# Predicting Words from their Description

**Troy O'Neal**
Stanford University
CS224D
SUNET: toneal

## Abstract

We attempt to predict words from phrases that describe them. This task is motivated by automatic summarization and the various work on matching descriptions to images, as well as intrinsic curiosity of the author. Since this appears to be a novel task without an existing state-of-the-art, we explore neural network models ranging from the extremely basic up through models of medium complexity in an attempt to glean the difficulty of the problem. Models were trained on a mixture of Princeton Wordnet and Webster's Unabridged Dictionary. An attempt was made to compensate for the relatively small number of training examples per class by leveraging the similarity metric built in to GloVe pretrained vectors. Over the models tested, the best performance was obtained by averaging a phrase's distributed word vectors and feeding that into one-hidden-layer neural network. This trained network achieved a test accuracy of 28% on a vocabulary size of 100, and an accuracy of 2.9% on a vocabulary size of 4000. While low in an absolute sense, these results greatly outperform random guessing and a simple cosine similarity model.

## 1 Introduction

Words can be powerful entities. For example, consider the word "squirm", or "skyrocket." Any given word is capable of succinctly encoding layers of meaning, emotion, and connotation. In NLP today, it is accepted practice to encode words as numerical vectors. However, when we humans wish to explicitly communicate representations of words, we use descriptions, of which dictionary definitions are a special case. Very little existing work focuses on the nature of the relationship between words and their "human summaries": descriptions. This project aimed to explore this relationship. We can divide the relationship into two directions. The first is generating a description given a word, and the second is predicting a word given its description. This project focuses on the latter task, which should be much easier. For example, given the statement "a small to medium-sized primate that typically has a long tail," our system should output a probability distribution with a high value for "monkey".


Figure 1. The problem under study

# 2 Background

This task as proposed appears to be novel, but has significant relationships to existing tasks in NLP and machine learning.

## 2.1 Automatic Summarization

The problem of identifying words from their definition is similar in a sense to automatic summarization, in that we are taking a group of words and mapping them to a smaller group of words. Kageback et al. (2014) discuss automatic summarization and were one of the first to apply distributed word vector representations and deep learning to the problem. They used Word2Vec vectors, recursive neural nets, and cosine similarity to measure similarity between sentences. This provided the inspiration to use cosine similarity for zero-shot learning. In this project, techniques from zero and one-shot learning are applicable since we have so few examples for each class.

## 2.2 Zero and One-Shot Learning

Socher et al. (2013) discusses an approach to zero-shot learning of image categories. They use a separate "novelty" variable to switch between classifiers for seen and unseen classes. To determine whether an example is seen or unseen, an outlier detection function is applied based on a Gaussian prior. In this project, the zero-shot model (fuzzy) did not use separate classifiers for seen versus unseen, rather, a single classifier attempted to a predict a "fuzzy" distribution over the vocabulary based on similarity in the GloVe vector space. It was hoped that the similarity relationships encoded in GloVe were enough to enable generalization of the model. Fei-Fei et al. (2006) discuss one-shot learning as applied to image classification. Their approach involves learning new categories based on prior distribution of model parameters learned from old categories. Although much of their analysis is image-specific, the rough analogue of their prior distribution is the Glove pretrained distributed vector representation. Romera-Paredes and Torr (2015) provide a summary of zero-shot learning and provide an approach that learns both unseen classes and the mapping between seen and unseen classes. Applying their approach of learning the mapping between seen and unseen might have yielded a better similarity metric than the one used.

## 2.3 Long Short-Term Memories

Hochreiter and Schmidhuber (1997) first described Long Short-Term Memories (LSTMs). LSTMs are used to mitigate the vanishing and exploding gradient problem when training long recurrent neural networks. Since definitions could reach 20 words or more, an LSTM model was thought to be appropriate.

# 3 Approach

Before any of the trainable models were conceived, a simple non-trainable baseline of cosine similarity between average GloVe vector and the entire vocabulary was run. This yielded an accuracy of just 0.06%. This extremely low number provided an initial baseline and suggested that the task was hard. More models were then set up for the main experiments.

There were two overall experiments done. The first was to train a softmax classifier based on various underlying neural network architectures using test words that were seen in training. The second was to explore a classifier that could handle test words that were unseen in training.

