COMPOSITIONAL LANGUAGE MODELING

By

KUSHAL ARORA

A THESIS PRESENTED TO THE GRADUATE SCHOOL
OF THE UNIVERSITY OF FLORIDA IN PARTIAL FULFILLMENT
OF THE REQUIREMENTS FOR THE DEGREE OF
MASTER OF SCIENCE

UNIVERSITY OF FLORIDA

2015

ACKNOWLEDGMENTS

The time spent on pursuing this work in the form of my master's thesis, has not only been challenging but a great learning experience as well. I am thankful to my advisor Prof Anand Rangarajan for continuously motivating me and guiding me to accomplish this work. Without his able guidance, this thesis would not have been possible. I would also like to thank my other committee members, Dr. Arunava Banerjee and Dr. Alireza Entezari for accepting my thesis proposal and becoming a part of this research.

I am also very grateful to my family and friends for constant support. I am indebted to them for discussion of ideas, consistent feedback as well as for their belief in me. Their encouragement has given me the strength and perseverance to keep at it.

TABLE OF CONTENTS

page

ACKNOWLEDGMENTS . . . 3

LIST OF TABLES . . . 6

LIST OF FIGURES . . . 7

ABSTRACT . . . 8

CHAPTER

1 INTRODUCTION . . . 10

2 BACKGROUND . . . 13

2.1 Traditional Language Modeling . . . 13
2.1.1 N-gram Language Models . . . 13
2.1.2 Long-Context Dependencies . . . 14
2.1.3 Generalization . . . 14
2.1.4 Extensions of N-gram Models . . . 15
2.1.4.1 Class based language model . . . 15
2.1.4.2 Structured language model . . . 15
2.2 Neural Network Language Models . . . 16
2.2.1 Feed Forward Neural Language Model(NNLM) . . . 16
2.2.2 Recurrent Neural Language Model(RNNLM) . . . 18
2.3 Probabilistic Context Free Grammars . . . 20
2.3.1 Inside-Outside Algorithm . . . 21
2.3.1.1 Inside algorithm . . . 21
2.3.1.2 Outside algorithm . . . 23
2.3.1.3 Calculating $\mu(r)$ . . . 26
2.4 Compositional Approach to Language Processing . . . 28
2.4.1 Composition Function and Scoring . . . 28
2.4.2 Compositional Vector Grammar(CVG) . . . 30

3 COMPOSITIONAL LANGUAGE MODELING . . . 32

3.1 Introduction . . . 32
3.2 Recurrent Neural Network Language Model as a Compositional Model . . . 34
3.3 Unconditioned Compositional Language Model . . . 36
3.4 Tree Representation . . . 37
3.5 Computing Sentence Probability . . . 38
3.6 Training . . . 41
3.6.1 E step . . . 41
3.6.1.1 Outside algorithm . . . 43
3.6.1.2 Calculating $\mu(r)$ . . . 44
3.6.2 M step . . . 45

3.7 Modeling compositional probability $p(r)$ 45
3.8 Phrasal Representation 47
3.9 Calculating $\partial E(r;\alpha)/\partial\alpha$ 47
3.10 Summary 50

4 EVALUATION AND RESULTS 52

4.1 Perplexity 52
4.2 Contrastive Entropy 55
4.3 Results 56

5 CONCLUSION 65

REFERENCES 67

BIOGRAPHICAL SKETCH 69

LIST OF TABLES

Table page

4-1 Example sentence with 20% and 40% distortion 58

4-2 Comparing n-gram models and RNNLM model perplexity for different level of distortion levels. 59

4-3 Comparison of current state of the art language model techniques to Compositional Language Model. CompositionalLM-30 and CompositionalLM-50 r refers to Compositional Language Model with embedding space dimension 30 and 50. 63

LIST OF FIGURES

Figure page

2-1 Neural Network Language Model architecture 17

2-2 Recurrent Neural Network Language Model architecture 18

2-3 Visualizing of $\pi(i, j, A)$ 21

2-4 Visualizing $\pi(A \rightarrow BC, i, k, j)$ 23

2-5 Visualizing $\beta(A, i, j)$ 24

2-6 *Outside Algorithm with right child left out* $\beta(B \rightarrow CAk, i-1, j)$ 25

2-7 *Outside Algorithm with left child left out* $\beta(B \rightarrow AC, i, j+1, k)$ 25

2-8 Visualizing $\mu(A, i, j)$ 27

2-9 Example of a recursive composition structure 29

3-1 Probability space on words for sentence *My dog likes eating pizza.* 32

3-2 Probability space on n-grams for sentence *My dog likes eating pizza.* 33

3-3 Single stage of RNNLM 35

3-4 Unrolled RNNLM showing the compositional structure 35

3-5 Parse tree for sentence W_{12345} 37

4-1 Contrastive Perplexity and Perplexity for all models, all distortion level 60

4-2 Perplexity Vs Distortion 60

4-3 Contrastive Perplexity vs percentage distortion 61

4-4 Correlation between Contrastive Perplexity vs Perplexity over models 61

4-5 Correlation between Contrastive Perplexity and Perplexity over percentage distortion. 62

4-6 Comparison of current state of the art language model techniques to Compositional Language Model. CompositionalLM-30 and CompositionalLM-50 r refers to Compositional Language Model with embedding space dimension 30 and 50. 64

Abstract of Thesis Presented to the Graduate School
of the University of Florida in Partial Fulfillment of the
Requirements for the Degree of Master of Science

COMPOSITIONAL LANGUAGE MODELING

By

Kushal Arora

December 2015

Chair: Anand Rangarajan
Major: Computer Engineering

One of the underlying assumption of most of the current state-of-art language models, is that, language can be modeled as a linear chain on probability space of words. This is a very strong assumption for modeling something inherently as complex as natural languages. We propose a different approach to this problem by modeling language as a composition on words and phrases. We start by showing how we can view the current state-of-art Recurrent Neural Network Language Model as a compositional model with a sequential tree structure assumption. We then remove this assumption by modeling joint probability of a sentence and compositional tree and then marginalizing it over all such trees. In this endeavor, we use phrasal-structure grammar or consistency grammar as a framework to define composition trees and to build probability space on them. The marginalization can be carried out efficiently using a modified version of Inside algorithm. The training is formulated as a maximum likelihood estimation problem and uses EM algorithm to carry out the maximization. We also present an efficient and exact way to compute gradients using Inside-Outside Algorithm. As perplexity cannot be calculated for phrase based language models due to intractability of partition function, a new in-vivo evaluation technique has been proposed, to compare our model with the current state-of-the-art. As a byproduct of this model, we get a word representation in the latent space and a compositional model to build phrasal embeddings. We hypothesize that these embeddings and the model can be seen as a language representation in the latent space

and can be used as a base for other natural language tasks like sentiment detection, NER, parsing and paraphrase detection.

CHAPTER 1
INTRODUCTION

The work of Hector J.Levesque,'On Our Best Behavior'[1], has been the motivation behind this work. In this work, the author deals with the topic how the technology encompassing Artificial Intelligence(AI) has taken over the science of AI, how we have forsaken the search of solutions which exhibit or mimic intelligent behavior for good enough solutions, that employ cheap tricks like heuristics or linear models tweaked specifically to certain benchmarks. Language modeling is a great example of this behavior. The cheap tricks(n-gram models) dominate the field due to ease of engineering solutions and easy access to increasing amounts of data. It is not that there has not been criticism of this simplistic statistical approach. One of the most prominent of such criticism has been by Noam Chomsky in a series of influential papers[2, 3]. The fallacies of n-gram models have been well known but instead of looking beyond the finite state model the research has been focused on plugging the shortcomings heuristically to address these underlying problems. Two most notable approaches in this regard are are Structured Language Models and Neural Network based language model. The former tries to address the long context dependency issues, by introducing headword for calculation of context and later embeds the history or context into the latent space to improve generalization. One of the most important property of natural language is its recursivity and existence of underlying compositional structure. We don't form phrases and sentence one word at a time, but by recursively building phrases from words, sentences from phrases and documents from sentences. The underlying structure of these compositions should play an important role in modeling something as complex as language. The computational linguist community has been working on understanding the underlying structure of language in the form of 'Grammar'. It is a belief that a step in the right direction would be to build a language model that uses these composition frameworks to assign probability to words and hence a sentence, than a simple frequency estimation.

The work discussed in this thesis explores the aforementioned compositional approach towards language modeling. This is accomplished by showing how the traditional models are basically compositional models with a strong assumption on composition structure. Furthermore, the research carried forward focuses on removing the assumptions. As the current in-vivo evaluation techniques too assume the finite state machine nature of language, a new evaluation technique has been presented, that works for both traditional and compositional models. Finally, the new technique has been employed to evaluate the proposed model as well as the traditional model, reporting the results on both.

Chapter 2 details the traditional approaches to language modeling. We highlight two fallacies of traditional approaches, lack of generalization and inability to capture long term dependencies. We also briefly discuss the work done in realms of traditional model to solve these issues. We then go on to discuss the background work that we will use to build our compositional language modeling, such as, Probabilistic Context Free Grammar(PCFGs), Inside-Outside Algorithm and recursive composition functions. We conclude Chapter 2 by discussing Compositional Vector Grammar model by Socher et al.[4] which takes a similar approach to ours, but does the inexact inference.

In Chapter 3, we discuss the proposed Compositional Language Modeling approach. We start by formulating how traditional language model assumes a sequential tree compositional structure. We follow this up formally showing that recursive definition of Recurrent Neural Network Language Model is indeed a composition framework built on sequential tree. We then remove this assumption by marginalizing joint probability of sentence and compositional tree over all trees. Further discussion shows how this marginalization can be efficiently performed using Inside Algorithm and also develop an efficient algorithm for likelihood maximization by using the Inside Algorithm along with Outside algorithm. A summary of our approach and discussion about the computation complexity of the model, concludes Chapter 3.

Chapter 4 throws light on the current evaluation techniques for language modeling: Word Error Rate and Perplexity. These approaches are unsuitable for us due to many constraints, such as data and pipeline unavailability, need for exact probability computation, etc. This led to us defining a new contrastive approach in Section 4.2. This enables us to evaluate compositional models without partition function computation. Towards the end, a comparative analysis of results for traditional language models and compositional language model on this new evaluation criteria is reported.

Finally, in Chapter 5 we conclude our work by stating the contribution of this work, as well as defining the future work that can build on top of this model.

CHAPTER 2
BACKGROUND

2.1 Traditional Language Modeling

2.1.1 N-gram Language Models

Traditional language models usually model language as linear chain. By this linear chain assumption, the probability of a sentence $W = w_1 w_2 w_3 .... w_{n-1} w_n$, could be factorized as:

$$p(W) = \prod_{i=1}^{N} p(w_i | w_1, .., w_{i-1}) \tag{2–1}$$

In this paradigm, the definition of language modeling can be seen as predicting next word given the history or the context. In its simplest formulation, these statistical models count the frequency of the word given the context, and build a sparse probability table i.e. $P(w_i | w_1 ... w_{i-1})$ based on them. One of the most apparent problem with this approach is that the number of parameters i.e. words in context grow exponentially. Traditional language models solve this problem by imposing Markovian assumption on conditional probability, i.e. the distribution of next word depends upon a fixed number of previous words. With this assumption, the conditional probability $p(w_i | w_1, ..., w_{i-1})$ can be modeled as

$$\begin{aligned} p(w_i | w_1, .., w_{i-1}) &= p(w_i | w_{i-n}, ..., w_{i-1}) \\ &= \frac{p(w_1, .., w_i)}{p(w_1, .., w_{i-1})} \end{aligned}$$

The most popular and widely used formulation is known as N-gram language model, where N is the order of the Markov chain.

As the number of parameters rise exponentially with the order of the chain, a lot of n grams would not be seen in training and would be assigned a zero probability. This leads to a sparsity problem famously known as the 'curse of dimensionality'. An effective solution used by the N-gram models to cope up with the data sparseness problem is to apply smoothing techniques. This helps redistribute the probability mass from the examples seen in the training data as well as to the unseen examples. A number of such smoothing techniques have been extensively studied in context of statistical language modeling. A good summary of these techniques and their effectiveness can be found in [5].

Despite their success in language modeling, due to their ability to scale up, these smoothened language models still fail on two accounts; generalization and capturing context dependencies beyond the Markov boundary. In the following section, we detail these issues and understand what exactly they mean in terms of language modeling and why are they of such importance.

2.1.2 Long-Context Dependencies

Let's consider the following sentence

The sky above our head is blue.

In this sentence, as we can see, the word *blue* is dependent on the word sky. We can re-formulate the same sentence in multiple ways, where this dependency is preserved. For example, *Sky this morning is blue*. A good model for language should be able to capture these regularities in language and assign *blue* a higher probability if *sky* was in context. The n-gram models with interpolation would need a context size of at-least 6, to be able to capture this relationship leading to huge probability tables and costly interpolation.

2.1.3 Generalization

Now, lets consider another sentence.

Party will be on Tuesday

Ideally, a language model, if shown two sentences *Party will be on Tuesday* and *Party will be on Friday*, with days of week in similar context, should be able to identify the pattern, generalize and assign a higher probability to *Party will be on Wednesday,* or for that matter *Party will be on WEEKDAY*, where *WEEKDAY* could be one of the classes covering all the days in a week. A good model should have inherent ability to do implicit clustering to understand that *Tuesday* and *Friday* are used in similar context at multiple places, so is

Wednesday. This is something that n-gram language models cannot do, unless, they come across the very same sentence or smaller n-grams for interpolation.

2.1.4 Extensions of N-gram Models

There have been a lot of attempts to improve generalization and long-context dependency in realm of traditional n-gram language models. Two notable approaches in the literature are Class Model[6] and Structured Language Model[7, 8] . The Class Model tries to address the generalization issue whereas the latter addresses long term context dependencies.

2.1.4.1 Class based language model

The underlying idea behind class-based model unfolds probabilistically mapping each word in vocabulary to one or more predetermined class and then build aN-gram model on these classes. These classes can either by manually created by experts or can be learnt statistically from the training data. For example, *WEEKDAY* can be one of the classes where we map all the days of the week. A detailed overview of various class-based model could be found in [9]. However this model brings its own upside; with their prime disadvantage being either the need of expert knowledge or a computationally expensive inference to build classes and word mapping. In addition to this, these models add an extra hyper-parameter; number of classes, which require optimization.

2.1.4.2 Structured language model

Structured Language models challenge n-order Markov assumption by adding syntactic cues like phrase trees headwords and/or POS tags to n-gram context. Most notable structured models by Chelba's[7] and Charnaik's[8] follow this approach. As the headword(word in phrase that can represent it) and POS tag of the phrase might lie beyond the Markovian boundary of the context, these models can capture longer context dependencies. The disadvantage of this approach is that despite using the syntactic structure, they don't go beyond the finite state model of language. Another interesting structured model that needs to be discussed here, is Chelba's[10]. In this work, they model probability at sentence level and use dependency parse trees as priors. However, instead of marginalizing over all trees, it is assumed that the probability on trees in concentrated mostly on a single tree i.e. the parse tree, thus avoiding the marginalization.

2.2 Neural Network Language Models

The reason why generalization is so difficult for traditional language models is because they treat words as discrete random variables. Due to lack of implicit notion of relatedness or closeness on a discrete set it is hard to define a smooth probability function on it. If we can embed words in a continuous space with some smoothness constraint then we can easily build a smooth probability distribution on this space, solving the generalization issue. This is the exact problem neural network based language models try to solve. By using the context similarity for distributed representation and by building a smooth density function on this space, automatic generalization is provided. In this section we discuss two famous neural network based approaches, a feed-forward based approach by Bengio et al.[11] that formalizes this idea, along with Recurrent Neural Network based approach by Mikolov et al.[12] that extends the NNLM to capture longer dependencies.

