Training Word2Vec with Negative Sampling

Applied Deep Learning in Natural Language Processing

Alexander Koenig

GitHub: axkoenig/word_embeddings

Word Embeddings

- Common preprocessing step in NLP
- Maps discrete words to continuous embeddings space
- Word2Vec word embedding presented by Google in 2013
- Captures semantic similarities between linguistic items

Introduction to Word2Vec

Efficient Estimation of Word Representations in Vector Space

Tomas Mikolov
Google Inc., Mountain View, CA
tmikolov@google.com

Kai Chen
Google Inc., Mountain View, CA
kaichen@google.com

Greg Corrado
Google Inc., Mountain View, CA
gcorrado@google.com

Jeffrey Dean
Google Inc., Mountain View, CA
jeff@google.com

Abstract

We propose two novel model architectures for computing continuous vector representations of words from very large data sets. The quality of these representations is measured in a word similarity task, and the results are compared to the previously best performing techniques based on different types of neural networks. We observe large improvements in accuracy at much lower computational cost, i.e. it takes less than a day to learn high quality word vectors from a 1.6 billion words data set. Furthermore, we show that these vectors provide state-of-the-art performance on our test set for measuring syntactic and semantic word similarities.

Sep 2013

Introduction to Word2Vec

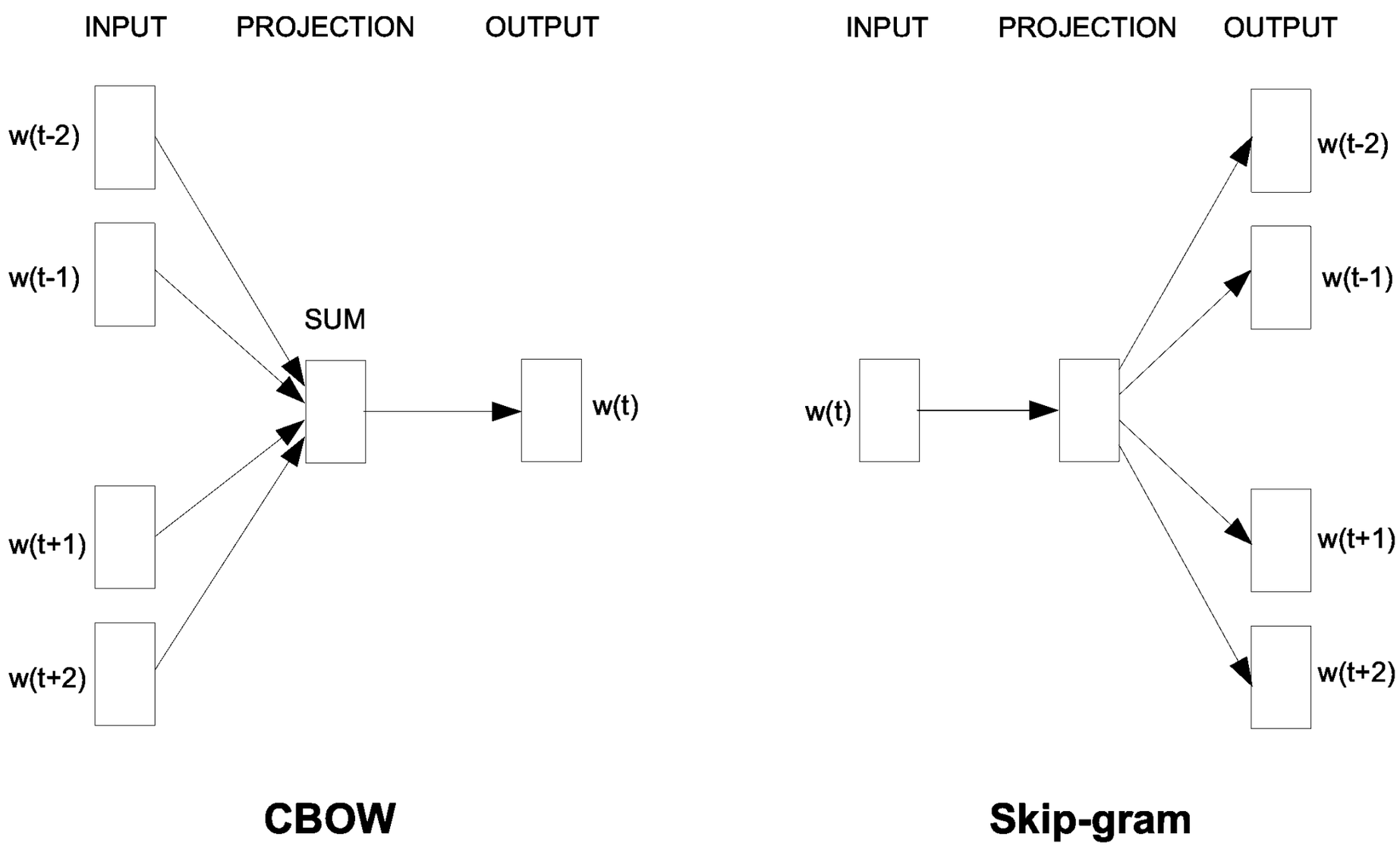


Figure 1: Architectures presented by Mikolov et al. [1]

Introduction to Word2Vec

- Word2Vec designed for “huge datasets” - (billions of words in dataset, millions of words in vocabulary)

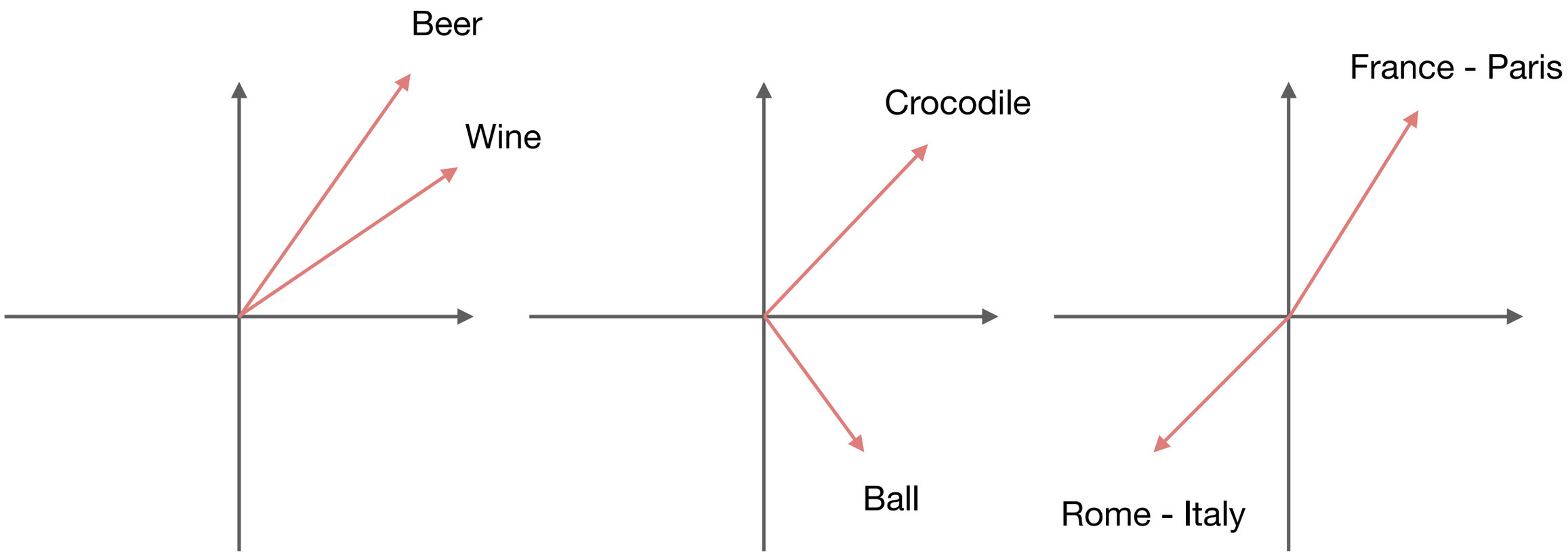
- Objective function forces words that occur in similar contexts to produce similar embeddings

- Relationships between words can be investigated with simple algebraic operations in vector space

$$\overrightarrow{e}(Berlin) + \overrightarrow{e}(Germany) = \overrightarrow{e}(Uruguay) + \overrightarrow{e}(X)$$

$$\overrightarrow{e}(X) = \overrightarrow{e}(Berlin) + \overrightarrow{e}(Germany) - \overrightarrow{e}(Uruguay)$$

Cosine Similarity



high similarity

$cos(\theta) \approx 1$

uncorrelated

$cos(\theta) \approx 0$

high dissimilarity

$cos(\theta) \approx -1$

Figure 2: Cosine similarity measure [2]

