Critique: Efficient Estimation of Word Representations in Vector Space

Archit Sengupta: 904013900

September 2024

1 Problems Addressed

This paper highlights the primary limitations of older SOTA NLP models and proposes new architectures to tackle the challenges. The key issues include:

- Computational Efficiency: Traditional Feedforward and Recurrent Neural Networks, which often utilize non-linear hidden layers, require significant computational resources. This high cost prevents training on massive datasets, containing billions of words. The scalability issue limits their effectiveness in large-scale applications.
- Word Representation: Conventional models typically treat words as discrete units, often using one-hot vector representations. This approach fails to capture crucial relationships between words, including:
  - Semantic connections (synonyms and antonyms)
  - Contextual similarities
  - Varying degrees of relatedness

  Thus, these models struggle to capture semantic and syntactic regularities, such as the relationships between words like "king" and "queen" or "man" and "woman". The lack of representation at morphological and grammatical levels affects performance in tasks like machine translation and automatic speech recognition.
- Evaluation Complexities: Assessing the quality of word vectors generated by these models presents significant challenges. Effective word representations need to model a broad spectrum of word relationships, which older models do not capture too accurately. This results in a costly evaluation processes.
- Domain-Specific Challenges: In the field of machine translation, the limited availability of training data for numerous languages increases the problem. Just increasing model size within the current constraints fails to improve performance too much.

2 Solutions

2.1 New Ideas

The authors have presented 2 new model architectures to solve preexisting issues

1. **Continuous Bag-of-Words (CBOW) Model**:

- Similar to feedforward neural network language model, however it doesn't implement the non linear hidden layer
- Projects all words into the same position (their vectors are averaged)
- Uses words in the context, to predict the target word.
- Training complexity is reduced compared to neural network language models
- The complexity is given by

$$Q = N \times D + D \times \log_2(V)$$

2. **Continuous Skip-gram Model**:

   - Similar to CBOW, but instead of predicting the current word based on context, it does the opposite. It tries to predict the context words using the target words.
   - Uses the current word as input to predict words within a certain range before and after it
   - More distant words from the target are given less weight since they are sampled less
   - The complexity is proportional to:

$$Q = C \times (D + D \times \log_2(V))$$

     C is max distance of words from target word

These models are considerably better than earlier ones.

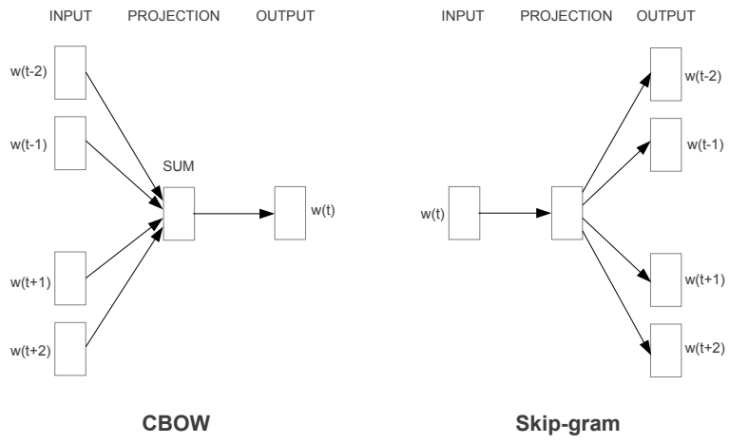

Figure 1: Architecture

2.2 Strengths

- **Efficiency, Scalability and Reduced Computational Complexity:** Both models significantly reduce computational complexity compared to traditional neural network language models. This reduction allows for training on much larger datasets, potentially billions of words, with a vocabulary of up to 1 million words

- **Innovative Word Vector Representations:** The Word2Vec approach presented in the paper learns to predict words appearing in similar contexts, automatically aligning words through syntax and semantics. This results in a vector space that embeds linguistic relationships, revealed through vector arithmetic operations. For example, the models can capture analogies like "$king - man + woman \approx queen$" and syntactic transformations like "$walking - walked + swimming \approx swam$". These relationships constitute a crucial advantage of the Word2Vec model.

