A Compositional Approach to Language Modeling

Kushal Arora
Department of CISE,
University of Florida
karora@cise.ufl.edu

Anand Rangarajan
Department of CISE
University of Florida
anand@cise.ufl.edu

Abstract

Traditional language models treat language as a finite state automaton on a probability space over words. This is a very strong assumption when modeling something inherently complex such as language. In this paper, we show how this linear chain assumption can be translated into a sequential composition tree and overcome this structural assumption by proposing a new model that marginalizes over all possible composition trees. We use PCFGs to model the composition trees and reduce the problem to modeling conditional probability distribution of a sentence given the structure. We then model this conditional distribution using continuous space representation. The marginalization is carried out efficiently using Inside-Outside algorithm. We evaluate our new model using Perplexity on Penn and Brown corpus..

1 Introduction

The objective of language modeling is to build a probability distribution over sequences of words. The traditional approaches, inspired by Shannon's game, has molded this problem into merely predicting the next word given the context. This leads to a linear chain model on words. In its simplest formulation, these conditional probabilities are estimated using frequency tables of word w_i following the sequence w_{i-1}^{1}. There are two big issues with this formulation. First, the number of parameters rises exponentially with the size of the context. Second, it is impossible to see all such combinations in the training set, however large it may be. In traditional models, the first problem, famously called the *curse of dimensionality,* is tackled by limiting the history to the previous $n-1$ words leading to an n-gram model. The second problem, one of sparsity, is tackled by redistributing the probability mass over seen and unseen examples usually by applying some kind of smoothing or interpolation techniques. A good overview of various smoothing techniques and their relative performance on language modeling tasks can be found in (Goodman, 2001).

Smoothed n-gram language models fail on two counts: their inability to generalize and their failure to capture the longer context dependencies. The first one is due to the discrete nature of the problem and the lack of any kind of implicit measure of relatedness or context based clustering among words and phrases. The second problem—the failure to capture longer context dependencies—is due to the n-order Markov restriction applied to deal with the curse of dimensionality. There have been numerous attempts to address both the issues in the traditional n-gram framework. Class based models (Brown et al., 1992; Baker and McCallum, 1998; Pereira et al., 1993) try to solve the generalization issue by deterministically or probabilistically mapping words to one or multiple classes based on manually designed or probabilistic criteria. The issue of longer context dependencies has been addressed using various approximations such as cache models (Kuhn and De Mori, 1990), trigger models (Lau et al., 1993) and structured language models (Charniak, 2001; Chelba et al., 1997; Chelba and Jelinek, 2000).

Neural network based language models take an entirely different approach to solving the generalization problem. Instead of trying to solve the difficult task of modeling the probability distribution over discrete sets of words, they try to embed these words into a continuous space and then build a smooth probability distribution over it. Feedforward neural network based models (Bengio et al., 2006; Mnih and Hinton, 2009; Morin and Bengio, 2005) embed the concatenated n-gram history in this latent space and then use a *softmax* layer over these embeddings to predict the next word. This solves the generalization issue by building a smoothly varying probability distribution but is still unable to capture longer dependencies beyond the Markovian boundary. Recurrent neural network based models (Mikolov et al., 2011; Mikolov et al., 2010) attempt to address this by recursively embedding history in the latent space, predicting the next word based on it and then updating the history with the word. Theoretically, this means that the entire history can now be used to predict the next word, hence, the network has the ability to capture longer context dependencies.

All the models discussed above solve the two aforementioned issues to varying degrees of success but none of them actually challenge the underlying linear chain model assumption. Language is recursive in nature, and this along with the underlying compositional structure should play an important role in modeling language. The computational linguistics community has been working for years on formalizing the underlying structure of language in the form of grammars. A step in the right direction would be to look beyond simple frequency estimation-based methods and to use these compositional frameworks to assign the probability to words and sentences.

We start by looking at n-gram models and show how they have an implicit sequential tree assumption. This brings us to the following questions: Is a sequential tree the best compositional structure to model language? If not, then, what is the best compositional structure? Further, do we even need to find one such structure, or can we marginalize over all structures to remove any underlying structural assumptions?

In this paper we take the latter approach. We model the probability of a sentence as ***the marginalized joint probability of words and composition trees*** (over all possible rooted trees). We use a probabilistic context-free grammar (PCFG) to generate these trees and build a probability distribution on them. As generalization is still an issue, we use distributed representations of words and phrases and build a probability distribution on them in the latent space. A similar approach but in a different setting has been attempted in (Socher et al., 2013) for language parsing. The major difference between our approach and theirs is the way we handle the breaking of PCFG's independence assumption due to the distributed representation. Instead of approximating the marginalization using the n-best trees as in (Socher et al., 2013), we restore this independence assumption by averaging over phrasal representations leading to a single phrase representation. This single representation for phrases, in turn, allows us to use an efficient Inside-Outside (Lari and Young, 1990) algorithm for exact marginalization and training.