Skip-Gram Softmax Problem

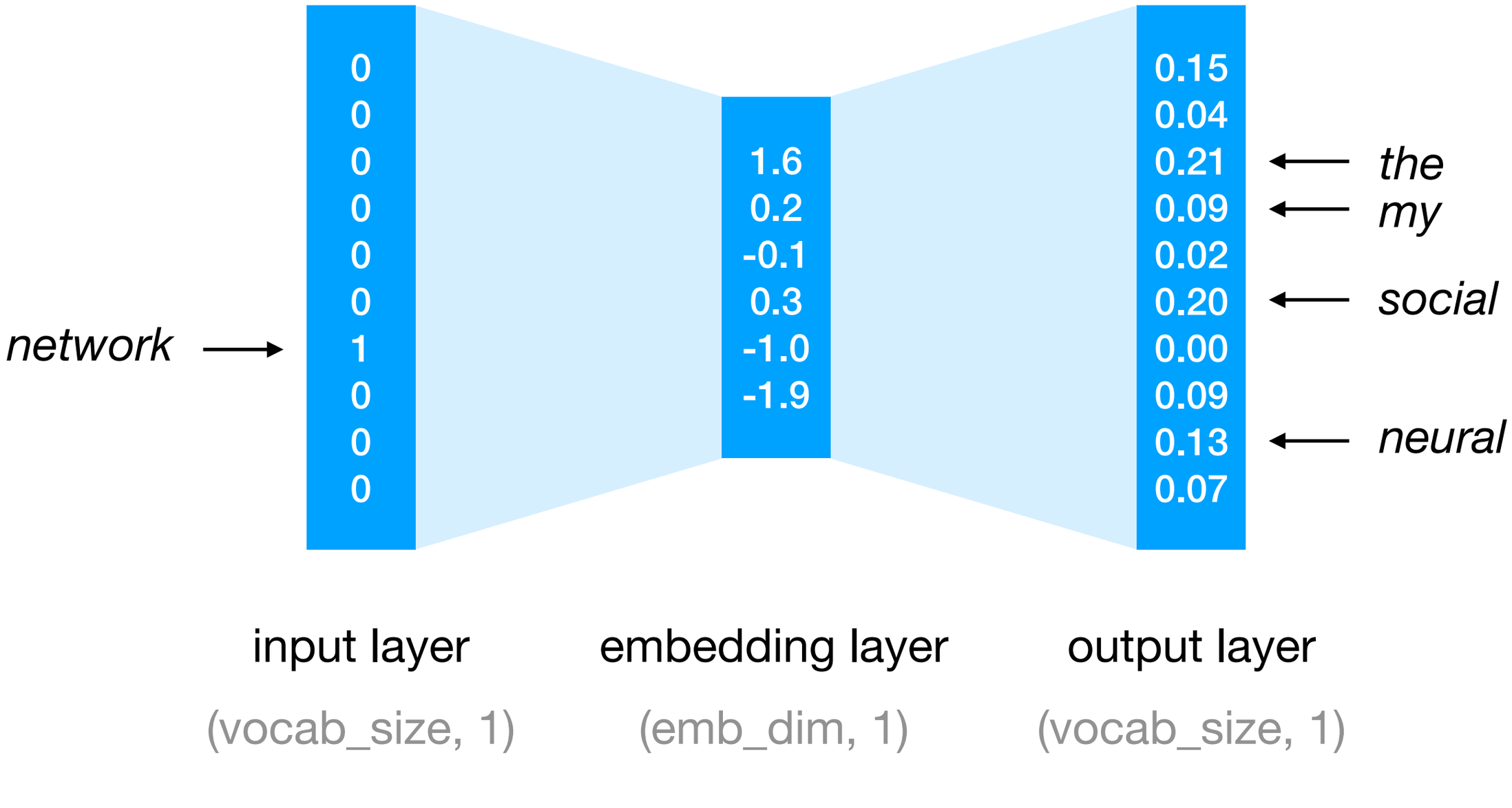


Figure 3: Naive approach

Negative Sampling

Distributed Representations of Words and Phrases and their Compositionality

Tomas Mikolov
Google Inc.
Mountain View
mikolov@google.com

Ilya Sutskever
Google Inc.
Mountain View
ilyasu@google.com

Kai Chen
Google Inc.
Mountain View
kai@google.com

Greg Corrado
Google Inc.
Mountain View
gcorrado@google.com

Jeffrey Dean
Google Inc.
Mountain View
jeff@google.com

Abstract

The recently introduced continuous Skip-gram model is an efficient method for learning high-quality distributed vector representations that capture a large number of precise syntactic and semantic word relationships. In this paper we present several extensions that improve both the quality of the vectors and the training speed. By subsampling of the frequent words we obtain significant speedup and also learn more regular word representations. We also describe a simple alternative to the hierarchical softmax called negative sampling.

An inherent limitation of word representations is their indifference to word order and their inability to represent idiomatic phrases. For example, the meanings of

Oct 2013

Network Architecture

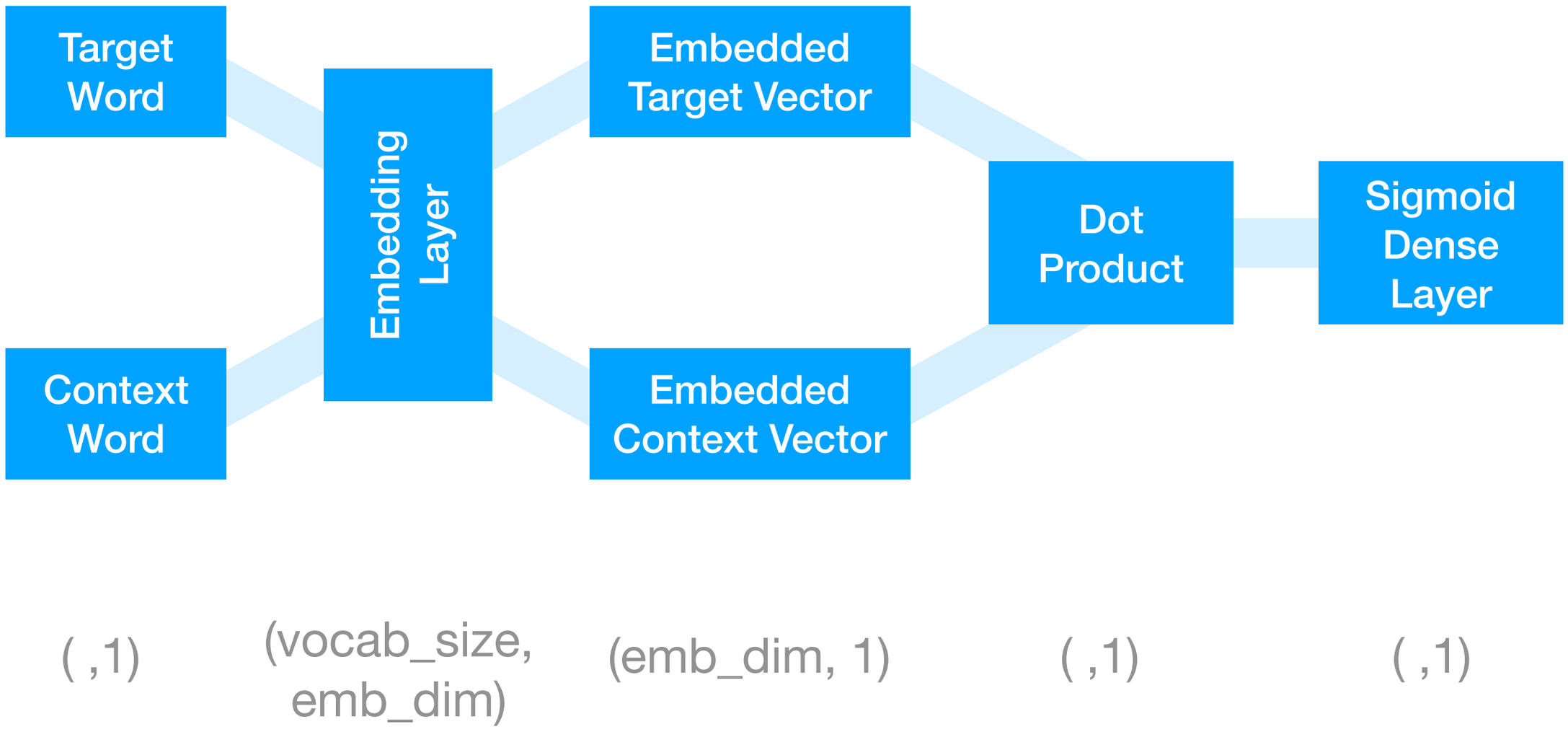


Figure 4: Network Architecture from Tutorial [3]

Results of Tutorial

Iterations = 0:

Nearest to eight: much, holocaust, representations, density, fire, senators, dirty, fc

Iterations = 50,000:

Nearest to eight: six, finest, championships, mathematical, floor, pg, smoke, recurring

Iterations = 200,000:

Nearest to eight: six, five, two, one, nine, seven, three, four

Figure 5: Output of blog post [3]

Results - 1

Data	Bible + Alice
Words	820.571
Unique Words	12.652
Vocabulary Size	10.000
Window Size	3
Embedding Dimension	300
Iterations	500.000
Batch Size	1
Training Time	≈ 3h