- **Parallel Training** The models were implemented on a large-scale distributed framework called DistBelief, allowing for parallel training using multiple model replicas. This approach enables efficient training on massive datasets

- **Performance and Applications** The new models show better performance in almost all experiments done by the authors. Examples:

  - The Microsoft Sentence Completion Challenge: There were 1,040 sentences with one missing word, and a choice needed to be made from five options. Past methods, including N-grams and recurrent neural networks, got a result of 54% accuracy. The authors evaluated the performance of a 640-dimensional model using the Skip-gram architecture, trained on a dataset of 50 million words. Although this model on its own did not surpass the performance of Latent Semantic Analysis (LSA), its results complemented those obtained from Recurrent Neural Network Language Models (RNNLMs). By combining the scores from both models in a weighted manner, they achieved a new SOTA accuracy of 58.9% and 59.2%.
  - The authors contrast the CBOW model against a test set of Semantic-Syntactic Relationships. Accuracy was measured in terms of exact matches; synonyms were considered errors since perfect accuracy is unattainable due to the limited amount of information. The accuracy was better than most older architectures.

  Performance improvements in various tests and challenges have been shown in the following section

3 Results

3.1 Semantic-Syntactic Word Relationship Test

Dimensionality / Training words	24M	49M	98M	196M	391M	783M
50	13.4	15.7	18.6	19.1	22.5	23.2
100	19.4	23.1	27.8	28.7	33.4	32.2
300	23.2	29.2	35.3	38.6	43.7	45.9
600	24.0	30.1	36.5	40.8	46.6	50.4

Figure 2: CBOW : Semantic-Syntactic Word Relationship test subset accuracy

Model	Vector Dimensionality	Training words	Accuracy [%]		
			Semantic	Syntactic	Total
Collobert-Weston NNLM	50	660M	9.3	12.3	11.0
Turian NNLM	50	37M	1.4	2.6	2.1
Turian NNLM	200	37M	1.4	2.2	1.8
Mnih NNLM	50	37M	1.8	9.1	5.8
Mnih NNLM	100	37M	3.3	13.2	8.8
Mikolov RNNLM	80	320M	4.9	18.4	12.7
Mikolov RNNLM	640	320M	8.6	36.5	24.6
Huang NNLM	50	990M	13.3	11.6	12.3
Our NNLM	20	6B	12.9	26.4	20.3
Our NNLM	50	6B	27.9	55.8	43.2
Our NNLM	100	6B	34.2	**64.5**	50.8
CBOW	300	783M	15.5	53.1	36.1
Skip-gram	300	783M	**50.0**	55.9	**53.3**

Figure 3: results of publicly available vectors on the test set compared

3.2 Training with different Epochs

In the following experiment, the authors conclude that training the models with 1 epoch is more efficient than using 3 epochs

Model	Vector Dimensionality	Training words	Accuracy [%]			Training time [days]
			Semantic	Syntactic	Total	
3 epoch CBOW	300	783M	15.5	53.1	36.1	1
3 epoch Skip-gram	300	783M	50.0	55.9	53.3	3
1 epoch CBOW	300	783M	13.8	49.9	33.6	0.3
1 epoch CBOW	300	1.6B	16.1	52.6	36.1	0.6
1 epoch CBOW	600	783M	15.4	53.3	36.2	0.7
1 epoch Skip-gram	300	783M	45.6	52.2	49.2	1
1 epoch Skip-gram	300	1.6B	52.2	55.1	53.8	2
1 epoch Skip-gram	600	783M	56.7	54.5	55.5	2.5

Figure 4: Epoch comparison

3.3 Google News Dataset

The following shows implementation in DistBelief.The performance of several models trained on the Google News dataset, which contains 6 billion words, was evaluated using a method called mini-batch asynchronous gradient descent. This approach also utilized an adaptive learning rate technique known as Adagrad.

Model	Vector Dimensionality	Training words	Accuracy [%]			Training time [days x CPU cores]
			Semantic	Syntactic	Total	
NNLM	100	6B	34.2	64.5	50.8	14 x 180
CBOW	1000	6B	57.3	68.9	63.7	2 x 140
Skip-gram	1000	6B	66.1	65.1	65.6	2.5 x 125

Architecture	Accuracy [%]
4-gram [32]	39
Average LSA similarity [32]	49
Log-bilinear model [24]	54.8
RNNLMs [19]	55.4
Skip-gram	48.0
Skip-gram + RNNLMs	**58.9**

Figure 5: Google News Test

4 Weaknesses

- Limited evaluation on downstream NLP tasks: The paper focuses on word analogy and similarity tasks to evaluate the word embeddings, but does not thoroughly evaluate performance on downstream NLP applications. They could've evaluated the word embeddings on a wider range of NLP tasks such as sentiment analysis, named entity recognition, machine translation, etc.

- Lack of Contextual embeddings: The paper deals with static embeddings. So the words are assumed to mean the same even in different use cases. Embeddings like ELMo and BERT use contextual embeddings to solve this.

