Journal of High School Science

Optimizing GloVe in lower dimensions with principal component analysis and birch clustering

Kacheria N

Submitted: April 15, 2023, Revised: version 1, June 25, 2023
Accepted: June 28, 2023

Abstract

Natural Language Processing systems such as GloVe are necessary for computer interpretations of human language. Previous research has used these systems to predict genomes or stocks, which provided motivation for this study to reduce its runtime and memory usage. Creative methods for determining the relative definition of a word as well as designing a method of database reduction helped shape the path of this investigation. Clustering methods as well as dimensionality reduction techniques were comparatively analyzed to reduce the dataset dimensionality while preventing data loss. These methods yielded 143 instead of 300 dimensions necessary with the same amount of data, thus halving memory requirements, and demonstrated significant runtime improvements of up to 8 minutes for $O(n^3)$ complexities. This work also revealed word associations that reflected racial and gender bias, and outlined how to find these biases in the future. Finally, a framework was created for determining the necessary dimensionality sizes of specialized corpuses in the future that have less words and dimensions being used for extraneous information.

Keywords

Natural language processing, GloVe, Optimization, Dimensionality, Database, Runtime, Memory, Specialized corpuses, Word embedding, Word vectors, t-distributed stochastic neighbor embedding

---

Nishka Kacheria, Interlake High School, 16245 NE 24th St, Bellevue, WA 98008, USA.
nishkamk2@gmail.com

Summary

From the video game Semantle to ChatGPT (Chat Generative Pre-trained Transformer), Natural Language Processing (NLP) systems are used to communicate with humans, answer questions and classify texts (1). GloVe (Global Vectors for Word Representation) is one such system that improves on Word2Vec (2). By determining semantic similarity indices and mapping words as vectors in a cosine distance-based space, GloVe uses the strengths of CBOW (Continuous Bag Of Words) and skip-gram models. It can be used to find synonyms, sort documents, and for machine translation. It has been used to identify correlations in the human genome and to predict stock prices.

This research's optimization of GloVe thus signifies improvements in all of the above fields. The process outlined in this paper includes using dimensionality reduction techniques on GloVe, and then using clustering algorithms to ensure that words still retain their meaning, which is a new concept that does not exist in the literature. The research concluded that GloVe's 300 dimensions can be reduced to 143 dimensions without data loss. As a result, GloVe; which inherently runs slow and is memory greedy (3, 4) is expected to have a faster runtime, and lower memory constraints. This makes headway in improving LSTMs (Long Short-Term Memory) so they can more effectively mimic human language.

The work presented here is available on the Github Repository https://github.com/the-Fish2/Optimizing-GloVe.git. The folder 'WordFiles' contains word clusters, and the Jupyter Notebooks contains the complete set of code.

A major challenge encountered in this paper was the time complexity. Due to the size of the database (400,000 words, 300 dimensions), and the limited computing power of the local machine used for this research, even simple code took a long time to execute. Further, the framework outlined in this paper for dimensionality reduction and clustering could also be used in NLPs for AIs to continuously refine their own dataset, thus necessitating fast executing algorithms. As a result, several types of clustering and dimensionality reduction techniques were evaluated, with emphasis placed on the criterion of runtime. In determining the optimal dimension for GloVe to avoid losing data from the status quo of 300 dimensions, a binary search was also used because that was the fastest.

Materials

The GloVe Database of 400k words with 300 dimensions was used, available from http://nlp.stanford.edu/projects/glove . The GloVe database has been cross-referenced with the list of the most common 10k words in the English language, which determined the dataset size used. An improvement currently worked upon is expanding the size of the database to 20k words. Jupyter Notebooks, available from https://jupyter.org/ was used to incorporate horizontal scaling in executing the code; else the code would run sequentially. Scikit Learn, available from https://scikit-learn.org/ was used for easier implementation of clustering algorithms. MatplotLib, available from https://matplotlib.org/ was used for the visualization of words

Methods

First, clustering algorithms and then dimensionality reduction techniques were comparatively evaluated. Then, after having chosen Birch and PCA (Principal Component Analysis), a binary search checked each word's clusters to confirm that the word retained its relative definition in 300 dimensions. For the purposes of this study, the maximum dimensions necessary for words to remain in the same cluster was considered the optimal reduction. In addition, PCA creates eigenvectors that were used to replace the previous vectors determined by the AI, because the PCA eigenvectors stored more data per dimension.

Clustering Algorithms

In this context, a 'cluster' is a group of words such that one can infer the definition of one word in the cluster based on the surrounding words' definitions. For example, if Monday, Tuesday, and Wednesday were in a cluster, the semantic definition of Tuesday could be inferred as the 2nd day of the week since it is mathematically adjacent to the 1st and 3rd days of the week.

The idea of the cluster is also a key piece of intuition. To determine if a word still means the same thing even in a different number of dimensions, the natural response is to check a verbal definition. However, since words are stored as vectors, the definition is not understandable or quantifiable. Instead, with clusters where definitions can be inferred based on surrounding words, there is a way to "relatively" define a word.

There were several potential avenues that were considered in lieu of clustering. First, a triangulation strategy to determine that the mathematical relationships between words were preserved was attempted. However, the triangulation always succeeded because there was only one point at the intersection of three circles, which was the word in question. Thus, triangulation would infer that definitions are preserved in any dimension.