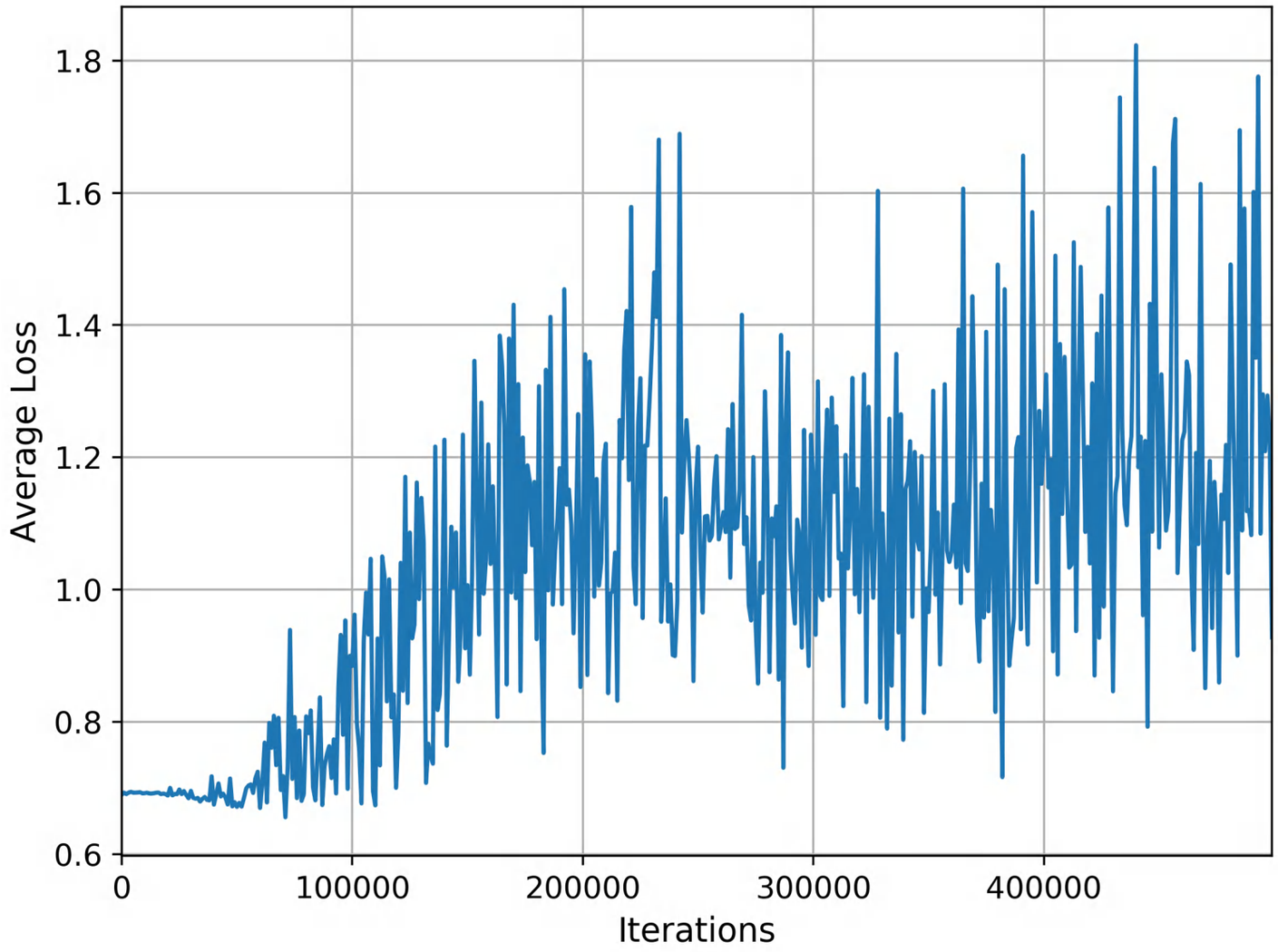


Figure 6: Average loss

Results - 2

Data	**Tutorial**
Words	**17.005.207**
Unique Words	**253.854**
Vocabulary Size	10.000
Window Size	3
Embedding Dimension	300
Iterations	500.000
Batch Size	1
Training Time	≈ 3h

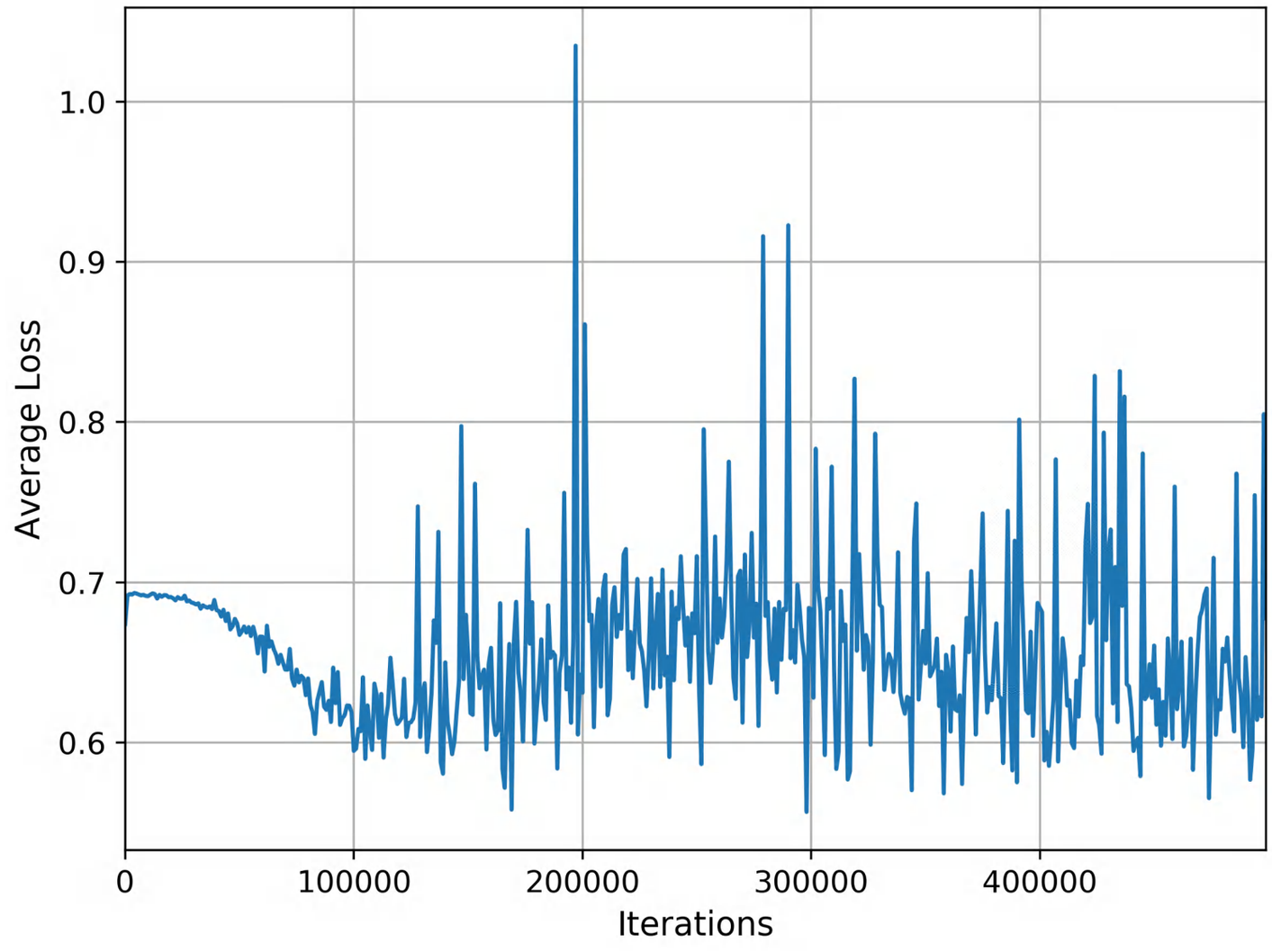


Figure 7: Average loss

Results - 3

Data	Bible + Alice
Words	820.571
Unique Words	12.652
Vocabulary Size	10.000
Window Size	**10**
Embedding Dimension	300
Iterations	500.000
Batch Size	1
Training Time	≈ 3h