2.2.1 Feed Forward Neural Language Model(NNLM)

Feed forward neural network based model was proposed by [11] in 2003. The approach is straightforward and details of the approach are discussed next. The words are encoded in 1-hot vector representation and are projected into a continuous space using the projection matrix X, i.e. if e is the 1-of-V vector for a word, then Xe is its continuous space embedding. Let N be the length of the context. The embedding layer would take these N words, embed them into the latent space of dimension d and produce a vector of size Nd. The embedding layer is followed by a hidden layer which projects this Nd vector in a high dimensional space of typical dimension 200-300 and an output layer of size V, which assigns the probability over next word. This is akin to embedding history in latent space and building a smooth conditional distribution function over vocabulary on this space.

Formally this model can be described using the following equations:

$$y = b + Wx + U \tanh(d + Hx)$$

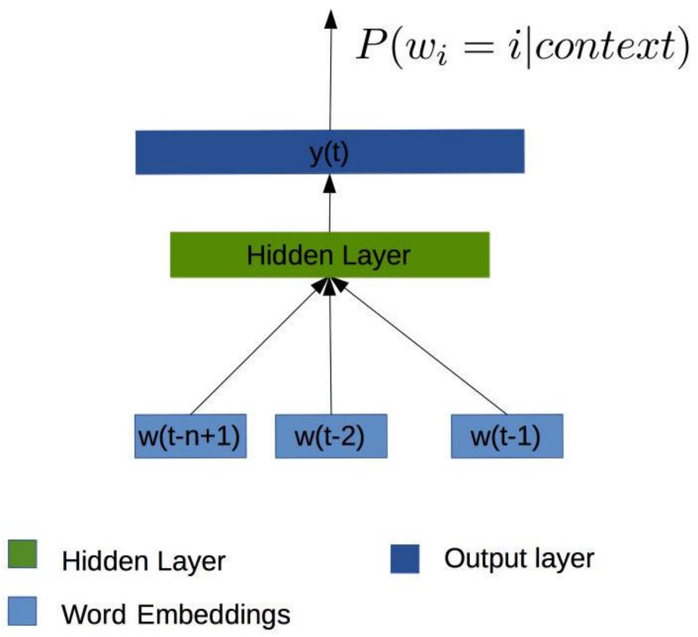


Figure 2-1. Neural Network Language Model architecture

$$\tilde{P}(w_t|w_{t-1}, w_{t-n+1}) = \frac{e^{y_{w_t}}}{\sum_i e^{y_{w_i}}}$$

where$x$ is the context embedding of length $(n-1)d$, b is the output layer biases of dimension V, W is a latent space to output layer weights of dimension $V \times (n-1)m$, H is a hidden layer weight matrix of dimension $h \times (n-1)d$, where where h is the dimension of hidden layer and d is a hidden layer biases of size h.

The main advantage of NNLM over the classical n-gram models and their extensions is that the prediction doesn't depend on exact context of N words rather on a projection of the sequence in a lower dimensional space. This drastically reduces parameter space of the model from $O(|V|^n)$ to $O(|V|)$ due to inherent clustering of similar histories. On the other hand, the biggest drawback of this model is the training time which prohibited this model to be trained for very large sequences. Various improvements in this regard were made by Mnih[13] and Morin[14], who proposed hierarchal model for training and Mikolov[15] which factorizes output layer to decrease the output layer size.

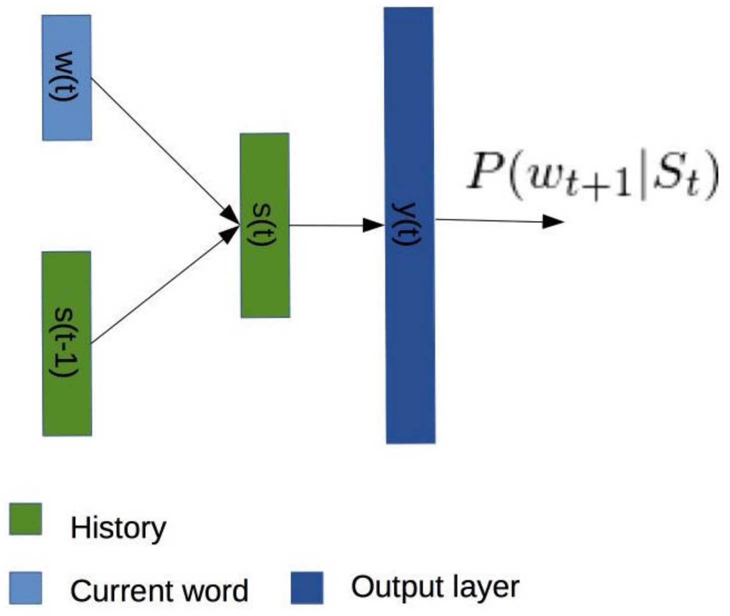


Figure 2-2. Recurrent Neural Network Language Model architecture

2.2.2 Recurrent Neural Language Model(RNNLM)

Though the NNLM model was able to better generalize than the n-gram models, due to distributed representation of context, yet, it still relied on Markovian assumption for defining the context. This led to its incapability of capturing long term dependencies. The Recurrent neural net based language model, proposed by Mikolov in [12, 16] improved this by learning the better representation of history during training. In their work, they treated history as a continuous vector $S(t)$ embedded in the space as word. This effectively meant that theoretically entire history can be used predicting the next word.

The recurrent neural model can be formulated using the following recursive equations:

$$x(t) = \left[w(t)^T s(t-1)^T\right]^T \tag{2–2}$$
$$s(t) = f(Ux(t)$$

The conditional probability density over vocabulary on this latent space is be defined as.

$$y(t) = g(Ws(t))$$

where non linearity like *tanh* is used for f and g is *softmax* function $w(t)$ is 1-of-V encoding of the word and $s(t-1)$ is hidden layer from previous time step. U, W are the parameters of the model of dimension $2n \times n$ and $n \times 1$. The output layer $y(t)$ represents $P(w(t)|s(t-1))$.

The model is trained using back-propagation through time(BPTT), proposed in [17] which treats a temporal recurrent neural network as an unfolded deep neural network and limits the error propagation to a few steps to avoid vanishing exploding gradient problem.

The RNN is currently the state of art technique for language modeling. Despite its huge success, and ability to address both the shortcomings of N-gram model, RNNLM still models the language as a finite state automaton and treats it as a problem of predicting the next word given the context. In Chapter 3, we will return to RNNLM and show how recursive equation 2–2 can be seen as a compositional model with a sequential tree assumption.

Now, we focus towards the background work that we would need, while defining Compositional Language Model in Chapter 3. We begin by looking as Probabilistic Context Free Grammars which is used to build compositional structures and assign the probability. We follow that by looking at Inside-Outside Algorithm which plays an essential part in both marginalization and gradient computation. Following that, we discuss the composition and scoring function for building phrases recursively and scoring them. We conclude this Chapter 2 with the discussion of Compositional Vector Grammar[4] model proposed by Socher et al. for language parsing. This model takes a similar compositional approach to ours, by building a joint probability over composition structures and sentence and then marginalizing over these structures, but instead of doing the exact inference, uses N-best trees to do the marginalization.

2.3 Probabilistic Context Free Grammars

A Probabilistic Context Free Grammar (G, θ) can be defined as a quintuple:

$$(G, \theta) = (N, T, R, S, P)$$

where

- N is set of non-terminal symbols
- T is the set of terminal symbols
- R is the finite set of production rules
- S is special start symbol
- P is the set of probabilities on production rule.

Here, θ can be defined as a real-valued vector of length $|R|$ indexed by production rules, where $\theta_r \in P$ and $r \in R$. As is with probabilities of an outcome,

$$\theta_r \geq 0$$

and

$$\sum_{A \to \beta \in R} \theta_{A \to \gamma} = 1$$

where $A \in N$ and $\gamma \in \{N \cup T\}^+$.

Using set of probabilities P on productions, we can model the probability of a tree $p(t)$ as a multinomial distribution as

$$p(t; \theta) \propto \prod_{r \in R} \theta_r^{f_r(t)} \tag{2–3}$$

where t is tree generated by G and $f_r(t)$ is the number of times production r is used in derivation of tree t.

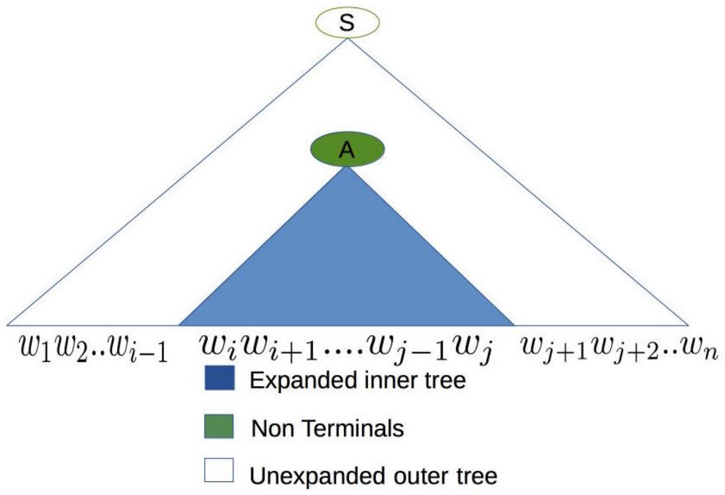


Figure 2-3. Visualizing of $\pi(i, j, A)$

2.3.1 Inside-Outside Algorithm

2.3.1.1 Inside algorithm

Having defined PCFGs and described a way to define probability distribution over trees $p(t)$, we now describe the well known dynamic programming based algorithm to calculate probability of sentence over all parse tree with assumption that $P(W|t) = 1$.

Let $\mathcal{T}_G(W)$ be the set of all parse trees for sentence W generated using grammar G. We can define $p(W)$ i.e. probability that grammar G generated sentence W, as

$$p(W) = \sum_{t \in \mathcal{T}_G(W)} p(W|t)p(t) \tag{2–4}$$

$$= \sum_{t \in \mathcal{T}_G(W)} p(t)$$

$$= K \sum_{t \in \mathcal{T}_G(W)} \prod_{r \in R} \theta_r^{f_r(t)} \tag{2–5}$$

Let $W = w_1w_2....w_n$, for any $A \in N$, span i, j such that $1 \leq i \leq j \leq n$, we define $\mathcal{T}_G(i, j, A)$, to be set of all parse trees for phrase $w_iw_{i+1}....w_j$ with A as a non-terminal root of the tree.

We define

$$\pi(i, j, A) = \sum_{t \in \mathcal{T}(i,j,A)} p(t) \tag{2–6}$$

This can be seen as probability of deriving sequence $w_i w_{i+1} \ldots . w_{j-1} w_j$ starting with non terminal A by recursively applying production rules in R to generate span $w_i \ldots w_j$. With $p(W|t) = 1$ assumption, this can be seen as probability of phrase $w_i w_{i+1} \ldots w_j$.

Using 2–6, we can define $p(W)$ as

$$p(W) = \pi(1, n, S) = \sum_{t \in \mathcal{T}(1,n,S)} p(t)$$

where S is special start symbol and is always the root node of a full parse tree. Using the definition similar 2–6, we can visualize $\pi(1, n, S)$ as probability of deriving $W = w_1 w_2 \ldots . w_n$ from S using production rules in grammar G. Having seen the relation between 2–6 and $p(W)$, we turn our focus to recursive definition of Inside Algorithm.

Base Case of the recursion is defined for leaf nodes as

$$\pi(i, i, A) \quad = \quad \begin{cases} \theta_{A \to w_i} & A \to w_i \in R \\ 0 & otherwise \end{cases} \tag{2–7}$$

i.e. probability of tree spanning a word rooted at non terminal A is nothing but the probability of unary production $\theta_{A \to w_i}$ itself.

Before defining probability of longer span(phrases) i, j rooted at non terminal A, lets define an intermediate term $\pi(A \to BC, i, k, j)$ that we will use in our computation.

We defining $\pi(A \to BC, i, k, j)$ as

$$\pi(A \to BC, i, k, j) = \theta_{A \to BC} \times \pi(i, k, B) \times \pi(k + 1, j, C) \tag{2–8}$$

This can be seen as probability of tree rooted at A spanning (i, j), splitting at k, such that subtrees (i, k) is rooted at B and $(k + 1, j)$ is rooted at C. This can be seen in Figure 2-4.

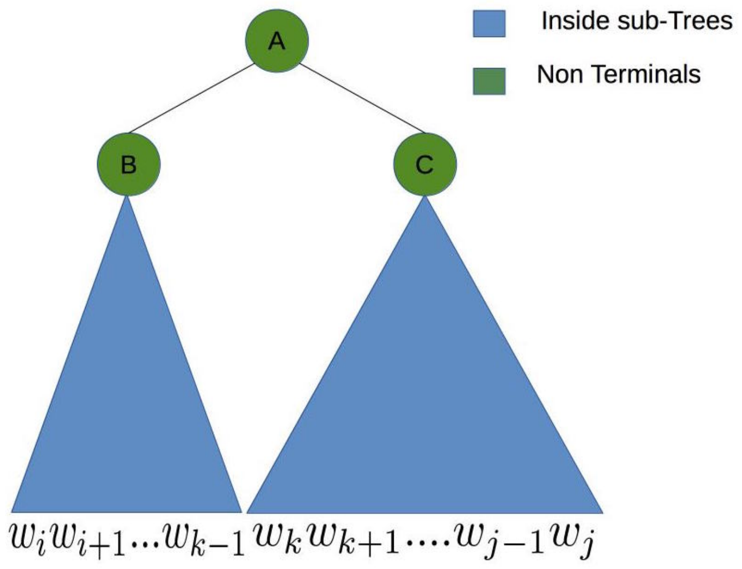


Figure 2-4. Visualizing $\pi(A \rightarrow BC, i, k, j)$

We can then calculate $\pi(i, j, A)$ by marginalizing over all splits and all binary rules rooted at A as follows:

$$\pi(i, j, A) = \sum_{A \rightarrow BC \in R} \sum_{i \leq k < j} \pi(A \rightarrow BC, i, k, j) \quad (2\text{–}9)$$

2.3.1.2 Outside algorithm

The Inside algorithm was concerned about the following problem. Given a non terminal $A \in N$, and a span (i, j), calculate the probability, $\pi(A, i, j)$, i.e. probability that A spans the sequence $w_i ... w_j$. The outside term β does the very opposite of this. The outside term $\beta(A, i, j)$ is the sum of all trees, starting with S as root spanning $1, n$ such that non terminal $A \in N$ spanning i, j is not expanded. Figure 2-5 shows the outside score $\beta(A, i, j)$ computation for span $w_i w_{i+1} .. w_j$ rooted at A.