Constructing mathematical equations out of words was also considered, but due to the number of variables, this required a lot of complex calculations.

Thus, the selected idea was clustering, and the following algorithms were comparatively evaluated: Agglomerative Clustering, DBScan, K-Means, Birch, and a Breadth-First Search. Sci-Kit Learn's clustering modules were used for all the algorithms except for the breadth-first search; which was coded independently. The Birch algorithm was also optimized by implementing a cosine distance feature. The objective for comparing clustering algorithms was to determine at which dimension the clusters ended up becoming different from previous clusters, and thus lost their meaning.

There were also prerequisites to the clusters: they needed to be performed by a clustering algorithm that used cosine distances (because that is what GloVe uses), and the clusters needed to be of relatively equal size – some words might be more spaced out than others because they were more unique, but they needed to belong in a cluster. All the clusters were analyzed with 300 dimensions and attached images included a representative sample of the clusters that accurately reflected the larger scale of clusters. The complete list of

clusters can be found at the Github repository. Three hundred dimensions were used because this research attempted to condense the clusters and definitions of 300 dimensions in lower dimensions. Hence, the 300 dimension clusters were compared to lower dimension clusters.

The first clustering algorithm analyzed was Birch. It does not offer a cosine distance scale. However, due to Birch's efficiency and suitability for this project, a cosine feature was added that can be found at Github. Birch was a good fit for the program, as the code demonstrated, and created a set of clusters as seen in Figure 1, posted in the WordFiles repository at Github. Words were actually correlated, as shown with how 'working', 'workers', 'worked' and 'works' were clustered together, as were the words, 'elections' and 'democratic'.

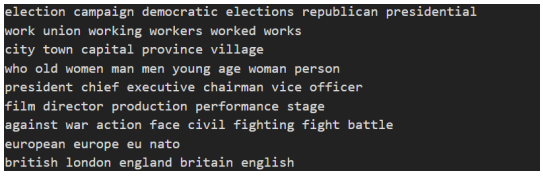

Figure 1: Birch clusters

The next clustering method analyzed was Agglomerative Clustering, which groups words by first finding each word's closest neighbor, and then joining the groups of two based on the distances between those groups. This method failed as it left words on the outskirts of the graph without any similarities or clusters at all, and the more close-together words were all grouped in the same cluster (Figure 2). The results of the Agglomerative Clustering returned one large cluster and over 100 clusters with just one word, which was unhelpful to the study objective. It essentially associated all words in high density regions of the graph and left out the words in low density regions.

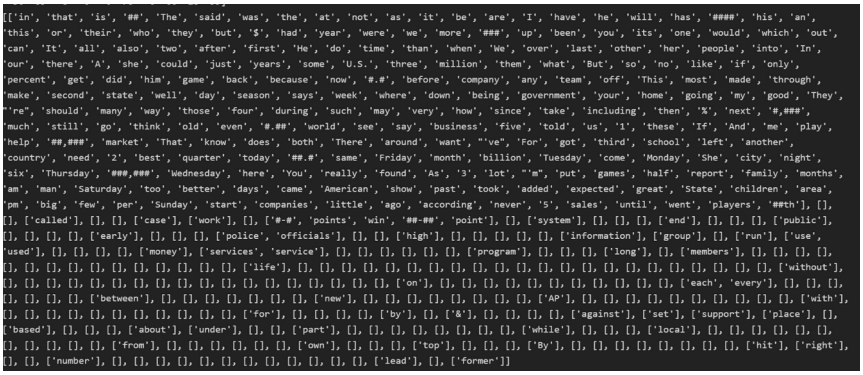

Figure 2: Agglomerative clustering

K-means clustering, although it was able to provide relatively accurate clusters with lower sizes of data sets, took over 10 minutes to execute on the 1000 element data set, and thus was deemed unsuitable for working with the larger dataset. Even if it had been used, the initial set of clusters exhibited the same problems as those of agglomerative clustering; returning one large cluster containing the majority of words and several one-word clusters.

The DBScan clustering algorithm's clusters are presented in Figure 3. Each line is a new cluster, and evidently, the last, huge cluster is not of use. All of k-means, Agglomerative, and DBScan searched primarily for high-density locations and ignored the rest of the words, making them ineffective.

Figure 3: DBScan clustering

One problem, however, was common to all the aforementioned clustering methods, and all other existing ones. Words could only belong to one cluster. At first glance, this might not be perceived as being detrimental toward the objective, because clustering is meant to sort words into groups. However, words can have multiple meanings such that a word could belong in multiple clusters. For example, BIRCH, Agglomerative Clustering, and DBscan all grouped "history, record, career, division, class" and "music, album, release, band, song" into different clusters, even though the word 'record' was correlated with both music and history; as a CD record and as a record of a previous event. To fit one word with two definitions, i.e, attempting to fit 'record' with both 'music' and 'history', would imply that the two clusters would also need to be combined. This is however not correct, as 'history' and 'band' are unrelated words. In this way, the clustering was ineffective for the purpose.

