The Classification of Posts on Reddit with Binary Network Reduction

Travis Barton

I. Introduction

Reddit, is an anonymous social media website where users are responsible for all posted content. It has grown to be the 25^{th} most popular website worldwide and 13^{th} in the United States. It has had over 1.5 billion views and over 100 million active users. [1] There is a treasure trove of information to be had inside its database, and countless oppertunities for automation. Inside of the greater umbrella of Reddit, there are organizational structures called subreddits. Each post must be submitted to a single distinct subreddit, and each subreddit has its own rules and content. The subreddit r/Aww, for example, is a subset of Reddit dedicated to images of cute things. It does not allow certain file formats and has limitations on allowable content. The subreddit r/baseball is a subreddit dedicated to the sport of baseball, and does not allow any posts that are not related to such. The people who enforce these rules are volunteers called moderators, and they can often be overwhelmed. The subreddit r/funny has over 21 million active users, but only 10 moderators to enforce the rules. With such daunting numbers there is a need for automation. Thus the creation of 'bots' becomes necessary. Bots are automated Reddit users designed to fulfill a specific function. They can perform all of the actions that a normal user can, (comment, post, vote, report, etc) but they are doing so based on a script. This project will be focused on the creation of an algorithm for a bot to help moderate the subreddit r/askscience. Particularly, in the automatic labeling of posts.

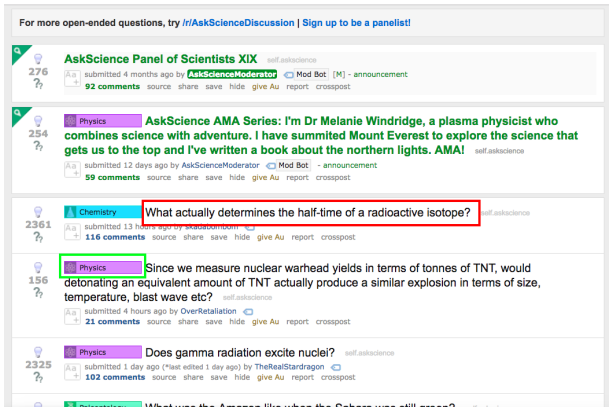


Fig. 1. Example of a page on r/askscience. The red box shows an example of a title, while the green box is an example of a tag

A. r/askscience

R/askscience is a subreddit dedicated to the answering of science related questions posed by its users. Each post can be binned into one of 13 tags: Physics, Biology, Chemistry, Sociology, Astronomy, Planetary Sciences, Medicine, Mathematics, Computing, Engineering, Psychology, or Neuroscience. After each post is uploaded, a moderator must actively label it as one of these categories. With over 16 million readers and only 10 moderators, there is a back log for posts that are awaiting review. [2] With the implementation of the right algorithm and subsequent bot, the process could be automated to the point where moderators only need attend the occasional wrongly classified post.

I have created a technique that can aid in classifying posts with a fair degree of accuracy that, to the best of my knowledge, has not been created before. It utilizes the variable reduction of the data through a series of binary neural networks that I call the 'feed' networks. While they are not yet adequate enough to be implemented directly into a bot, with more data and more time, I believe that they can offer the solution to the problem of backlogged posts.

B. Data Description

Because Reddit does not want to stress their servers, they only allow the accessing of the last 1000 posts from any given subreddit. To obtain the data I need I have been pulling the contents from their page daily for two months, and currently have 4,503 posts. I am limited by the rate of new posts and the quality of content. To ensure quality data, I am only pulling posts that have been deemed successful by Reddit users. I am basing my metric of success on 'karma', which are the points obtained from Reddit's voting system. Posts that have enough 'karma' to make it into the list of top posts for the day were considered. All others were ignored. My database will grow with time, and thus my model accuracy will improve. To pull the data necessary, I used the python package 'Praw'.[3]

1) Toy Data: To do my preliminary testing, I needed a balanced, easily obtained, and easily separated data set. My solution was to pull the titles of posts from 3 different subreddits: r/Aww, r/Politics, and r/Showerthoughts. As described above, r/Aww is a subreddit dedicated to images of cute things. r/Politics is a subreddit dedicated to articles, and discussions of politics, both national and international. r/showerthoughts is a subreddit dedicated to the sharing of random, insightful thoughts. Each of these subreddits have very different goals, cultures, and rules, and as such should have very different titles. Below is an example of a typical post from each subreddit:

Title	Subreddit
Imagine wakingup to this!	r/Aww
Arkansas Democrat was left off one countys ballot as early voting began.	r/Politics
If I see Google in a show/movie I think nothing of it, but if I see Bing, I know it's product placement.	r/Showerthoughts

This data will be used in order to establish if there is a meaningful relationship between my variable reduction and the accuracy of predictions. I expected this data to do well regardless of the algorithm due to its easily separable nature. I am more interested in how well it does in reference to my baseline.

2) Full Data: The full data, as described above, is a collection of 4,503 posts consisting of a title and a tag. Here is an example of a couple of representative points:

Title	Tag
Why do we use pillows now when we sleep? Did we need this during the prehistoric/ancient age? What changed?	soc
Why are Primates incapable of Human speech, while lesser animals such as Parrots can emulate Human speech?	bio
If each day is only 23h56m4s, over the course of 4 years, we accumulate 95.7 hours of unaccounted time when approximating each day to 24 hours. We give ourselves one extra day in February, which accounts for only 24 hours of that extra time, but where does that extra 71.7 hours go?	astro

There are 13 possible tags, but many of them are very sparse. To avoid a sample size issue, I condensed the smaller values into one 'other' category.

physics	983			
bio	814			
med	706			
geo	545		physics	983
chem	401		bio	814
astro	380		med	706
eng	281	→	geo	545
neuro	107		chem	401
soc	92		astro	380
maths	82		other	739
psych	81			
computing	72			
Meta	24			

This may cause a different issue of the 'other' posts being very different from one another, but I feel it would be worse to have extremely sparse data inflating my accuracy.

Since many of these topics are rarely asked about on r/askscience, it may be practical for the final bot to assign these groups as 'other', and simply direct those posts to the moderators for active labeling. Cutting the number of posts by a large portion is not as good as a complete labeling algorithm, but it will still save a lot of time for the moderators.

II. Methods

A. Vectorization

For the data to interact with the models, it needs to be converted into numeric vectors. There are a number of methods that can do this, for example, bag of words has been a standard for years.[9] [10] [11] However, I will be using a newer method that avoids the sparsity that bag of word often produces called word embeddings. Word embeddings take an established collection of documents (like wikipedia, or twitter) and examines the relationship between the words within it. Based on the relationships seen by this examination, each word can be mapped to a point in N dimensional space, where its neighbors are words that are observed to be similar in context. [11]. Word embeddings have been shown to be more representative than many older state of the art methods by long margins, and is quickly gaining traction as the leading technique for word and sentence vectorization. [9] [10] [11] There are many variations to word embeddings, but we will only focus on the methods implemented by the python package Spacy. [7] [12] Spacy utilizes the skip-gram model that was first introduced by the creators of the textual analysis package word2vec at Google. Given a sequence of training words $w_1, w_2, ..., w_T$, one can find the optimal word representations by maximizing the log probability:

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

where c is the size of the training context window (the words to the left and right of w_i that are examined for context). Increasing c tends to increase accuracy, but also greatly increases training time. [12] the generalized $p(w_O|w_t)$ is defined using the softmax function:

$$p(w_O|w_t) = \frac{\exp {v'_{wo}}^T v_{wI}}{\sum_{w=1}^{W}\exp {v'_{wo}}^T v_{wI}}$$

[12] Where v_w and v'_w are the input and output vector representations of w and W is the size of the vocabulary. Since this is proportional to W, and since the vocabulary is often more than hundreds of millions, approximations to the above equation have to be used. These approximation methods and the creation of v and v' can be seen in Yoav and Levy's paper. [12] The main reason that I chose Spacy over other methods is that it is the easiest to implement without sacrificing effectiveness. One of Spacy's most attractive features is its pre-compiled vectorization method. It comes with a number of pre-built models that can vectorize words, as well as sentences. I choose to use their en_vectors_web_lg model, which is their largest and is trained on the common crawl data base. [7] When completed, the output of the Spacy's vectorization model is a 300 dimensional vector for each title. [7]

B. Feed Networks

After the vectorization process is complete, I can begin my variable reduction technique. It involves training a number of 'feed networks' equal to the number of unique labels, in our case 7. These are the 7 binary neural networks whose only job is to determine if a given sentence belongs to their category or not. The structure of each of these networks is the same, and as such, I will only describe one without a loss of generality.

