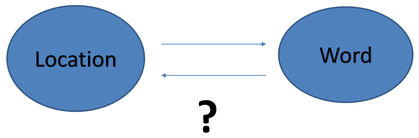
Billion Word Imputation Challenge

Aayush Mudgal [12008] Shruti Bhargava [13671]
with assistance from Prof. Amitabha Mukherjee, for the course CS365 : Artificial Intelligence Programming

Introduction

- The problem is a slight variation from the classical language modelling task. From every English sentence in the test, exactly one word (randomly) has been removed.



- Working upon the billion word benchmark corpus provided by *Chelba et al*, the task is to develop a language model that has the ability to predict the missing word and its location in the sentences from the test data. Submissions are scored using an edit distance metric.
- The problem of word imputations in words seems an interesting and challenging task which involves the application of natural language processing, machine learning and sequential data.

Related Work

- The problem is seemingly a more difficult modification of the Microsoft Research's Sentence Completion Challenge. The data in this consists of 1,040 sentences, each of which has four impostor sentences and the task is to determine from a set of five choices which word is the correct one.
- The Kaggle's challenge is different from this challenge, as herein we have to predict the missing word and its location.

Our Approach

- Our approach to the problem involves the creation of a language model, that will provide a decisive score that a given imputed sentence is generated by the learn language model. This involves modifying the following approaches

N-gram Model
(using KenLM language modelling toolkit)

Recurrent Neural Network
(using RNNLM toolkit by T. Mikolov)

- The second approach that we are trying is to develop a language model that will predict the possible word using the context information of its neighboring words.

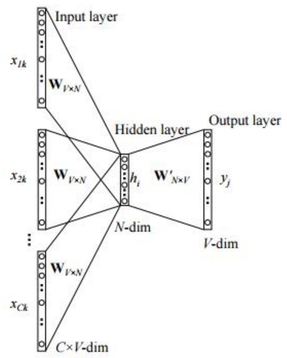
Word2Vec (CBOW)
(modifying Mikolov's word2vec code)

Preliminary Results

- After a basic modification of the CBOW approach, most of the times determiners (most frequent) words are predicted. Sentences with imputation of determiners, can be easily captured by it.
- The N-gram approach is able to capture imputations over small sentences. We plan to integrate brown clustering with this approach to yield better results from the sparse training set

Word2Vec (CBOW)

- The Continuous Bag of Words model proposed by Mikolov is similar to the feed forward NNLM where the non-linear hidden layer is removed and the projection layer is shared for all words.



Vocabulary size is V
Hidden layer size is N
Input Vector : One-hot encoded vector, i.e. only one node of $\{X_{\{1\}}, X_{\{2\}}, \ldots, X_{\{v\}}\}$ is 1 and others 0
Weights between the input layer and the output layer is represented by a VxN matrix W
$h = x^T W = v_{Wi}$
v_{Wi} is the vector representation of the input word w_i
Given a context, formed by words $x_1, x_2, \ldots, x_c$
While computing the hidden layer output, the model takes the average of the vectors of the input context words, where c is the number of words in the context, and are the word vectors corresponding to the words in the context

Source: http://arxiv.org/pdf/1411.2738v1.pdf

- **Hierarchal softmax :** Firstly a Huffman tree is build based on word frequencies. As a result each word is a leaf of that tree, and enjoys a path from the root to itself. The probability is calculated using the formula, where w is a word in the vocabulary

$$p(w|w_I) = \prod_{j=1}^{L(w)-1} \sigma([n(w,j+1) = ch(n(w,j))] \cdot v'_{n(w,j)}{}^T v_{w_I})$$

Modified approach CBOW

Training
- Billion Word Corpus : A 4 gigabyte corpus of English sentences
- Subsampling of frequent words
- Phrasal vector representation

Modified CBOW approach
- During the hierarchical softmax, the probability of a word given a context hierarchical softmax (as a binary tree in word2vec)
- Sum / Average the vectors to form the 'hidden' layer state
- The computed vector is used to compute the probabilities of nodes in the softmax tree (already implemented in the code)
- Compute product of these probabilities and output them

Imputed Sentence
- After the training, the a pseudo CBOW model is run, without backpropagation for scoring the test sentence at each position of imputation
- This will output a set of possible imputations

Scoring Metric
- Each of the possible candidate solution will be parsed with an NLP parser, to predict the best solution

N-Gram Model

- The understanding of natural languages is very complex. Words essentially combine in a non-random order. Therefore, language models can be learnt from the word and its neighbours, and exploit this structure for sentence completion.
- The n^{th} order model assumes that the probability distribution of the n_{th} word depends on only the previous n-1 words. For a 2-gram model,

$$P(w_1)P(w_2|w_1)P(w_3|w_2) \ldots\ldots P(w_{n-1}|w_n)$$

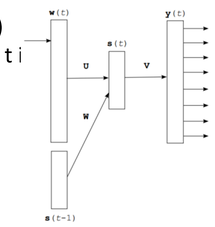
- A maximum likelihood of the conditional probabilities is given below. where the count is trained over the billion word corpus (provided as dataset)

$$P(w_2|w_1) = \frac{count(w_1, w_2)}{count(w_1)}$$

- Since the training data is very sparse, the calculated N-grams may not be accurate (since N-grams not seen in the training set would be assigned 0 probability). Modified Kneser-Ney smoothing technique is used
- Smoothing is used to generate probabilities when we have sparse statistics. The KenLM library implements two data structures for efficient language queries, namely the Probing and TRIE.
- Kneser-Ney Smoothing is used for generalization and estimation.

Recurrent Neural Network Language Modelling

- Unlike other neural network models, which require a fixed length context, recurrent neural networks can cause information to cycle inside these networks
- Input to the input layer is a concatenation of vector w representing current word and output from neurons in context layer at time t-1
- Training is standard backpropagation with stochastic gradient descent
- This has the inherent advantage over feedforward networks as having fewer parameters. Only the size of the hidden context needs to be selected in case of RNN-LM
- The network has an input layer x, hidden (context) layer s and output layer y. Input to the vector in time t i denoted as y(t) and s(t) is the hidden layer.
- Initialization of s(0) is not crucial for processing large amounts of data and can be set to a vector of small values
- Input vector x(t) represents the word at time t, encoded using 1-of-N coding and previous context layer. Size of the vector x being equal to the size of the vocabulary, and the size of the context layer being 30-500 hidden units, as mentioned in Mikolov's paper [1]
- Due to limited computational resources the model can only be trained for 1 epochs. The convergence is expected to be achieved after about 10-20 epochs [2]



Source: http://www.coling-2014.org/

References

- Mikolov, Tomas, et al. "Efficient estimation of word representations in vector space." *arXiv preprint arXiv:1301.3781* (2013).
- . Mikolov, Tomas, et al. "Distributed representations of words and phrases and their compositionality." *Advances in Neural Information Processing Systems*. 2013.
- Microsoft Sentence Completion Challenge http://research.microsoft.com/apps/pubs/default.aspx?id=157031