As a result, a breadth-first search (BFS) on all the clusters was used, terminating the search once it reached a certain set of parameters of threshold and size, and allowing each word to populate more than one cluster if needed. The parameters of the size of clusters were calculated from Pennington et. al. (2), wherein the accuracy when plotted against the window size showed a plateau at 5. This would correspond to a cluster of average size 5, hence the parameters were modified to yield clusters of average size 5.

```
curr_cluster=[]
visited = []
queue = []
queue.append(word)

while len(curr_cluster) < 7 and queue:
    curr_word = queue.pop(0)

    if curr_word in curr_cluster:
        break

    curr_cluster.append(curr_word)
    visited.append(curr_word)

    #alg:
    for i in range(4):

        closest_word = find_most_similar(curr_word, data, visited)

        if not (closest_word in curr_cluster):
            queue.append(closest_word)

        if closest_word == word:
            break

        visited.append(closest_word)

    #alg2:
    visited = []
```

Figure 4: BFS code

Since every single word was associated with a cluster, the clusters were significantly more thorough, with the cluster containing the word ‘first’ being sorted with ‘1’ and ‘one’, and also being sorted with ‘second’ and ‘third’ and fourth’ as expected. This code was quite inefficient, as it took four minutes to run for the 300-dimensional 1000 word database, because it multiplied algorithm runtime by n for confirming that every element had at least one cluster. The code for this is shown in Figure 4.

In addition, the BFS had very similar clusters to the clusters created by Birch for a few of their clusters. However, BFS for certain points would result in strange clusters, such as a cluster with both the ‘album’ and ‘history’ mixed in the cluster meant to be centered around record, which is not an association that could be preserved in lower dimensions. This provided evidence of reduced cluster quality, even in 300 dimensions. Birch also exhibited occasional idiosyncrasies like the one

mentioned; although those occurred less frequently when compared with BFS.

All this code, such as the BFS run and the evaluation of different clustering algorithms, can be viewed in the file Optimization2.ipynb. The resulting clusters are stored in gloveWordsClusters.txt. The chart in Figure 5 summarizes this information. In this figure, Red to Green is a spectrum, with Red as most negative and Green as most positive. The Birch algorithm was chosen for optimization, because although it had similar advantages and disadvantages (not allowing words to be reused, slow runtime) as BFS, the Birch algorithm had a higher cluster quality than BFS. Some of the breadth-first search algorithm ended up selecting words from different clusters.

Algorithm	Criterion				
	Does it allow words to be reused?	Cosine Distances?	Runtime (rank)	Cluster Quality (rank)	Memory Use (rank)
Birch	N	Y	1	1	1
Agglomerative	N	Y	3	4	4
K-means	N	N	5	3	2
DBScan	N	Y	4	5	5
Breadth-First Search	Y	Y	2	2	3

Figure 5: Comparison of clustering algorithms

For additional information on data collection, runtime was based on the value that the Jupyter notebook returned for runtime, and cluster quality was measured based on the number of clusters with incorrect associations. For example, both the Birch Clusters and the Breadth First Search clusters returned incorrect clusters surrounding the word ‘york’. BFS resulted in “new, York, old,” seemingly grouping New and Old but being thrown off by New York; while the Birch clusters grouped “new, York, center’ and left ‘central’ in a different cluster. After counting both cluster errors, Birch Clusters had fewer incorrect clusters, and thus it was ranked higher in terms of cluster quality. Therefore, the Birch clustering algorithm was chosen and used for dimensionality reduction.

Dimensionality Reduction Method

PCA was used instead of t-SNE (t -distributed Stochastic Neighbor Embedding) for dimensionality reduction of the vector database. The current vector database attempts to store different “qualities” of each word in different dimensions of the vector. A common example used to explain such vectorized NLPs is given by the formalism,

$$king - man + woman = queen$$

This implies that different vector values are associated with different properties. For example, if the first column is ‘man’, the second is ‘royal’ and the third column is ‘fruit’

(something random) with cosine's range, this would be

$$(1 \quad 1 \quad 1)-(1 \quad 0 \quad 0)+(-1 \quad 0 \quad 0)=(-1 \quad 1 \quad 0)$$

showing that the result is a female royal that is not fruit, or queen.

PCA will then project all these points onto planes that preserve the maximum amount of values. For example, if PCA wanted to make this go from 3 to 2 dimensions, it would project onto the xy plane, ignoring the z values because all z values in this case are trivially zero. Thus, using PCA allows for a significant optimization of GloVe.

The most significant part of this dimensionality reduction is that words with a high semantic similarity should remain close together, while different ones should be far apart.

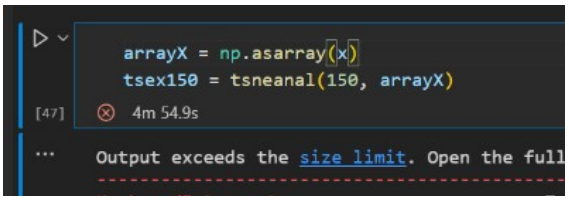

Figure 6: TSNE runtime

Running t-SNE is inefficient due to the long runtime and thus its unsuitability for larger datasets. Using t-SNE, which took over 5 minutes for 1000 words, was thus impractical, especially for the full set of 400k words. Assuming it did take 5 minutes, using a binary search algorithm and then clustering for every new dimension would give $400 \times 5\log_2 n = \log_2 n^{2000}$, which is too long a time.

