Scholastic Aptitude of Neural Networks:
Improving the performance of word embeddings on analogy questions

Felipe Campos **Robert Foster** **Zachary Ingbretsen**

Abstract

Understanding the relationships between abstract concepts is a valuable skill for both humans and computers. A somewhat surprising property of word embeddings trained by CBOW or skip-gram neural network models is that linear relationships form between words. We are putting word embeddings trained on different corpora to the test. First, we show how Word2vec and GloVe embeddings perform on a standardized set of analogy questions. Then, we explore methods of improving the performance of a subset of these word embeddings, including using ELMo representations of words and a deep neural network to predict the correct answer to analogy questions.

1 Introduction

This paper studies different approaches to computationally process analogies. A good understanding of analogies is crucial for better understanding the meaning and usage of words in human-written texts and may be generalizable to other areas of AI development. First, we use pre-trained word embeddings (Word2vec and GloVe) to establish a baseline of how word embeddings perform at this problem. We establish performance differences based on the training corpus, number of dimensions, and distance metric used. Given that understanding, we attempt to improve on the performance of the embeddings by analyzing their shortcomings and augmenting them. Specifically, we use ELMo representations of our words to improve on the GloVe embeddings, and we train a deep neural net on analogy questions.

2 Background

The analogy task from Turney, et. al. has been subjected to various approaches since the early 2000's. There are many approaches that exceed the baseline performance of random guessing, however no automated approaches excel at this task. Somewhat consolingly, humans also do not excel at the task; The average US college applicant only scores a 57% on this particular test (Turney and Littman, 2005). The current state of the art on this task is a hybrid ConceptNet implementation which scores a 56.1%, approaching the performance of the average college applicant (Speer et. al., 2017).

2.1 Challenges

Understanding analogies requires identifying the correct meaning of individual words and the relationships between pairs of words. Correctly answering a seemingly simple analogy could involve first disambiguating homonyms, and then identifying complex/non-linear relationships between words, among other challenges. Our work tries to address the following problems.

2.2 One-to-Many Relationships

A relationship such as capital-to-country has a straightforward interpretation. Italy has only one capital, and that capital is Rome. Such a one-to-one relationship can be applied to all countries in order to find the capital. However, there is no such relationship between any arbitrary city and its country. Naples is a city in Italy, but the city's relationship in high-dimensional space does not occupy a privileged position, as does the capital city. For this reason, a linear transformation may not be effective at correctly answering this type of analogy. One solution idea to this problem was to use a web search to find the words (Bollegala et. al., 2008). Overall this was only moderately effective.

2.3 Polysemy

Having just one embedding per word can be problematic if a word has more than one meaning.

Over the course of training the embedding, all meanings of the word will contribute to the final embedding. This will reduce the usefulness of the embedding for all senses of the word.

2.4 Out of Vocabulary Words

Word2vec and GloVe are both trained on a large corpus, and filter out words that do not pass a certain count threshold. If a word in the test set was not found in the original corpus or did not surpass the count threshold, it will be impossible to predict an embedding vector for that word.

3 Methods

Given the challenges inherent in this task, we first benchmarked the performance of pre-trained Word2vec and GloVe vectors. In order to get a direct comparison of Word2vec and GloVe on this task, we then trained our own embeddings on a full dump of English Wikipedia. We used these benchmarks to identify strong and weak aspects of the different embeddings.

Using this knowledge, we decided that adding some context to our vectors would likely help with the performance on the task. Using pre-trained vectors, one can further train on a new corpus, and derive new vectors for those words in context, as is done with CoVe or ELMo representations (McCann, et. al., 2017; Peters, et al., 2018). We modified our embeddings via ELMo, as well as trained a neural network to use the word embeddings to capture non-linear relationships in the analogies.

3.1 Embedding Evaluation

As noted by Mikolov et. al., word embeddings can encode relational similarities in their vector representations and, when well-trained, analogies in the form "*king* is to *man* as *queen* is to ..." can be answered with some simple arithmetic. Therefore, in a word analogy of form a:b::c:d, where (a,b,c,d) are words, it is expected that the vector resulting from the operation d - c would share some similarity with the vector resulting from b - a. Therefore, we could answer an analogy question such as a:b::c:? by calculating:

$$argmax(sim(d, b - a + c))$$

for every dV, where *sim* can be any similarity function such as cosine similarity. Other works usually refer to the equation by the name *3CosAdd* (Levy and Goldberg, 2014). In this paper, we use both cosine similarity and Euclidean distance functions for experimentation purposes.

