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 challenge this by showing how the linear chain assumption inherent in previous work can be translated into a sequential composition tree. We then propose a new model that marginalizes over all possible composition trees thereby removing any underlying structural assumptions. As the partition function of this new model is intractable, we use a recently proposed sentence level evaluation metric Contrastive Entropy to evaluate our model. Given this new evaluation metric, we report more than 100% improvement across distortion levels over current state of the art recurrent neural network based language models.

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 tack-

led 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. Feedfor-

ward 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., 2011a; 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 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
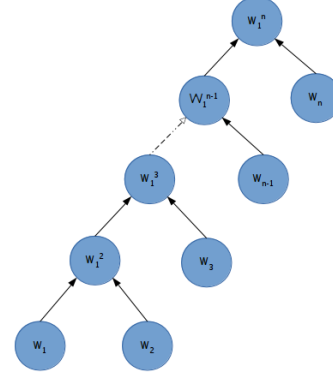

Figure 1: Sequential tree of a linear chain model

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.

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:

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

All the models discussed in 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

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

If we move the probability space to sequences of words, the same factorization in equation (2)

can be written as:

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

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:

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

Having shown the sequential compositional structure assumption of the n-gram model, in the next section we try to remove this conditional density assumption by modeling the joint probability of the sentences and the compositional trees.

3 The Compositional Language Model

In this section, we build the framework to carry out the marginalization over all possible trees. 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. We define a PCFG as a quintuple:

$$(G, \theta) = (N, T, R, S, P) \quad (6)$$

where N is the set of non-terminal symbols, T is the set of terminal symbols, R is the finite set of production rules, S is the special start symbol which is always at the root of a parse tree and P is the set of probabilities on production rules.[1] θ is a

---

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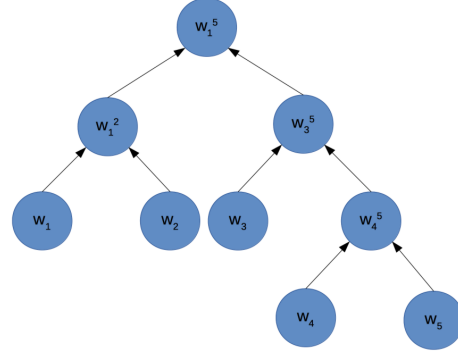

Figure 2: Parse tree for sentence W_{12345}

real valued vector of length $|R|$ with the rth index mapping to rule $r \in R$ and value $\theta_r \in P$.

Using this definition, we can write the probability of any tree t, $p(t)$, as the product of the production rules used to derive the sentence W from S i.e.

$$p(t) = \prod_{r \in \tilde{R}_t(W)} \theta_r \quad (7)$$

where $\tilde{R}_t(W) \in R$ is the set of production rules used to derive tree t. As we are only interested in the compositional structure, $\tilde{R}_t(W)$ only contains the binary rules for tree t.

3.1 The Composition Tree Representation

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

$$p(w_1^5|t) = p(w_1^5|w_1^2, w_3^5) p(w_1^2|w_1, w_2)$$
$$p(w_3^5|w_3, w_4^5) p(w_4^5|w_4, w_5)$$
$$p(w_1) p(w_2) p(w_3) p(w_4) p(w_5). \quad (8)$$

We now seek to represent any arbitrary tree t such that it can be factorized easily as we did in (8). We do this by representing the compositional tree t for a sentence W 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

$$R_t(w_1^5) = \{w_1^5 \leftarrow w_1^2 \, w_3^5,$$
$$w_3^5 \leftarrow w_3 \, w_4^5, w_3,$$
$$w_1^2 \leftarrow w_1 \, w_2, w_1, w_2,$$
$$w_4^5 \leftarrow w_4 \, w_5, w_4, w_5\}. \quad (9)$$

We now rewrite the factorization in (8) as

$$p(w_1^5|t) = \prod_{r \in R_t(w_1^5)} p(r) \quad (10)$$

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

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

$$p(W|t) = \prod_{r \in R_t(W)} p(r). \quad (11)$$

3.2 Computing the Sentence Probability

Using the definition of $p(t)$ from (7) and the definition of $p(W|t)$ from (11), we rewrite the joint probability $p(W, t)$ as

$$p(W, t) = \prod_{c \in R_t(W)} p(c) \prod_{r \in \tilde{R}_t(W)} \theta_r. \quad (12)$$