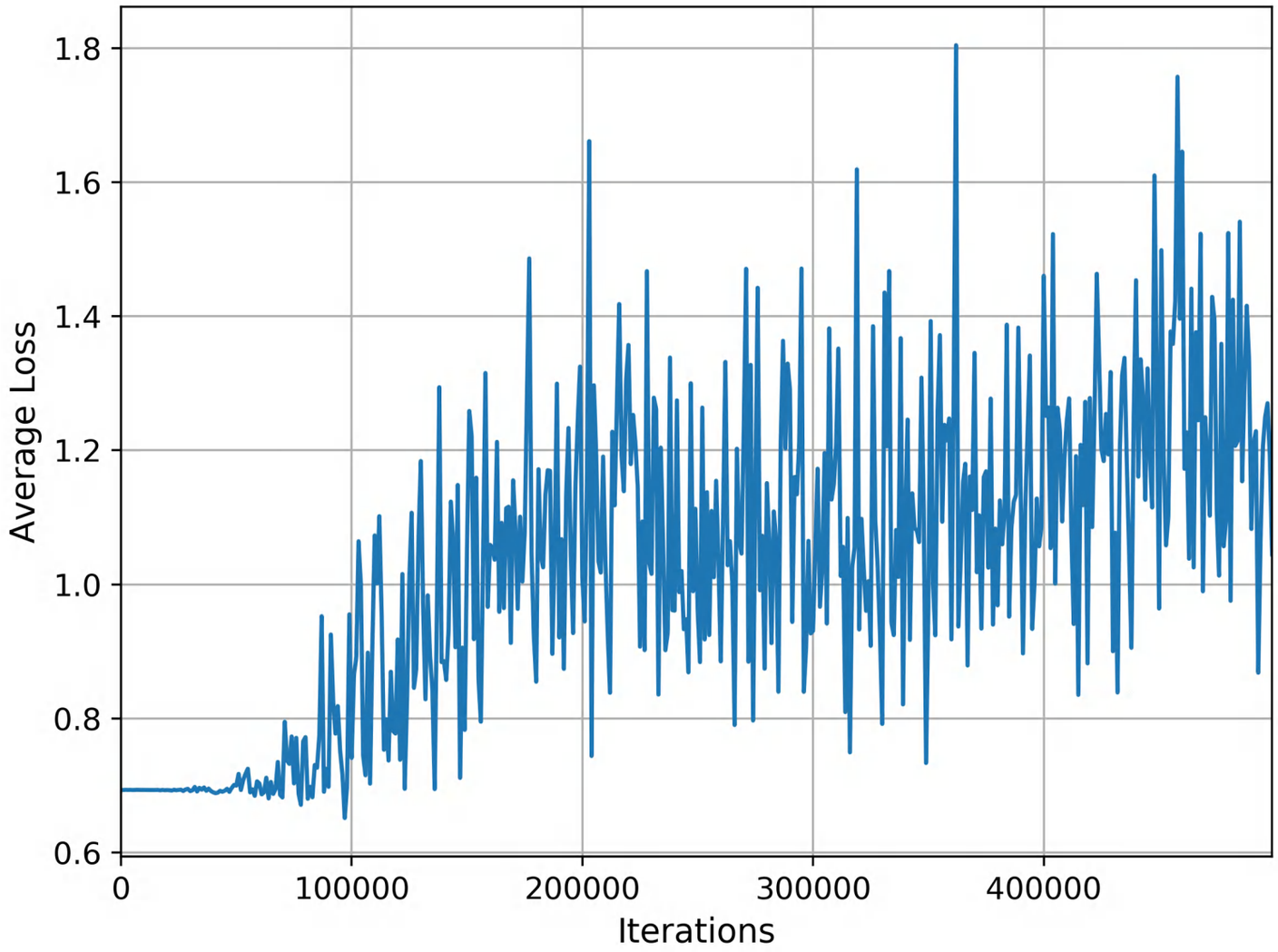


Figure 8: Average loss

Results - 4

Data	Bible + Alice
Words	820.571
Unique Words	12.652
Vocabulary Size	10.000
Window Size	**10**
Embedding Dimension	**50**
Iterations	500.000
Batch Size	1
Training Time	≈ 2h

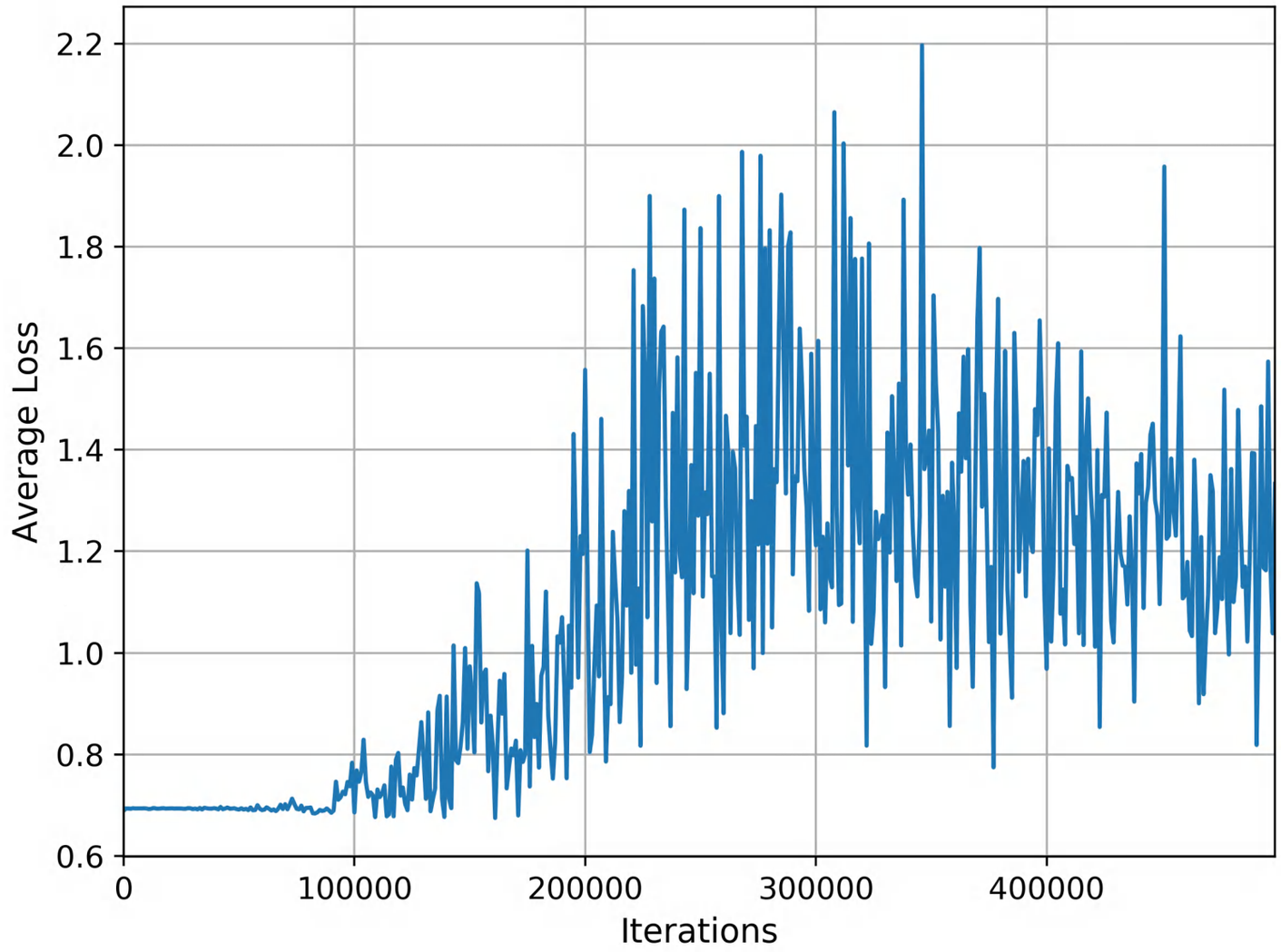


Figure 9: Average loss

Results - 5

Data	Bible + Alice
Words	820.571
Unique Words	12.652
Vocabulary Size	10.000
Window Size	**15**
Embedding Dimension	**300**
Iterations	500.000
Batch Size	1
Training Time	≈ 3h