For the first experiment, named *classical*, there were five primary architectures tested. These models mostly serve to explore and see how hard the problem is. Each of these architectures accept a definition as input and produce a softmax probability distribution over the vocabulary. The first two architectures are toy structures, meant to be used as a trivial baseline. The third architecture consists of a simple linear projection. The fourth architecture consists of a one-hidden-layer neural network. The fifth and final architecture uses LSTMs. The details of the various architectures are now provided.

## 3.1 Models

The first architecture, named *random*, outputs a uniform probability distribution over the vocabulary. This model served as a trivial baseline.

$$\hat{y} = \left[\frac{1}{V}, \frac{1}{V}, \frac{1}{V}, \dots\right]$$

The second architecture, named *toy-bias*, consists of a simple trainable bias vector. This served to find the prior distribution of the outputs. Where $\theta$ represents all trainable parameters,

$$\theta = \{b\}$$

$$\hat{y} = softmax(b)$$

The third architecture, named *glove-mean-linear*, consists of an average of the input Glove vectors fed through a single linear projection layer. Where $N$ is the number of words in the input, and $u_i$ is the Glove word vector at word index $i$, and $m$ is the mean of the Glove vectors.

$$m = \frac{1}{N}\sum_i u_i$$

$$\theta = \{W, b\}$$

$$\hat{y} = softmax(mW + b)$$

The fourth architecture, named *glove-mean-deep*, consists of an average of Glove vectors fed into a one-hidden-layer network.

$$\theta = \{W_1, b_1, W_2, b_2\}$$

$$h = ReLU(mW_1 + b_1)$$

$$\hat{y} = softmax(hW_2 + b_2)$$

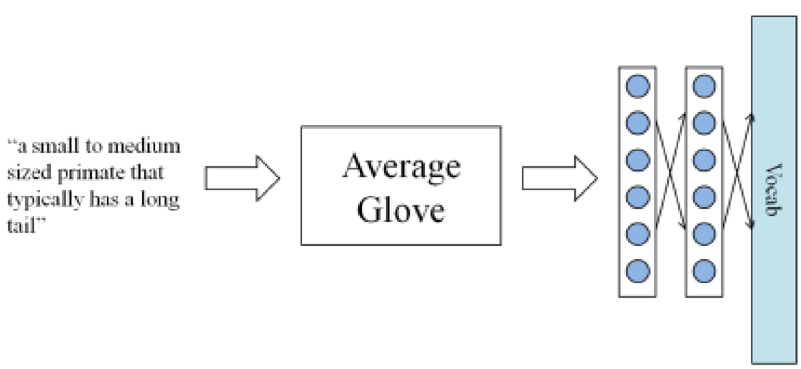



Figure 2. *glove-mean-deep* architecture

The fifth and final architecture, named *lstm*, consists of a recurrent LSTM model which accepted a sequence of Glove words from the input. A single linear projection layer is then applied to the LSTM cell's output, where the output is taken from the cell corresponding to the last word of the input. For a input definition of length $N$ (and $h_i$ is the output of LSTM cell $i$) this architecture computes

$$\theta = \{\theta_{LSTM}, W, b\}$$

$$\hat{y} = softmax(h_N W + b)$$

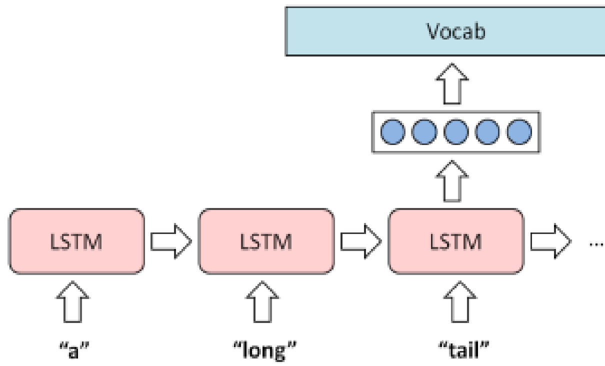



Figure 3. *lstm* architecture

### 3.2 Loss

For the preceding five architectures, loss is standard cross entropy loss between the correct word (the one-hot vector $y$), and the prediction $\hat{y}$.