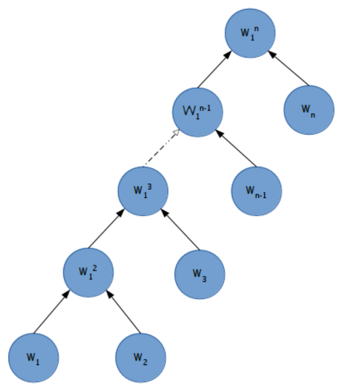

Figure 1: Sequential tree of a linear chain model

2 Compositional View of an N-gram model

Let us consider a sequence of words w_1^n. A linear chain model would factorize the probability of this sentence $p(w_1^n)$ as a product of conditional probabilities $p(w_i|h_i)$ leading to the following factorization:

$$p(w_1^n) = \prod_{i=1}^{n} p(w_i|h_i). \tag{1}$$

All the models discussed in the previous section differ in how the history or context h_i is represented. For a linear chain model with no assumptions, the history h_i would be the previous $i-1$ words, so the factorization is

$$p(w_1^n) = \prod_{i=1}^{n} p(w_i|w_1^{i-1}). \tag{2}$$

If we move the probability space to sequences of words, the same factorization in equation (2) can be written as:

$$p(w_1^n, \dots, w_1^i, \dots, w_1^2, w_n, \dots, w_i, \dots, w_1) = \prod_{i=1}^{n} p(w_1^i | w_1^{i-1}, w_i) p(w_i). \quad (3)$$

Figure 1 shows the sequential compositional structure endowed by the factorization in equation (3). As the particular factorization is a byproduct of the underlying compositional structure, we can rewrite (3) as a probability density of sequence w_1^n conditioned on this sequential tree t as follows:

$$p(w_1^n | t) = \prod_{i=1}^{n} p(w_1^i | w_1^{i-1}, w_i) p(w_i). \quad (4)$$

As the cardinality of set of all possible compositional trees $\mathcal{T}(w_1^n)$ is 1 for n-gram models, equation (4) would also be the probability of the sentence i.e.

$$p(w_1^n) = \sum_{t \in \mathcal{T}(w_1^n)} p(w_1^n | t) p(t) = p(w_1^n | t)$$

3 The Compositional Language Model

Let W be the sentence and $\mathcal{T}(W)$ be the set of all compositional trees for sentence W. The probability of the sentence $p(W)$ can then be written in terms of the joint probability over the sentence and compositional structure as

$$p(W) = \sum_{t \in \mathcal{T}(W)} p(W|t) p(t) \quad (5)$$

Examining (5), we see that we have two problems to solve: i) enumerating and building probability distributions over trees $p(t)$ and ii) modeling the probability of sentences conditioned on compositional trees $p(W|t)$.

Probabilistic Context-Free Grammars (PCFGs) fit the first use case perfectly. Using PCFGs we can write the probability of tree $p(t)$ as:

$$p(t) = \frac{1}{Z(\theta, W)} \prod_{r \in \tilde{R}_t(W)} \theta_r \quad (6)$$

where θ is a real valued vector of length $|R|$ with the rth index mapping to rule r in production and value θ_r is the probability of that rule r[1] .

[1]We restrict our grammar to Chomsky Normal Form (CNF) to simplify the derivation and explanation.

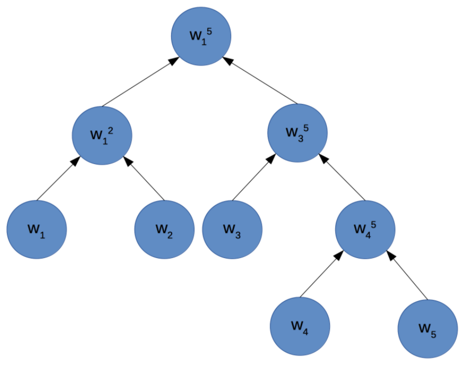

Figure 2: Parse tree for sentence W_{12345}

$\tilde{R}_t(W)$ is the set of all rules used in derivation for tree t and $Z(\theta, W)$ is the inside score for sentence W calculated as follows:

$$Z(\theta, W) = \sum_{t \in \mathcal{T}_G(W)} \prod_{r \in \tilde{R}_t(W)} \theta_r$$

$\mathcal{T}_\mathcal{G}$ here is the set of all the trees generated by that PCFG $\mathcal{G}$.