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. The compositional rules in this new set can be re-written as

$$A, w_i^j \rightarrow BC, w_i^k w_{k+1}^j. \quad (13)$$

Using this new rule set, we can rewrite $p(W, t)$ from (12) as

$$p(W, t) = \prod_{r \in R_t(W)} \zeta_r \quad (14)$$

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

$$p(W) = \sum_{t \in \mathcal{T}(W)} \prod_{r \in R_t(W)} \zeta_r \quad (15)$$

where

$$\zeta_r = \begin{cases} p(r)\theta_r & \text{binary rules} \\ p(r) & \text{unary rules} \end{cases}. \quad (16)$$

The marginalization formulation in (15) is similar to one solved by the Inside algorithm.

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

$$p(W) = \pi(S, w_1^n). \quad (17)$$

We can now recursively build $\pi(S, w_1^n)$ using a dynamic programming (DP) based Inside algorithm 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

$$\pi(r) = \zeta_r \pi(B, w_i^k)\pi(C, w_{k+1}^j). \quad (18)$$

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

$$\pi(A, w_i^j) = \sum_{i < k < j} \sum_{r_{A,i,k,j} \in R} \pi(r_{A,i,k,j}). \quad (19)$$

Examining equations (15) and (16), we see that we have reduced the problem of modeling $p(W)$ to modeling $p(r)$, i.e. modeling the probability of compositional rules and leaf nodes. In the next section we carefully examine this problem. The approach we follow here is similar to the one taken in (Socher et al., 2010). We embed the words in a vocabulary V in a latent space of dimension d, use a compositional function f to build phrases in this latent space and then build a probability distribution function p over the leaf nodes and compositional rules.

3.3 Modeling the compositional probability

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 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.

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

$$pa = f\left(W \begin{bmatrix} c_1 \\ c_2 \end{bmatrix} \right) \qquad (20)$$

where W is a parameter with dimensions $d \times 2d$ and f is a non-linear function like *tanh* or a *sigmoid*. The probability distribution $p(r)$ over rule $r \in R_t(W)$ is modeled as a Gibbs distribution

$$p(r) = \frac{1}{Z} \exp\left\{ -E(r) \right\} \qquad (21)$$

where $E(r)$ is the energy for a compositional rule or a leaf node, and is modeled as

$$E(r) = g(u^T pa). \qquad (22)$$

Here u is a scoring vector of dimension $d \times 1$ and g is the *identity* function. From (20), (21) and (22), the parameters α of $p(r; \alpha)$ are (u, X, W).

In the next section we derive an approach for parameter estimation. We achieve this by formulating training as a maximum likelihood estimation problem and estimating α by maximizing the probability over the training set D.

4 Training

Let D be the set of training sentences. We can write the likelihood function as

$$\mathcal{L}(\alpha; D) = \prod_{W_d \in D} p(W_d; \alpha). \qquad (23)$$

This leads to the negative log-likelihood objective function

$$\mathcal{E}_{\mathrm{ML}}(\alpha; D) = -\sum_{W_d \in D} \ln(p(W_d; \alpha)). \qquad (24)$$

Substituting the definition of $p(W; \alpha)$ from (15) in (24), we get

$$\mathcal{E}_{\mathrm{ML}}(\alpha; D) = -\sum_{W_d \in D} \ln\left(\sum_{t \in \mathcal{T}(W_d)} \prod_{r \in R_t(W_d)} \zeta_r(\alpha) \right) \qquad (25)$$

The formulation in equation (25) 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.

$$\mathcal{Q}(\alpha; \alpha^{old}, W) =$$
$$- \sum_{t \in \mathcal{T}_G(W)} p(t|W; \alpha^{old}) \ln(p(t, W; \alpha)). \qquad (26)$$

Substituting $p(t, W)$ from (14) in (26), we can rewrite $\mathcal{Q}(\alpha; W)^2$ as

$$\mathcal{Q}(\alpha; W) =$$
$$- \sum_{t \in \mathcal{T}_G(W)} p(t|W) \sum_{r \in R_t(W)} \ln(\zeta_r(\alpha)). \qquad (27)$$

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

$$\mathcal{Q}(\alpha; W) = -\frac{1}{p(W)} \sum_{r \in R} \mu(r) \ln(\zeta_r(\alpha)) \qquad (28)$$

where

$$\mu(r) = \sum_{t \in \mathcal{T}_G(W): r \in R_t(W)} p(t, W). \qquad (29)$$

The term $\mu(r)$ sums over all trees that contain rule r and can be calculated using the inside term π and a new term—the outside term β. Before computing $\mu(r)$, let's examine how to compute this outside term β.