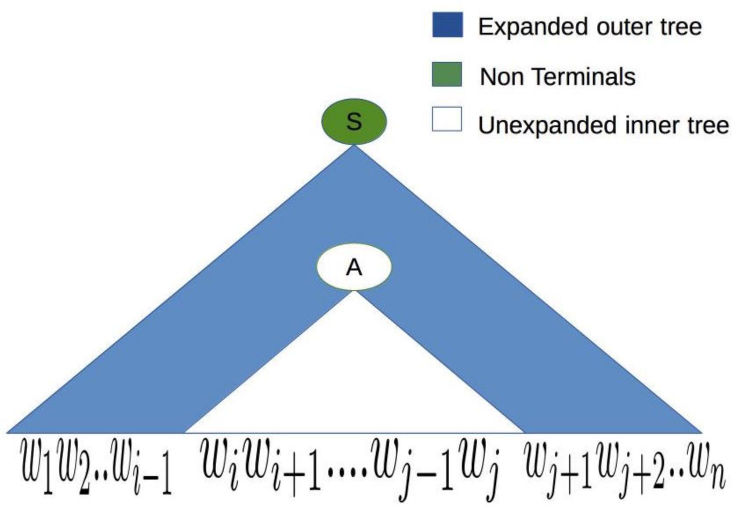

Expanded outer tree
Non Terminals
Unexpanded inner tree
S
A
$w_1 w_2 .. w_{i-1}$ $w_i w_{i+1} .... w_{j-1} w_j$ $w_{j+1} w_{j+2} .. w_n$

Figure 2-5. Visualizing $\beta(A, i, j)$

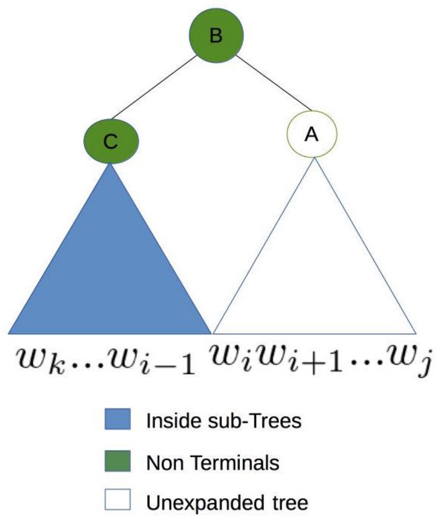
The base case $\beta(S, 1, n)$ is 1, as the probability of the root node being S, and the whole sentence not extended is 1. Moreover, as any other non-terminal $A \in N$ cannot be the root of full parse tree, hence $\beta(A, 1, n)$ is 0.

$$\begin{aligned} \beta(S, 1, n) &= 1 \\ \beta(A, 1, n) &= 0 \end{aligned}$$

Now, the recursive definition of $\beta(A, i, j)$ is as follows. Similar to Inside Algorithm, we define intermediate terms $\beta(B \to CA, k, i-1, j)$ and $\beta(B \to AC, i, j+1, k)$. In these two terms, we are trying to keep span (i, j) rooted at A unexpanded. We can calculate these terms by using recursive definition of inside and outside term as follows:

$$\beta(B \to CA, k, i-1, j) = \theta_{B \to CA} \times \pi(C, k, i-1) \times \beta(B, k, j) \tag{2–10}$$

$$\beta(B \to AC, i, j+1, k) = \theta_{B \to AC} \times \pi(C, j+1, k) \times \beta(B, i, k) \tag{2–11}$$



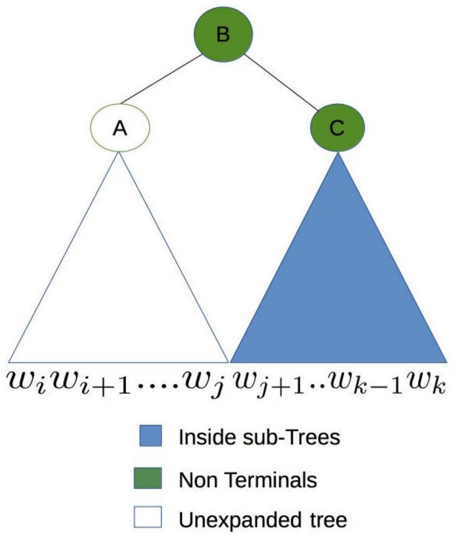


Figure 2-6. *Outside Algorithm with right child left out* $\beta(B \rightarrow CAk, i-1, j)$

Figure 2-7. *Outside Algorithm with left child left out* $\beta(B \rightarrow AC, i, j+1, k)$

Let's deconstruct 2–10 and 2–11. We start with outside tree spanning (k, j) rooted at B. $\beta(B \rightarrow CA, k, i-1, j)$ can be seen as outside tree rooted at B which splits at index i where span $(k,\ i-1)$ rooted at C is expanded whereas span (i, j) is left unexpanded. $\beta(B \rightarrow AC, i, j, k)$ can be expanded in similar way. Figure 2-6 and 2-7 shows this process.

We can now calculate outside probability of tree rooted at A spanning (i, j) by marginalizing over all these binary rules and all splits.

$$\beta(A, i, j) = \sum_{k=1}^{i-1} \sum_{B \to CA \in R} \beta(B \to CA, k, i-1, j) + \tag{2–12}$$

$$\sum_{k=j+1}^{n} \sum_{B \to AC \in R} \beta(B \to AC, i, j+1, k) \tag{2–13}$$

2.3.1.3 Calculating $\mu(r)$

During our training, we would need to add probability of all trees that contain a specific rule $r \in R$. Let's call this quantity $\mu(r)$. Formally, $\mu(r)$ can be defined as follows.

$$\mu(r) = \sum_{t \in \mathcal{T}_G(W) : r \in t} p(t)$$

i.e. sum of all trees in set of trees $\mathcal{T}_G(W)$ built on grammar G which contains rule r. Restricting production rules to CNF form, we need to compute the following terms $\mu(A, i, i)$ and $\mu(A \to BC, i, k, j)$ corresponding to unary $A \to w_i$ and binary rules $A \to BC$ respectively.

Before computing $\mu(r)$, we look at a related term $\mu(A, i, j)$. This term will help us better understand how we can compute $\mu(r)$ using the Inside-Outside Algorithm. $\mu(A, i, j)$ can be defined as sum of all trees that where span $w_i w_{i+1}..w_j$ is spanned by non terminal A. This can be calculated using the inside term $\pi(A, i, j)$ and $\beta(A, i, j)$ as follows:

$$\mu(A, i, j) = \pi(A, i, j) \times \beta(A, i, j)$$

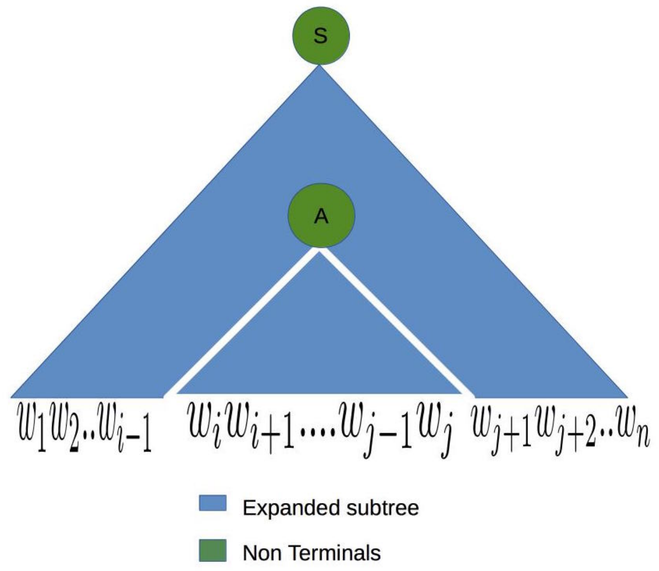


Figure 2-8. Visualizing $\mu(A, i, j)$

To understand this, let's recall $\beta(A, i, j)$ which is the sum of all probabilities where tree rooted at A spanning i, j is not expanded, whereas, $\pi(A, i, j)$ gives probability of non terminal A spanning i, j. Their product is all the trees rooted at S, spanning $1, n$ which contain non terminal A such that it spans (i, j).

On the similar lines $\mu(A, i, i)$ and $\mu(A \rightarrow BC, i, k, j)$ can be computed from inside outside probabilities as follows:

$$\mu(A \rightarrow w_i, i, i) = \pi(A, i, i) \times \beta(A, i, i) \tag{2–14}$$

$$\mu(A \rightarrow BC, i, k, j) = \beta(A, i, j) \times \pi(A \rightarrow BC, i, k, j) \tag{2–15}$$

Equation 2–14 can be the sum of the probability of all trees where non terminal A spans the leaf node i. Equation 2–15 is little more involved and can be seen as the sum of the probabilities of all trees where rule $A \rightarrow BC$ is used to build inside tree spanning $w_i w_{i+1} \ldots w_j$, rooted at A and split at index k.

2.4 Compositional Approach to Language Processing

Recently, compositional approaches have been used for various language processing tasks like sentiment analysis[18, 19], paraphrase detection[20], language parsing[4], and relation classification[21] to name a few. Our approach to composition is similar to Socher et al.'s work in Recursive Neural Network for language processing. This section begins by discussing the composition and scoring function that we borrow from Socher's work, followed by an overview of Compositional Vector Grammar model used by him for language parsing.

2.4.1 Composition Function and Scoring

Let's say we have a composition structure as shown in Figure 2-9. The basic component of a recursive neural network is a composition function h, which composes parent pa using children a, b.

$$pa = h(a, b)$$

and a scoring function g which assigns the score to each phrase(including the leaf nodes)

$$s = g(p)$$

where p can either be a leaf node or a phrase or a non leaf node.

In our case and Socher's, the composition and scoring function are defined as follows:

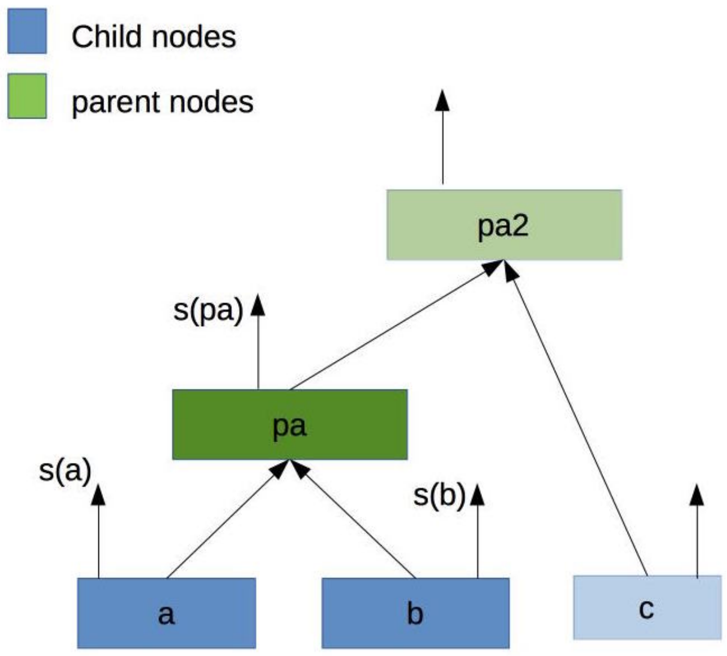


Figure 2-9. Example of a recursive composition structure

$$
\begin{aligned}
p &= f\left(W\begin{bmatrix} a \\ b \end{bmatrix}\right) \\
s &= k(u^T pa)
\end{aligned}
$$

where W is a affine transformation matrix with dimension $2n \times n$, f is a non linearity like *tanh* or *sigmoid*, and k is usually an identity function and u is a column vector of dimension n.

In our approach, we use the score s, generated by this architecture as energy of the composition, which is used to model the probability of the phrase using Gibbs distribution

$$p(pa) = \frac{1}{Z} exp(-s)$$

where Z is a partition function computed by summing over all unnormalized probability of all possible sentences.

2.4.2 Compositional Vector Grammar(CVG)

Most language parsers capture syntactic information but ignore the distributional hypothesis of language. If we could build a model that can capture the syntactic and semantic information, we might be able to do a better job at predicting the underlying parse structure of the sentence. CVG uses this institution to build a compositional model on top of syntactic parsers.

Training is done in a supervised fashion using max-margin training objective by trying to minimize incorrect span predictions similar to Tasker et al. The max margin objective can be formulated as :

$$s(CVG(\theta, x_i, \hat{y}_i)) \geq s(CVG(\theta, x_i, y_i)) + \Delta(x_i, \hat{y}_i)$$

where,

$$\Delta(y_i, \hat{y}_i) = \sum_{d \in N(\hat{y_i})} k1\{subTree(d) \notin Y(x_i)\}$$

x_i be a set of sequence of words,

y_i is the true parse tree,

$\hat{y}_i$ is the candidate tree generated by the RNN.

$Y(x_i)$ be the sequence of ground truth tree for the given sequence,

$N((\hat{y_i}))$ be set of all non terminal nodes in $\hat{y}_i$ and

$subTree(d)$ gives the subtree rooted at non terminal node$d$.

The scoring function for the tree $S(CVG(\theta, x_i, y_i)$ is defined as

$$S(CVG(\theta, x_i, y_i) = \sum_{r \in N(y_i)} \tilde{s}(p_r)$$

where r is a production rule in tree set $N(y_i)$ and $\tilde{s}(r)$ is defined as

$$\tilde{s}(r) = s(p) + log(p(r))$$

where p is phrase or node and r is the production rule that composes node p. Score s is computed in a similar way as mentioned in Section 2.4.1.

Scoring function $s(CVG(\theta, x_i, y_i))$ used by CVG can be seen as a log of joint probability over sentence and compositional tree.

$$\begin{aligned} s(CVG(\theta, x_i, y_i)) &= log(P(x_i, y_i)) \\ &= log(P(x_i|y_i) + log(p(y_i)) \\ &= \sum_p s(p) + \sum_r log(p(r)) \\ &= \sum_{r \in N(y_i)} (s(p) + log(p(r))) \end{aligned}$$

The first term, $P(x_i|y_i)$, is modeled using the composition function we defined in previous section. The second term $p(y_i)$, is obtained using PCFG described in Section 2.3. As the joint probability model breaks the context free assumption, hence DP based marginalization cannot be not used out of the box. To overcome this, CVG generates *k-best* candidate trees, using beam search and use this set to marginalize over all trees.

Our approach is similar to this approach in the sense that both use similar composition function to model conditional probability as CVG as well as use PCFGs to calculate the probability over trees. The differences mainly is in the problems we are trying to solve. We are actually interested in calculating $P(x_i)$, hence we need a different approach towards marginalization. Moreover, instead of generating *k-best tree,* we restore the context free assumption by ensuring single representation for each phrase. Due to this, we were able to use efficient DP based Inside-Outside Algorithm for exact marginalization . Lastly, we use maximum likelihood estimation for training instead of max margin objective.

CHAPTER 3
COMPOSITIONAL LANGUAGE MODELING

3.1 Introduction

Lets start by an considering a sample sentence, *My dog likes eating pizza.* Traditional language model considers language as a Markov linear chain on probability space of words. It would factorize the sentence in the following way

$$\begin{aligned} P(My\ dog\ likes\ eating\ Pizza) &= P(Pizza|My, dog, likes, eating) \\ &\quad P(eating|My, dog, likes) \\ &\quad P(likes|My, dog) \\ &\quad P(dog|My)P(My) \end{aligned}$$

This can be visualized as a finite state machine on the word space as shown in Figure 3-1

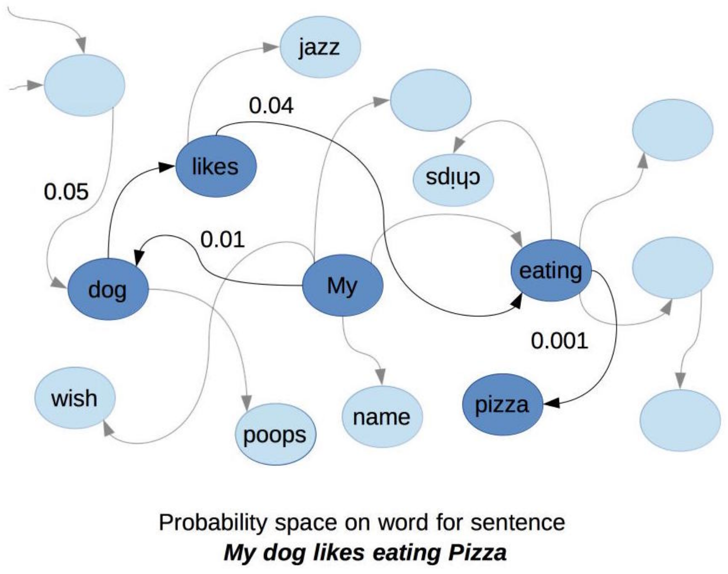

jazz
0.04
likes
chips
0.05
0.01
eating
dog
My
0.001
wish
poops
name
pizza
Probability space on word for sentence
My dog likes eating Pizza

Figure 3-1. Probability space on words for sentence *My dog likes eating pizza.*

If we move our probability space on n-grams or phrases, we will obtain the following factorization

$$\begin{aligned}
&P(My\ dog\ likes\ eating\ Pizza, My, dog, likes, eating, Pizza) = \\
&P(My\ dog\ likes\ eating\ Pizza|My\ dog\ likes\ eating, Pizza)P(Pizza) \\
&P(My\ dog\ likes\ eating|My\ dog\ likes, eating)P(eating) \\
&P(My\ dog\ likes|My\ dog, likes)P(likes) \\
&P(My\ dog|My, dog)P(My)P(dog)
\end{aligned} \tag{3–1}$$

This factorization leads to sequential composition tree as shown in Figure 3-2.