3.1 The Composition Tree Representation

We now focus our attention to the problem of modeling $p(W|t)$. Let t be the tree shown in Figure 2. The factorization of w_1^5 given t is

$$\begin{aligned} p(w_1^5|t) = & p(w_1^5 | w_1^2, w_3^5) p(w_1^2 | w_1, w_2) \\ & p(w_3^5 | w_3, w_4^5) p(w_4^5 | w_4, w_5) \\ & p(w_1) p(w_2) p(w_3) p(w_4) p(w_5). \quad (7) \end{aligned}$$

We now seek to represent any arbitrary tree t such that it can be factorized easily as (7). We do this by representing the compositional tree t as a set of compositional rules and leaf nodes. Let us call this set $R_t(W)$. Using this abstraction, the rule set for the sentence w_1^5 with compositional tree t, $R_t(w_1^5)$ is

$$\begin{aligned} R_t(w_1^5) \quad = \quad & \{w_1^5 \rightarrow w_1^2 \; w_3^5, \\ & w_3^5 \rightarrow w_3 \; w_4^5, w_3, \\ & w_1^2 \rightarrow w_1 \; w_2, w_1, w_2, \\ & w_4^5 \rightarrow w_4 \; w_5, w_4, w_5\}. \quad (8) \end{aligned}$$

We now rewrite the factorization in (7) as

$$p(w_1^5|t) = \prod_{r \in R_t(w_1^5)} p(r) \quad (9)$$

where $p(pa \rightarrow c_1 \; c_2) = p(pa | c_1, c_2)$.

In more general terms, let t be a compositional tree for the sentence W. We can write the conditional probability $p(W|t)$ as

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

3.2 Computing the Sentence Probability

Using the definition of $p(t)$ from (6) and the definition of $p(W|t)$ from (10), we rewrite the joint probability $p(W,t)$ as[2]

$$p(W,t) = \prod_{c \in R_t(W)} p(c) \prod_{r \in \tilde{R}_t(W)} \theta_r. \tag{11}$$

Now, as the compositional tree t is the same for both the production rule set $\tilde{R}_t(W)$ and the compositional rule set $R_t(W)$, there is a one-to-one mapping between the binary rules in both the sets.

Adding corresponding non-terminal tags to phrase w_i^j, we can merge both these sets. Let $R_t(W)$ be the merged set, A, B and C are non-terminals. The compositional rules and leaf nodes in this new set can be re-written as

$$A, w_i^j \rightarrow BC, w_i^k w_{k+1}^j, \tag{12}$$

and

$$A \rightarrow w_i.$$

Using this new rule set, we can rewrite $p(W,t)$ from (11) as

$$p(W,t) = \prod_{r \in R_t(W)} \zeta_r \tag{13}$$

and $p(W)$ from (5) as

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

where $\zeta_r = p(r)\theta_r$.

The marginalization formulation in (14) is similar to one solved by the Inside algorithm.

Inside Algorithm: Let $\pi(A, w_i^j)$ be the inside probability of a non-terminal A spanning w_i^j. For $W = w_1^n$, we can rewrite $p(W)$ in terms of the inside probability as

$$p(W) = \pi(S, w_1^n). \tag{15}$$

[2]Henceforth, we are dropping the term $Z(\theta, W)$ for brevity. We will assum θ_r is normalized such that $\sum_{t \in \mathcal{T}_\mathcal{G}(W)} \prod_{r \in \tilde{R}_t(W)} \theta_r = 1$.

We can now recursively build $\pi(S, w_1^n)$ in the following way:

***Base Case*:** For the unary rule, the inside probability $\pi(A, w_i)$ is the same as the production rule probability $\zeta_{A \rightarrow w_i}$.

***Recursive Definition*:** Let $\pi(B, w_i^k)$ and $\pi(C, w_{k+1}^j)$ be the inside probabilities spanning w_i^k rooted at B and w_{k+1}^j rooted at C respectively. Let $r = A, w_i^j \rightarrow BC, w_i^k w_{k+1}^j$ be the rule which composes w_i^j from w_i^k and w_{k+1}^j, each one rooted at A, B and C respectively.

The inside probability of rule r, $\pi(r)$, can then be calculated as

$$\pi(r) = \zeta_r \pi(B, w_i^k) \pi(C, w_{k+1}^j). \tag{16}$$

Let $r_{A,i,k,j}$ be the rule rooted at A spanning w_i^j with a split at k. We can now calculate $\pi(A, w_i^j)$ by summing over all possible splits between i, j.

$$\pi(A, w_i^j) = \sum_{i<k<j} \sum_{r_{A,i,k,j} \in R} \pi(r_{A,i,k,j}). \tag{17}$$

Examining equations (14), we see that we have reduced the problem of modeling $p(W)$ to modeling probability over words and the compositional rules i.e. $p(r)$. In the following section we carefully examine this problem.

3.3 Modeling the compositional probability

The input to our model is a sentence. We represent it as an array of integers, each integer referring to the index of the word w in our vocabulary V. As a first step, we project each word into a continuous space using X, a $d \times |V|$ embedding matrix, to obtain a continuous space vector $x_i = X[i]$ corresponding to word w_i. d here is the dimension of embedding space.

A nonterminal parent node pa is composed of children nodes c_1 and c_2 as

$$pa = f\left(W \begin{bmatrix} c_1 \\ c_2 \end{bmatrix}\right) \tag{18}$$

where W is a parameter with dimensions $d \times 2d$ and f is a non-linear function like *sigmoid*.