PCA, on the other hand, was much faster. It provided quick results in lower dimensions because it only uses matrix multiplication. Additionally, it preserves the close points and the far-away points, so that similarities can be measured even in the lower dimensions. Thus, the code for PCA was run instead of that using t-SNE, for the most optimal results. A visualization on 2-D is seen in Figure 7. The days of the week are clustered and highlighted in yellow, as are "us", "each" and "own" in blue.

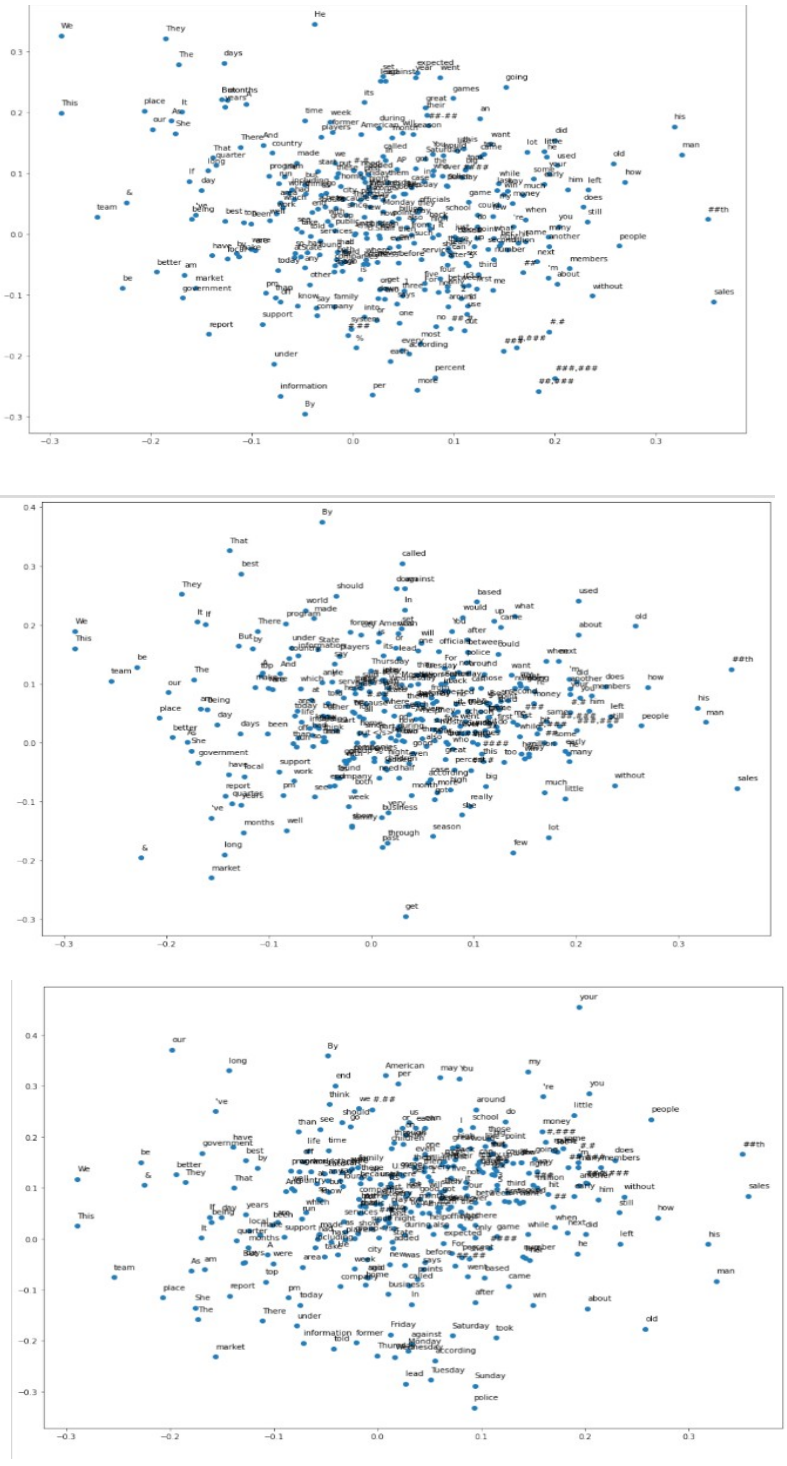

Figure 7: PCA projection in 3D, projection of points on the xy, yz and xz planes.