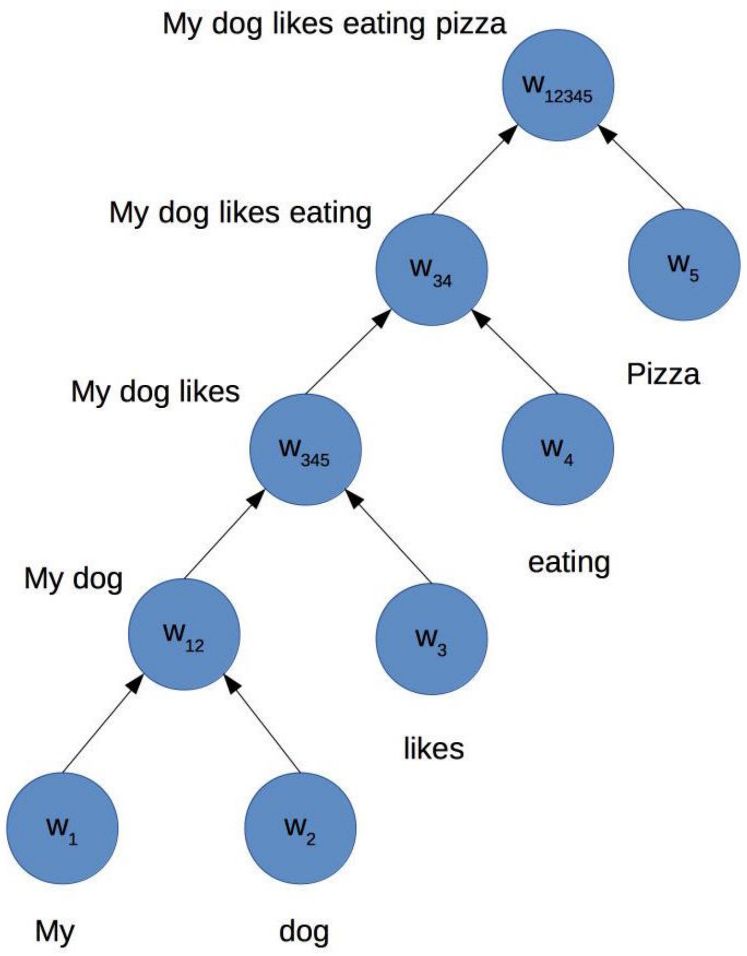


Figure 3-2. Probability space on n-grams for sentence *My dog likes eating pizza.*

Let$t$ be the composition tree shown Figure 3-2, we can re-write 3–18 as a conditional distribution on this tree as

$$\begin{gathered}P(My\ dog\ likes\ eating\ Pizza|t) = \\ P(My\ dog\ likes\ eating\ Pizza|My\ dog\ likes\ eating, Pizza)P(Pizza) \\ P(My\ dog\ likes\ eating|My\ dog\ likes, eating)P(eating) \\ P(My\ dog\ likes|My\ dog, likes)P(likes) \\ P(My\ dog|My, dog)P(My)P(dog) \qquad (3\text{–}2)\end{gathered}$$

This shows the implicit sequential tree assumption inherent in traditional language modeling approaches. This is not surprising as, it is just a reformulation of linear chain assumption in probability space on phrases. To formalize this notion, we now show how recurrent neural network language model can be seen as a compositional model on sequential tree.

3.2 Recurrent Neural Network Language Model as a Compositional Model

Let $W = w_1 w_2 w_3 w_4 w_5$ be the example sentence where w_x are words in vocabulary. RNNLM discussed in Section 2.2.2 models this problem using following recursive equations:

$$x(t) = \left[{w_{t-1}}^T s(t-1)^T\right]^T$$

$$s(t) = f(Ux(t)) \qquad (3\text{–}3)$$

$$y(t) = g(Ws(t)) \qquad (3\text{–}4)$$

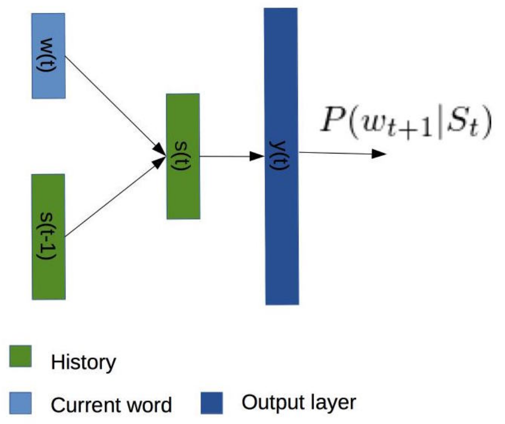
where

$w(t)$ is the word at position t,

$s(t)$ is the current context, and

$y(t) = p(w(t)|s(t-1))$ i.e. probability distribution over next word given the history.

This formulation is akin to linear chain assumption with $P(W)$ factorization of the form:



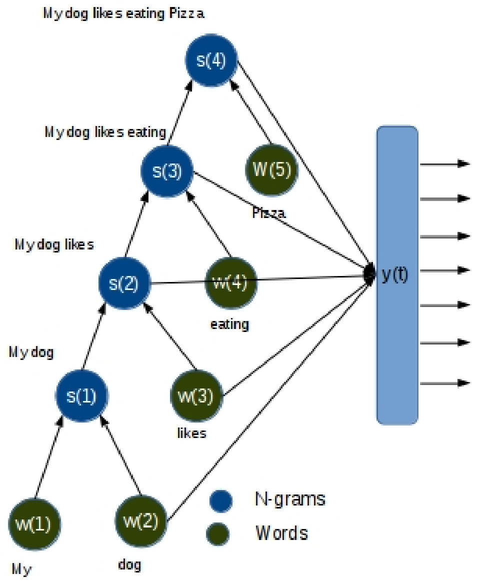


Figure 3-3. Single stage of RNNLM

Figure 3-4. Unrolled RNNLM showing the compositional structure

$$\begin{aligned} p(w_1 w_2 w_3 w_4 w_5) &= p(w_1)p(w_2|w_1)p(w_3|w_1, w_2) \\ &\quad p(w_4|w_1, w_2, w_3)p(w_5|w_1, w_2, w_3, w_4) \end{aligned} \tag{3–5}$$

Figure 3-3 shows the RNNLM architecture and Figure 3-4 shows RNNLM unrolled for sentence W. The unrolled RNNLM, can be seen as a compositional architecture with probability space built over n-grams or phrases. In this context, 3–3 can be seen as a composition operation between history $s(t-1)$ and current word $w(t-1)$ and3–4 can be seen as a scoring equation where$y(t) = p(s(t)|s(t-1), w(t-1))$. This composition formulation is similar to one we discussed in Section 2.4.2 and with underlying composition tree t being a sequential tree leading to following factorization on W

This factorization and underlying composition is similar to one we saw in Figure 3-2.

$$\begin{aligned} p(w_{12345}|t) &= p(w_{12345}|w_{1234}, w_5)p(w_{1234}|w_{123}, w_4) \\ &\quad p(w_{123}|w_{12}, w_3)p(w_{12}|w_1, w_2) \\ &\quad p(w_1)p(w_2)p(w_3)p(w_4)p(w_5) \end{aligned} \tag{3–6}$$

3.3 Unconditioned Compositional Language Model

Having shown the sequential tree composition structure of RNNLM model, we now try to remove this conditional assumption on the composition structure. We do this by marginalizing joint probability of the composition tree and sentence of all such trees. Let $\mathcal{T}(W)$ be a set of all composition trees of the sentence W, we can formulate this marginalization as

$$P(W) = \sum_{t \in \mathcal{T}(W)} P(W, t) = \sum_{t \in \mathcal{T}(W)} p(W|t)p(t) \tag{3–7}$$

Now, we have two problems at hand.

- Enumerate all composition trees
- Build a framework that can assign a probability measure to these trees.

Probabilistic grammars like Context Free Grammars described in Section 2.3 fits perfectly to our use case. They not only constrain the possible number of trees using production rules but also assign probabilities to these derivations. Also, as we have seen in Section 2.3.1 this context free assumption of PCFG would save us from the trouble of explicitly marginalization over all the trees as we can use Dynamic Programming based Inside Algorithm for the same.

One of the important thing we need to mention here is that we reduce our grammar into Chomsky Normal Form to simplify the derivations. Due to this our grammar would now only be composed of the rules of type

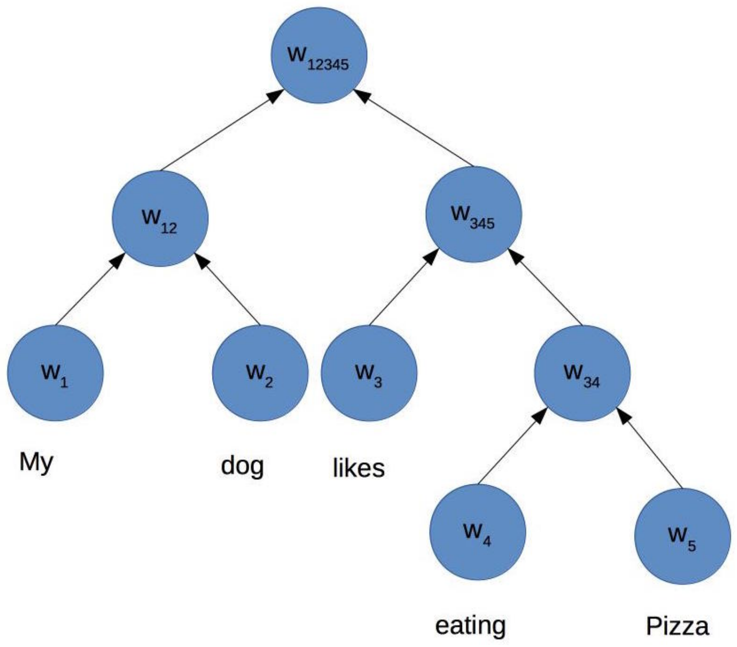


Figure 3-5. Parse tree for sentence W_{12345}

$$A \rightarrow BC$$
$$A \rightarrow a$$

where $A, B, C \in N$ and $a \in T$, i.e. A, B and C are non terminals or POS tags and a is a terminal symbol or a word.

3.4 Tree Representation

Let t be the composition tree for sentence w_{12345} shown in Figure 3-5. The factorization of w_{12345} given t as a composition structure can be re-written as

$$\begin{aligned} p(w_{12345}|t) = p(w_{12345}|w_{12}, w_{345})p(w_{12}|w_1, w_2)p(w_{345}|w_3 w_{45}) \\ p(w_{45}|w_4, w_5)p(w_1)p(w_2)p(w_3)p(w_4)p(w_5) \end{aligned} \tag{3–8}$$

We would now like to build a notion of representation of any arbitrary tree t. The goal of this representation is to build a framework to write factorization like in 3–8 in abstract terms. To achieve this, we represent tree t as a set of composition rules and leaf nodes. The composition rules in our set are of the form $p \rightarrow c_1 c_2$, where p is the parent or composed node and c_1, c_2 are composing children nodes. Using this abstraction, the rule set $R_t(w_{12345})$ for tree in Figure 3-5 will be

$$\begin{aligned} R_t(w_{12345}) &= \{ \\ &\quad w_{12345} \rightarrow w_{12}\ w_{345}, \\ &\quad w_{345} \rightarrow w_3\ w_{45}, \\ &\quad w_{45} \rightarrow w_4\ w_5, \\ &\quad w_{12} \rightarrow w_1\ w_2, \\ &\quad w_1, w_2, w_3, w_4 \} \end{aligned} \tag{3–9}$$

Now we can use this definition of tree in 3–9 to rewrite 3–6 as

$$p(w_{12345}|t) = \prod_{r \in R_t(w_{12345})} p(r)$$

where $p(pa \rightarrow c_1\ c_2) = p(pa|c_1, c_2)$ for binary rules.

In a more general terms, let $R_t(W)$ be the set of rules for any arbitrary composition tree t for a sentence W, we can write conditional probability of sentence W as

$$p(W|t) = \prod_{t \in R_t(W)} p(r) \tag{3–10}$$

3.5 Computing Sentence Probability

The representation we formalized in last section leads to full binary trees and also conforms to CNF form we discussed in Section 3.3. We will now see how this will help us in reducing the joint probability as a binomial distribution similar to 2–3. Recalling from Section

2.3 we can rewrite $p(t)$ using PCFGs as a multinomial distribution on production rules

$$p(t;\theta) = K \prod_{r \in R} \theta_r^{f_r(t)} \tag{3–11}$$

We will drop the multinomial coefficient K from equation 3–11 as we won't need this in our computation.

We can rewrite 3–7 by substituting 3–10 and 3–11 as

$$P(W) = \sum_{t \in \mathcal{T}(W)} p(W|t)p(t) = \sum_{t \in \mathcal{T}_G(W)} \prod_{c \in R_t(W)} p(c) \prod_{r \in R} \theta_r^{f_r(t)} \tag{3–12}$$

Looking at 3–11 we can make two observations. First, not all rules will exist in a tree, hence for some rules $f_r(t) = 0$. Second, some rules, especially binary rules might appear more than once in a tree and for these rules $f_r(t) > 1$. Using this information we define a multi-set $\tilde{R}_t(W)$ for each tree which contains only the rules present in the current production. We can rewrite 3–11 as

$$p(t;\theta) = \prod_{r \in \tilde{R}_t(W)} \theta_r$$

and 3–12 as

$$p(W) = \sum_{t \in \mathcal{T}(W)} \prod_{c \in R_t(W)} p(c) \prod_{r \in \tilde{R}_t(W)} \theta_r \tag{3–13}$$

Now, for unary rule $A \rightarrow w_i$, we add the index i corresponding to word spanned to the rule and for a binary rule of the form $A \rightarrow B\ C$, we add to each of these rules, i, k, j where i, j is the span of non terminal A and k is the index in the span where it factorizes into B and C. This takes care of the productions that are repeated more than once as each one of them will have different spans attached. As each span is attached to a single rule, we can observe that $\tilde{R}_t(W)$ is nothing but a re-write of such composition rules in $R_t(W)$ with the POS tags and the $R_t(W)$ and $\tilde{R}_t(W)$ would have one to one mapping. Using this fact we can rewrite 3–13 as

$$p(W) = \sum_{t \in \mathcal{T}(W)} \prod_{r \in R_t(W)} p(r)\theta_r \tag{3–14}$$

where each rule in $R_t(W)$ contains both the production rule and span it covers in sentence.