We also evaluate the performance of word embeddings when used in a neural network trained and focused on solving analogy questions, based on a deep neural network model used for visual analogy-making in a previous work (Reed et. al., 2015).

3.2 Datasets

An evaluation dataset of 373 SAT questions was obtained from Peter Turney. Each analogy question has four or five choices as well as part of speech tagging. We start with benchmarks for the dataset as a whole, but looked at accuracies for part of speech and question source as well, using the algorithms below.

Common pre-trained word embeddings (GloVe 6B, 42B, and 840B, and Google News word2vec) were downloaded from different sources, and we converted them into the standard Word2vec format to standardize the model scoring interface.

For training the analogy-focused neural network models, we have used the Google Analogy Dataset as a smaller training sample (with approximately 20,000 analogies) and the Bigger Analogy Test Set as a more complete training sample (with more than 2,000,000 analogy questions). The trained neural network was then benchmarked against the SAT dataset.

3.3 Code

The code is written in Bash and Python scripts and is available at: *https://github.com/zingbretsenb/w266_final*
Please email *zingbretsen@berkeley.edu* if you do not have access to the repository.

4 Results

4.1 Baseline Algorithms

We used pre-trained word embeddings (word2vec and GloVe trained on different sources) as a baseline using both Euclidean and cosine distance measurements. The summary of the baseline models of each are in Table 1 below.

As a side note, using the GloVe embeddings trained on 42 billion tokens from Twitter, the accuracies were 26% and 22% for cosine and Euclidean distance, respectively. While more words generally means better word embeddings, the content of the corpus is important, and the authors do

Embeddings	Dims	Euclidean	Cosine
Twitter.42B	200	21.72%	26.01%
GloVe.6B	50	31.90%	35.66%
GloVe.6B	100	34.32%	37.27%
GloVe.6B	200	34.05%	40.75%
GloVe.6B	300	34.05%	41.55%
GloVe.42B	300	37.80%	44.50%
GloVe.840B	300	36.19%	49.06%
Word2vec	300	31.90%	42.36%

Table 1: Baseline accuracy of embeddings with different distance metrics.

Embedding	Dims	Diff. PoS	Same PoS
Twitter.42B	200	31.8%	34.0%
Glove.6B	50	35.6%	36.6%
Glove.6B	100	32.6%	40.9%
Glove.6B	200	35.6%	44.8%
Glove.6B	300	37.0%	45.3%
Glove.42B	300	41.0%	46.8%
Glove.840B	300	43.2%	52.8%
Word2vec	300	40.7%	44.8%

Table 2: Accuracy of embeddings when the two words in the analogy were the same or different part of speech.

not recommend using Twitter to train future SAT models.

From this baseline, we decided to do a direct comparison between corpora, embeddings and distance metrics. Since the pre-trained Word2vec and GloVe vectors were all trained on different corpora, we decided to train both algorithms on the same dataset: the latest full dump from Wikipedia. This included all text that could be parsed from the site, and totaled about 4.2 billion tokens. GloVe outperformed word2vec by a modest margin: 40.5% to 37% accuracy. We believe that this provides a rationale for using GloVe embeddings for this particular task going forward.

We have so far found the best performance from embeddings with higher dimensionality, trained on more data, and using cosine distance as the distance metric. However, there may be cases when the Euclidean distance is more appropriate to use (e.g., when the analogy has more to do with the magnitude of the difference between words than direction).

4.2 Error Discovery

Next we examined errors from the best performing embeddings, the GloVe.840B, 300 dimension embedding. The model appeared to have more difficulty with a couple of relationships more often than others. For example, all of the embeddings got a question wrong where the analogy was sandal:footwear. All models chose child:parent as the answer, but the answer was watch:timepiece.

A linear relationship does not capture this relationship well. A sandal is a type of footwear. However, there are many types of footwear, and all may have different types of linear relationships. The same is true for the correct answer: a watch is a type of timepiece. We will need a different approach to capture relationships like this.