The probability distribution $p(r)$ over rule $r \in R_t(W)$ is modeled as a Gibbs distribution. The rule r in our case can either be a leaf node or a composition rule. Let's start with r being a leaf node. In this case the probability of $r = w_i$ is modeled as

$$p(r;\alpha) = \frac{\exp\{-E(w_i;\alpha)\}}{\sum_{w \in V} \exp\{-E(w;\alpha)\}} \tag{19}$$

where w_i is the ith word in vocabulary V. E here is the energy function which is defined as

$$E(r;\alpha) = g(u^T w_i). \tag{20}$$

Now, let's focus at modeling composition rule probability. Let $r = w_i^j \rightarrow w_i^k w_{k+1}^j$ be the composition rule. The probability that w_i^j was formed by phrases w_i^k and w_{k+1}^j is modeled as

$$p(r;\alpha) = \frac{\exp\{-E(w_i^j \rightarrow w_i^k, w_{k+1}^j;\alpha)\}}{\sum_{i<l<j} \exp\{-E(w_l^j \rightarrow w_l^k, w_{l+1}^j;\alpha)\}} \tag{21}$$

Here, the energy function $E(r;\alpha)$ is modeled as

$$E(r;\alpha) = g(u^T w_i^j + h_1^T w_i^k + h_2^T w_{k+1}^j). \tag{22}$$

Vector u in equation (20) and (22) is a scoring vector of dimension $d \times 1$, g is the *identity* function. h_1 and h_2 are $d \times 1$ dimensional scoring vectors corresponding to children vectors. Looking at the definition of the energy function, equation (20) and (22) are the same as children vectors will be zero for leaf node. From (18), (20) and (22), the parameter α is (u, h_1, h_2, X, W).

4 Training

Let D be the set of training sentences. We can write the negative log-likelihood objective function as

$$\mathcal{E}_{\text{ML}}(\alpha;D) = -\sum_{W_d \in D} \ln(p(W_d;\alpha)). \tag{23}$$

Substituting the definition of $p(W;\alpha)$ from (14) in (23), we get

$$\mathcal{E}_{\text{ML}}(\alpha;D) = \\ -\sum_{W_d \in D} \ln\left(\sum_{t \in \mathcal{T}(W_d)} \prod_{r \in R_t(W_d)} \zeta_r(\alpha)\right) \tag{24}$$

The formulation in equation (24) is very similar to the standard expectation-maximization (EM) formulation where the compositional tree t can be seen as a latent variable.

4.1 Expectation Step

In the E-step, we compute the expected log-likelihood $\mathcal{Q}(\alpha;\alpha^{old},W)$ as follows.

$$\mathcal{Q}(\alpha;\alpha^{old},W) = \\ -\sum_{t \in \mathcal{T}_G(W)} p(t|W;\alpha^{old}) \ln(p(t,W;\alpha)). \tag{25}$$

Substituting $p(t,W)$ from (13) in (25), we can rewrite $\mathcal{Q}(\alpha;W)$[3] as

$$\mathcal{Q}(\alpha;W) = \\ -\sum_{t \in \mathcal{T}_G(W)} p(t|W) \sum_{r \in R_t(W)} \ln(\zeta_r(\alpha)). \tag{26}$$

We can simplify the expression further by taking summations over the trees inside leading to the following expression

$$\mathcal{Q}(\alpha;W) = -\frac{1}{p(W)} \sum_{r \in R(W)} \mu(r) \ln(\zeta_r(\alpha)) \tag{27}$$

where $R(W)$ is a set of all possible rules for sentence W i.e. $\cup_{t \in \mathcal{T}_G(W)} R_t(W)$ and $\mu(r)$ is sum over all trees that contain rule r i.e.

$$\mu(r) = \sum_{t \in \mathcal{T}_G(W): r \in R_t(W)} p(t,W). \tag{28}$$

The term $\mu(r)$ can be calculated using the inside term π and a new term—the outside term β.

Outside Algorithm: The Outside term $\beta(A, w_i^j)$ is the probability of expanding S to sentence w_i^n such that sub-sequence w_i^j rooted at A is left unexpanded. Similar to the inside probability, the outside probability can be calculated recursively as follows:

Base Case: As the complete sentence is always rooted at S, $\beta(S, w_1^n)$ is always 1. Moreover, as no other non-terminal A can be the root of the parse tree, $\beta(A, w_1^n)$, $A \neq S$ is zero.

Recursive Definition: To compute $\beta(A, w_i^j)$ we need to sum up the probabilities both to the left and right of w_i^j. Let $k < i$ and $l > j$ be the indexes to the left and right of the span w_i^j. Let $\beta(r_L)$ be the probability of expanding subsequence w_k^j rooted at B using rule $r_L = B, w_k^j \rightarrow$

[3]Henceforth we drop the term α^{old} in all our equations for the sake of brevity.

