Lin Qiu and Jungang Xu. A chinese word clustering method using latent dirichlet allocation and k-means. In *International Conference on Advances in Computer Science and Engineering*, 2013.

[120] Benjamin Recht, Christopher Re, Stephen J. Wright, and Feng Niu. Hogwild: A lock-free approach to parallelizing stochastic gradient descent. In *Advances in Neural Information Processing Systems*, pages 693–701, 2011.

[121] Klaus Ries, Lori Levin, Liza Valle, Alon Lavie, and Alex Waibel. Shallow discourse genre annotation in callhome Spanish. In *in Proceecdings of the International Conference on Language Ressources and Evaluation*, 2000.

[122] Tony Robinson and Frank Fallside. Word recognition from the DARPA resource management datadata with the Cambridge recurrent error propagation network speech recognition system. *Computer Speech and Language*, 5:362–367, 1991.

[123] Tony Robinson, Mike Hochberg, and Steve Renals. The use of recurrent networks in continuous speech recognition. In C. H. Lee, K. K. Paliwal, and F. K. Soong, editors, *Automatic Speech and Speaker Recognition – Advanced Topics*, pages 233–258. Kluwer Academic Publishers, 1996.

[124] Ronald Rosenfeld. *Adaptive Statistical Language Modeling: A Maximum Entropy Approach*. PhD thesis, School of Computer Science, Carnegie Mellon University, 1994.

[125] Ronald Rosenfeld. A maximum entropy approach to adaptive statistical language modeling. *Computer, Speech and Language*, 10(3):187–228, 1996.

[126] Ronald Rosenfeld. Two decades of statistical language modeling: Where do we go from here? *Proceedings of the IEEE*, 88(8):1270–1278, 2000.

[127] Ronald Rosenfeld, Stanley F. Chen, and Xiaojin Zhu. Whole-sentence exponential language models: a vehicle for linguistic-statistical integration. *Computer Speech and Language*, 15(1):55 – 73, 2001.

[128] Leon J. M. Rothkrantz and Dann Nollen. Speech recognition using elman neural networks. In *Text, Speech and Dialogue*, pages 146–151, 1999.

[129] David E. Rumelhart, James L. McClelland, and CORPORATE PDP Research Group, editors. *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*. MIT Press, Cambridge, MA, USA, 1986.

[130] Ralf Salomon and J. Leo Van Hemmen. Accelerating backpropagation through dynamic self-adaptation. *Neural Networks*, 9:589–601, 1996.

[131] Marina Santini. A shallow approach to syntactic feature extraction for genre classification. In *7th Annual CLUK Research Colloquium*, 2004.

[132] Marina Santini. Some issues in automatic genre classification of web pages. In *In JADT 2006 - 8mes Journes*, 2006.

[133] Mike Schuster and Kuldip K. Paliwal. Bidirectional recurrent neural networks. *IEEE Transactions of Signal Processing*, 45(11):2673–2681, 1997.

[134] Holger Schwenk. Efficient training of large neural networks for language modeling. In *Proceedings of IEEE International Joint Conference on Neural Networks*, pages 3059–3064.

[135] Holger Schwenk. Continuous space language models. *Computer Speech and Language*, 21(3):492–518, 2007.

[136] Claude Elwood Shannon. Prediction and entropy of printed english. *Bell Systems Technical Journal*, 30:50–64, 1951.

[137] Yangyang Shi, Pascal Wiggers, and Catholijn M Jonker. Language modelling with dynamic bayesian networks using conversation types and part of speech information. In *The 22nd Benelux Conference on Artificial Intelligence*, pages 154–161, 2010.

[138] Yangyang Shi, Pascal Wiggers, and Catholijn M. Jonker. Combining topic specific language models. In *Proceedings of the International Conference on Text, Speech and Dialogue*, pages 99–106, 2011.

[139] Yangyang Shi, Pascal Wiggers, and Catholijn M. Jonker. Socio-situational setting classification based on language use. In *IEEE workshop on automatic speech recognition and understanding*, pages 455 – 460, 2011.

[140] Yangyang Shi, Pascal Wiggers, and Catholijn M Jonker. Towards recurrent neural networks language models with linguistic and contextual features. In *Proceedings of Interspeech*, pages 1664–1667, 2012.