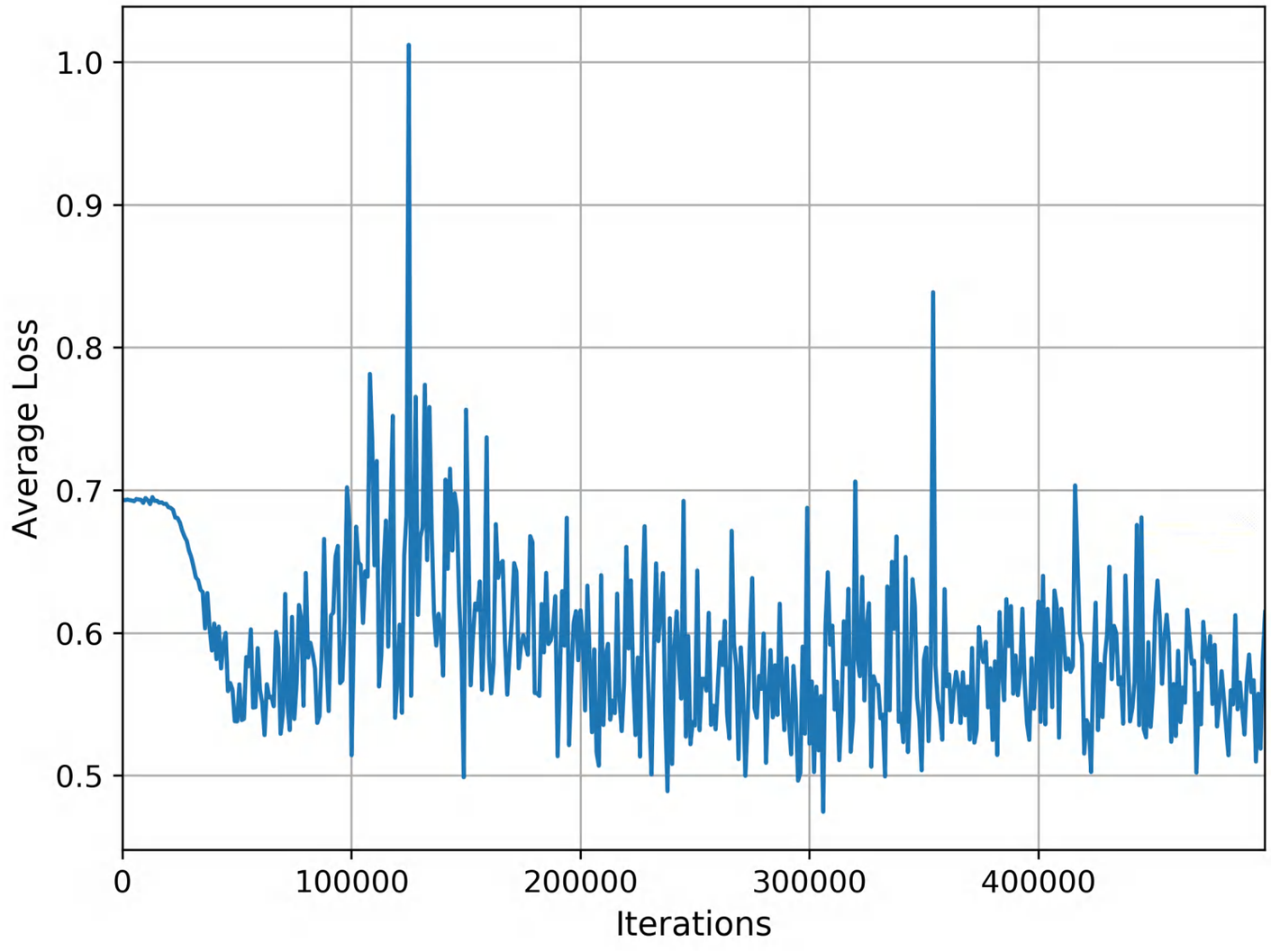


Figure 10: Average loss

Effect of Batch Size

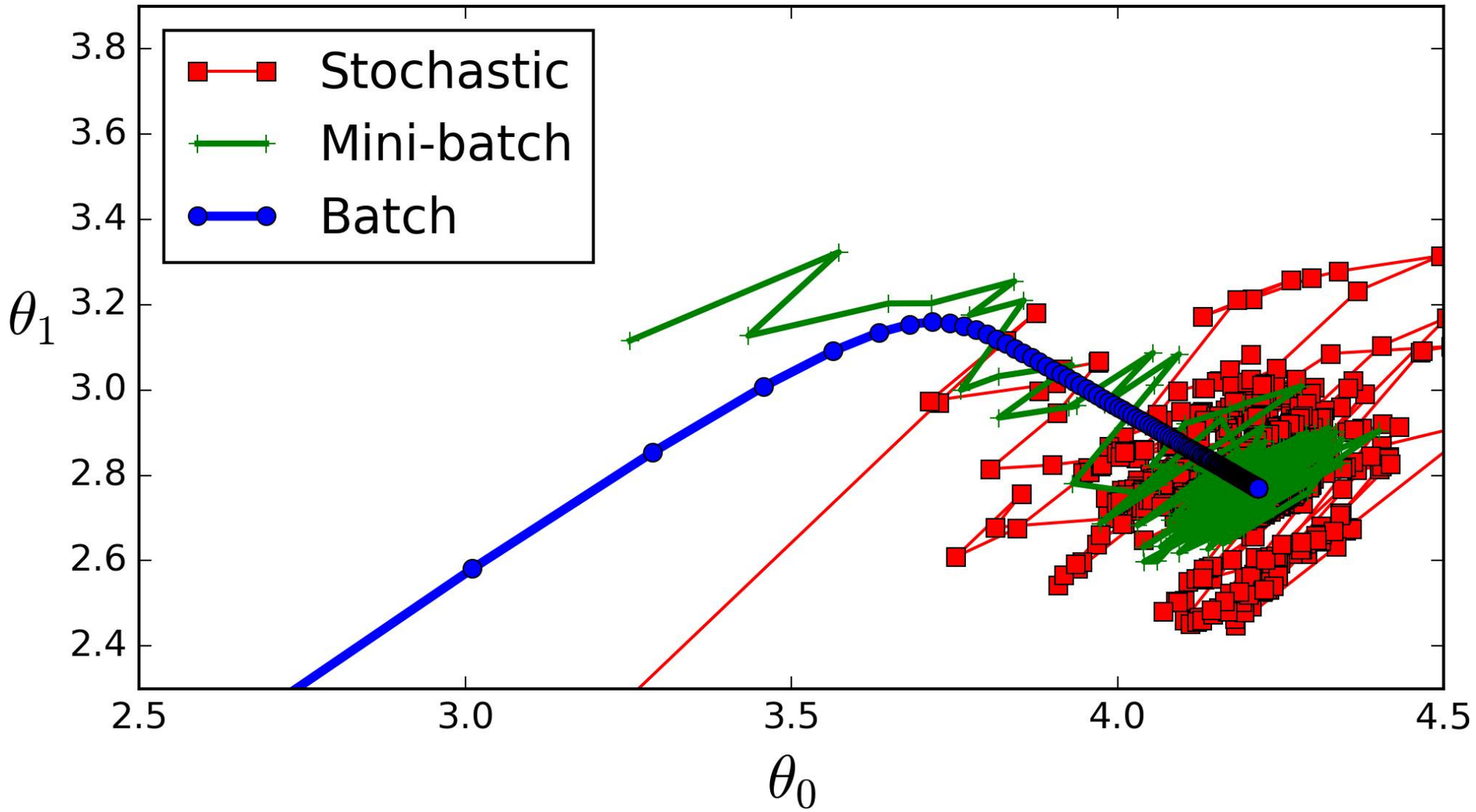


Figure 11: Effect of batch size on optimisation [4]

Results - 6

Data	Bible + Alice
Words	820.571
Unique Words	12.652
Vocabulary Size	10.000
Window Size	3
Embedding Dimension	300
Batch Size	**128**
Training Time	≈ 20m