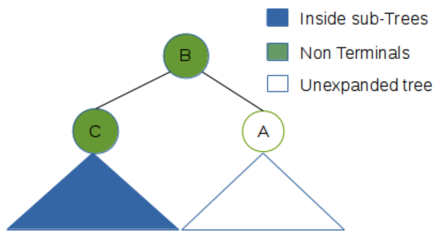

Figure 3: Calculating $\beta(B, w_k^j \to CA, w_k^{i-1} w_i^j)$, the outside probability of nonterminal A spanning w_i^j such that rule $B, w_k^j \to C\,A, w_k^{i-1}\,w_i^j$ was used expand the subsequence to its left

$C\,A, w_k^{i-1}\,w_i^j$ such that w_i^j rooted at A is left unexpanded.

We can write $\beta(r_L)$ in terms of its parent's outside probability $\beta(B, w_k^j)$, left sibling's inside probability $\pi(C, w_k^{i-1})$ and ζ_{r_L} as

$$\beta(r_L) = \zeta_{r_L}\pi(C, w_k^{i-1})\beta(B, w_k^j). \tag{29}$$

Similarly, on the right hard side of w_i^j, for rule $r_R = B, w_i^l \to AC, w_i^j w_{j+1}^l$ $\beta(r_R)$ will be

$$\beta(r_R) = \zeta_{r_R}\pi(C, w_{j+1}^l)\beta(B, w_i^l). \tag{30}$$

Let $r_{A,k,i,j}$ and $r_{A,i,j,l}$ be the rules spanning w_k^j and w_i^l such that nonterminal A spanning w_i^j is its left and right child respectively. $\beta(A, w_i^j)$ can then be calculated by summing $r_{A,k,i,j}$ and $r_{A,i,j,l}$ over all such rules and splits, i.e.

$$\beta(A, w_i^j) = \sum_{k=1}^{i-1} \sum_{r_{A,k,i,j}\in R} \beta(r_{A,k,i,j}) + \sum_{l=j+1}^{n} \sum_{r_{A,i,j,l}\in R} \beta(r_{A,i,j,l}). \tag{31}$$

Now, let look at the definition of $\mu(r)$. The rule $r = A, w_i^j \to BC, w_i^k w_{k+1}^j$ is rooted at A and spans w_i^j with a split at k. $\pi(r)$ contains probabilities of all the trees rooted at A spanning w_i^j that uses production rule r in their derivation. $\beta(A, w_i^j)$ would contain the probability of expanding non-terminal S to w_1^n leaving A spanning w_i^j unexpanded. Hence, their product contains the probability of all parse trees that have rule r in them, i.e. $\mu(r)$.

4.2 Minimization Step

In the M-step, the objective is to estimate α^* such that

$$\alpha^\star = \arg\min_\alpha \sum_{W_d\in D} \mathcal{Q}(\alpha; \alpha^{old}, W_d) \tag{32}$$

We carry out minimization using mini-batch gradient descent with the gradient $\partial Q/\partial\alpha$ calculated as

$$\frac{\partial\mathcal{Q}}{\partial\alpha} = -\frac{1}{P(W)} \sum_{r\in R(W)} \left\{\mu(r)\frac{\partial\ln(\zeta_r(\alpha))}{\partial\alpha}\right\}. \tag{33}$$

Substituting the definition of $\zeta_r(\alpha)$ and the value of $p(r;\alpha)$ for leaf nodes and composition rules from equation (19) and (21) respectively, we can re-write equation (33) as:

$$\frac{\partial\mathcal{Q}}{\partial\alpha} = \frac{1}{P(W)} \sum_{r_{ijk}\in R_C(W)} \mu(r_{ijk})\frac{\partial ln(p(r_{ijk};\alpha))}{\partial\alpha} + \frac{1}{P(W)} \sum_{r_i\in R_L(W)} \mu(r_i)\frac{\partial ln(p(r_i;\alpha))}{\partial\alpha} \tag{34}$$

where R_C and R_L are set of composition rules and leaf nodes respectively i.e.

$$R_C = \cup_{i,j,k} w_i^j \to w_i^k w_{k+1}^j$$

and

$$R_L = \cup_{i<n} w_i$$

$\mu(r_{ijk})$ and $\mu(r_i)$ are $\mu(r)$ summed over non-terminal i.e.

$$\mu(r_{ijk}) = \sum_{A\to BC\in R} \mu(A, w_i^j \to BC, w_i^k w_{k+1}^j)$$

and

$$\mu(r_i) = \sum_{A\in R} \mu(A \to w_i).$$

The gradients $\partial ln(p(r_{ijk};\alpha))/\partial\alpha$ and $\partial ln(p(r_i;\alpha))/\partial\alpha$ can be computed as follows:

$$\frac{\partial ln(p(r_{ijk};\alpha))}{\partial\alpha} = \frac{\partial E(r_{ijk};\alpha)}{\partial\alpha} - \boldsymbol{E}_l\left[\frac{\partial E(r_{ijl};\alpha)}{\partial\alpha}\right]$$

and

$$\frac{\partial ln(p(r_i;\alpha))}{\partial\alpha} = \frac{\partial E(w_i;\alpha)}{\partial\alpha} - \boldsymbol{E}_V\left[\frac{\partial E(w;\alpha)}{\partial\alpha}\right].$$

Here, $\boldsymbol{E}_l$ and $\boldsymbol{E}_V$ are expectation over split index l and the whole vocabulary V respectively.