Outside Algorithm: The Inside term $\pi(A, w_i^j)$ is the probability of $A \in N$ spanning sub-sequence w_i^j. The Outside term $\beta(A, w_i^j)$ is just the opposite. $\beta(A, w_i^j)$ 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's consider summing over the left side span w_1^{i-1}. Figure 3 shows one of the

---

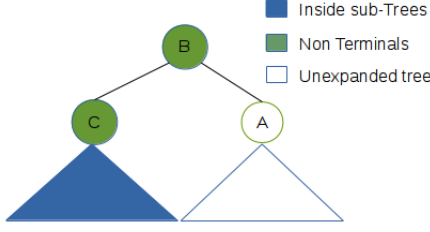[2] Henceforth we drop the term α^{old} in all our equations for the sake of brevity.

Figure 3: Calculating $\beta(B, w_k^j \to CA, w_k^{i-1} w_i^j)$, the outside probability of non terminal 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

intermediate steps. Let $\beta(r_L)$ be the probability of expanding subsequence w_k^{i-1} rooted at B using rule $r_L = B, w_k^j \to C A, w_k^{i-1} w_i^j$ 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

$$\beta(r_L) = \zeta_{r_L} \pi(C, w_k^{i-1}) \beta(B, w_k^j). \qquad (30)$$

Similarly, on the right hard side of w_i^j, we can calculate $\beta(r_R)$ in terms of its parent's outside probability $\beta(B, w_i^l)$, rule $r_R = B, w_i^l \to AC, w_i^j w_{j+1}^l$ probability ζ_{r_R} and inside probability of right sibling $\pi(C, w_{j+1}^l)$ as

$$\beta(r_R) = \zeta_{r_R} \pi(C, w_{j+1}^l) \beta(B, w_i^l). \qquad (31)$$

Let $k < i$ and $l > j$ be the indexes to the left and right of the span w_i^j. 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 non terminal 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.

$$\beta(A, w_i^j) = \sum_{k=1}^{i-1} \sum_{r_{A,k,i,j} \in R} \beta(r_{A,k,i,j})$$
$$+ \sum_{l=j+1}^{n} \sum_{r_{A,i,j,l} \in R} \beta(r_{A,i,j,l}). \qquad (32)$$

Now, let's look at the definition of $\mu(r_{A,i,k,j})$. Let $r_{A,i,k,j}$ be a rule rooted at A and span w_i^j. $\pi(r_{A,i,k,j})$ contains probabilities of all the trees that uses production rule $r_{A,i,k,j}$ in their derivation. $\beta(A, w_i^j)$ would contain the probability of

everything except for A spanning w_i^j. Hence, their product contains the probability of all parse trees that have rule $r_{A,i,j}$ in them, i.e. $\mu(r_{A,i,j})$.

4.2 Minimization Step

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

$$\alpha^\star = \arg\min_{\alpha} -\frac{\sum_{r \in R} \ln(\zeta_r(\alpha)) \mu(r)}{P(W)}. \qquad (33)$$

Substituting the definition of $\zeta_r(\alpha)$ and the value of $p(r; \alpha)$ from equation (21) and differentiating $\mathcal{Q}(\alpha; \alpha^{\mathbf{old}})$ w.r.t. to α, we get

$$\frac{\partial \mathcal{Q}}{\partial \alpha} = \frac{1}{P(W)} \sum_{r \in R} \left\{ \mu(r) \frac{\partial E(r; \alpha)}{\partial \alpha} \right\}. \qquad (34)$$

Using the definition of the energy function from (22), the partials $\frac{\partial E}{\partial u}$, $\frac{\partial E}{\partial W}$ and $\frac{\partial E}{\partial X}$ are

$$\frac{\partial E(r; u, W, X)}{\partial u} = g'(u^T pa) pa, \qquad (35)$$

$$\frac{\partial E(r; u, W, X)}{\partial W} = g'(u^T pa) u^T \frac{\partial pa}{\partial W}, \qquad (36)$$

and

$$\frac{\partial E(r; u, W, X)}{\partial X} = g'(u^T pa) u^T \frac{\partial pa}{\partial X}. \qquad (37)$$

The derivatives $\frac{\partial pa}{\partial W}$ and $\frac{\partial pa}{\partial X}$ can be recursively calculated as follows:

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

Recursive Definition: Let $\frac{\partial c_1}{\partial w_{ij}}$ and $\frac{\partial c_2}{\partial w_{ij}}$ be the partial derivatives of children c_1 and c_2 respectively w.r.t to scalar w_{ij} at index (i, j) in W. The derivative of parent embedding $\frac{\partial pa}{\partial w_{ij}}$ can then be computed as

$$\frac{\partial pa}{\partial w_{ij}} = f' \circ \left\{ c_j^{12} \mathbf{1}_i + W \begin{bmatrix} \frac{\partial c_1}{\partial w_{ij}} \\ \frac{\partial c_2}{\partial w_{ij}} \end{bmatrix} \right\} \qquad (38)$$

where $\circ$ is the Hadamard product and $c^{12} = \begin{bmatrix} c_1 \\ c_2 \end{bmatrix}$ and $\mathbf{1}_i$ is an indicator vector of size $d \times 1$.
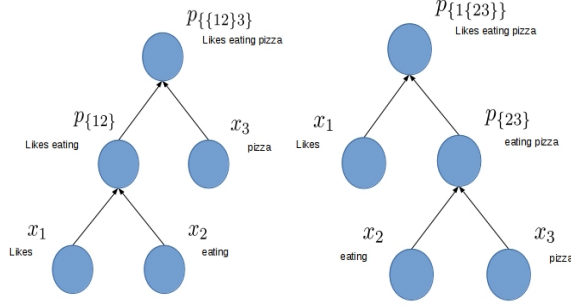
Figure 4: Two different compositional trees for w_1^3 leading to different phrasal representations.

$\partial\mathbf{pa}/\partial\mathbf{X}$: **Base Case:** Let x_i be an embedding vector in X. 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 the identity matrix $I_{d\times d}$ otherwise, it is zero.

Recursive Definition: Let $\frac{\partial c_1}{\partial x_i}$ and $\frac{\partial c_2}{\partial x_i}$ be the partial derivatives of children c_1 and c_2 respectively w.r.t. x_i. The derivative of the parent embedding pa can be built using $\frac{\partial c_1}{\partial x_i}$ and $\frac{\partial c_2}{\partial x_i}$ as

$$\frac{\partial pa}{\partial x_i} = f' \circ W \left[\begin{array}{c} \frac{\partial c_1}{\partial x_i} \\ \frac{\partial c_2}{\partial x_i} \end{array} \right]. \quad (39)$$

4.3 Phrasal Representation

One of the inherent assumptions we made while using the Inside-Outside Algorithm was that each span has a unique 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.

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 $\pi(C, w_{k+1}^j)$. As our compositional probability $p(r)$ depends upon the phrase embeddings, so multiple possible phrase representations would mean multiple values for ζ_r leading to multiple 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 upon how it's children were composed.

We restore the assumption by taking the expected value of phrasal embeddings w.r.t. its compositional inside probability $\pi(w_i^j \to w_i^k w_{k+1}^j)$ [3]

$$X(i, j) = E_\pi[X(i, k, j)]. \quad (40)$$

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

$$X(1, 3) = p_{\{1\{23\}\}}\pi(w_1^3 \to w_1 w_2^3) +$$
$$p_{\{\{12\}3\}}\pi(w_1^3 \to w_1^2 w_3). \quad (41)$$

This representation is intuitive as well. If composition structures for a phrase leads 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.

In the above three sections we have highlighted the inherent sequential tree assumption of the traditional models, proposed a compositional model and derived efficient algorithms to compute $p(W)$ and to train the parameters. In the next section we look at how to evaluate our compositional model. For this we use a recently proposed discriminative metric Contrastive Entropy (Arora and Rangarajan, 2016) which doesn't require the explicit computation of the partition function.

5 Evaluation

The most commonly used metric for benchmarking language models is perplexity. Despite its widespread use it cannot evaluate sentence level models like ours due to its word level model assumption and reliance on exact probabilities. A recently proposed discriminative metric Contrastive Entropy (Arora and Rangarajan, 2016) fits our evaluation use case perfectly. The goal of this new metric is to evaluate the ability of the model to discriminate between test sentences and their distorted version.