[141] Yangyang Shi, Pascal Wiggers, and Catholijn M. Jonker. Adaptive language modeling with a set of domain dependent models. In *Proceedings of the International Conference on Text, Speech and Dialogue*, pages 472–479, 2012.

[142] Yangyang Shi, Mei-Yuh Hwang, Kaisheng Yao, and Martha Larson. Speed up of recurrent neural network language models with sentence independent subsampling stochastic gradient descent. In *Proceedings of Interspeech*, page to appear, 2013.

[143] Yangyang Shi, Martha Larson, and Catholijn M Jonker. K-component recurrent neural network language models using curriculum learning. In *IEEE workshop on automatic speech recognition and understanding*, page to appear, 2013.

[144] Yangyang Shi, Martha Larson, Pascal Wiggers, and Catholijn M. Jonker. Exploiting the succeeding words in recurrent neural network language models. In *Proceedings of Interspeech*, page to appear, 2013.

[145] Yangyang Shi, Pascal Wiggers, and Catholijn M. Jonker. Classifying the socio-situational settings of transcripts of spoken discourses. *Speech Communication*, 55 (10):988 – 1002, 2013.

[146] Bengt Sigurd, Mats Eeg-Olofsson, and Joost Van Weijer. Word length, sentence length and frequency—Zipf revisited. *Studia Linguistica*, 58(1):37–52, 2004.

[147] Man-Hung Siu and Herbert Gish. Evaluation of word confidence for speech recognition systems. *Computer Speech & Language*, 13(4):299–319, 1999.

[148] Rohini Srihari and Charlotte Baltus. Combining statistical and syntactic methods in recognizing handwritten sentences. In *AAAI symposium: Probabilistic Approaches to Natural Language*, pages 121–127, 1992.

[149] Efstathios Stamatatos, Nikos D. Fakotakis, and George K. Kokkinakis. Text genre detection using common word frequencies. In *Proceedings of the 18th conference on Computational linguistics - Volume 2*, pages 808–814, 2000.

[150] Volker Steinbiss, Bach-Hiep Tran, and Hermann Ney. Improvements in beam search. In *Proceedings of the International Conference on Spoken Language Processing*, pages 2143–2146, 1994.

[151] Panos Stinis. Stochastic global optimization as a filtering problem. *Journal of Computational Physics*, 231(4):2002 – 2014, 2012.

[152] Andreas Stolcke, Yochai Knig, and Mitchel Weintraub. Explicit word error minimization in n-best list rescoring. In *Proc. EUROSPEECH*, pages 163–166, 1997.

[153] Yi Su. *Knowledge Integration Into Language Models: A Random Forest Approach*. BiblioBazaar, 2011.

[154] Yik-Cheung Tam and Tanja Schultz. Dynamic language model adaptation using variational bayes inference. In *Proceedings of Interspeech*, pages 5–8, 2005.

[155] Sergios Theodoridis and Konstantinos Koutroumbas. *Pattern Recognition*. Academic Press, 4th edition edition, 2009.

[156] Simon Tong and Daphne Koller. Support vector machine active learning with applications to text classification. *Journal of Machine Learning Research*, 2:45–66, March 2002.

[157] G. Udny Yule. On sentence-length as a statistical characteristic of style in prose: With application to two cases of disputed authorship. *Biometrika*, 30(3/4):363–390, 1939.

[158] Joerg P. Ueberla. More efficient clustering of n-grams for statistical language modeling. In *Proceedings of EUROSPEECH*, pages 1257–1260, 1995.

[159] Alan van den Bosch. Scalable classification-based word prediction and confusible correction. *Traitement Automatique des Langues*, 46(2):39–63, 2006.

[160] Frank Van Eynde. Part of speech tagging en lemmatisering van het corpus gesproken nederlands. Technical report, K.U.Leuven, 2004.

[161] Sofie Van Gijsel, Dirk Speelman, and Dirk Geeraerts. Locating lexical richness : a corpus linguistic, sociovariational analysis. *Les journes internationales danalyse des donnes textuelles JaDT Proceedings of the 8th International Conferene on the statistical analysis of textual data JADT*, 2:961–972, 2006.