Let $s = u^T pa + h_1^T c_1 + h_2^T c_2$. $\partial E/\partial \alpha$ will then be $g'(s)\partial s/\partial \alpha$. The derivative of s w.r.t to u, h_1 and h_2 will simply be pa, c_1 and c_2 respectively. The derivative w.r.t to W and X will be

$$\frac{\partial s}{\partial w_{lm}} = u^T \frac{\partial pa}{\partial w_{lm}} + h_1^T \frac{\partial c_1}{\partial w_{lm}} + h_1^T \frac{\partial c_2}{\partial w_{lm}}, \quad (35)$$

and

$$\frac{\partial s}{\partial x_i} = u^T \frac{\partial pa}{\partial x_i} + h_1^T \frac{\partial c_1}{\partial x_i} + h_1^T \frac{\partial c_2}{\partial x_i}. \quad (36)$$

Here, w_{lm} is the scalar at index (l, m) in W and x_i be an embedding vector in X.

Let p be a phrase or a word. The derivatives $\frac{\partial p}{\partial w_{lm}}$ and $\frac{\partial p}{\partial x_i}$ can be recursively calculated as follows:

$\partial \mathbf{p}/\partial \mathbf{W}$: ***Base Case:*** For terminal node p, $\frac{\partial p}{\partial w_{lm}}$ is zero as there is no composition involved.

Recursive Definition: Let $\frac{\partial p_1}{\partial w_{lm}}$ and $\frac{\partial p_2}{\partial w_{lm}}$ be the partial derivatives of children p_1 and p_2 of node p w.r.t to w_{lm}. The derivative of parent embedding $\frac{\partial p}{\partial w_{lm}}$ can then be computed as

$$\frac{\partial p}{\partial w_{lm}} = f' \circ \left\{ p_m^{12} \mathbf{1}_l + W \begin{bmatrix} \frac{\partial p_1}{\partial w_{lm}} \\ \frac{\partial p_2}{\partial w_{lm}} \end{bmatrix} \right\} \quad (37)$$

where $\circ$ is the Hadamard product, $p^{12} = \begin{bmatrix} p_1 \\ p_2 \end{bmatrix}$, p_m^{12} is the scalar at m th index of w^{12} and $\mathbf{1_l}$ is an indicator vector of size $d \times 1$.

$\partial \mathbf{p}/\partial \mathbf{X}$: ***Base Case:*** For a terminal node p, there are two possibilities, either p is equal to x_i or it is not. If $p = x_i$, $\frac{\partial p}{\partial x_i}$ is a d dimension identity matrix otherwise, it is zero.

Recursive Definition: Let $\frac{\partial p_1}{\partial x_i}$ and $\frac{\partial p_2}{\partial x_i}$ be the partial derivatives of children p_1 and p_2 of phrase p. The derivative of the parent embedding p can be built using $\frac{\partial p_1}{\partial x_i}$ and $\frac{\partial p_2}{\partial x_i}$ as

$$\frac{\partial p}{\partial x_i} = f' \circ W \begin{bmatrix} \frac{\partial p_1}{\partial x_i} \\ \frac{\partial p_2}{\partial x_i} \end{bmatrix}. \quad (38)$$

4.3 Phrasal Representation

One of the inherent assumptions we made while using the Inside-Outside Algorithm was that each span has a single representation. This is an important assumption because the *states* in the Inside and Outside algorithms are these spans. A distributed representation breaks this assumption. To understand this, let's consider a three-word sentence w_1^3. Figure 4 shows possible derivations of this sentence. Embeddings for $p_{\{1\{23\}\}}$ and $p_{\{\{12\}3\}}$ are different as they follow different compositional paths despite both representing the same phrase w_1^3. Generalizing this, any sentence or phrase of length 3 or greater would suffer from the multiple representation problem due to multiple possible compositional pathways.

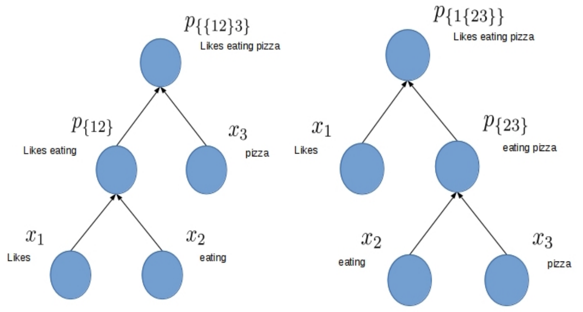

Figure 4: Two different compositional trees for w_1^3 leading to different phrasal representations.