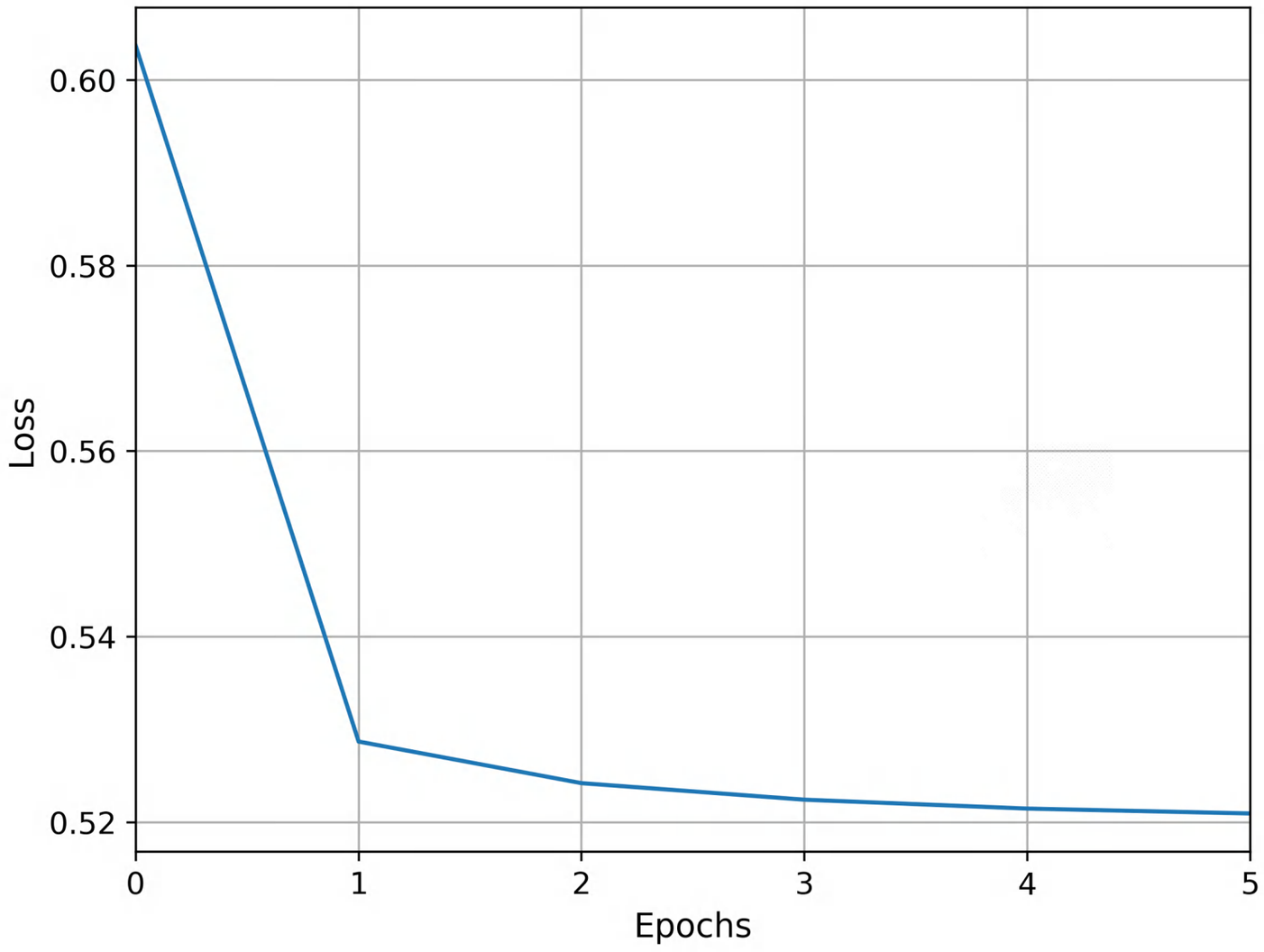


Figure 12: Loss for batched-training

Comparison

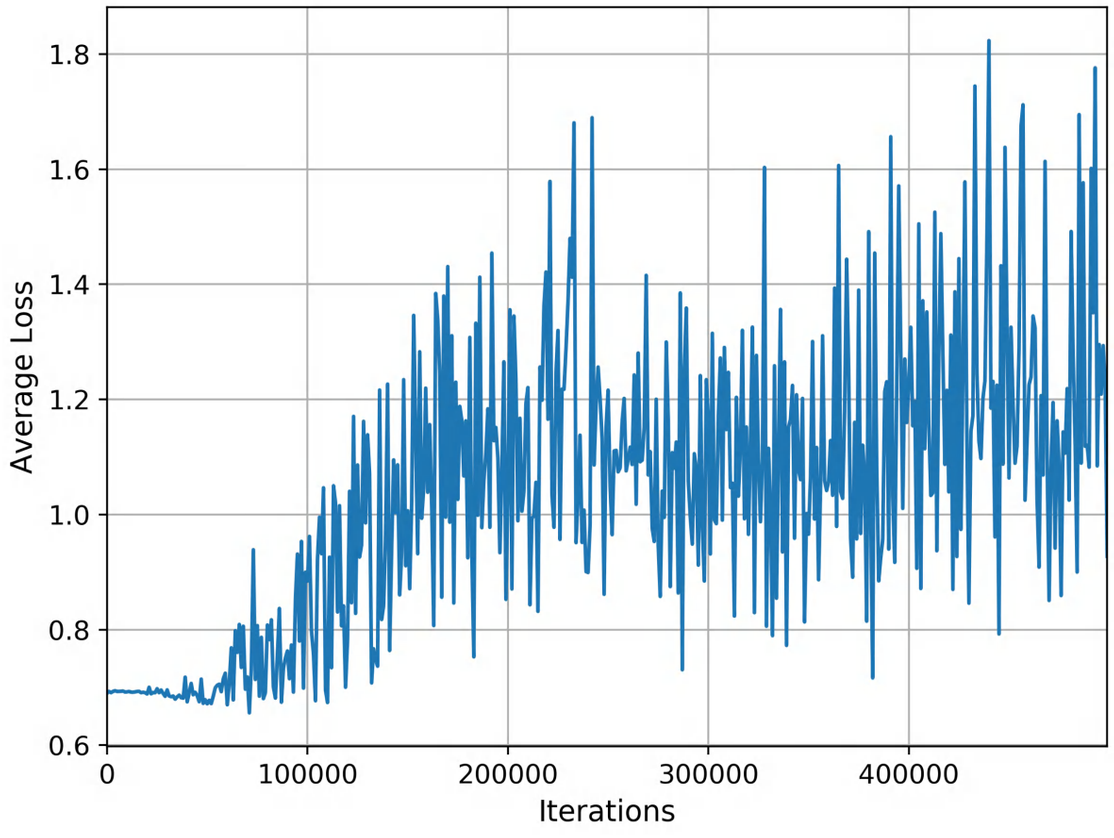


Figure 13: Their approach (Run 1)

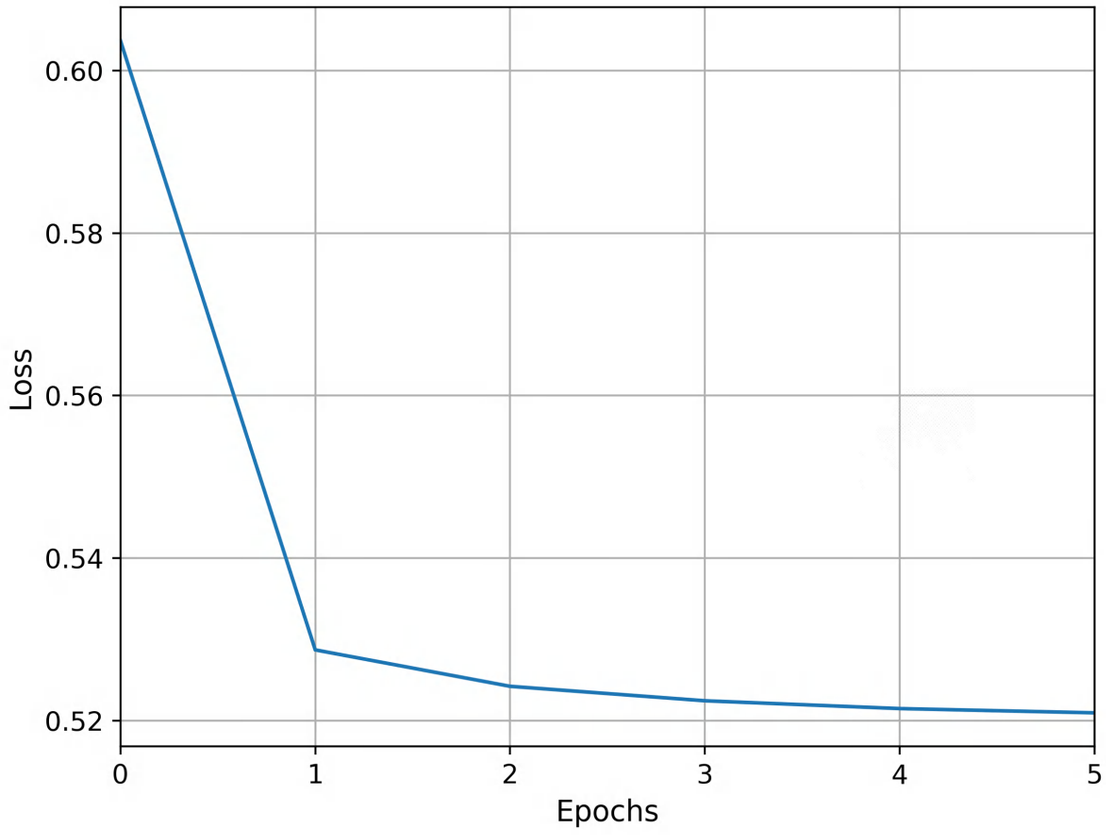


Figure 14: My approach (Run 6)

Further Changes

vocab_size = 10.000
emb_dim = 300

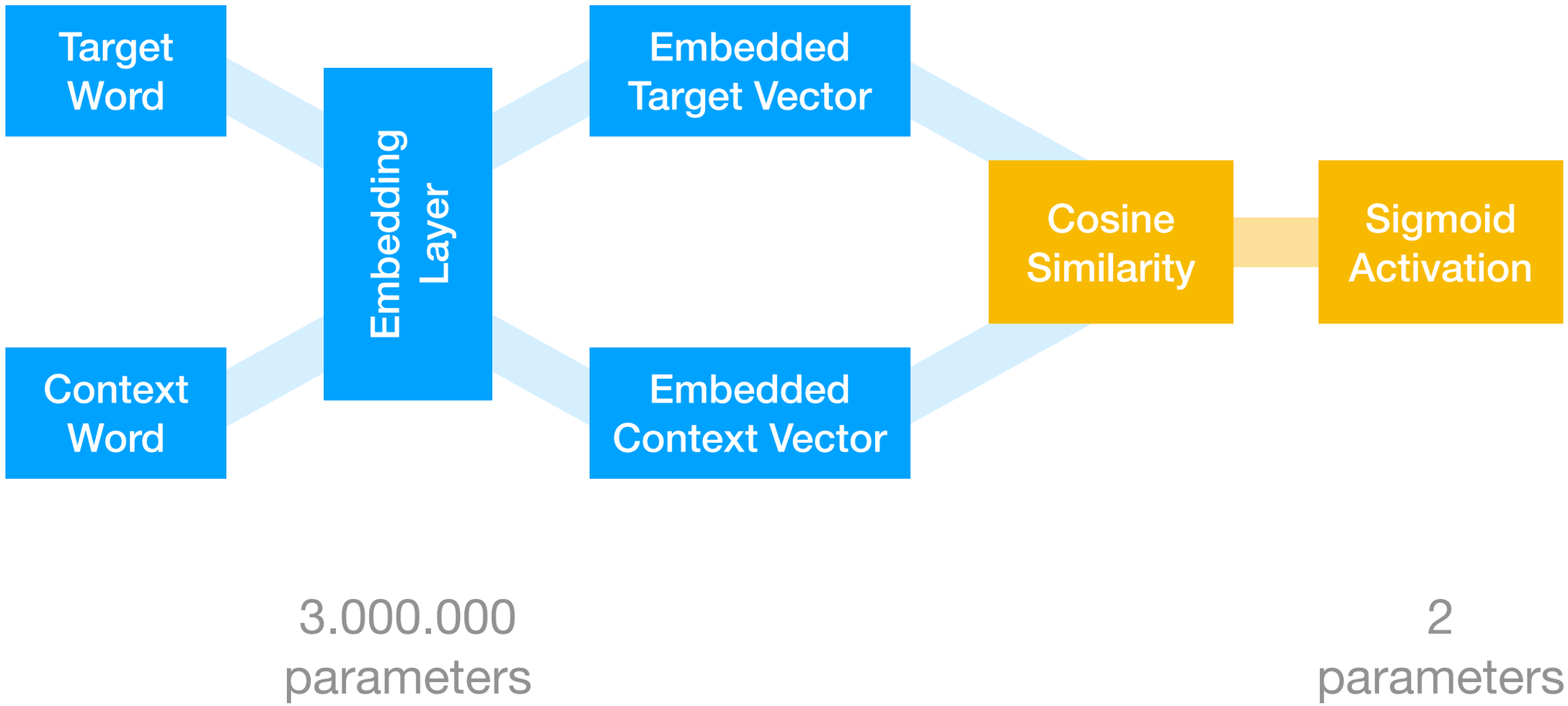
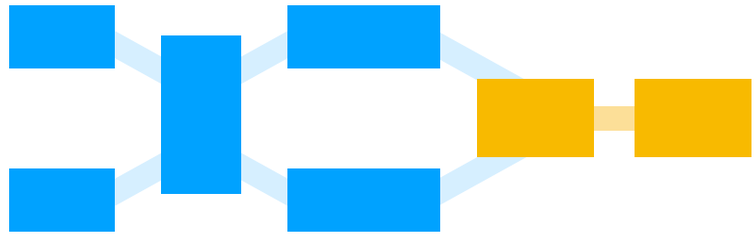