The physics network has a 300 dimensional input, a binary output, and three hidden layers, all with a 40% dropout between them to prevent overfitting. The details of which are below:

input	300
output	2
neurons per layer	300, 50, 100, 50
hidden activation	Leaky-Relu
loss function	binary crossentropy
optimizer	adam

The Leaky-Relu (leaky rectified linear units) activation function was used because I wanted to ensure that no value was allowed to be too negative, but still allowing values to stray below zero. This will aid in preventing the 'dying' of neurons in my network. When using traditional Relu, the formula is the maximum between 0 and the input z:

$$g(z) = max(0, z)$$

[5] However, this means that if a neuron is set to zero (via a negative bias + Relu), then it can have no more influence over the dot product, and may be dropped when it should have influence over the model. When a neuron is set to 0 like this it is called a 'dead' neuron. The Leaky-Relu activation avoids this by instead of simply taking the max and potentially outputting 0, it follows this piece-wise function:

$$g(z) = \begin{cases} 0.01z & z < 0 \\ z & z \geq 0 \end{cases}$$

[5] The loss function Binary Crossentropy measures how many correctly classified points the model has vs how many points it was fed. Formally, the function looks like so:

$$\text{loss}(x) = -(y \log(p) + (1 - y) \log(1 - p))$$

[13] With y being the label of the point x, and p being the probability that it lies in class y.

The Adam optimizer is very similar to the stochastic gradient decent method, except that it processes mini-batches and changes the gradient based on them, instead of using the full batch. This results in a more dynamic, and stable result. Its creator describes the process as "...many objective functions... composed of a sum of subfunctions evaluated at different subsamples of data; in this case optimization can be made more efficient by taking gradient steps w.r.t. individual subfunctions..." [4] Allowing each mini batch to have influence over the decent allows for more versatile, and less drastic movement. This is highly beneficial, as the data from r/askscience is highly volatile and inseparable.

When the feed networks are created, the training data and the test data are fed through them. The outputs from each network corresponding to the probability that a given point belongs to their category is taken as the new data frame. The process can be visualized as such:

Sentence 1	label 1
Sentence 2	label 2
⋮	⋮
Sentence N	label N

↓

Sentence Embedding

↓

$V_{1,1}$	$V_{1,2}$	…	$V_{1,300}$	label 1
$V_{2,1}$	$V_{2,2}$	…	$V_{2,300}$	label 2
⋮	⋮			⋮
$V_{N,1}$	$V_{N,2}$	…	$V_{N,300}$	label N

↓

physics network
bio network
⋮
other network

↓

$P(\text{physics})_1$	$P(\text{bio})_1$	…	$P(\text{other})_1$	label 1
⋮	⋮		⋮	⋮
$P(\text{physics})_N$	$P(\text{bio})_N$	…	$P(\text{other})_N$	label N

So where we started with an N by 300 dimensional matrix of sentence embeddings, we now have an N by 7 dimensional matrix of probabilities. There was no need to save both probabilities from each feed network, as each output is just one minus the other. The data is now ready to be fed into the final neural network and SVM model.