To understand why this is an issue, let's examine the Inside algorithm recursion. Dynamic programming works while calculating $\pi(A, w_i^j)$ because we assume that there is only one possible value for $\pi(B, w_i^k)$ and $\pi(C, w_{k+1}^j)$. Also, our compositional probability $p(r)$ depends on the phrase embeddings, so multiple possible phrase representations would mean multiple values inside probabilities for each span. This can also be seen as breakage of the independence assumption of CFGs as now the probability of the parent node also depends on how its children were composed.

We restore the assumption by taking the expected value of phrasal embeddings w.r.t. its compositional inside probability $\pi(w_i^j \to w_i^k w_{k+1}^j)$ [4]

$$X(i, j) = E_\pi[X(i, k, j)]. \quad (39)$$

With this approximation, the phrasal embedding for w_1^3 from Figure 4 is

$$X(1,3) = p_{\{1\{23\}\}}\pi(w_1^3 \to w_1 w_2^3) + p_{\{\{12\}3\}}\pi(w_1^3 \to w_1^2 w_3). \quad (40)$$

[4]Inside probability of composition $\pi(w_i^j \to w_i^k w_{k+1}^j)$ is Inside rule probability $\pi(A, w_i^j \to BC, w_i^k w_{k+1}^j)$ marginalized for all nonterminals

This representation is intuitive as well. If composition structures for a phrase lead to multiple representations then the best way to represent that phrase would be an average representation weighted by the probability of each composition. Now, with the context-free assumption restored we can use the Inside-Outside algorithm for efficient marginalization and training.

4.4 Computational Complexity

Let's start by analyzing the time complexity of computing $P(W)$. Inside score computation in equation (14) can be seen as computing $p(r)$, multiplying it with θ_r and then summing for all rules and splits. For binary rules, the number of operation for each step will be $\Theta(nd^2 + n|G|)$ where $|G|$ is the size of the grammar and n is the length of the sentence. We will have to do these operation for all $i, j|1 < i < j < n$. So, the total time complexity of computing probability $P(W)$ will be $\Theta(n^3d^2 + n^3|G|)$. Now, for the unary rules, the cost of computing energy function $E(w_i; \alpha)$ will be $\Theta(d)$ and for a sentence of length n, the total cost will be $\Theta(nd)$. The normalization constant in equation (19) is independent of word w_i and can be calculated once for the whole training batch at the cost of $\Theta(dV)$. So, the amortized cost of computing $P(W)$ for a training batch of size N will be $\Theta(n^3d^2 + n^3|G| + \frac{1}{N}dV)$.

Now, let's turn our focus to the training phase. Starting with the leaf node, we do not need to compute $\partial E/\partial h_1$, $\partial E/\partial h_2$ and $\partial E/\partial w_{lm}$, and $\partial E/\partial u$ and $\partial E/\partial x_i$ can be computed in $\Theta(d)$ and $\Theta(d^2)$, so the overall complexity of $\partial E/\partial \alpha$ is $\Theta(d^2)$. The expectation over whole vocabulary $\boldsymbol{E}_V$ can then be calculated in $\Theta(Vd^2)$.

Important thing to note here is that as $\boldsymbol{E}_V$ is independent of index i, j, k and hence we can sum $\mu(r)$ over all spans and split and this value will be $nP(W)$. So, effectively $\boldsymbol{E}_V$ can be seen as a overall vocabulary normalization constraint with Lagrange multiplier equal to number of words in the batch and can be calculated once per batch. Summing it up, for a sentence of length n, the overall complexity of computing $\partial p(r_i; \alpha)/\partial \alpha$ for leaf nodes will be $\Theta(nd^2 + \frac{1}{N}d^2V)$ where N is the batch size.

For composition rules, the computational complexity of computing $\partial p/\partial w_{lm}$ and $\partial p/\partial x_i$ will be $\Theta(d^2)$ and $\Theta(d^3)$. $\partial E/\partial \alpha$ can then computed at an additional cost of $\mathcal{O}(d^2)$. So, overall cost of computing $\partial E/\partial \alpha$ will be of the order of $\Theta(d^3)$. We need to sum up $\partial p(r_{ijk})/\partial \alpha$ over all $ijk|1 \leq i < k < j \leq n$, so $\boldsymbol{E}_l$ can be computed as a part of this step. So, the overall complexity of computing $\partial p(r_{ijk})/\partial \alpha$ for all spans and splits will be $\Theta(n^3d^3)$. $P(W)$ and $\mu(r)$ can be calculated in $\Theta(n^3d^2 + n^3|G|)$. So, the overall computational complexity of training a sentence of length n is $\Theta(n^3d^3 + n^3|G| + \frac{1}{N}d^2V)$[5].