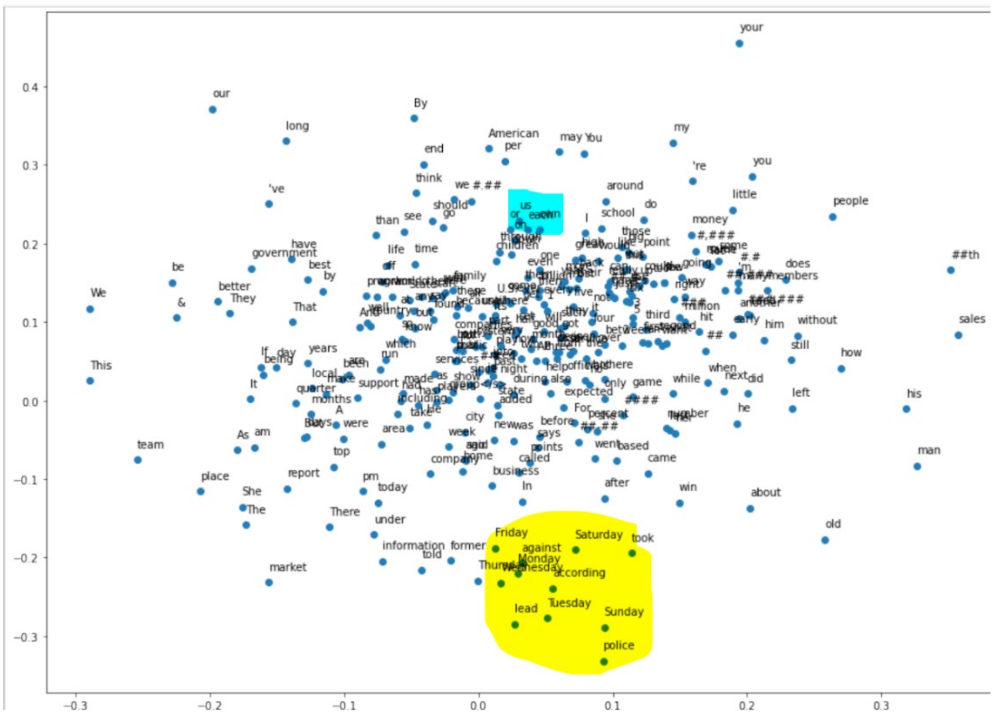

Figure 8: 2D PCA projection

The PCA also had another advantage. Even though the PCA execution could run slower with significantly large datasets like the 400k words in 300 dimensions, it only needed to be run once. PCA uses eigenvalues in sorted order to find the plane projections of the dataset, and thus after computing these eigenvalues for the first time, PCA need never compute them again, but merely only include the important planes. As a result, running PCA to move the dataset to 299 dimensions is all of the computation that is necessary, thereby speeding up the process when compared with t-SNE.

PCA used to simplify the dataset to 2 dimensions produced Figure 8. PCA was also used to simplify the dataset to 3 dimensions, where projecting these points onto the xy, yz, and xz planes yields the three graphs, as shown in Figure 7. Clearly, the first graph projection in 3-D is the same as the 2-D projection, demonstrating how the data points on the planes remain constant throughout dimensionality reduction. This was also tested and demonstrated to be true for four dimensions, with graphs in Plots.pdf and at github. This allowed for a significant optimization of the time complexity of code, to reduce the GloVe dimension size rapidly and then test out the clustering algorithms.

Complete Code

When the best algorithms to lower dimensionality of the GloVe database had been selected, the rest of the code was compiled.

To confirm that clusters would be preserved in lower dimensions, a HashMap was used that associated keys of relative "centers" of clusters, and then checked to see if the keys were still associated with either the closest or second closest of their initial cluster. Initially, just the closest dimension was used, but since this was recognized to not always be optimal, it was later extended to second-closest relationships as well.

If almost every cluster could be preserved, then the dimension was taken to satisfy the condition. Subsequently, a binary search was used that used these checks to find the lowest possible dimension that would still satisfy this condition. The binary search iterated through all possible dimensions in an efficient manner. The data was then simplified using PCA analysis to a lower dimension, and Birch was used to create clusters. These clusters were then checked for associations with the Hashmaps. This iterative code can be accessed in the file Optimization2.ipynb in Github. Excerpts are presented in Figure 9. Overall, the optimal dimension for each word was determined and is accessible in WordFiles/Dimensions.txt in Github. The optimum dimension for each word tended to converge at around 50, a surprisingly low number.

```
def binary_search(n, s):
    low = 1
    high = n
    mid = 0

    while low <= high:
        mid = (high+low)//2
        pcaxNew = pcaanal(mid, keepX)
        if run_birch3(pcaxNew, mid, s):
            high = mid - 1
        else:
            low = mid + 1

    return mid
```

```
def run_birch3(x, dim, s):
    brc = Birch(threshold=0.7, n_clusters=142)
    x2 = brc.fit_predict(x)
    clusters = [[] for x in range(142)]
    index = 0
    for i in range(1, 1000):
        clusters[x2[i]].append(z[i])
        if (z[i] == s):
            index = i
    if(clusters[x2[index]].__contains__(associations[s])):
        return True
    else:
        return False
```

```
associations = {}
with open("WordFiles/gloveWordsBirch300.txt", 'r') as f:
    for i in range(142):
        s = next(f).split(" ")
        s = s[:len(s) - 1]
        if (len(s) >= 2):
            associations[str(s[0])] = str(s[1])

print(associations)
```

```
ind = 0
for key in associations2:
    ans = binary_search(300, key)
    print(ans)
    with open ("WordFiles/Dimensions2.txt", 'a') as f:
        if (ans < dims[ind]):
            f.write(str(ans))
        else:
            f.write(str(dims[ind]))
        f.write("\n")

    ind = ind + 1
```

Figure 9: Complete code excerpts

An example of this process would be the word ‘story’. Initially, it was associated with love (300 dimensions), because of the common phrase ‘love story’. However, when reduced to 150 dimensions, it coalesced with another cluster, that included the word ‘people’, to show how stories talk about people. When the dimensions were further reduced to 75, it joined a cluster with ‘film’ and ‘book’; arguably a better cluster than those formed with 150 and 300 dimensions. However, when reduced to ~56 dimensions, the number of clusters plateaued; but its clusters excluded the word ‘love’.