- If a word doesn't show up in the training corpus, the model won't be able to deduce its representation. Newer models, such as FastText, can handle this through the use of subword information.

- Skip-gram's classification model even though better than earlier Neural networks based systems, is still computationally expensive in large vocabulary setups. Since the objective function is proportional to the size of the vocabulary. This can be improved by sampling techniques discussed in future studies.

- Interpretability: The vector representations are extremely hard to interpret since its just a set of values. What each dimension represents is not clear.

Critique: Distributed Representations of Words and Phrases and their Compositionality

Archit Sengupta: 904013900

September 2024

1 Problems Addressed

The paper tackles several interesting challenges in the field of NLP, particularly in how computers understand and represent words and phrases They reference the Skip-gram model, an efficient, effective way to learn vector representations from text data. This was discussed in the earlier paper. These representations capture precise syntactic and semantic relationships between words, making them useful for various tasks. They tackle issues like:

- Computational complexity with large vocabularies in Skipgram
- Sub-optimal quality in the representations
- Lack of depth in word and phrase relationships

They have decided to implement several extensions to Skip-gram and improve the system through novel techniques. They aim to reduce computational complexity as well as improve the effectiveness of the model through new methodologies like Negative Sampling, Subsampling frequent words and phrase representation.

2 Prior work references

2.1 Original Skip-Gram model

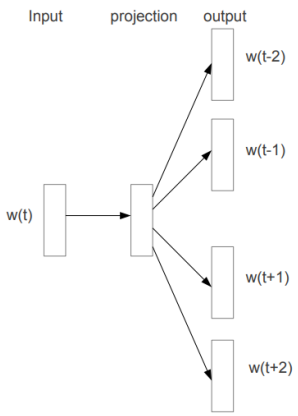

Figure 1: Model Architecture

The objective of Skipgram is to maximize the log probability

$$\frac{1}{T}\sum_{t=1}^{T}\sum_{-c\leq j\leq cj\neq 0}\log p(w_{t+j}|w_t)$$

where the softmax is

$$p(w_O|w_I)=\frac{\exp\left(\mathbf{v'}_{w_O}^{\top}\mathbf{v}_{w_I}\right)}{\sum_{w=1}^{W}\exp\left(\mathbf{v'}_{w}^{\top}\mathbf{v}_{w_I}\right)}$$

But as we can see this is computationally very expensive. The denominator calculates the expectation across the entire vocabulary which can consist of millions of words. The complexity is proportional to W($10^5 - 10^7$).

2.2 Hierarchical Softmax

Hierarchical softmax is an approximation of the softmax function which is computationally cheaper, where words are organized in a binary tree. The model doesn't calculate probabilities for all words in the vocabulary, instead it traverses a path from the root to a word in the tree. The probability is

$$p(w|w_I)=\prod_{j=1}^{L(w)-1}\sigma\left([[n(w,j+1)=ch(n(w,j))]]\cdot\mathbf{v'}_{n(w,j)}^{\top}\mathbf{v}_{w_I}\right)$$

Since only $log(W)$ words are evaluated it makes it much faster. However the performance of this model purely by itself was shown to be poorer than what the authors designed, as we will discuss later.

3 New Ideas and Strengths

Here we will discuss the new ideas and strengths as presented by the paper.

1. **Improving the quality and efficiency of word vector representations**: The researchers aim to enhance the Skip-gram model and add several extensions to it. They introduce techniques to improve both the quality of the vectors and the training speed.

2. **Phrase Recognition** The researchers recognized that many phrases, like "New York Times," mean more than just the sum of their individual words. They developed techniques to capture these unique meanings.

3. **Dealing with common words** Words like "the" and "a" appear so frequently that they can overshadow less common but more meaningful words. The team developed a clever way to balance this out.

3.1 Noise Contrast Estimation and Negative Sampling

Noise Contrastive Estimation (NCE) is a method used to approximate the softmax function by transforming the probability estimation problem into a binary classification task. The authors of the paper extend this idea for their use case. Skipgram is only requires learning high quality word vectors, so they can simplify NCE with negative sampling (NEG). They give the objective function:

$$\log \sigma \left(\mathbf{v'}_{w_O}^{\top} \mathbf{v}_{w_I} \right) + \sum_{i=1}^{k} E_{w_i \sim P_n(w)} \left[\log \sigma \left(-\mathbf{v'}_{w_i}^{\top} \mathbf{v}_{w_I} \right) \right]$$

where k datapoints are randomly sampled. The intention is that words in the context should be closer to each other. So sampling random elements would place them much farther away in vector space (Since they would be dissimilar), hence improving the quality of the vector. Thus NEG does not need the noise distribution as needed by NCE.