$$J = -\sum_i y_i \log(\hat{y}_i)$$

The second experiment, named *fuzzy*, attempted to learn unseen words using a non-standard model. When learning from zero, one, or very few examples per class, a standard softmax-based classifier will have poor generalization performance. However, it should still be possible to learn in this situation. For example, we should be able to learn definitions for "gorilla" if we have learned a definition for "monkey" and we know that "monkey" is similar to "gorilla". In our case, GloVe encodes similarity information between words. To implement this, we use techniques from one-shot and zero-shot learning. The basic idea is to create conceptual supercategories, such as in (Salakhudinov 2012), each of which has output classes as (potentially fuzzy) members. To evaluate whether an input example belongs to a class, we evaluate the example's membership in the supercategory and the class's membership in the supercategory and make a prediction from that. This requires some sort of similarity metric between classes (words) and supercategories (which we use here loosely to refer to regions in the distributed word vector space). In this project, model *fuzzy* used a cosine similarity metric between a target class (word) and the rest of the distributed word vector space. This meant that definitions were trained to match to not only the target word, but all words close to the target. Since in this model $\hat{y}$ is no longer a mutually exclusive probability distribution, squared loss was used instead of CE loss. When $y_{vec}$ is the distributed (not one-hot) representation of the correct word, we compute

$$s = \frac{y_{vec} V^T}{\|y_{vec}\|} \circ \frac{1}{\|V\|}$$

$$J_{fuzzy} = (s - \hat{y})^2$$

To generate $\hat{y}$, in theory any of the preceding five architectures could be used, however the one-hidden-layer neural network was selected to evaluate *fuzzy*.

## 4 Experiment

### 4.1 Dataset

For both *classical* and *fuzzy*, the primary dataset is a aggregation of two dictionaries, the Webster's Unabridged Dictionary (early 1900s) from Project Gutenberg [7], and the Princeton Wordnet dictionary (2006) [8]. Synonyms were treated by aggregating all definitions under the same word key. Wordnet has 60k words and 102k distinct definitions. Websters Unabridged has 95k words and 244k distinct definitions. To reduce computational cost, the top 10k most common words (as determined by GloVe) were intersected with the words in each dictionary. This preprocessing led to a final combined vocabulary of 4411 words, for which there are 66930 distinct definitions. This meant that there were 15.2 training examples, on average, per word. Each training example consists of a (definition, word) tuple. Pretrained word vectors, used as fixed distributed representations of the definitions, are GloVe 6B, 100 dimensional (from Socher et. al. [9]).

### 4.2 Training and test split

To generate the training data, 80% of the aggregate dataset was randomly sampled. Validation and test data was randomly sampled as the remaining 10% and 10% respectively. For the limited-vocabulary instances, the aggregate dataset was filtered by a set of $|V|$ randomly sampled words before the train/test split. The same dataset split was used for both *classical* and *fuzzy*. If *fuzzy* was successful on this split, we planned to create a train/test split that included unseen words in the test set, a much tougher learning problem.

### 4.3 Hyperparameters

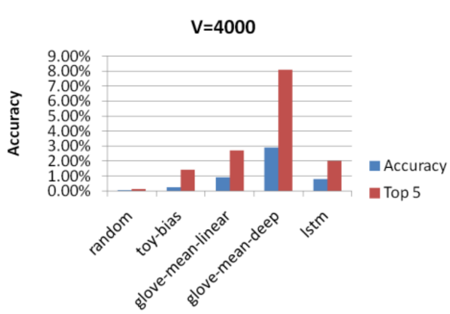
For all models, a fixed learning rate of 0.01 was used, a dropout rate of 0.5, and a L2 regularization constant of 0.001. For the one-hidden-layer network, 100 hidden units were used. These values were selected based on informal experimentation, and were not optimized in any formal way. Training was performed using standard gradient descent, and continued for each model until the author noticed a qualitative plateau in validation accuracy over multiple epochs.

### 4.4 Evaluation

Accuracy of the trained models on the test set was measured. F1 score could have been used, but was not due to this project mainly being exploratory in nature. The lack of F1 can be justified by noting that there is a baseline model (*toy-bias*) that accounts for learning the prior output distribution.