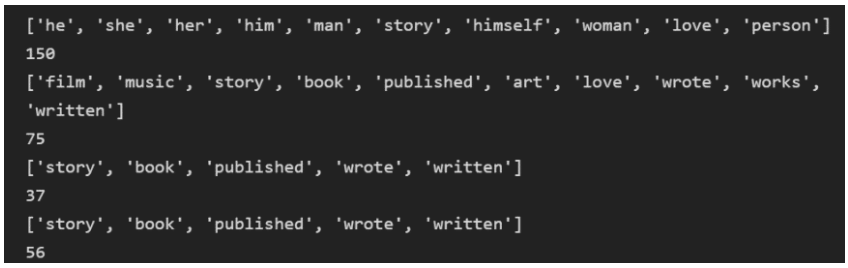

Figure 10: Example with the word “story”

Even though the correlation between story and book from a human’s perspective is high, the GloVe database initially presumed the best connection with the word story was that of the word love. Using GloVe’s initial calculated similarity based on what people type and google searches, the answer would be the smallest dimension where love and story are both in the same cluster – which happens in the 75th dimension but not the 56th dimension. The binary search determined that the 68th cluster as the last one where love and story were in the same cluster, yielding an output of 68.

However, it is also worth considering that the word ‘love’ might make just as much sense as the word ‘book’ when next to the word ‘story.’ The minimum number of dimensions that preserved the words ‘story’ and ‘book’ in the same cluster was ~30, which is less than a third of the size of the cluster with the words ‘love’ and ‘story’ in the same cluster. As a result, the association detection was expanded to include the second-closest word (story in this case) as well.

Results

The procedure on all of the possible clusters yielded a list of numbers associated with the minimum dimension necessary for each cluster to be maintained mostly intact. This is listed at Github as well.

There were two separate rounds of analysis. First, only the 1st level of associations were used – the two closest words in the center of the cluster needed to remain isolated in their own cluster – yielding a mean of 78 and a median of 66 dimensions. Subsequently, another round of analysis was completed where both the closest word to the “center” and

the second-closest word to the "center" of the cluster were specified to stay with the cluster – as with the example of the words 'story', 'book' and 'love'. Even though the cluster itself could be changing, the cluster still made sense, hence this 2nd level of association allowed the code to recognize that and allow such clusters that made sense, to still exist.

Using both the 2nd (second closest and closest words are both allowed to stay in the cluster) and 1st layer of association, yielded totally different results. The 2nd layer of association resulted in a mean of 58 and a median of 33. This almost halved the number of necessary dimensions when compared with the 1st level of association.

However, the standard deviation for both data sets was still relatively large. There was a skew towards 1s – some elements were put in a cluster with such a large size that they would of course be with the other element in their cluster. An example was in the cluster "love story he him her she their …" – even though love and story were unrelated, they were still in the same cluster in lower dimensions, but also in a larger cluster with other words. Upon examination, there were 15 '1' clusters that fell into this issue, and those were removed. Similarly, the words in more than two clusters would have a very high dimension of change, and thus those words were also removed from the evaluation because repeating the process with 3 associations yielded strange clusters. This essentially removed outlying cases that were inconsistent with the dataset. With this change, the standard deviation decreased by 30, showing how the values did tend to approach 33.

The mean was also not an appropriate metric to choose the magnitude of dimensionality reduction , as one or two larger values could result in a significant increase in the number of necessary dimensions. The median, similarly, was unhelpful in that it only preserved ~50% of data.

However, the value that preserved the complete dataset – such that every single word was included, except for outlying cases – was 143 dimensions. This is revealed by the analysis in Figure 11, where the maximum dimension necessary for any word to have the same association is 143. This means that 143 dimensions ensured that every single word retained its cluster and thus its relative definition.

This is additionally significant in comparison with the initial statistics presented by Pennington et al. (2). They saw a decrease of approximately 8% in accuracy from 300 to 150 dimensions. This is significantly more than this paper's accuracy change, which would yield a decrease of 0% accuracy from 300 to 150 dimensions. This accuracy preservation is because of the compression using PCA, and the newer method to determine if words preserve their relative definition. The additional advantage of the method presented in this paper is that it is less restrictive than the method proposed in Pennington et. al., yet still yields the same results when put to use, because it only requires the relative definition of the words to be used. The graph of accuracy *versus* vector dimension with this paper's compressed dataset, as compared to the overall curve from Pennington et. al., is shown in Figure 12.

Clustering Checking Mechanism	Statistics					
	Mean	Median	Mode	Maximum	Minimum	Standard Deviation
1st Association	78	66	1	247	1	71.5
1st, 2nd Association	58	33	1	247	1	65.1
1st, 2nd Association without Outliers	45	33	5	143	3	38.9

Figure 11: Dimensional statistics

As is visible, the framework outlined in this paper creates the refined GloVe accuracy, which is significantly better than the GloVe Accuracy as determined by the paper in Pennington et. al, most visible in the stark difference in accuracy at 66 dimensions. The values in this accuracy map for the refined GloVe accuracy were determined by finding the percentage of words that still retained their relative cluster in a dimension lower than the one provided.