Substituting $\zeta_r = p(r)\theta_r$, we rewrite joint probability $p(W, t)$ and 3–14 as

$$\begin{aligned} p(W,t) &= \prod_{r \in R_t(W)} \zeta_r & (3–15) \\ p(W) &= \sum_{t \in \mathcal{T}(W)} \prod_{r \in R_t(W)} \zeta_r & (3–16) \end{aligned}$$

The formulation of 3–15 is similar to 2–5 for computing $P(W)$ but without $P(W|t) = 1$ assumption. In Section 2.3.1 we saw how 2–4 can be efficiently computed using Inside Algorithm, we now do the same for 3–15.

$$\begin{aligned} p(W) &= \sum_{t \in \mathcal{T}_G(W)} p(W|t)p(t) \\ &= \sum_{t \in \mathcal{T}_G(W)} \prod_{r \in R_t(W)} p(r)\theta_r \\ &= \sum_{t \in \mathcal{T}_G(W),} \prod_{r \in R_t(W)} \zeta_r \\ &= \pi(1, n, S) \end{aligned}$$

Now we can compute $\pi(1, n, S)$ recursively adapting equation 2–7 and 2–9 to our case as

$$\pi(i, i, A) = \begin{cases} \zeta_{A, i \to w_i} & A \to w_i \in R \\ 0 & otherwise \end{cases}$$

and

$$\pi(A \to BC, i, k, j) = \zeta_{A \to B\ C\ i,k,j} \pi(i, k, B)\pi(k+1, j, C)$$

$$\pi(i,j,A) = \sum_{A\rightarrow BC\in R}\sum_{i\leq k<j}\pi(A\rightarrow BC,i,k,j)$$

where

$\zeta_{A,i\rightarrow w_i} = p(w_i)\theta_{A\rightarrow w_i}$ and

$\zeta_{A\rightarrow B\ C,i,k,j} = p(w_{i..j}\rightarrow w_{i..k}\ w_{k+1..j})\theta_{A\rightarrow BC}$.

3.6 Training

Having computed $P(W)$, we now focus our attention to training. We formulate training as a maximum likelihood estimation problem on the training set. Let D be the set of training sentences, α be the parameters of the composition,we can write likelihood function as

$$\mathcal{L}(\alpha;D) = \prod_{W_d\in D} p(W_d)$$

Taking negative log both sides

$$-\mathcal{LL}(\alpha;D) = -\sum_{W_d\in D} ln(p(W_d)) \tag{3–17}$$

Substituting 3–7 in 2–9, we get

$$\begin{aligned}-\mathcal{LL}(\alpha;D) &= -\sum_{W_d\in D} ln(\sum_{t\in\mathcal{T}(W)} p(W_d|t;\alpha)p(t))\\ &= -\sum_{W_d\in D} ln(\sum_{t\in\mathcal{T}(W)}\prod_{r\in R_t(W)}\zeta_r(\alpha))\end{aligned}$$

This formulation is very similar to standard EM formulation where composition tree t can be seen as latent variable.

Expectation Maximization formulation:.

3.6.1 E step

In E step we compute $P(t|W,\alpha)$ and $\mathcal{Q}(\alpha;\alpha^{old})$ as follows

$$
\begin{aligned}
p(t|W;\alpha) &= \frac{p(t,W;\alpha)}{p(W;\alpha)} = \frac{p(t,W;\alpha)}{\sum_{t\in\mathcal{T}_G(W)} p(t,W;\alpha)} && (3\text{–}18)\\
\mathcal{Q}(\alpha;\alpha^{old}) &= -\sum_{t\in\mathcal{T}_G(W)} p(t|W;\alpha^{old}) ln(p(t,W;\alpha)) && (3\text{–}19)
\end{aligned}
$$

Substituting $p(t,W)$ from 3–15 in 3–19, we get

$$
\mathcal{Q}(\alpha;\alpha^{old}) = -\sum_{t\in\mathcal{T}_G(W)} \{\sum_{r\in R_t} ln(\zeta_r(\alpha))\} \times p(t|W;\alpha^{old}) \qquad (3\text{–}20)
$$

Looking at $\mathcal{Q}(\alpha;\alpha^{old})$, it still looks intractable due to summation over all trees, but actually it is not. We can make this term tractable by introducing an indicator variable $z_{tr}(W)$ which is 1 if rule r is present in tree t and 0 otherwise i.e.

$$
z_{tr}(W) = \begin{cases} 1 & r \in R_t(W) \\ 0 & otherwise \end{cases}
$$

We rewrite inner summation of 3–20 with indicator $z_{tr}(W)$ as follows:

$$
\begin{aligned}
ln(p(t,W;\alpha)) &= ln(\prod_{r\in R_i(W)} \zeta_r(\alpha)) \\
&= \sum_{r\in R_t(W)} ln(\zeta_r(\alpha)) \\
&= \sum_{r\in R} z_{tr}(W) ln(\zeta_r(\alpha)) && (3\text{–}21)
\end{aligned}
$$

Substituting 3–21 in 3–20, we get

$$
\mathcal{Q}(\alpha;\alpha^{old},W) = -\sum_{t\in\mathcal{T}_G(W)} \{\sum_{r\in R} z_{tr}(W) \times ln(\zeta_r(\alpha))\} \times p(t|W;\alpha^{old}) \qquad (3\text{–}22)
$$

As summation over production rules R is independent of both composition tree and sentence we can move it out and move summation over trees inside leading to 3–23

$$\mathcal{Q}(\alpha;\alpha^{old},W) = -\sum_{r\in R} ln(\zeta_r(\alpha)) \times \{ \sum_{t\in\mathcal{T}_G(W)} z_{tr}(W) \times p(t|W;\alpha^{old})\} \tag{3–23}$$

Expanding $p(t|W;\alpha^{old})$ in 3–23, we get

$$\begin{aligned}\mathcal{Q}(\alpha;\alpha^{old},W) &= \frac{-\sum_{r\in R} ln(\zeta_r(\alpha)) \times \{\sum_{t\in\mathcal{T}_G(W)} z_{tr}(W) \times p(W,t;\alpha^{old})\}}{\sum_{t\in\mathcal{T}_G(W)} p(W,t;\alpha^{old})} \\ &= \frac{-\sum_{r\in R} ln(\zeta_r(\alpha)) \times \{\sum_{t\in\mathcal{T}_G(W)} z_{tr}(W) \times p(W,t;\alpha^{old})\}}{\pi(1,n,S)} \end{aligned} \tag{3–24}$$

We now take a look at inner summation in 3–24 $\sum_{t\in\mathcal{T}_G(W)} z_{tr} \times p(t,W;\alpha^{old})$ and call it $\mu(r)$. We can also rewrite $\mu(r)$ as

$$\begin{aligned}\mu(r) &= \sum_{t\in\mathcal{T}_G(W)} z_{tr} \times p(t,W;\alpha^{old}) \\ &= \sum_{t\in\mathcal{T}_G(W):r\in R_t(W)} p(t,W;\alpha^{old})\end{aligned} \tag{3–25}$$

The quantity $\mu(r)$, the posterior probability of rule r, is nothing but sum of probability of all trees which contain rule r. This is the same quantity that we derived in the Section 2.3.1.3 and can be computed as a product of outside and inside probability of the rule. Now, we need to adapt Outside algorithm for for joint probability case.

3.6.1.1 Outside algorithm

As in the case with Inside Algorithm, we can adapt Outside Algorithm for the joint probability formulation by using ζ_r instead of θ_r and using inside score of joint probability calculated in Section 3.5. The base case remains the same as the original Outside Algorithm i.e. outside score for tree spanning whole sentence and rooted at S is 1 and for other non terminals is 0.

$$\begin{aligned}\beta(S,1,n) &= 1\\ \beta(A,1,n) &= 0\end{aligned}$$

Now, the recursive definition of $\beta(A,i,j)$ is as follows

$$\beta(A,i,j) = \sum_{k=1}^{i-1} \sum_{B\to C\ A,k,i,j\in R} \beta(B\to CA,k,i-1,j) + \tag{3–26}$$

$$\sum_{k=j+1}^{n} \sum_{B\to A\ C,i,j,k\in R} \beta(B\to AC,i,j+1,k) \tag{3–27}$$

where

$\beta(B\to CA,k,i-1,j) = \zeta_{B\to C\ A,k,i,j} \times \pi(C,k,i-1) \times \beta(B,k,j)$ and

$\beta(B\to AC,i,j+1,k) = \zeta_{B\to A\ C,i,j,k} \times \pi(C,j+1,k) \times \beta(B,i,k)$.

3.6.1.2 Calculating $\mu(r)$

Again, we can calculate $\mu(r)$ is similar way as in Section 2.3.3.1 by multiplying the outside probability of rule r with its inside probability

$$\mu(A\to w_i) = \pi(A,i,i)\times\beta(A,i,i) \tag{3–28}$$

$$\mu(A\to B\ C,i,k,j) = \beta(A,i,j)\times\pi(A\to BC,i,k,j) \tag{3–29}$$

Now, substituting the definition of $\mu(r)$ in equation 3–24 we can rewrite 3–24 as

$$\begin{aligned}\mathcal{Q}(\alpha;\alpha^{old},W) &= -\frac{\sum_{r\in R} ln(\zeta_r(\alpha))\mu(r)}{\pi(1,n,S)}\\ &= -\frac{\sum_{r\in R} ln(\zeta_r(\alpha))\mu(r)}{P(W)}\end{aligned}$$

3.6.2 M step

In M step, the objective is to minimize $\mathcal{Q}(\alpha; \alpha^{old})$ to find the α^* such that

$$\begin{aligned} \alpha^\star &= \underset{\alpha}{arg\ min} \mathcal{Q}(\alpha; \alpha^{old}, W) \\ &= \underset{\alpha}{arg\ min} - \frac{\sum_{r \in R} ln(\zeta_r(\alpha))\mu(r)}{P(W)} \end{aligned} \tag{3–30}$$

Differentiating $\mathcal{Q}(\alpha; \alpha^{\text{old}})$ w.r.t. to α, we get

$$\frac{\partial \mathcal{Q}}{\partial \alpha} = -\frac{1}{P(W)} \sum_{r \in R} \left\{ \mu(r; \alpha^{old}) \times \frac{\partial ln(\zeta_r(\alpha))}{\partial \alpha} \right\} \tag{3–31}$$

Substituting the original definition of $\zeta_r(\alpha)$ we get

$$\frac{\partial \mathcal{Q}}{\partial \alpha} = -\frac{1}{P(W)} \sum_{r \in R} \left\{ \mu(r; \alpha^{old}) \times \frac{\partial \left[ln(p(r; \alpha)) + ln(\theta_r) \right]}{\partial \alpha} \right\} \tag{3–32}$$

As θ_r term is independent of optimization parameter α, we can rewrite 3–32 as

$$\frac{\partial \mathcal{Q}}{\partial \alpha} == -\frac{1}{P(W)} \sum_{r \in R} \left\{ \mu(r; \alpha^{old}) \times \frac{\partial ln(p(r; \alpha))}{\partial \alpha} \right\} \tag{3–33}$$

3.7 Modeling compositional probability $p(r)$

Before proceeding further we would need to define $p(r; \alpha)$ as we haven't done so yet. Let's start by taking a closer look at our architecture. The input to our model is an array of integers, each referring to the index of the word w in our vocabulary V. As a first step, we project this word into continuous space using embedding matrix X which is a $n \times V$ matrix to obtain a continuous space vector $X[i]$ corresponding to word w_i. In further discussion we never mention this step again and assume each word or sequence is in this latent space. Now, we turn our focus to modeling $p(r)$, which we model as a Gibbs distribution. Let's recall $p(r) = p(w_{i...j} | w_{i...k}, w_{k+1...j})$. We call $w_{i...j}$ as pa, the composed parent node and $w_{i...k}$ and

$w_{k+1,j}$ as c_1 and c_2, the child nodes. The probability $p(pa|c_1, c_2)$ is modeled as

$$p(pa|c_1, c_2; \alpha) = \frac{1}{Z} exp(-E(pa, c_1, c_2; \alpha)) \tag{3–34}$$

Composition and scoring method is similar to one discussed in Section 2.4.1 and would be done as follows:

$$pa = f\left(W\begin{bmatrix} c_1 \\ c_2 \end{bmatrix}\right) \tag{3–35}$$

where W is a parameter with dimensions $n \times 2n$ and f is a non linearity like *tanh*. The energy E for this composition is calculated as follows:

$$\begin{aligned} E(pa, c_1, c_2) &= g(u^T pa) \\ &= \begin{cases} g(u^T x) & x = Xw \; \& \; w \in V \\ g\left(u^T f\left(W\begin{bmatrix} c_1 \\ c_2 \end{bmatrix}\right)\right) & otherwise \end{cases} \end{aligned} \tag{3–36}$$

where X is embedding matrix for of all word in corpus and V is set of all those words and g is again a non linearity like *sigmoid* or *tanh*. Looking at composition and energy function we can deduce that parameters of the model α would be (W, u, X).

For notational simplicity, we denote $E(pa|c_1, c_2; \alpha)$ as $E(r; \alpha)$ where $r \implies pa \rightarrow c_1 \; c_2$. Now, we can rewrite $\zeta_r(\alpha)$, as

$$\zeta_r(\alpha) = p(r; \alpha)\theta_r = exp(-E(r; \alpha))\theta_r = exp(-E(r; \alpha) + ln(\theta_r)) \tag{3–37}$$

Now, substituting $p(t, W; \alpha)$ from 3–37, in equation 3–30, we get

$$\alpha^\star = \underset{\alpha}{arg\ min} \sum_{r \in R} (E(r; \alpha) - ln\theta_r) \times \mu(r; \alpha^{old}) \tag{3–38}$$

The gradient of $Q(\alpha; \alpha^{old})$ w.r.t.αcalculated in equation 3–33 can be rewritten as

$$\frac{\partial Q}{\partial \alpha} = \sum_{r \in R} \frac{\partial E}{\partial \alpha} \times \mu(r; \alpha^{old}) \tag{3–39}$$

Some important things to note here are that the term Z in 3–34 is intractable as it is sum of all possible sequences of all possible lengths. We will never explicitly calculate Z, hence will never have probability in true sense but a unnormalized probability or score.

3.8 Phrasal Representation

One of the not so apparent issue of our composition model is that it leads to a different composition vector for each factorization. Due to this the context free assumption of PCFG is broken. This is the same issue faced by Socher et al. in CVG[4] parsing that was discussed in Section 2.4.2. This is a big issue as both Inside and Outside algorithm used for marginalization and training depends upon this assumption and need a single representation of phrase. We achieve this by taking the expected value of each continuous representation of the phrase w.r.t. its inside probability of the phrase i.e.

$$X(i,j) = \mathbb{E}_{\pi}[X(i,k,j)]$$

This representation is intuitive too. If composition structures for a phrase leads multiple representation, then the best way to represent that phrase would be an average representation weighted by likelihood of each structure. Now, with context free assumption restored, we would not need to resort to*k-best*list marginalization as done by Socher et al. in [4].

3.9 Calculating $\partial E(r; \alpha)/\partial \alpha$

We now focus our attention to computing the derivative $\frac{\partial E}{\partial \alpha}$ where $\alpha = \{W, u, X\}$. We now compute derivative of composition energy with respect to each of these parameter.

We start by calculating gradients w.r.t. to u.