Figure 15: Altered Network Architecture [3]

Sigmoid Activation



- Problem: $cos(\theta) \in [-1,1]$

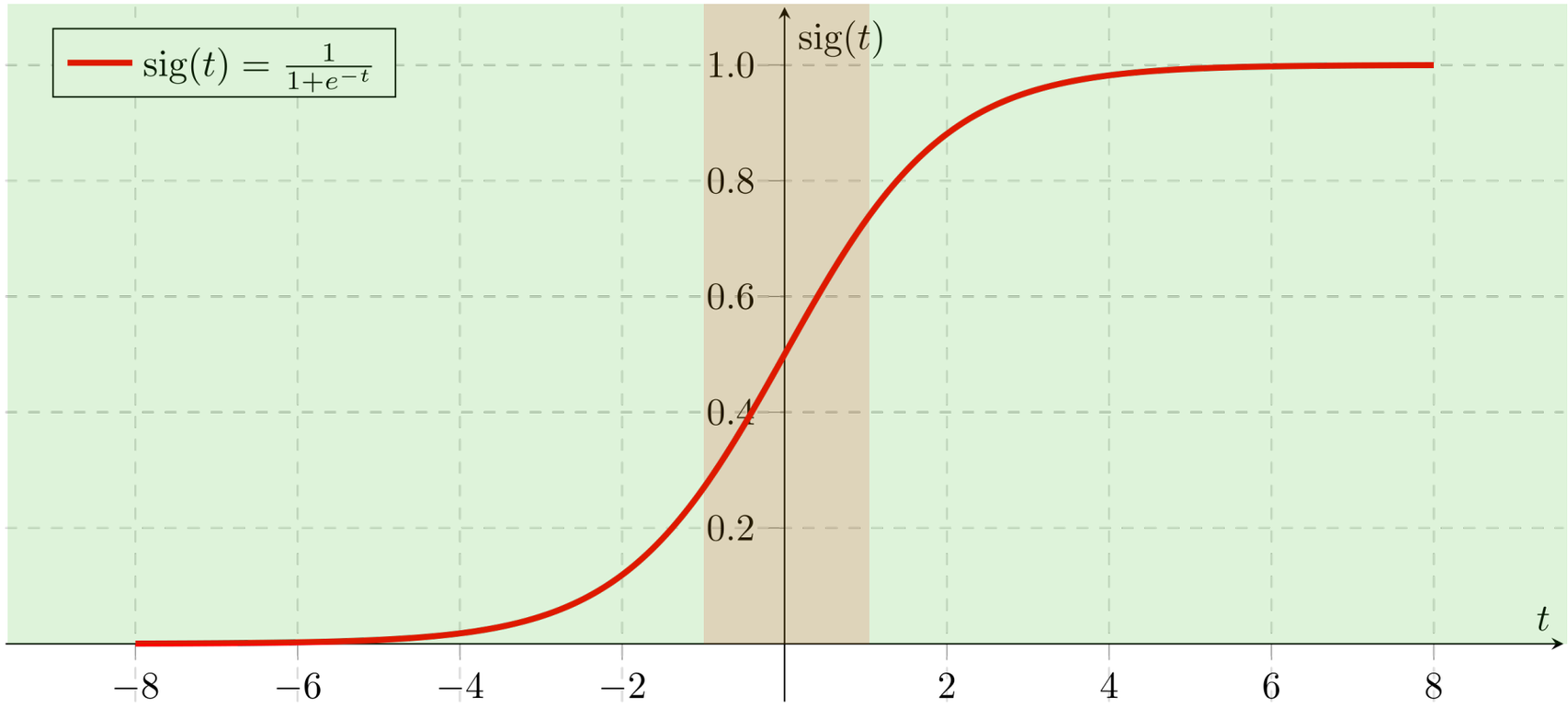


Figure 16: Sigmoid function [5]

Final Architecture

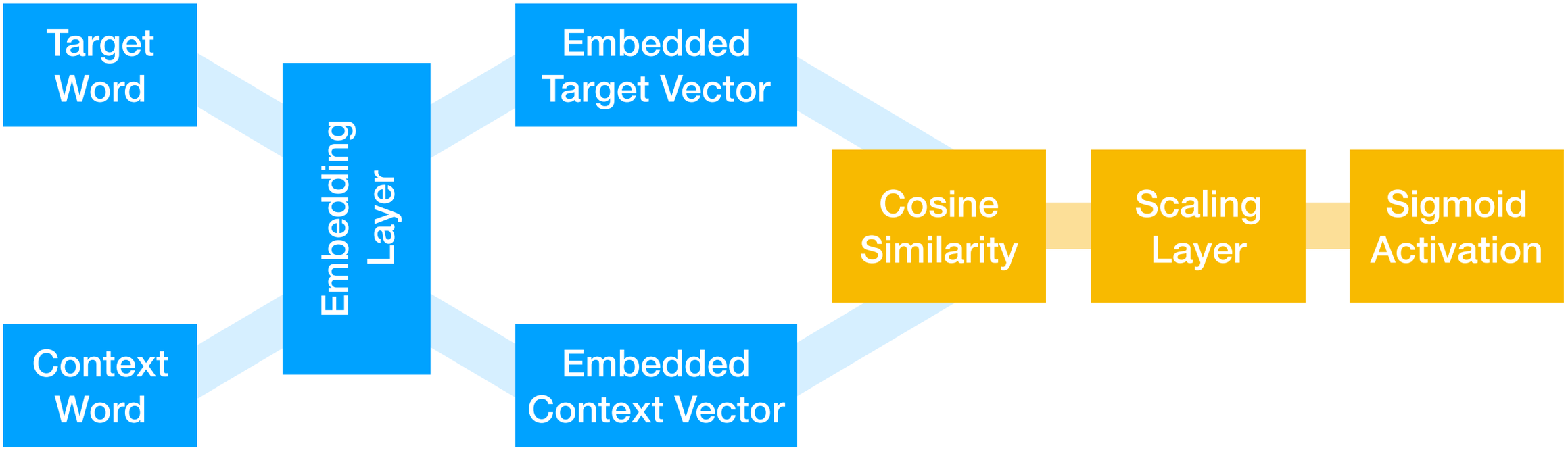


Figure 17: Final Network Architecture [3]

Results - 7

Data	Bible + Alice
Words	820.571
Unique Words	12.652
Vocabulary Size	10.000
Window Size	3
Embedding Dimension	300
Batch Size	128
Training Time	≈ 20m