3.2 The Mystical choice

The authors explored various options for $P_n(w)$ and discovered that the unigram distribution $U(w)$ raised to the power of 3/4 (i.e., $U(w)^{3/4}/Z$) always outperformed the unigram as well as uniform distributions across all tasks (including NCE and NEG0 particularly in language modeling. Why 3/4 works well is not clear , even to this day. However it seems to work better than most other factors.

3.3 Subsampling frequent words

While training there is always an imbalance between frequent and rare words in large sets. To solve this, a subsampling technique was applied. Each word w_i was discarded with a probability given by:

$$P(w_i) = 1 - \sqrt{\frac{t}{f(w_i)}}$$

where $f(w_i)$ is the frequency of the word and t is a chosen threshold. This approach helps reduce the impact of very frequent words, speeding up training and improving rare word representations.

3.4 Learning Phrases

The authors decided to generate vector representations for phrases to better reflect their weightage. In order to do this they picked words which occur together frequently, but less frequently in different contexts. These were identified as unique tokens. Eg. “New York Times” is replaced by its own unique token in the training data, while “this is” remains unchanged.

Newspapers			
New York	New York Times	Baltimore	Baltimore Sun
San Jose	San Jose Mercury News	Cincinnati	Cincinnati Enquirer
NHL Teams			
Boston	Boston Bruins	Montreal	Montreal Canadiens
Phoenix	Phoenix Coyotes	Nashville	Nashville Predators
NBA Teams			
Detroit	Detroit Pistons	Toronto	Toronto Raptors
Oakland	Golden State Warriors	Memphis	Memphis Grizzlies
Airlines			
Austria	Austrian Airlines	Spain	Spainair
Belgium	Brussels Airlines	Greece	Aegean Airlines
Company executives			
Steve Ballmer	Microsoft	Larry Page	Google
Samuel J. Palmisano	IBM	Werner Vogels	Amazon

Figure 2: Examples for the phrases

This way, phrases could be found without increasing the vocabulary size by too much.

4 Results

- Setting the vector dimensionality to 300, and the context size as 5, yielded good performance. They compared the Negative Sampling (NEG) and Hierarchical Softmax (HS) methods, through subsampling as well as without it. They showed that Negative Sampling achieves good accuracy with $k = 5$, but using $k = 15$ leads to even better performance. While the Hierarchical Softmax performed worse without subsampling, it became the best method when frequent words were downsampled. This suggests that subsampling can enhance training speed and accuracy in certain scenarios.

Table 1: Accuracies of the Skip-gram models

Method	Dimensionality	No Subsampling [%]	10^{-5} Subsampling [%]
NEG-5	300	24	27
NEG-15	300	27	42
HS-Huffman	300	19	47

- For the phrase vectors task, the training dataset was increased to around 33 billion words, using hierarchical softmax and entire sentences for context, which achieved an accuracy of 72%. Reducing the dataset to 6 billion words dropped the accuracy to 66%, demonstrating that a larger dataset is important. The best representations were learned using hierarchical softmax with subsampling.

- Additive Factor: Skip-gram model learns vector representations in a linear structure, so there is effective reasoning through vector arithmetic. These representations can be combined by adding their vectors. This additive property comes from the training objective, where word vectors predict surrounding words, representing the context in

which a word appears. The relationship between the vectors and the output probabilities means that the sum of two vectors is related to the product of their context distributions. Eg. if "Volga River" appears with "Russian" and "river" frequently, the combined vector will be similar to the vector representing "Volga River."

5 Weaknesses

- Lack of Theoretical Depth: There seems to be a lack of theoretical depth in the paper. Some things have just been stated as fact, while deeper analysis of why certain methodologies are better haven't been discussed. For example The exponential of 3/4 as discussed before has been shown to show superior results, however the actual analysis of why this is so hasn't been discussed.

- Simplistic phrase detection: The method used for identifying phrases is relatively simple, based only on frequency statistics. This may miss important phrases that are less frequent but semantically significant. Dependency parsing or named entity recognition techniques could've been utilized to improve phrase detection.

- Potential bias in word and phrase representations: The paper doesn't address potential biases in the learned representations, which could be problematic when these vectors are used in downstream applications. The learned representations should be analyzed for social biases and debiasing techniques should be explored.