From 3–36, we know the energy of composed (or leaf) node pa is given by

$$E(r; u, W, X) = g(u^T pa) \tag{3–40}$$

Derivate w.r.t. u can be computed as

$$\frac{\partial E(r;u,W,X)}{\partial u} = \frac{\partial E}{\partial u} = g\prime(u^T pa)pa$$

Now we move to computing gradient w.r.t. W. Differentiating 3–33 w.r.t. W, we get

$$\frac{\partial E(r;u,W,X)}{\partial W} = g\prime(u^T pa)\frac{\partial pa}{\partial W} \tag{3–41}$$

Here, we have reduced the problem of differentiating E w.r.t. to W to differentiating node pa w.r.t. to W. Now pa can either be a leaf node or a non terminal node composed of two children c_1 and c_2. In first case, derivate w.r.t to W would be 0. For the later composition would look like

$$pa = f\left(W\left[c^{12}\right]\right)$$

where $c^{12} = \begin{bmatrix} c_1 \\ c_2 \end{bmatrix}$

Differentiating pa w.r.t. to w_{ij}, we get

$$\begin{aligned}
\frac{\partial pa}{\partial w_{ij}} &= f\prime\left(W\left[c^{12}\right]\right) \circ \frac{\partial W c^{12}(W)}{\partial w_{ij}} \\
&= f\prime\left(W\left[c^{12}\right]\right) \circ \left\{ \begin{bmatrix} 0 \\ . \\ c_j^{12}\ (W) \\ . \\ 0 \end{bmatrix} + \begin{bmatrix} \sum_u w_{1u}\frac{\partial c_u^{12}(W)}{\partial w_{ij}} \\ . \\ \sum_u w_{iu}\frac{\partial c_u^{12}(W)}{\partial w_{ij}} \\ . \\ \sum_u w_{du}\frac{\partial c_u^{12}(W)}{\partial w_{ij}} \end{bmatrix} \right\} \\
&= f\prime\left(W\left[c^{12}\right]\right) \circ \left\{ \mathbf{1}_j \circ c^{12} + W\frac{\partial c^{12}(W)}{\partial w_{ij}} \right\} \\
&= f\prime\left(W\left[c^{12}\right]\right) \left\{ \mathbf{1}_j \circ c^{12} + W \begin{bmatrix} \frac{\partial c_1(W)}{\partial w_{ij}} \\ \frac{\partial c_2(W)}{\partial w_{ij}} \end{bmatrix} \right\}
\end{aligned} \tag{3–42}$$

where c_j^{12} is jth element in column vector c^{12}, $\circ$ refers to Hadamard product and $\mathbf{1}_j$ is a indicator vector with entry at jth index 1.

Looking at equation 3–42, $\frac{\partial c_1}{\partial w_{ij}}$ and $\frac{\partial c_2}{\partial w_{ij}}$ are nothing but derivatives of children node c_1 and c_2 w.r.t. to w_{ij}. We can build up the expression in 3–42 recursively using a dynamic programming based algorithm very similar to Inside algorithm as follows:

Base Case is $pa \in T$ i.e. pa is a leaf node

$$\frac{\partial pa}{\partial W} = 0$$

Recursively, $\frac{\partial pa}{\partial w_{ij}}$ can be built using derivative of composing children $\frac{\partial c_1(W)}{\partial W}$ and $\frac{\partial c_2(W)}{\partial W}$ using the definition in 3–42

$$\frac{\partial pa}{\partial W} = f\prime\left(W\left[c^{12}\right]\right)\left\{\mathbf{1}_j \circ c^{12} + W\begin{bmatrix} \frac{\partial c_1(W)}{\partial W} \\ \frac{\partial c_2(W)}{\partial W} \end{bmatrix}\right\}$$

Summarizing the recursion

$$\frac{\partial pa}{\partial W} = \begin{cases} 0 & pa \in T \\ f\prime\left(W\left[c^{12}\right]\right)\left\{\mathbf{1}_j \circ c^{12} + W\begin{bmatrix} \frac{\partial c_1(W)}{\partial W} \\ \frac{\partial c_2(W)}{\partial W} \end{bmatrix}\right\} & pa \rightarrow c_1c_2 \in R \end{cases} \tag{3–43}$$

where T is a set of terminal nodes or words and R is a set of composition rules for sentence.

The derivative $\frac{\partial E}{\partial W}$ can be computed by substituting $\frac{\partial pa}{\partial W}$ computed in 3–43 in 3–41.

Finally, we look at computing gradient w.r.t. to X. Differentiating E w.r.t .X we get

$$\frac{\partial E}{\partial X} = g\prime(u^T pa)u^T\frac{\partial pa}{\partial X} \tag{3–44}$$

$\partial pa/\partial X$ can be computed in a similar manner as $\partial pa/\partial W$. Let's start with the case where c is a leaf node. Let x be a row vector of dimension $1 \times n$ in X. The base case of recursion will be

$$\frac{\partial c}{\partial x} = \begin{cases} \mathbb{I}_{n\times n} & x = c \\ \\ 0 & otherwise \end{cases}$$

where $\mathbb{I}$ is an identity matrix.

Now, we consider more general case of a non terminal node pa composed by c_1 and c_2

$$\begin{aligned} \frac{\partial}{\partial x} pa &= \frac{\partial}{\partial c} f\left(W \begin{bmatrix} c_1 \\ \\ c_2 \end{bmatrix} \right) \\ &= f\prime\left(W \begin{bmatrix} c_1 \\ \\ c_2 \end{bmatrix} \right) \circ \frac{\partial}{\partial x} W \begin{bmatrix} c_1 \\ \\ c_2 \end{bmatrix} \\ &= f\prime\left(W \begin{bmatrix} c_1 \\ \\ c_2 \end{bmatrix} \right) \circ W \begin{bmatrix} \frac{\partial c_1}{\partial x} \\ \\ \frac{\partial c_2}{\partial x} \end{bmatrix} \end{aligned}$$

We can summarize the recursion in similar fashion to 3–43

$$\frac{\partial pa}{\partial x} = \begin{cases} \mathbb{I}_{n\times n} & x \in T \& pa = x \\ \\ 0 & x \in T \& pa \neq x \\ \\ f\prime\left(W \left[c^{12} \right] \right) \circ W \begin{bmatrix} \frac{\partial c_1}{\partial x} \\ \\ \frac{\partial c_2}{\partial x} \end{bmatrix} & x \notin T \end{cases} \tag{3–45}$$

where T is a set of terminal nodes or words.

Similarly we can substitute 3–45 in 3–44 to obtain $\frac{\partial E}{\partial X}$.

3.10 Summary

We here summarize Chapter 3 by stating all the points talked in Section 3-1 to 3-9

- Traditional language models use linear n-order Markov assumption which translates into sequential tree when we build our probability space on phrases.
- We believe this to be strong assumption and attempt to address it by building a joint probability distribution over composition tree and sentence and marginalizing over all such trees.

- Probabilistic context free grammar is used to build the composition tree and to assign the probability to such trees
- The joint distribution is factorized in probability of tree and conditional probability of sentence give the composition tree
- The conditional distribution over sentence is modeled as Gibbs distribution with energy function being sum of all compositions and nodes.
- The context free assumption broken by the joint model is restored by using a single phrasal representation which is expected representation of phrases over probability distribution of all compositions i.e. inside probabilities.
- The marginalization is done using efficient dynamic programming based Inside Algorithm.
- Training is formulated as a maximum likelihood estimation problem solved using EM algorithm where E step is again calculated using Inside-Outside Algorithm.
- The derivatives can be efficiently calculated using dynamic programming based approach.

CHAPTER 4
EVALUATION AND RESULTS

There are two standard evaluation metrics for language models: perplexity or word error rate(WER). The simpler of these measures is word error rate which is simply the percentage of erroneously recognized words E (deletions, insertions, substitutions) to total number of words N, in a speech recognition task i.e.

$$WER = \frac{E}{N} \times 100\%$$

In many ways, this is a better evaluation metric as any improvement on language modeling metric is meaningful if it translates in to improvements in the tasks like Automatic Speech Recognition(ASR) or Machine Translation. The problem with WER, is that it needs a complete ASR pipeline to evaluate. It is noteworthy to mention that, all evaluation datasets are behind pay-wall, hence not readily available for benchmarking. Due to lack of both the existing pipeline and access to these datasets we were not able to report WER for our model.

4.1 Perplexity

The second metrics is Perplexity. Let p be the original distribution of the test sequence $W = w_1 w_2 .. w_n$. Let m be the probability distribution built on the language model being evaluated. The Perplexity(per word) of the test sequence, W, can be defined as:

$$PPL(W) = \sqrt[N]{\frac{1}{m(w_1 w_2 ... w_n)}} = 2^{-\frac{1}{N} lg(m(w_1 w_2 ... w_n))} \tag{4–1}$$

Equation 4–1 can also be seen as exponentiated cross entropy with cross-entropy $H(p, m)$, being approximated as:

$$H(p, m) = \frac{1}{n} lg(m(w_1 w_2 ... w_n)) \tag{4–2}$$

Herein, we discuss how we approximate cross entropy to 4–2. While doing so, we borrow notation and methodology from [22].

Entropy H, is a measure of uncertainty or disorder of a random variable. For a random variable X, with probability distribution p, the entropy can is defined as

$$H(X) = -\sum_{x \in X} p(x) lg(p(x))$$

In general terms perplexity PPL is defined as exponentiated entropy i.e.

$$PPL(X) = 2^{H(X)} = 2^{-\sum_{x \in X} p(x) lg(p(x))}$$

Now, for a sequence of random variables of length l, entropy $H(x_1 x_2 ... x_l)$ can be defined as

$$H(x_1 x_2 .... x_l) = -\sum_{x_1 x_2 .. x_l \in X_1^l} p(x_1 x_2 .. x_l) lg(p(x_1 x_2 ... x_l))$$

where X_1^l is a set of all sequences of length l.

In case of language, we can formulate entropy in a similar way. So, for all sentence $w_1 w_2 .... w_l$ of length l, the entropy would be

$$H(w_1 w_2 .... w_l) = -\sum_{w_1 w_2 .. w_l \in W_1^l} p(w_1 w_2 .. w_l) lg(p(w_1 w_2 ... w_l))$$

where W_1^l is set of all possible sentences of length l.

The entropy rate or entropy per word, then can be defined as

$$\frac{1}{l} H(w_1 w_2 .... w_l) = -\frac{1}{l} \sum_{w_1 w_2 .. w_l \in L} p(w_1 w_2 .. w_l) lg(p(w_1 w_2 ... w_l))$$

Now, considering language as a ergodic source, we can approximate the summation over all possible length $0 \leq l < \infty$ with a single sequence of infinite length, hence

$$\begin{aligned} H(L) &= \lim_{l \to \infty} \frac{1}{l} H(w_1 w_2 .... w_l) \qquad (4\text{–}3) \\ &= \lim_{l \to \infty} -\frac{1}{l} \sum_{w_1 w_2 .. w_l \in L} p(w_1 w_2 .. w_l) lg(p(w_1 w_2 ... w_l)) \end{aligned}$$

Assuming language to be ergodic and stationarity, the Shannon-McMillan-Breiman Theorem[23] states that 4–3 can be approximated as a single sequence that is long enough, hence;

$$\begin{aligned} H(L) &= \lim_{l\to\infty} -\frac{1}{l} \sum_{w_1 w_2 .. w_l \in L} p(w_1 w_2 .. w_l) lg(p(w_1 w_2 ... w_l)) \\ &= \lim_{l\to\infty} -\frac{1}{l} lg(p(w_1 w_2 ... w_l)) \\ &= -\frac{1}{l} lg(p(w_1 w_2 ... w_l)) \end{aligned}$$

Now, lets look at the cross entropy term. The cross entropy of language model m on p would be defined as:

$$H(p, m) = \lim_{l\to\infty} -\frac{1}{l} \sum_{w_1 w_2 .. w_l \in L} p(w_1 w_2 .. w_l) lg(m(w_1 w_2 ... w_l))$$

Again, applying the Shannon-McMillan-Breiman Theorem, we can approximate this to 4–2.

Now we focus our attention to why should this work as an evaluation metric. From definition of entropy and cross entropy we know,

$$H(p) \leq H(p, m)$$

i.e. cross entropy is a upper bound on entropy. Hence the model that maximizes the lower limit of the upper bound models the language best.

Before we move to next section we need to point out the following:

- Traditionally PPL is calculated at test document level where N is the size of vocabulary in test set.
- The perplexity of dataset is modeled as n-grams, hence 4–2 can be reduced to

$$H(p, m) = \frac{1}{N} \sum_{i=1}^{N} lg(m(w_i | h_i))$$

where h_i is history and N is the vocabulary size.

This definition of perplexity is unsuitable for our case because, computing exact probability for phrase based models is impossible due to intractability of partition function. In the next section, we propose a new metric which eliminates the need to explicitly partition function calculation leading to a new in-vivo evaluation technique for comparing language models.

4.2 Contrastive Entropy

Let W be a sentence in a test set. We pass this test sentence through a noise channel, that distorts the sentence leading to $\hat{W}$. Now, we define Contrastive Entropy(not entropy rate) of this sentence as:

$$\begin{aligned}
\hat{H}_C(W) &= -lg\left(\frac{p(\hat{W})}{p(W)}\right) \\
&= -lg\left(\frac{\frac{\tilde{p}(\hat{W})}{Z}}{\frac{\tilde{p}(W)}{Z}}\right) \\
&= -lg\left(\frac{\tilde{p}(\hat{W})}{\tilde{p}(W)}\right)
\end{aligned}$$

Here, Contrastive Entropy Rate or Contrastive Entropy (per word) for the whole test set D with vocabulary size N can be defined as:

$$\begin{aligned}
H_C(D) &= \frac{1}{N}\sum_{d\in D}\hat{H_C}(W_D) \\
&= -\frac{1}{N}\sum_{d\in D} lg\left(\frac{\tilde{p}(\hat{W}_d)}{\tilde{p}(W_d)}\right) \\
&= H(\hat{D}) - H(D)
\end{aligned}$$

Now, using the definition of Contrastive Entropy Rate, we calculate Contrastive Perplexity as:

$$\begin{aligned} PPL_C(W) &= 2^{-H_c(D)} \\ &= \sqrt[N]{\frac{\prod_{d\in D}\tilde{p}(W_d)}{\prod_{d\in D}\tilde{p}(\hat{W}_d)}} \\ &= \sqrt[N]{\frac{\tilde{p}(D)}{\tilde{p}(\hat{D})}} \end{aligned}$$

The intuition behind our evaluation technique is that the distorter sentence $\hat{W}$, should be seen as a out of domain text and that a better probability model should be able to distinguish between an in-domain sentence from the language versus a malformed sentence that is less likely to be generated by the language. This formulation is very similar to Contrastive Estimation by [24], the difference being that we use this intuition for testing whereas they use the assumption to do max margin training on language.

Now we look at distortion generation mechanism. We allow only two type of distortions: substitution i.e. replacing a word from the vocabulary with a random word from our vocabulary and transpositions i.e swapping two words from the sentence at random. We model this distortion in following way: For each word in a sentence there are three possible outcomes: no distortion with probability $x_{\mathcal{N}}$, substitution with probability x_S and transposition with probability x_T such that $x_{\mathcal{N}} + x_S + x_T = 1$. So, distortion probability would be a multinomial distribution;

$$p(D) = \frac{\Gamma(n+1)}{\prod_{i\in\{S,T,\mathcal{N}\}}\Gamma(\alpha_i+1)} p(x_S)^{\alpha_S} p(x_T)^{\alpha_T} p(x_{\mathcal{N}})^{\alpha_{\mathcal{N}}}$$

where D is distortion process and $\alpha_S + \alpha_T + \alpha_{\mathcal{N}} = n$, where n is the length of the sentence.

4.3 Results

We will start this section by discussing the datasets used by us to conduct the evaluation experiments. We use two datasets for evaluation purposes. Standard Pen-TreeBank section of WSJ corpus for proposed evaluating metric with respect to Perplexity and example data set provided in [25](RNNLM dataset). Pen TreeBank corpus is one of the most widely used dataset in statistical modeling community to report perplexity results. We use Pen TreeBank dataset with following split and preprocessing: Sections 0-20 were used as training data, sections 21-22 for validation and 23-24 for testing. The training, validation and testing token sizes are 930k, 74k and 82k respectively. Vocabulary is limited to 10k words with all words outside this set mapped to a special token $< unk >$.The example data set by [25] on the other hand in smaller in size with 10000, 1000, 1000 sentences for training, validation and testing respectively. Training file contains 81350 tokens and with vocabulary of 3720 words. Test set contains 7498 tokens with 216 Out of Vocabulary words. We use Pen-TreeBank corpus for evaluation metric benchmarking and example dataset for comparing our model to existing approaches.