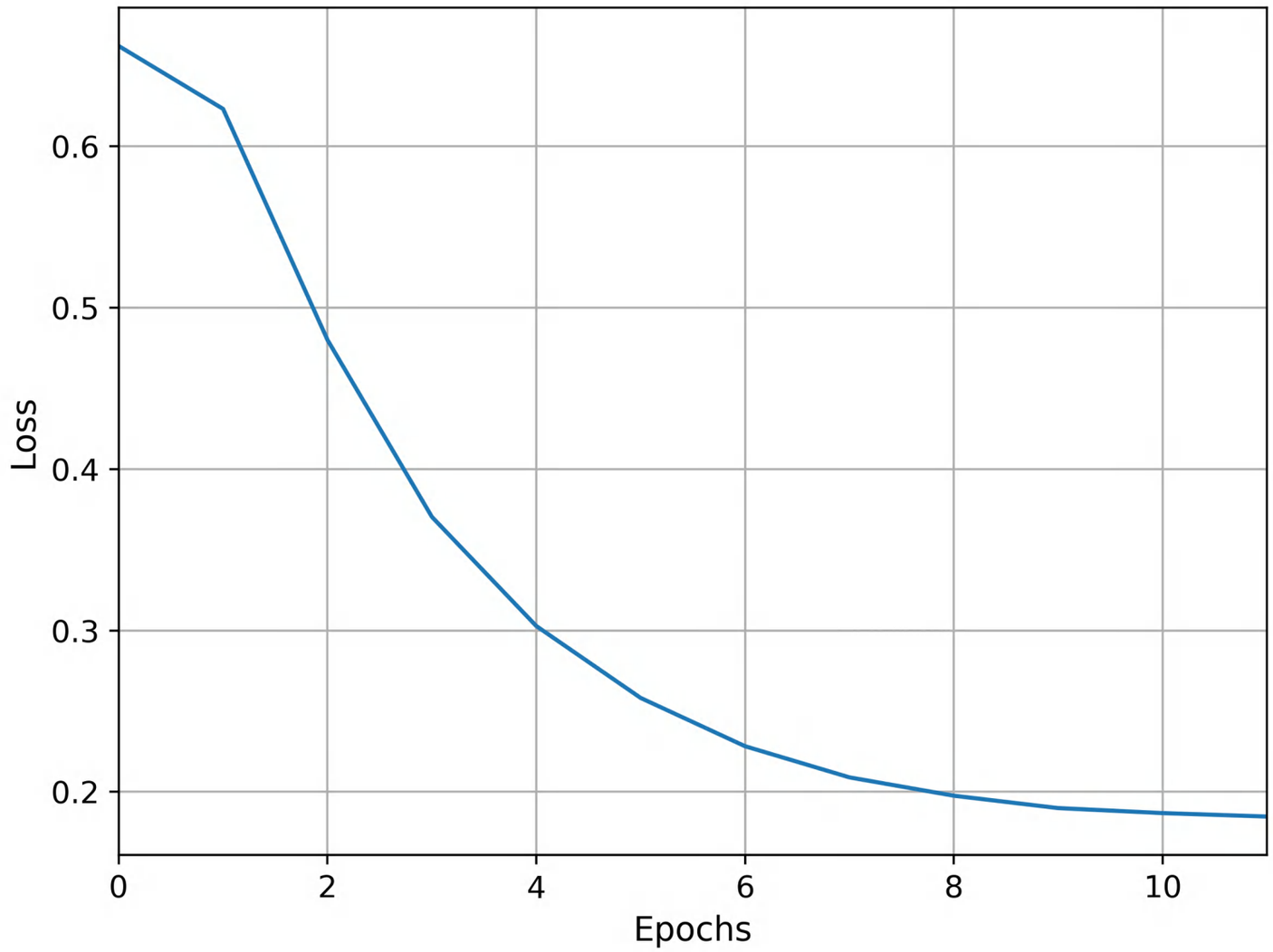


Figure 18: Loss of new architecture

Comparison

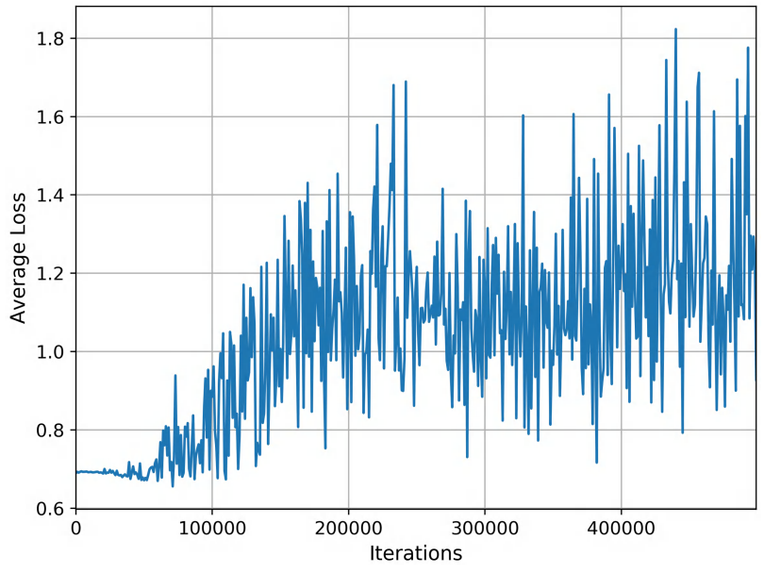


Figure 19: Their approach

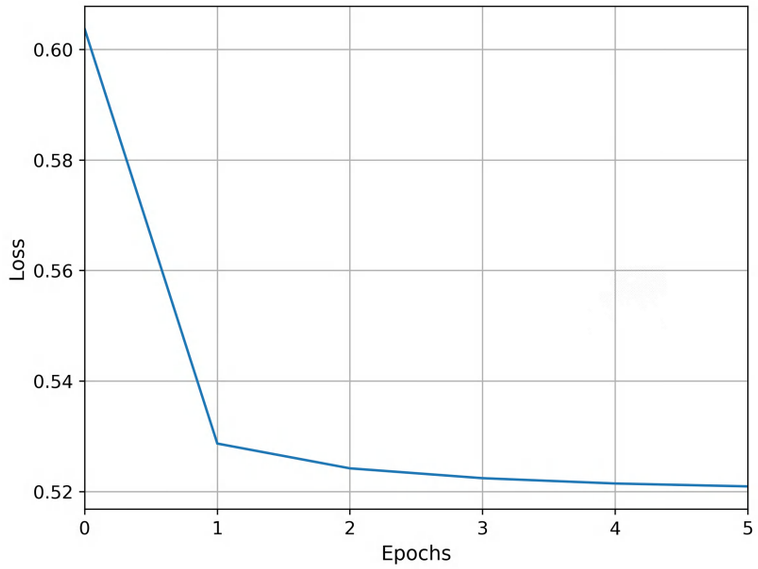


Figure 20: Batched training

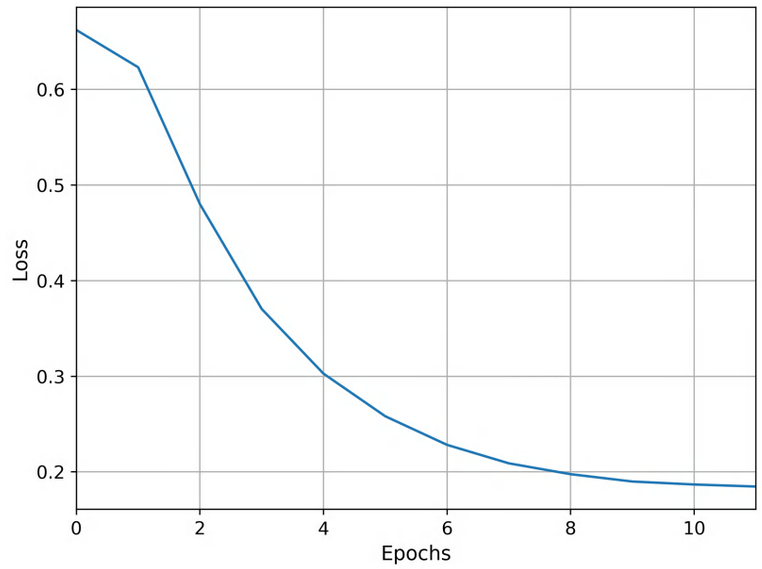


Figure 21: New architecture

Using the Network

Data	“Islam”
Words	3.246.352
Unique Words	22.934
Vocabulary Size	10.000
Window Size	5
Embedding Dimension	300
Batch Size	128
Epochs	12

- Trained on translations of the Quran and the Hadith
- Future work: compare with Bible translations

Live Demo

```
In [59]: # evaluate cosine similarity
         print("---COSINE SIMILARITIES---")
         for word in test_words:
             word_id = word2id[word]
             print("\n" + word)
             print(f"-{male_word}: \t", cos_similarity(word_id, male_word_id, model).numpy())
             print(f"-{female_word}: \t", cos_similarity(word_id, female_word_id, model).numpy())

         ---COSINE SIMILARITIES---

         good
         -he:     0.15852162
         -she:    0.03075388

         power
         -he:     0.12747066
         -she:    -0.057414874

         mighty
         -he:     0.14584734
         -she:    0.06545416

         bad
         -he:     0.1051482
         -she:    0.08670533

         evil
         -he:     0.12937585
         -she:    0.0074573746

         lord
         -he:     0.3070649
         -she:    0.07154995

         god
         -he:     0.31637102
         -she:    0.10531166
```

Figure 22: Excerpt from Live Demo

Future Work

- Determine best hyper parameters with grid search
    - find optimal batch sizes
    - find optimal embedding dimension
    - find optimal window size
- Using the embeddings to find out about gender bias
- Investigate validation loss

References

1. Mikolov, Tomas & Corrado, G.s & Chen, Kai & Dean, Jeffrey. (2013). Efficient Estimation of Word Representations in Vector Space. 1-12.
2. https://datascience-enthusiast.com/DL/Operations_on_word_vectors.html
3. https://adventuresinmachinelearning.com/word2vec-keras-tutorial/
4. https://stats.stackexchange.com/questions/153531/what-is-batch-size-in-neural-network
5. https://de.wikipedia.org/wiki/Sigmoidfunktion#/media/Datei:Sigmoid-function-2.svg
6. Datasets
    1. Experimental dataset: form nltk Gutenberg corpus
    2. Bible texts: https://bible4u.net/en/download#en
    3. Quran texts: http://tanzil.net/trans/
    4. Hadith text: https://www.holybooks.com/the-complete-hadith-all-9-volumes-in-one-pdf/