We also investigated errors in the difference between a variety of relationships using the Google analogies dataset published by Mikolov et al. The results are shown below in Table 3. As noted in the table, most Semantic Questions are answered correctly fairly accurately (with the exception of currency) while most Syntactic Questions are answered less accurately (Table 4).

For these two reasons, we thought that a purely linear approach to solving the problem was limiting the types of questions the model could get correct. We decided to try a couple of approaches to see if we could overcome this hurdle.

Question Type	ACC
Capitals of common countries	94.66%
Capital of the world	96.26%
Currency	5.77%
City in state	64.69%
Family	78.46%

Table 3: Accuracy of GloVe.840B 300D embeddings on semantic questions.

4.3 Improving on Baseline

To solve the analogy problem, we tried a few different solutions. The first was to train new embeddings that have more context included for the words using the ELMo embedding algorithm. The second was to train a deep neural network to understand the analogies and make better guesses at the answers.

Question Type	ACC
Adjective to adverb	29.33%
Opposite	31.16%
Comparative	77.33%
Superlative	38.24%
Present participle	59.00%
Nationality adjective	92.50%
Past tense	48.21%
Plural	82.28%
Plural verbs	51.38%

Table 4: Accuracy of GloVe.840B 300D embeddings on syntactic questions.

4.4 ELMo Encoding Algorithm

The ELMo algorithm is deep contextualized word representation that models both (1) complex characteristics of word use (e.g., syntax and semantics), and (2) how these uses vary across linguistic contexts (Peters et. al. 2018). The ELMo algorithm can be trained on words or sentences to produce word embeddings that include more context around each word. ELMo word representations are functions of the entire input sentence and are computed on top of two-layer bidirectional LSTMs with character convolutions as a linear function of the internal network states. (Peters et. al. 2018). Because ELMo operates on a per-character basis, we believed that even training on individual words would increase performance, particularly on the syntactic questions. It also means that ELMo can operate on out of vocabulary words, by deriving meaning from the structure of the words.

ELMo returns 3 layers for each word, where the lowest layer encodes more syntactic information, and the highest layer encodes more general, semantic information. Each layer of the ELMo encodings can be used as the embedding for our problem set. In order to test the effectiveness of ELMo, we tested the effectiveness of each layer from the ELMo representation separately, as well as the best option across all layers. We also tried training the representations on each word individually and in pairs.

ELMo uses the GloVe.6B 100D embeddings as a basis for its own vectors, so we used the performance of those embeddings as our baseline. As noted in Table 1 above, those embeddings had an accuracy of 37.27%. Table 5 contains the accuracy of the different ELMo layers and training styles.

Trained On	Layer	ACC
Single	Bottom	36.46%
Single	Middle	45.57%
Single	Top	42.35%
Single	All	41.28%
Pairs	Bottom	36.46%
Pairs	Middle	43.69%
Pairs	Top	39.94%
Pairs	All	40.75%

Table 5: Accuracy of ELMo encoded vectors on SAT test set. Trained on "single" denotes that ELMo was trained on each word separately. "Pairs" denotes that the question and answer pair words were trained as sentences.

From Table 5, the single word embeddings taken from the middle layer provide the biggest increase in accuracy with a total of 8.3 percentage points over the baseline accuracy. We validated this finding against the Google Analogy dataset (see Table 6) and found that performance increased compared to the GloVe baseline for all question types except for those related to countries/places. This is a significant improvement over the baseline.

Category	GloVe	ELMo L1
Common Country Capitals	93.87	6.72
Country Capitals	88.95	2.03
City in State	30.81	5.23
Currency	14.2	3.81
Family	81.62	84.58
Adjective to Adverb	24.4	47.18
Opposite	20.07	52.96
Comparative	79.13	92.57
Superlative	54.28	89.3
Present Participle	69.51	83.62
Nationality Adjective	87.87	30.52
Past Tense	55.45	72.05
Plural	72	92.94
Plural Verbs	58.39	96.9
Semantic Accuracy	65.34	8.07
Syntactic Accuracy	61.26	72.22
Total Accuracy	63.11	43.11

Table 6: Baseline accuracy of embeddings with different distance metrics.