We start by looking at the looking at the distortion our noisy channel produces for Pen Tree Bank dataset. Table 4-1 shows some example sentences for this dataset and corresponding output produced by the noisy channel. As we can see at 20% distortion the sentence is still coherent and meaning is still being conveyed. At 40% distortion, it is difficult even for human beings to discern what original sentence meant to say. This observation clearly indicates that a better language model should have considerably high contrastive perplexity for 40 % distortion as compared to 20%.

Now, we look at the contrastive perplexities of various well known models at different distortion rates. The objective here is to verify the following hypothesis about contrastive perplexity. For a good language model contrastive perplexity should rise faster with the distortion level. This is akin to saying that a good language model should do a lot better job at differentiating the language generated by the model and distorted language as the distortion level increases. At the same time contrastive perplexity should be somehow correlated to perplexity across the models. This means that for same distortion level, range of models should be ranked similarly on two metrics, perplexity and contrastive perplexity. Table 4-2 shows the results for our experiments. The results were generated using open source language modeling

Table 4-1. Example sentence with 20% and 40% distortion

Original Sentence	Sentence with 20% distortion	Sentence with 40% distortion
no it was n't black monday	no it deeply black n't monday/	no it generating proceeds black monday
at the end of the day N million shares were traded	nights the meantime of the day N million shares fourth traded	centrust fundamentals away too encourage to secretary government for stop them now
things have gone too far for the government to stop them now	things have gone too far charles goldberg government to stop them openly	concentrate did a n't get even chance to do we slight the wanted to bear
but stocks kept falling they never considered themselves to be anything else	but ride kept falling they never considered themselves to anything be else	falling stocks kept but promises never be themselves considered to anything else
businesses were borrowing at interest rates higher than their own earnings	businesses were borrowing their interest rates higher advise at own investigated	rates were borrowing intense interest businesses own than their equivalents ibm

SRILM toolkit[26] for n-gram models and RNNLM toolkit[25] for RNN based models. The results shown in Table 4-2 were averaged for 10 runs.

Figure 4-1 shows the results discussed in Table 4-2. First thing to observe is the perplexity is inversely correlated to contrastive perplexity. This is in line with what we expect. A better language model would lead to a lower perplexity score as it would to a better job of lowering the cross entropy of model and original distribution but contrastive perplexity should increase as it should be able to do a better job at differentiating between original and distorted language. Another interesting observation here is about increase in contrastive perplexity with distortion across models. Contrastive perplexity increase rapidly for models with lower perplexity like RNN-100, RNN-200 and 5-gram Max Entropy. This is in line with the hypothesis that state of the art models should do a lot better job at discriminating between good and bad language as compared to the bad models for example 5-grams. We can see here that increase in contrastive perplexity is minimal for it.

Table 4-2. Comparing n-gram models and RNNLM model perplexity for different level of distortion levels.

Model	Original	10% distortion	20% distortion	30% distortion	40% distortion	50% distortion
3 gram with no interpolation	1009.90	1.18	1.41	1.68	1.96	2.34
3 gram Good Turing	166.57	2.185	4.22	7.29	11.37	16.91
3 gram Kneser Ney	148.28	2.183	4.15	6.91	10.48	14.93
3 gram Max Entropy	166.57	2.185	4.22	7.29	11.37	16.91
5 gram with no interpolation	1177.85	1.112	1.28	1.49	1.711	2.02
5 gram Good Turing	169.33	2.17	4.19	7.20	11.25	16.70
5 gram Kneser Ney	141.46	2.22	4.254	7.08	10.72	15.26
5 gram Max Entropy	169.33	2.177	4.19	7.20	11.25	16.70
RNN-100	148.78	2.57	5.74	10.87	18.38	28.84
RNN-200	141.31	2.649	6.02	11.52	19.91	31.21

Figure 4-2 and Figure 4-3 shows this behavior clearly. Figure 4-1 and Figure 4-3 show the rate of increase of perplexity and contrastive perplexity with distortion across various models. We can clearly see in Figure 4-3 that contrastive perplexity rises considerably faster for better models. Another interesting thing to note here are two correlations between perplexity and contrastive perplexity across models and distortions. By looking at Figure 4-2 and 4-3 we can see a strong correlation between the two quantities across models. This observation is confirmed by Figure 4-4 where we plot correlation between contrastive perplexity and perplexity across different models.

Now, let's consider correlation across distortion levels. Figure 4-5 plots the correlation between perplexity and contrastive perplexity at various distortion levels. Firstly, as expected the correlation is negative which indicates the inverse relation between the two quantities across models. Second, and more interesting observation is the the increasing slope of correlation across the distortion levels. This can be explained by the same rationale that at

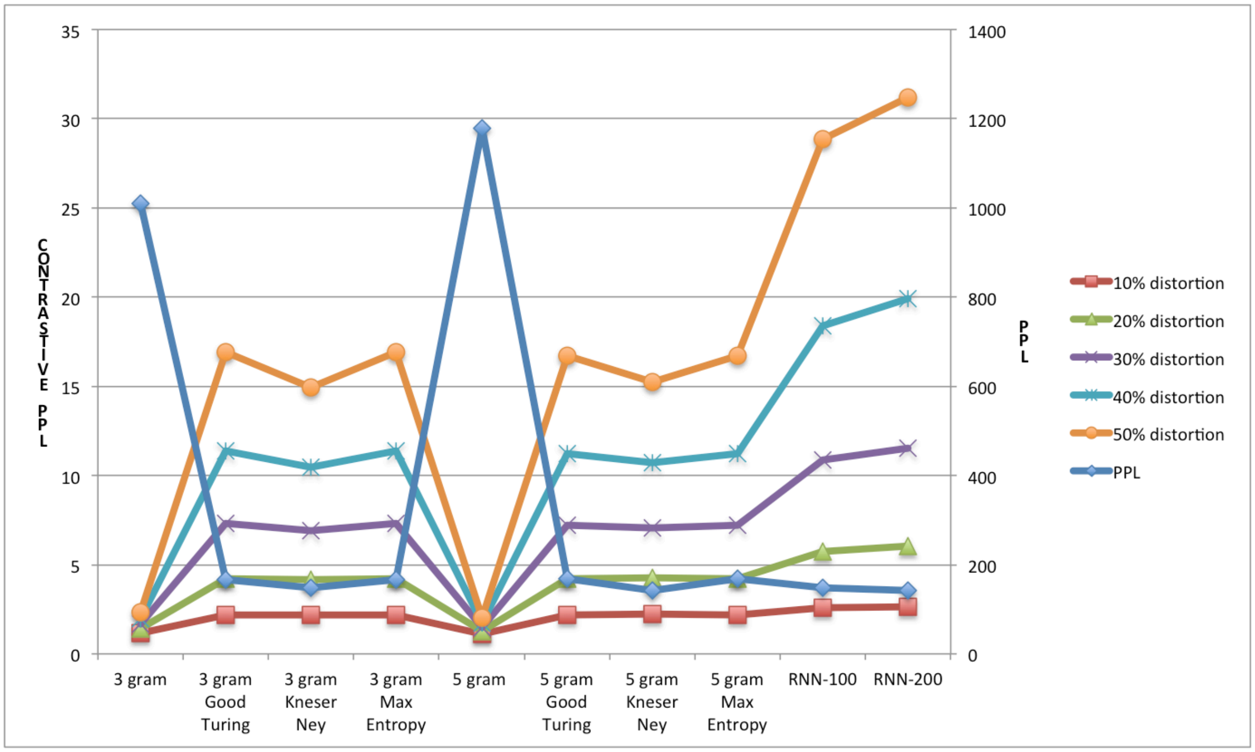


Figure 4-1. Contrastive Perplexity and Perplexity for all models, all distortion level

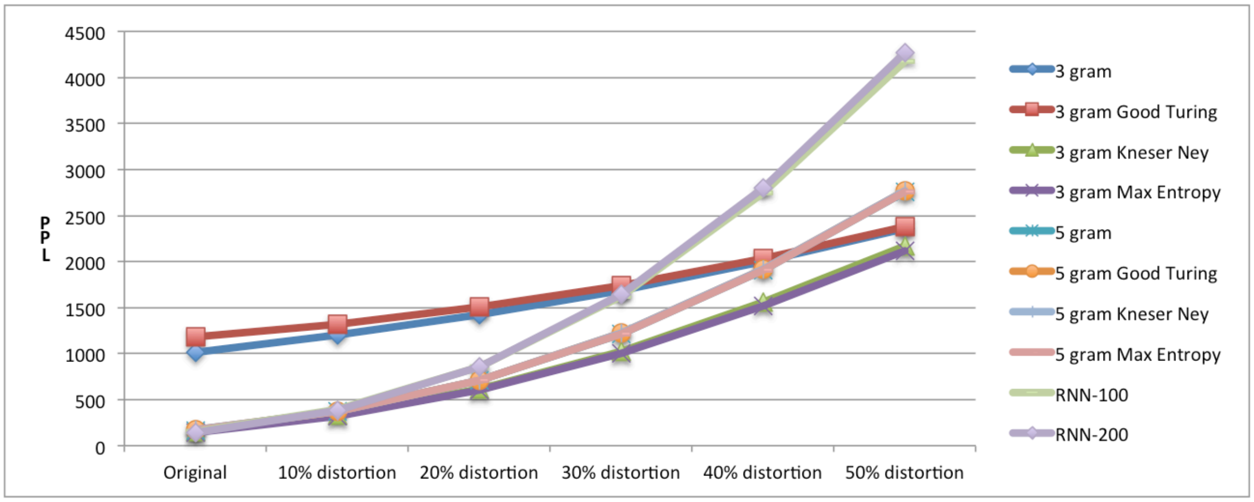


Figure 4-2. Perplexity Vs Distortion

higher distortion levels contrastive perplexity rises considerably faster as compared to decrease of perplexity, hence the slope. This indicates in many way contrastive perplexity might be a better metric to evaluation language models than perplexity alone.

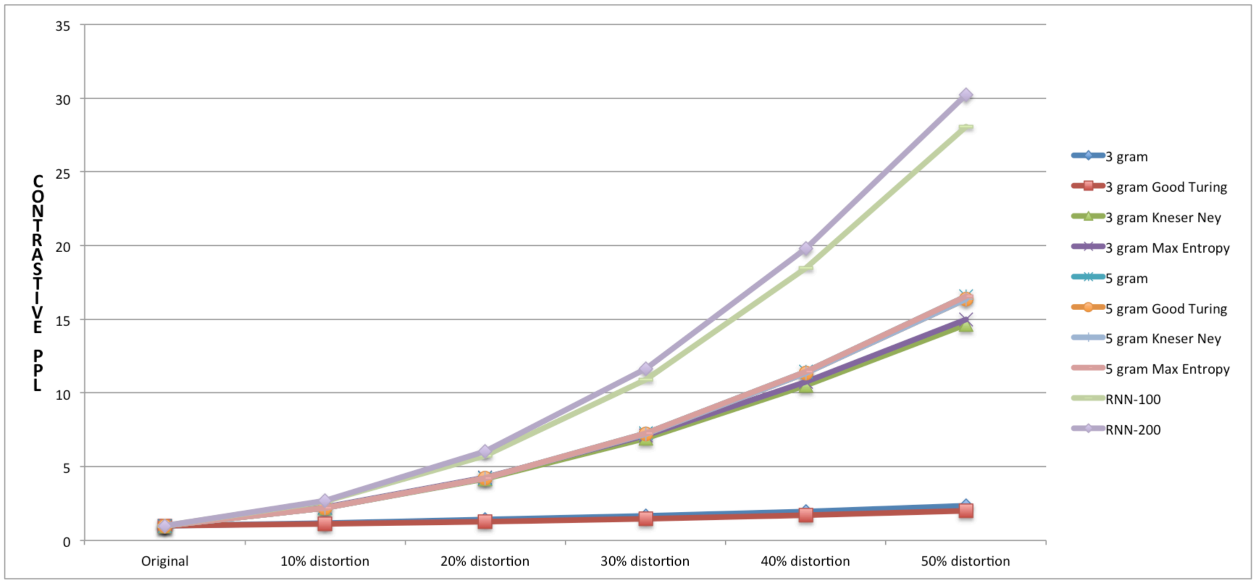


Figure 4-3. Contrastive Perplexity vs percentage distortion

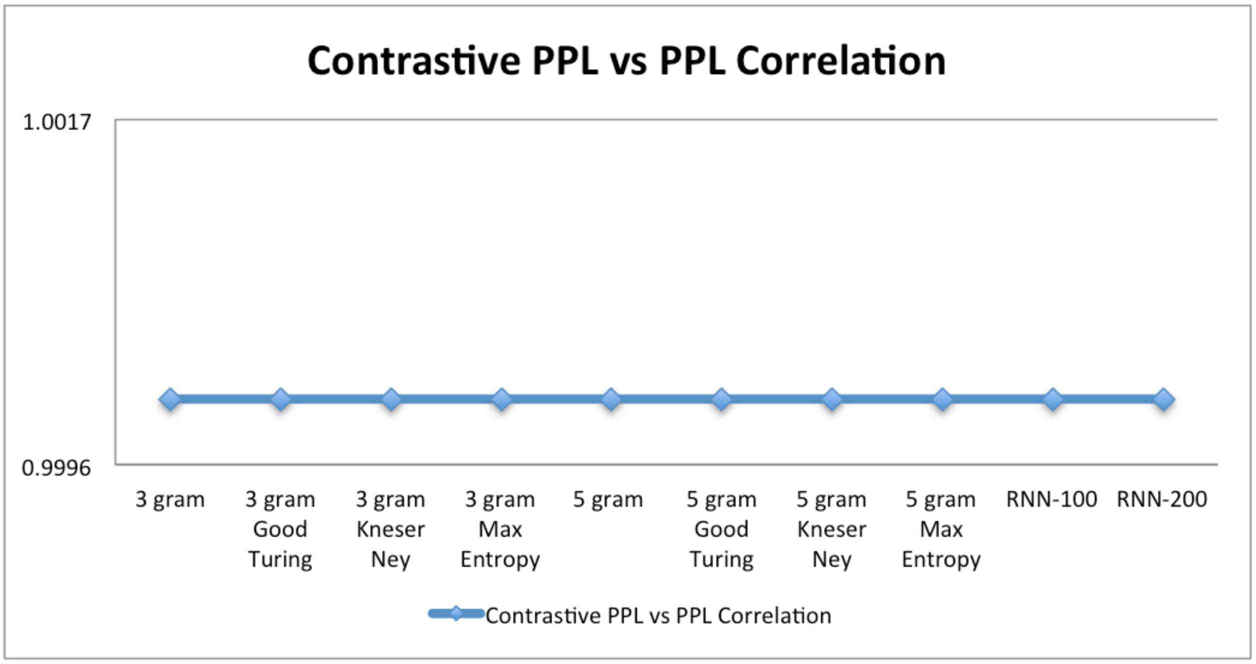


Figure 4-4. Correlation between Contrastive Perplexity vs Perplexity over models

Now, we turn our focus towards the evaluating Compositional Language model on this newly defined metric and comparing it with the models discussed above. As noted in the beginning of this section we will use RNNLM dataset for this comparison. Table 4-3