The Contrastive Entropy, H_C, is defined as the difference between the entropy of the test sentence

---

[3]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 non terminals
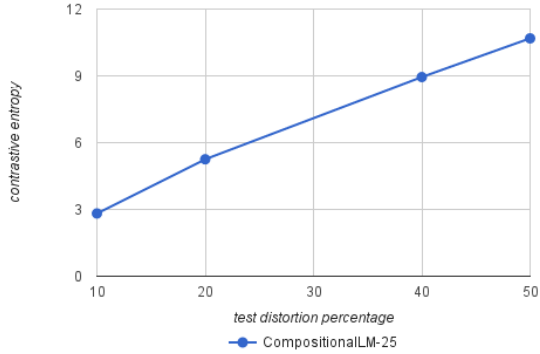
Figure 5: Contrastive entropy vs distortion levels

W_n, and the entropy of its distorted version $\hat{W}_n$ i.e.

$$H_C(D; d) = \frac{1}{N} \sum_{W_n \in D} H(\hat{W}_n; d) - H(W_n) \quad (42)$$

Here D is the test set, d is the measure of distortion and N is the number of words or sentences for word level or sentence level models respectively.

As this measure is not scale invariant we also report Contrastive Entropy Ratio H_{CR} w.r.t. a baseline distortion level d_b i.e.

$$H_{CR}(D; d_b, d) = \frac{H_C(D; d)}{H_C(D; d_b)}. \quad (43)$$

5.1 Results

We use the example dataset provided with the RNNLM toolkit (Mikolov et al., 2011b) for evaluation purposes. The dataset is split into training, testing and validation set of sizes 10000, 1000 and 1000 sentences respectively. The training set contains 3720 different words and the test set contains 206 out of vocabulary words. All reported values here are averaged over 10 runs.

Figure 5 shows the monotonic increase in contrastive entropy as the test distortion level increases. This is in line with hypothesis that the discriminative ability of a language model should increase with the test set distortion levels.

Table 1 compares our language model to standard language modeling techniques. The n-gram language models here use Kneser Ney (KN5) smoothing and were generated and evaluated using the SRILM toolkit (Stolcke, 2002). The recurrent neural network language model (RNN) has a hidden layer of size 400 and was generated using the RNNLM toolkit. The compositional language models (CLM) in Table 1 have latent space size of

Model	ppl	10%	20%	40%
3-gram KN	67.042	1.111	1.853	2.659
5-gram KN	66.641	1.107	1.846	2.652
RNN	65.361	1.322	2.231	3.227
cLM-25	-	2.818	5.252	8.945
cLM-50	-	2.833	5.441	9.179

Table 1: Contrastive entropy at distortion level 10%, 20% and 40%.

Model	20%/10%	40%/10%
3-gram KN	1.668	2.393
5-gram KN	1.667	2.395
RNN	1.688	2.441
cLM-25	1.864	3.174
cLM-50	1.921	3.240

Table 2: Contrastive entropy ratio at 20% and 40% distortion with baseline distortion of 10%.

25 and 50 and were trained using Adagrad (Duchi et al., 2011) with an initial learning rate of 1 and ℓ_2 regularization coefficient of 0.1. CLMs show more than 100% improvement over RNNLM, the best performing baseline model, across all distortion levels. Table 2 confirms that CLM outperforms all the baseline models on entropy ratio and isn't impacted by scaling issues.

6 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 non linear, better initialization, and improved 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

Kushal Arora and Anand Rangarajan. 2016. Contrastive entropy: A new evaluation metric for unnormalized language models. *arXiv preprint arXiv:1601.00248*.

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.

John Duchi, Elad Hazan, and Yoram Singer. 2011. Adaptive subgradient methods for online learning and stochastic optimization. *The Journal of Machine Learning Research*, 12:2121–2159.

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. 2011a. Extensions of recurrent neural network language model. In *Acoustics, Speech and Signal Processing (ICASSP), 2011 IEEE International Conference on*, pages 5528–5531. IEEE.

Tomas Mikolov, Stefan Kombrink, Anoop Deoras, Lukar Burget, and Jan Cernocky. 2011b. RNNLM-recurrent neural network language modeling toolkit. In *Proc. of the 2011 ASRU Workshop*, pages 196–201.

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, Christopher D Manning, and Andrew Y Ng. 2010. Learning continuous phrase representations and syntactic parsing with recursive neural networks. In *Proceedings of the NIPS-2010 Deep Learning and Unsupervised Feature Learning Workshop*, pages 1–9.

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.

Andreas Stolcke. 2002. Srilm-an extensible language modeling toolkit. In *INTERSPEECH*.