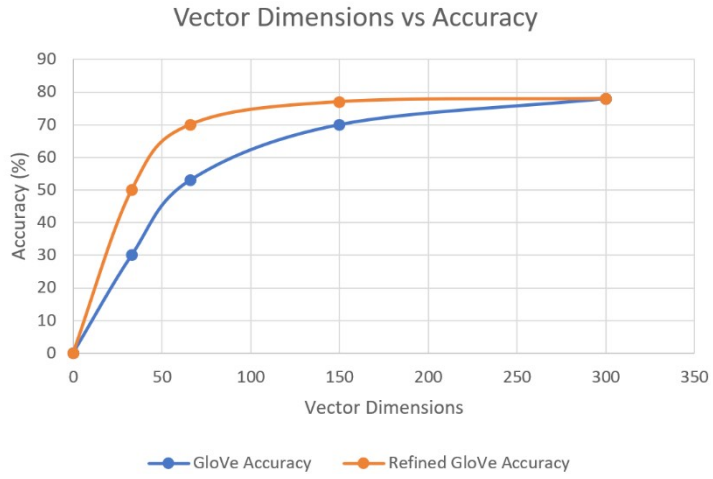

Figure 12: Improvement in accuracy using the refined GloVe algorithm

There were also significant runtime gains associated with these memory reductions. In 143 dimensions, there was as much of a 8 minute reduction in the amount of time it took for an $O((mn)^3)$ complexity, with no data loss. Note that the values in Figure 13 are measured in seconds, and this time is for dimensionality reduction, not the time for training the data. The lack of data loss was revealed by how all the words remained in the same clusters. For

already efficient algorithms, lag time was reduced significantly, allowing NLP users to have much faster computation and thus, response times.

	Runtime of Functions (s)																							
	300 dimensions						143 dimensions						87 dimensions						33 dimensions					
Complexity: O((mn)^x)	1	2	3	4	5	Avg	1	2	3	4	5	Avg	1	2	3	4	5	Avg	1	2	3	4	5	Avg
0	0.1	0.1	0.0	0.1	0.2	0.10	0.0	0.0	0.0	0.0	0.1	0.02	0.2	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
1	0.4	0.5	0.4	0.7	0.6	0.52	0.3	0.2	0.2	0.2	0.3	0.24	0.2	0.1	0.1	0.1	0.2	0.1	0.1	0.1	0.1	0.0	0.1	0.1
2	44.3	45.9	42.1	44.2	40.0	43.3	21.4	19.7	23.0	22.5	24.3	22.2	14.0	16.6	14.5	15.0	13.9	14.8	6.1	6.7	6.3	6.0	5.6	6.1
3	608	729	650	658	800	689	124	178	150	164	113	145.8	86	100	89	93	110	96	42	50	36	45	49	44
	Continuation on Smaller Dataset																							
4	7.0	7.5	6.9	7.0	6.9	7.1	3.4	3.2	3.5	3.4	3.4	3.4	2.1	2.0	2.1	2.1	2.1	2.1	1.1	0.9	0.8	1.0	0.9	0.9
5	768	832	750	790	706	769	272	293	206	290	266	265.4	167	189	167	170	180	175	85.2	83.9	80.6	81.0	76.3	81.4

Figure 13: Runtime data

Clustering biases

However, when the Breadth First Search was run in any number of dimensions, there were some startling associations that did not exist in BIRCH. Because BIRCH only sorted each word into one cluster, it picked the most prevalent association of the word. However, the breadth first search also analyzed the second most important association of the word, which reflected bias, as seen with these clusters, even in 300 dimensions. The Birch algorithm also began showing these biases when projected down to 10 dimensions, as shown in the Github in GloveWordsClusters.txt.

In Fig 14, 'white' is associated with Republican and Congress. Similarly, the word 'Pakistan' was associated with the word 'danger', and the word 'soldiers' was associated with the word 'killed'.

```
senate congress building white republican
```

Figure 14: Racial bias

The process of PCA analysis – projecting points onto a specific plane that preserves as much complexity as possible – revealed implicit biases in language. Since AI uses NLP and databases trained on the internet, these biases may continue to spread. People who are poor might end up being associated with perceived higher rates of crime as AI perpetuates this cycle, as also seen with facial recognition surveillance recently. The inclusion of AI biases even in the language AI is not surprising, given that correlations based on logic leave out the human element. Semantle, a Wordle variant, uses Word2Vec to identify

word similarity, a mechanism akin to GloVe. The thought that it is also being exposed to these biases already and might someday spur a word like ‘poor’ where the closest word is ‘bad,’ begins to ingrain biases in the mind of younger generations. This is a significant problem that must be solved, especially by demystifying AI.

Discussion and Conclusion

Implications

The first implication is also the most obvious one; another game of Semantle that is faster to locally run and implement.

In addition, an improvement of the GloVe database could be attempted.The optimized database could be used as a thesaurus, or for finding similar words to the one wanted, for helping people learn languages and expand their vocabularies. It can also be applied to ChatGPT, because the system uses an LSTM model to generate the next word, and this code can provide a wider range of next words due to two times the available space; as well as create a faster ability to autogenerate sentences.