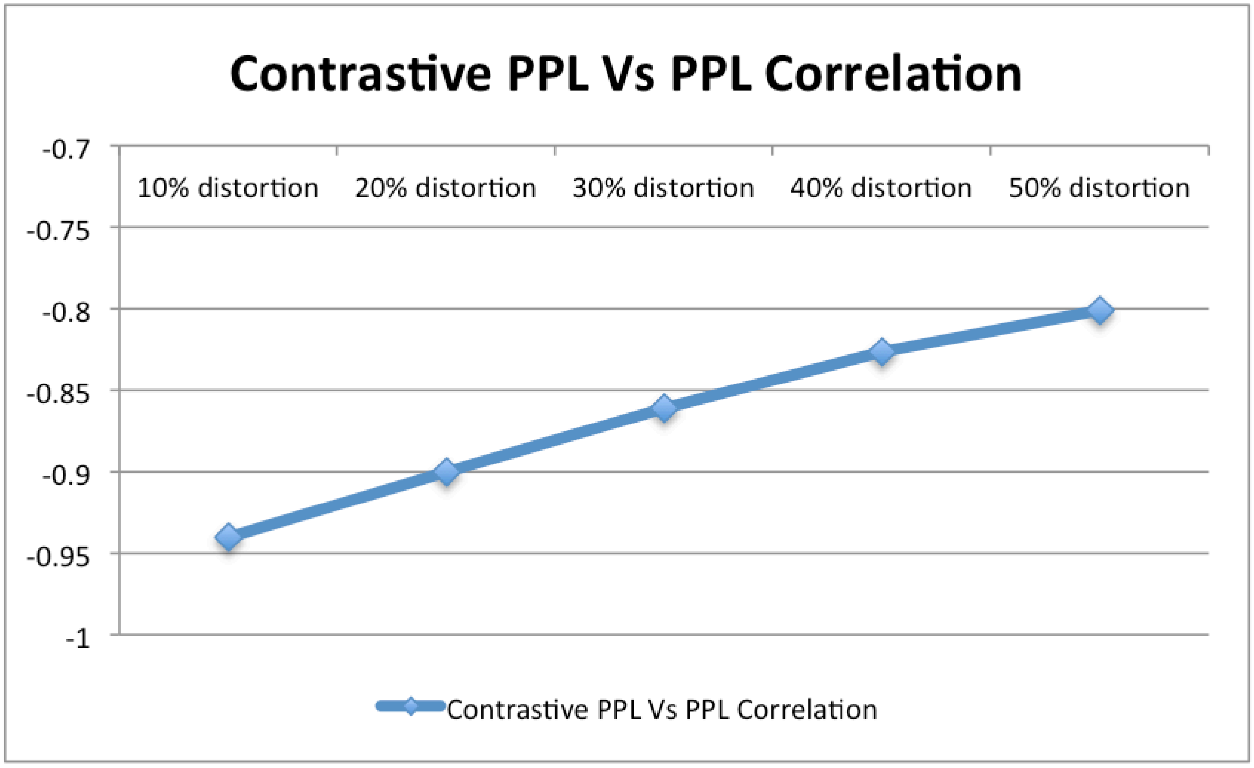


Figure 4-5. Correlation between Contrastive Perplexity and Perplexity over percentage distortion.

compares Compositional Language Model for dimension 30 and 50 with best results discussed above.Figure 4-6 shows the plot of these results.

As we can see from the Figure 4.6, our model consistently outperform all the other model on contrastive perplexity. Another thing to note is that gains RNN based model had on Pen-TreeBank dataset are not so pronounced here. This we think is due to underlying structure of data and out of word vocabulary handling. Though this shows the strength of Compositional Language Model over the traditional approaches, we firmly believe that Compositional LM should be evaluated on more standard dataset like Pen TreeBank. We were not able to do this in this work due to resource constrain.

We conclude these results by saying that though we have initial indication of our model being superior than the traditional approaches, we must take this with a pinch of salt. Firstly, the comparison data set is not a standard one, we need to use standard data set like PenTreeBank for comparison. Secondly, the true evaluation of any language model can

Table 4-3. Comparison of current state of the art language model techniques to Compositional Language Model. CompositionalLM-30 and CompositionalLM-50 r refers to Compositional Language Model with embedding space dimension 30 and 50.

Model	10% distortion	20% distortion	30% distortion	40% distortion	50% distortion
3gram Kneser Ney	1.6512	2.5713	3.6938	4.9501	6.5471
3 gram Good Turing	1.6463	2.5714	3.7581	5.1306	6.9668
5 gram Kneser Ney	1.6491	2.5650	3.6745	4.9241	6.5024
5 gram Good Turing	1.6253	2.5162	3.6554	4.9778	6.7481
RNN-100	1.6430	2.5601	3.6046	4.7551	6.0702
RNN-200	1.6841	2.6798	3.8256	5.0824	6.5905
CompositionLM-30	2.2549	4.5536	7.3683	10.2253	15.186742
CompositionLM-50	2.25052	4.87205	7.8000	12.06372	18.8168

only be done as a part of tasks like Speech Recognition and Machine Translation. We must evaluate these models on those standard tasks and report WER for the same.

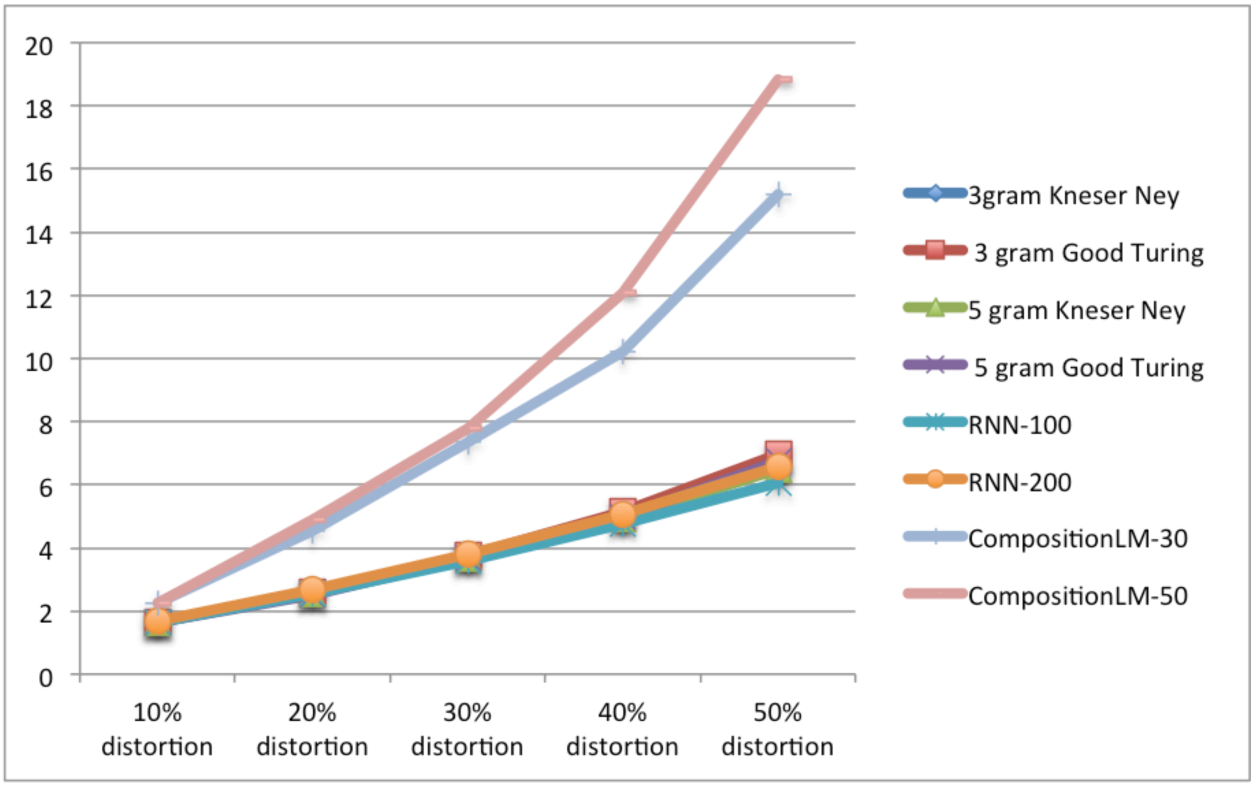


Figure 4-6. Comparison of current state of the art language model techniques to Compositional Language Model. CompositionalLM-30 and CompositionalLM-50 r refers to Compositional Language Model with embedding space dimension 30 and 50.

CHAPTER 5
CONCLUSION

In this thesis work, we tried to challenge the linear chain assumption of current language models. This was accomplished by moving probability space to phrases and showing how sequential tree assumption imposed a finite state view of language. We then removed these assumptions and proposed a neat framework to marginalize and train over all composition trees. We believe this work will encourage researchers in this field to look beyond the n-gram models and use compositional structures for language modeling. We also proposed a new intrinsic metric for comparing phrase based language models which cannot be benchmarked using Perplexity. This hopefully will lead growth of interest in phrase based models as one doesn't need costly datasets and pipelines anymore to evaluate these models. Though we have reported some initial comparative results for both the metric and the model, we believe a lot more still needs to be done. We need to benchmark our new metric against Perplexity on multiple data sets to eliminate the impact of underlying data distribution on correlation. In future, we must also study the correlation of Contrastive Perplexity with WER to see how well it benchmarks the performance on actual tasks. Though we have evaluated our model on an example dataset and shown good results there is a need to do comparative study on more standard dataset and extrinsic metrics like WER.

Apart from traditional use case of language models, language representation provided by this model can be used for building simple classification models for tasks like Sentiment Analysis, Word Sense Disambiguation, Name Entity Recognition etc. The assumption here would be that a lot of heavy lifting is done by the representation and a simple classifier should be able to give good performance on the evaluation benchmarks. Besides, another interesting area of research for these kind of models is in understanding of the representation itself. Due to the fact that this model uses the underlying composition structure for word and phrase embedding, we believe there would be interesting regularities in latent space that must be studied and this model could be extended to achieve better language representation. We believe this thesis work would help NLP community to challenge the implicit assumption in current state of the art models and see language as a composition instead of bag of words or a linear chain.

REFERENCES

[1] H. J. Levesque, “On our best behaviour,” *Artificial Intelligence*, vol. 212, pp. 27–35, 2014.

[2] N. Chomsky, “Three models for the description of language,” *Information Theory, IRE Transactions on*, vol. 2, no. 3, pp. 113–124, 1956.

[3] N. Chomsky and G. Miller, *Introduction to the formal analysis of natural languages*. Wiley, 1963. [Online]. Available: https://books.google.com/books?id=BIHNMgEACAAJ

[4] R. Socher, J. Bauer, C. D. Manning, and A. Y. Ng, “Parsing with compositional vector grammars,” in *In Proceedings of the ACL conference*. Citeseer, 2013.

[5] S. F. Chen and J. Goodman, “An empirical study of smoothing techniques for language modeling,” in *Proceedings of the 34th annual meeting on Association for Computational Linguistics*. Association for Computational Linguistics, 1996, pp. 310–318.

[6] P. F. Brown, P. V. Desouza, R. L. Mercer, V. J. D. Pietra, and J. C. Lai, “Class-based n-gram models of natural language,” *Computational linguistics*, vol. 18, no. 4, pp. 467–479, 1992.

[7] C. Chelba and F. Jelinek, “Structured language modeling,” *Computer Speech & Language*, vol. 14, no. 4, pp. 283–332, 2000.

[8] E. Charniak, “Immediate-head parsing for language models,” in *Proceedings of the 39th Annual Meeting on Association for Computational Linguistics*. Association for Computational Linguistics, 2001, pp. 124–131.

[9] J. T. Goodman, “A bit of progress in language modeling,” *Computer Speech & Language*, vol. 15, no. 4, pp. 403–434, 2001.

[10] C. Chelba, D. Engle, F. Jelinek, V. Jimenez, S. Khudanpur, L. Mangu, H. Printz, E. Ristad, R. Rosenfeld, A. Stolcke *et al.*, “Structure and performance of a dependency language model.” in *EUROSPEECH*. Citeseer, 1997.

[11] Y. Bengio, H. Schwenk, J.-S. Senécal, F. Morin, and J.-L. Gauvain, “Neural probabilistic language models,” in *Innovations in Machine Learning*. Springer, 2006, pp. 137–186.

[12] T. Mikolov, M. Karafiát, L. Burget, J. Cernockỳ, and S. Khudanpur, “Recurrent neural network based language model.” in *INTERSPEECH*, 2010, pp. 1045–1048.

[13] A. Mnih and G. E. Hinton, “A scalable hierarchical distributed language model,” in *Advances in neural information processing systems*, 2009, pp. 1081–1088.

[14] F. Morin and Y. Bengio, “Hierarchical probabilistic neural network language model,” in *AISTATS*, vol. 5. Citeseer, 2005, pp. 246–252.

[15] T. Mikolov, J. Kopecky, L. Burget, O. Glembek, and J. Cernocky, “Neural network based language models for highly inflective languages,” in *Acoustics, Speech and Signal*

Processing, 2009. ICASSP 2009. IEEE International Conference on. IEEE, 2009, pp. 4725–4728.

[16] T. Mikolov, S. Kombrink, L. Burget, J. Cernocky, and S. Khudanpur, "Extensions of recurrent neural network language model," in *Acoustics, Speech and Signal Processing (ICASSP), 2011 IEEE International Conference on*. IEEE, 2011, pp. 5528–5531.

[17] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *Cognitive modeling*, vol. 5, 1988.

[18] R. Socher, A. Perelygin, J. Y. Wu, J. Chuang, C. D. Manning, A. Y. Ng, and C. Potts, "Recursive deep models for semantic compositionality over a sentiment treebank," in *Proceedings of the conference on empirical methods in natural language processing (EMNLP)*, vol. 1631. Citeseer, 2013, p. 1642.

[19] R. Socher, J. Pennington, E. H. Huang, A. Y. Ng, and C. D. Manning, "Semi-supervised recursive autoencoders for predicting sentiment distributions," in *Proceedings of the Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, 2011, pp. 151–161.

[20] R. Socher, E. H. Huang, J. Pennin, C. D. Manning, and A. Y. Ng, "Dynamic pooling and unfolding recursive autoencoders for paraphrase detection," in *Advances in Neural Information Processing Systems*, 2011, pp. 801–809.

[21] R. Socher, B. Huval, C. D. Manning, and A. Y. Ng, "Semantic compositionality through recursive matrix-vector spaces," in *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*. Association for Computational Linguistics, 2012, pp. 1201–1211.

[22] D. Jurafsky and J. H. Martin, *Speech and language processing*. Pearson, 2014.

[23] P. H. Algoet and T. M. Cover, "A sandwich proof of the shannon-mcmillan-breiman theorem," *The annals of probability*, pp. 899–909, 1988.

[24] D. Okanohara and J. Tsujii, "A discriminative language model with pseudo-negative samples." in *ANNUAL MEETING-ASSOCIATION FOR COMPUTATIONAL LINGUISTICS*, vol. 45, no. 1, 2007, p. 73.

[25] T. Mikolov, S. Kombrink, A. Deoras, L. Burget, and J. Cernocky, "Rnnlm-recurrent neural network language modeling toolkit."

[26] A. Stolcke *et al.*, "Srilm-an extensible language modeling toolkit." in *INTERSPEECH*, 2002.

BIOGRAPHICAL SKETCH

Kushal Arora was born in New Delhi, India in April 1988. He received the B.Tech degree in electronics and communication engineering from National Institute of Technology(NIT), Allahabad, in 2010. After his bachelor's degree, he worked as a core member of the software development team of a start-up Chatimity in Bangalore for 2 years. He is currently working in Amazon as a Software Engineer, while he is also pursuing his Masters' thesis, from the University of Florida.

His research interests are natural language understanding, and representational learning. He is looking forward to pursue a PhD in the same or related fields.