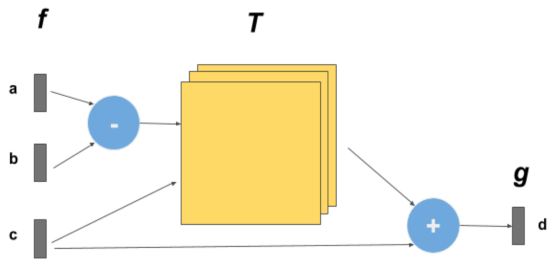

Figure 1: Model 1

4.5 Neural Network

We also wanted to test the behavior of embeddings when used in a Neural Network focused on analogy-making, under the rationale that it could perhaps find patterns between the encoding of words that is not obvious with simple vector operations. The neural network architecture was inspired by the paper "Deep Visual Analogy-Making", by Scott Reed et al, that used a neural network in the context of visual analogies (e.g. where images were rotated, changed color and/or changed size). This was done by encoding pictures using a function f and decoding using a function g. Using the a:b :: c:d framework for analogies, the pictures were inferred by the relationship between a and b by subtracting the encoded values and then used a transformation query T on c to then decode the result and get d. In the paper, they used three different types of transformation query for T: *3CosAdd* (as we have done previously), *3CosMul* (similar to *3CosAdd*, but instead of summing the difference of a and b to c a multiplication is used) and a deep neural network. We used the latter as inspiration for our neural network model.

We tried to reproduce a similar approach for words. For this, we changed the encoding function f to a word embedding lookup, based on the different embeddings we used above. We used the similar approach of inputting a concatenation of the embedding of the word (c) and the difference between the embeddings for two given words (a: b) into a fully-connected neural network, and then summed the results with the word (c) to achieve the final embedding. This embedding was then decoded back using decoder function g to a word (d) by picking the closest word in the embedding matrix through either cosine similarity or euclidean distance. The architecture diagram for the neural network is shown in figure 1.

The neural network was trained using the following objective function, similar to the one used in the original paper, which aims to reduce the sum of the square of norms for the vector resulting from the subtraction of the encoding of target word d and the decoded result as shown below:

$$L = \sum_{a,b,c,d \in V} \| f(d) - \{T[f(b) - f(a), f(c)] + f(c)\} \|_2^2$$

As the original paper did not mention the parameters used for the fully-connected layers, we tried several different network sizes. Initially, we tried with a single 600-dimensional layer, and then tried with a bigger model using 4096-2048-1024 dimensional layers. Training was performed for 4500 epochs with mini-batches of 50 using Adam Optimizer with gradient clipping with a maximum gradient norm of 1.0, a learning rate of 0.00001, and with a dropout rate of 0.5 using the Google Analogy dataset. The word embeddings were fixed and made not trainable in the model, as the interest of our study was to use the neural network to find word features encoded in the embeddings and not refine them with analogy data.

After the set amount of epochs, the training still showed room for loss reduction if more epochs had been executed. The resulting model, however, failed to produce useful results. It is worth mentioning that the model did produce word embeddings in the vicinity of the correct answer in vector space, which indicates that perhaps with some more training epochs to reduce loss the model could provide better results.

We also developed two other models, which haven't been as thoroughly tested. In our second model attempt, we tried a modified version by removing the subtraction between the embeddings of the two given words (a:b) and inputting a concatenation of all three known words (a:b :: c:) into the neural network to check whether it would behave better or worse than the previous model. The rationale was that maybe performing the subtraction before could be restricting the model to transformations that are visible in a linear space. The diagram for this architecture is depicted in figure 2.