## 5 Results

### 5.1 classical



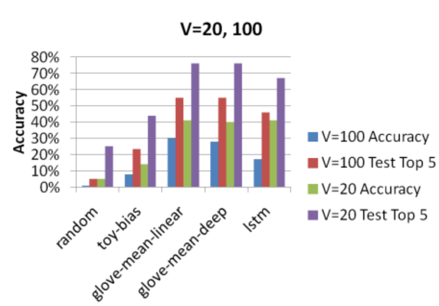



Figure 5. Primary results.

The results obtained for the full dataset ($V = 4000$) demonstrate accuracy significantly greater than random, especially for *glove-mean-deep*. It is noted that *toy-bias* also performed significantly better than random, which is indicative of a skewed training distribution (i.e., some words contain many more definitions than others, and are thus overrepresented in the training data). However, we also note that *glove-mean-deep* significantly outperforms *toy-bias*, indicating that learning is indeed taking place. The success of *glove-mean-deep* over *glove-mean-linear* implies that the decision boundaries within the distributed word vector space are complex and nonlinear. The failure of *lstm* was surprising, since due to its complexity, it was initially hypothesized to be the model that could most accurately capture the meaning of the input. While training *lstm*, we noted its ability to overfit more strongly than all the other models. However, increased regularization and dropout did not help *lstm* validation performance. The reason for the failure of *lstm* is suspected to be the LSTM's need for an amount of training examples far larger than used here.

### 5.2 fuzzy

| | *fuzzy* | Best *classical* accuracy |
|---|---|---|
| V=4000 | 0.09 | 2.9 |
| V=100 | 21 | 30 |
| V=20 | 41 | 41 |

Figure 6. *fuzzy* results and comparison to *classical*.

The results for *fuzzy* were not as high as hoped, even with the original training split. Although the accuracy is comparable to *classical* for limited vocabulary sizes (and, it should be noted, better than *random* and *toy-bias*), performance on the full vocabulary ($V = 4000$) was abysmal. The additional complexity of trying to learn a non-mutually exclusive distribution appears to have introduced too much noise into the system to be of use, although the precise reason for failure warrants further investigation.

## 6 Conclusion

The problem of predicting words from their descriptions appears to be, at the very least, non-trivial, although we venture that it is fairly difficult. The best "first-pass" approaches did not yield satisfying accuracy, although they did significantly outperform guessing. Inherent difficulties include handling synonyms, which introduce complex boundaries over the input space. In addition, even the best-written definitions cannot possibly encode all relevant connotative and contextual word information.

### 6.1 Future Work

To improve the results obtained here, one should obtain more definitions per word, and thus more training examples, by utilizing commercial dictionaries. This should allow for the successful training of complex models like LSTM. Other models might also be tried, such as recurrent neural networks on a parse tree of the definition, all the way up through extremely new models like dynamic memory networks. Such complex models should be able to better learn the meaning of the input phrases, especially when given many more examples per word.

The simplistic scheme used in this project of cosine similarity combined with squared loss did not yield good zero-shot generalization performance. Different similarity metrics and loss functions should be investigated to continue the goal of learning about multiple words from a single definition.

## References


[1] Kageback et al. *Extractive Summarization using Continuous Vector Space Models.* Proceedings of the 2nd Workshop on Continuous Vector Space Models and their Compositionality. 2014.

[2] Socher et al. *Zero-Shot Learning Through Cross-Modal Transfer.* NIPS. 2013.

[3] Fei-Fei et al. *One-Shot Learning of Object Categories.* IEEE Transactions on Pattern Analysis and Machine Intelligence. 2006.

[4] Romera-Paredes and Torr. *An embarrassingly simple approach to zero-shot learning*. Proceedings of the 32nd International Conference on Machine Learning. 2015.

[5] Hochreiter and Schmidhuber. *Long Short-Term Memory.* Neural Computation. 1997.

[6] Salakhutdinov, Tenenbaum, and Torralba. *One-Shot Learning with a Hierarchical Nonparametric Bayesian Model.* JMLR: Workshop and Conference Proceedings. 2012.

[7] *Webster's Unabridged Dictionary*. Project Gutenberg. 1996. <http://www.gutenberg.org/cache/epub/673/pg673.txt>

[8] Princeton University "About WordNet." WordNet. Princeton University. 2010. <http://wordnet.princeton.edu>

[9] Pennington, Socher, and Manning. *GloVe: Global Vectors for Word Representation.* 2014.