An important optimization to reduce training time will be to use pre-trained word embedding and not to learn them. This would eliminate $\Theta(d^3)$ factor originating from calculating $\partial p/\partial x_i$ hence reducing the time complexity to $\Theta(n^3d^2 + n^3|G| + \frac{1}{N}d^2V)$.

5 Conclusion

In this paper, we challenged the linear chain assumption of the traditional language models by building a model that uses the compositional structure endowed by context-free grammars. We formulated it as a marginalization problem over the joint probability of sentences and structure and reduced it to one of modeling compositional rule probabilities $p(r)$. To the best of our knowledge, this is the first model that looks beyond the linear chain assumption and uses the compositional structure to model the language. It is important to note that this compositional framework is much more general and the way this paper models $p(r)$ is only one of many possible ways to do so.

Also, this paper proposed a compositional framework that recursively embeds phrases in a latent space and then builds a distribution over it. This provides us with a distributional language representation framework which, if trained properly, can be used as a base for various language processing tasks like NER, POS tagging, and sentiment analysis. The assumption here is that most of the heavy lifting will be done by the representation and a simple classifier should be able to give good results over the benchmarks. We also hypothesize that phrasal embeddings generated using this model will be much more robust and will also exhibit interesting regularities due to the marginalization over all possible structures.

As the likelihood optimization proposed here is highly nonlinear, better initialization, and im-

[5]We here assume that the model level parallelism allows us to update w_{lm} and x_i in parallel. If that's not the case the computational complexity will be $\Theta(n^4d^3 + n^3d^4 + n^3|G| + \frac{1}{N}d^2V)$

proved optimization and regularization techniques being developed for deep architectures can further improve these results. Another area of research is to study the effects of the choice of compositional functions and additional constraints on representation generated by the model and finally the performance of the classification layer built on top

References

L Douglas Baker and Andrew Kachites McCallum. 1998. Distributional clustering of words for text classification. In *Proceedings of the 21st annual international ACM SIGIR conference on Research and development in information retrieval*, pages 96–103. ACM.

Yoshua Bengio, Holger Schwenk, Jean-Sébastien Senécal, Fréderic Morin, and Jean-Luc Gauvain. 2006. Neural probabilistic language models. In *Innovations in Machine Learning*, pages 137–186. Springer.

Peter F Brown, Peter V Desouza, Robert L Mercer, Vincent J Della Pietra, and Jenifer C Lai. 1992. Class-based n-gram models of natural language. *Computational linguistics*, 18(4):467–479.

Eugene Charniak. 2001. Immediate-head parsing for language models. In *Proceedings of the 39th Annual Meeting on Association for Computational Linguistics*, pages 124–131. Association for Computational Linguistics.

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

Ciprian Chelba, David Engle, Frederick Jelinek, Victor Jimenez, Sanjeev Khudanpur, Lidia Mangu, Harry Printz, Eric Ristad, Ronald Rosenfeld, Andreas Stolcke, et al. 1997. Structure and performance of a dependency language model. In *EUROSPEECH*. Citeseer.

Joshua T Goodman. 2001. A bit of progress in language modeling. *Computer Speech & Language*, 15(4):403–434.

Roland Kuhn and Renato De Mori. 1990. A cache-based natural language model for speech recognition. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 12(6):570–583.

Karim Lari and Steve J Young. 1990. The estimation of stochastic context-free grammars using the inside-outside algorithm. *Computer speech & language*, 4(1):35–56.

Raymond Lau, Ronald Rosenfeld, and Salim Roukos. 1993. Trigger-based language models: A maximum entropy approach. In *Acoustics, Speech, and Signal Processing, 1993. ICASSP-93., 1993 IEEE International Conference on*, volume 2, pages 45–48. IEEE.

Tomas Mikolov, Martin Karafiát, Lukas Burget, Jan Cernockỳ, and Sanjeev Khudanpur. 2010. Recurrent neural network based language model. In *INTERSPEECH*, pages 1045–1048.

Tomas Mikolov, Stefan Kombrink, Lukas Burget, JH Cernocky, and Sanjeev Khudanpur. 2011. Extensions of recurrent neural network language model. In *Acoustics, Speech and Signal Processing (ICASSP), 2011 IEEE International Conference on*, pages 5528–5531. IEEE.

Andriy Mnih and Geoffrey E Hinton. 2009. A scalable hierarchical distributed language model. In *Advances in neural information processing systems*, pages 1081–1088.

Frederic Morin and Yoshua Bengio. 2005. Hierarchical probabilistic neural network language model. In *AISTATS*, volume 5, pages 246–252. Citeseer.

Fernando Pereira, Naftali Tishby, and Lillian Lee. 1993. Distributional clustering of english words. In *Proceedings of the 31st annual meeting on Association for Computational Linguistics*, pages 183–190. Association for Computational Linguistics.

Richard Socher, John Bauer, Christopher D Manning, and Andrew Y Ng. 2013. Parsing with compositional vector grammars. In *In Proceedings of the ACL conference*. Citeseer.