Another significant implication of this research is its help in the creation of a framework for specialized corpuses. For example, a specialized chemistry version of ChatGPT would include only words related to chemistry, and the size of this chemical word bank would be smaller than the total range of vocabulary. However, using the same vectors is not feasible, since there will be dimensions separating these chemistry words from words like ‘dragon’ that are no longer necessary. This paper outlines a framework to remove these unnecessary dimensions.

The next implication is that of the biases – AI may be flawed in this regard. The GloVe words database was chosen because it used a training model based on what people said and claimed to be purely objective. However, as this research demonstrated, the database still preserved associations in several dimensions that associated discriminatory words with each other.

Future Improvements

One improvement is the expansion of this work to NLPs in different languages, such as Swedish. Since Swedish has 5% of the number of words in English, there are likely even more significant dimensionality reductions that can be exploited.

Another improvement would be the creation of specialized corpuses and implementation of this code on these corpuses. With a chemistry theme, only a set corpus of words is needed; for example: chemical compounds and not words like ‘dragon.’ For those processes, this paper’s framework can be applied to reduce the number of dimensions of the corpus and thus the runtime and the memory and eliminate unnecessary words.

Another improvement would be acting on debiasing the NLP, retraining it on multicultural literature and picture books. This change should be implemented in GloVe itself as well.

Comparison with Relevant Literature

There have been several manuscripts in which different attempts to reduce the dimensionality of GloVe, Word2Vec, or similar methods of text to vectors, have been implemented. An

example is in Singh et. al. (8), where dimensions are reduced, and several different reduction methods are comparatively analyzed. However, this research discussed and compared different dimensionality reduction techniques, accounting for non-linear method comparison as well.

A novel aspect of this research presented in this paper was the idea of measuring the accuracy of the reduced data set through using clustering algorithms to preserve relative definitions, something that – to the author's knowledge - has not been done before. Ordinarily, methods of determining the accuracy of a reduced set using PCA is simply as compared to the accuracy of a reduced set using other dimensionality reduction methods, using accuracy methods such as determining if tasks are completed successfully, such as in References 5 and 7. However, preserving how words are defined (as this paper does), instead of determining task completion, is more useful because it will work on any task, not just the ones that are planned for use. This research, thus, has a wider scope of application.

The notice of bias in these word systems is also more rare, since Reference 6 helped remove more explicit bias in these systems. However, the finding of bias due to moving the code to lower dimensions was also novel, and it helped unveil more deep-rooted forms of bias in NLPs.

Finally, this research is unique because of it's purpose and role in improving LLMs (Large Language Models) – where this work improves specifically a linguistic database where importance is defined on the correlations between words and thus relative definitions, rather than simply locations as vector points in space. No other use of work has also produced a similar work with a resulting lower number of 143 dimensions (or a lower amount), and there has not been significant work on outlining a framework for specialized corpuses like this paper does.

Thus, though there is literature discussing the use of GloVe with clustering algorithms for feature extraction (8, 9), or the use of GloVe with simply PCA for reduction, part of the novel approach in this paper is combining them, as well as the identification of bias in lower dimensions.

References

1. Mikolov, T et. al. (2013 January 16). *Efficient Estimation of Word Representations in Vector Space.* Cornell University. https://arxiv.org/pdf/1301.3781.pdf

2. Pennington, J. et. al. (2014 August) *GloVe : Global Vectors for Word Representation.* Stanford. https://nlp.stanford.edu/projects/glove/

3. Faruqui, M, Dodge, J, Jauhar, S.K, Dyer, C, Hovy, E, Smith, N.A. 2014. Retrofitting word vectors to semantic lexicons. https://doi.org/10.48550/arXiv.1411.4166

4. Ling, S, Song, Y, Roth, D. 2016. Word embeddings with limited memory. In The175 54th Annual Meeting of the Association for Computational Linguistics. Page 387. https://aclanthology.org/P16-2063.pdf

5. Raunak, V. 2017. Simple and Effective Dimensionality Reduction for Word Embeddings. *Cornell University.* Computation and Language, Computer Science. https://doi.org/10.48550/arXiv.1708.03629

6. Bolukbasi, T, Chang K.W, Zou, J.Y, Saligrama, V, Kalai, A.T. 2016. Man is to computer programmer as woman is to homemaker? debiasing word embeddings. In Advances in Neural Information Processing Systems. Pages 4349–4357. https://doi.org/10.48550/arXiv.1607.06520

7. Raunak, V, Gupta, V, Metze. F. 2019. Effective dimensionality reduction for word Embeddings. in *Proceedings of the 4th Workshop on Representation Learning for NLP (RepL4NLP-2019)*, pages 235–243, Florence, Italy. Association for Computational Linguistics. https://aclanthology.org/W19-4328.pdf

8. Singh, K.N, Devi, S.D, Devi, H.M, Mahanta, A.K. A novel approach for dimension reduction using word embedding: An enhanced text classification approach,Int. J. of Information Management Data Insights, 2(1), 2022. https://doi.org/10.1016/j.jjimei.2022.100061, (https://www.sciencedirect.com/science/article/pii/S2667096822000052

9. Gupta, A, Tripathy, B.K. "Implementing GloVe for context based k-means++ clustering," *2017 International Conference on Intelligent Sustainable Systems (ICISS)*, Palladam, India, 2017, pp. 1041-1046, http://dx.doi.org/10.1109/ISS1.2017.8389339