Our third model added a secondary, trainable embedding that is concatenated with the original, primary embedding through the secondary encoding function h. This aims to augment the original embeddings with data from training on analogies and perhaps make more complex analogies clearer

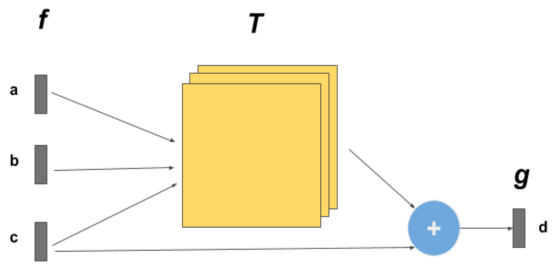

Figure 2: Model 2

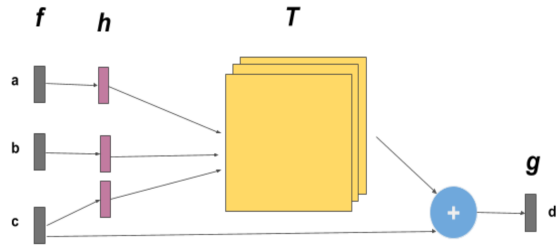

Figure 3: Model 3

on the embedding. The diagram for this architecture is depicted in figure 3. Unfortunately, both models also failed to produce a noteworthy result that was better than random chance.

5 Conclusion

There are a few lessons to be learned. Unsurprisingly, the quality and site of the training corpus is important for the performance of word embeddings on the analogy task, and using the cosine distance as the distance metric was universally more accurate. Given the results of our direct comparison of Word2vec and GloVe, we would recommend using GloVe as the base for future experimentation over Word2vec for this particular application. However, further improvements can be made on top of the embeddings alone.

Using ELMo to improve the GloVe embeddings proved to be particularly promising. The key insight was to use the middle layer of ELMo's deep net instead of the top layer. Since the lower layers encode more of the syntactic information, using the middle layer improved performance in nearly all syntactic questions, and in one semantic type of question. The ELMo representations did seem to perform poorly on questions relating to places (i.e., countries or states), but otherwise improved significantly on the performance of the GloVe 100 dimensional vectors.

The ELMo model that was trained on the Glove 6B 100D emeddings outperformed even the Glove 42B 300D embeddings. Given that particularly encouraging result, training an ELMo model on the Glove 840B 300D vectors would be a promising next step. Given the raw embeddings' performance on the task, the addition of the ELMo representation could offer performance that is competitive with the state of the art.

Finally, although the neural network approach ended up not providing satisfying results, we did find that it does return word embeddings close to the expected words. Given that there was still margin to reduce the loss with more training epochs, it is possible that with a larger analogy dataset for training, more training epochs and further tuning of the model parameters the neural network model could prove to be a useful method for solving analogy problems and thus worth the effort of more study and experimentation.

References

D. Bollegala, Y. Matsuo, M. Ishizuka 2008. *S.A.T. - Measuring Relational Similarity on the Web* https://dl.acm.org/citation.cfm?id=1567281.1567356

O. Levy and Y. Goldberg. 2014. *Linguistic Regularities in Sparse and Explicit Word Representations* http://www.aclweb.org/anthology/W14-1618

D. Li, D. Summers-Stay 2017. *Dual Embeddings and Metrics for Relational Similarity* http://www.aclweb.org/anthology/W17-6924

B. McCann, J. Bradbury, C. Xiong, R. Socher 2017. *Learned in Translation: Contextualized Word Vectors* https://arxiv.org/pdf/1708.00107.pdf

T. Mikolov, K. Chen, G. Corrado, J. Dean. 2013. *Efficient Estimation of Word Representations in Vector Space* https://arxiv.org/abs/1301.3781v3

M. E. Peters, M. Neumann, M. Iyyer, M. Gardener, C. Clark, K. Lee, L. Zettlemoyer 2018. *Deep contextualized word representations* https://arxiv.org/pdf/1802.05365.pdf

S. Reed, Y. Zhang, Y. Zhang, H. Lee 2015. *Deep Visual Analogy-Making* http://papers.nips.cc/paper/5845-deep-visual-analogy-making.pdf

R. Speer, J. Chin, C. Havasi 2016. *ConceptNet 5.5: An Open Multilingual Graph of General Knowledge* https://arxiv.org/abs/1612.03975

P. D. Turney 2008. *A Uniform Approach to Analogies, Synonyms, Antonyms, and Associations* https://arxiv.org/pdf/0809.0124.pdf

P. D. Turney 2002. *A Mining the Web for Synonyms: PMI-IR versus LSA on TOEFL* https://arxiv.org/ftp/cs/papers/0212/0212033.pdf