[162] Wen Wang and Mary P. Harper. The superARV language model: Investigating the effectiveness of tightly integrating multiple knowledge sources. In *Proceedings of Conference of Empirical Methods in Natural Language Processing*, pages 238–247, 2002.

[163] Paul J. Werbos. Backpropagation through time: what it does and how to do it. *Proceedings of the IEEE*, 78(10):1550–1560, 1990.

[164] Pascal Wiggers. *Modelling Context in Automatic Speech Recognition*. PhD thesis, 2008.

[165] Pascal Wiggers and Leon J. M. Rothkrantz. Dynamic bayesian networks for language modeling. In *Text and Speech and Dialogue*, page 555–562, 2006.

[166] Pascal Wiggers and Leon J. M. Rothkrantz. Topic-based language modeling with dynamic bayesian networks. In *Proceedings of the Ninth International Conference on Spoken Language Processing*, pages 1866–1869, 2006.

[167] Pascal Wiggers and Leon J. M. Rothkrantz. Exploratory analysis of word use and sentence length in the spoken Dutch corpus. In *Proceedings of the International Conference on Text, Speech and Dialogue*, pages 366–373, 2007.

[168] Youzheng Wu, Xugang Lu, Hitoshi Yamamoto, Shigeki Matsuda, Chiori Hori, and Hideki Kashioka. Factored language model based on recurrent neural network. In *Proceedings of International Conference of Computational Linguistics*, pages 2835–2850, 2012.

[169] Deyi Xiong, Min Zhang, and Haizhou Li. Enhancing language models in statistical machine translation with backward n-grams and mutual information triggers. In *Association for Computational Linguistics*, pages 1288–1297, 2011.

[170] Peng Xu and Frederick Jelinek. Random forests in language modeling. In *Proceedings of EMNLP*, pages 325–332, 2004.

[171] Puyang Xu, Asela Gunawardana, and Sanjeev Khudanpur. Efficient subsampling for training complex language models. In *Proceedings of EMNLP*, pages 1128–1136, 2011.

[172] Hirofumi Yamamoto and Yoshinori Sagisaka. Multi-class composite n-gram based on connection direction. In *Proceedings of 1999 IEEE International Conference on Acoustics, Speech, and Signal Processing*, pages 533–536, 1999.

[173] Kaisheng Yao, Geoffrey Zweig, Mei-Yuh Hwang, Yangyang Shi, and Dong Yu. Recurrent neural networks for language understanding. In *Proceedings of Interspeech*, pages 2524–2528, 2013.

[174] Dong Yu and Li Deng. Deep convex net: A scalable architecture for speech pattern classification. In *Proceedings of Interspeech*, pages 2285–2288, 2011.

[175] Carole Zangari, Lyle Lloyd, and Beverly Vicker. Augmentative and alternative communication: An historic perspective. *Augmentative and alternative communication*, 10(1):27–59, 1994.

[176] Martin Zinkevich, Alex Smola, and John Langford. Slow Learners are Fast. In *Advances in Neural Information Processing Systems 22*, pages 2331–2339, 2009.

[177] Martin Zinkevich, Markus Weimer, Alex Smola, and Lihong Li. Parallelized stochastic gradient descent. In *Advances in Neural Information Processing Systems 23*, pages 2595–2603, 2010.

[178] George K. Zipf. *Human Behavior and the Principle of Least Effort.* Addison-Wesley (Reading MA), 1949.

Curriculum vitae

Yangyang Shi was born on June 19, 1983 in city of Yancheng, Jiangsu, P. R. China. He obtained his BSc in Mathematics from Nanjing University of Information Science and Technology in 2006. He received his MSc in Mathematics from Southeast University in 2009.

From October, 2009 to October, 2013, he was a PhD student in Interactive Intelligence Group, Delft University of Technology, the Netherlands. He was supervised by Prof. Catholijn M. Jonker and Assist Prof. Pascal Wiggers from Interactive Intelligence Group, and Assist Prof. Martha Larson from Multimedia Computing Group. His research topics are data mining, machine learning and language modeling in automatic speech recognition. In 2012, Yangyang did an internship in Microsoft Asia on the use of multi-thread, multi-processor and GPU techniques to speed up the training of RNNLMs, mentored by Dr. Mei-Yuh Hwang and Dr. Kaisheng Yao.