C. Final Network

After the dimensionality reduction of the feed networks, our final network is allowed to have a lot fewer layers. Like the binary networks, it has 4 layers but this time without any dropout. Is has a 7 dimensional input and output, and has the following structure:

input	7
output	7
neurons per layer	12, 13, 15, 7
hidden activation	sigmoid
loss function	categorical crossentropy
optimizer	adam

Categorical cross-entropy is very similar to binary cross-entropy, with the function taking on the new form:

$$\text{loss} = -\sum_{c=1}^{M} y_o \log(p_{o,c})$$

Where $p_{o,c}$ is the probability that observation o falls into category c and y_o is the label of observation o. This change was necessary, as we no longer have a binary output. [13]

The sigmoid function was used for the hidden activation because it tested better than Leaky-Relu and Relu in empirical results. The sigmoid function is a smoother that sets our values between 0 and 1 by exponentiating the input to a given neuron like so:

$$g(z) = \frac{1}{1 + e^{-z}}$$

[5] The resulting output was able to perform better than a standalone network in a controlled data setting. (see simulations)

D. Feed SVM

An alternative to using another neural network is to use a support vector machines model. The benefit here lies mostly in time, fitting/tuning/modifying a SVM model is much faster and more intuitive than a neural network model. The best results were obtained with a linear kernel, and outperformed a similar SVM model with all 300 original dimensions.

III. Simulations

While testing, I will use 25% of the data as validation data, and the rest as a training set. This section will also be split into two subsections, toy data, and full data. Recall that the toy data comes from distinct subreddits and thus should be very easily separable.

A. Toy Data

To establish a baseline of performance, I ran the word embeddings from the toy data through an SVM model with a linear kernel (it did the best in empirical results) and a Neural network resembling the one I will use in the final step of the feed network (I did increase the number of neurons per layer to account for a larger input dimension). The results were immediately spectacular:

Toy Data

Method	Validation results
SVM	93.5%
Neural Network	94.3%

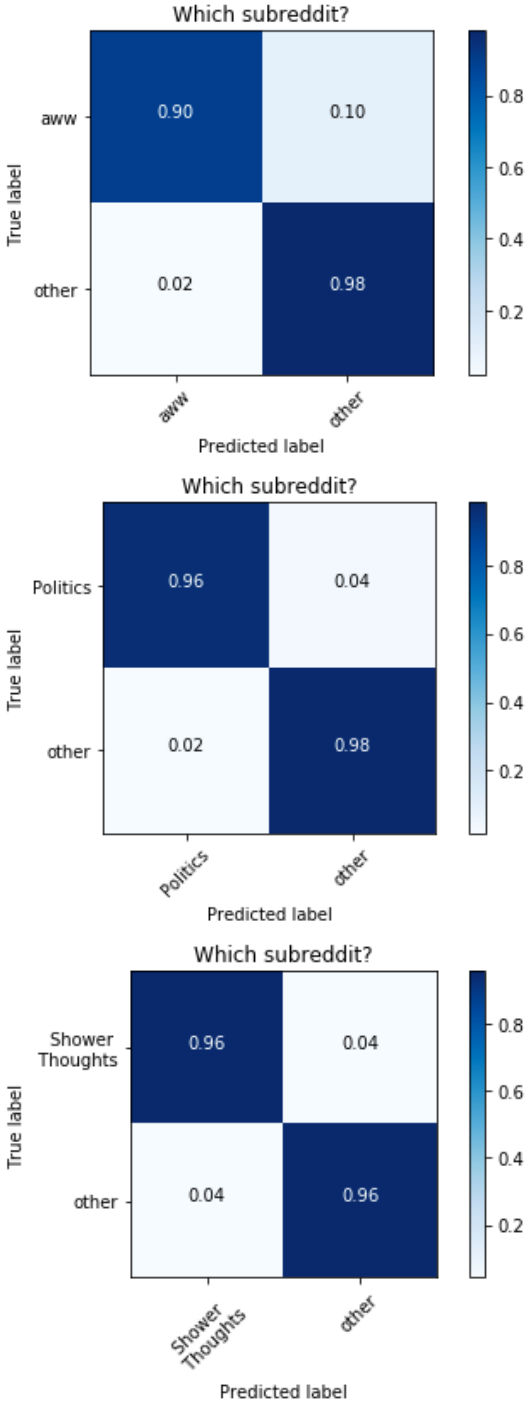
Thus setting a high bar for the feed networks to reach.

After running the embeddings through the three binary neural networks (which correspond to the number of unique labels) I obtained the following validation results:

Validation Results

Binary Label	Accuracy
r/Aww	95.1%
r/Politics	97.6%
r/Showerthoughts	95.6%

Corresponding to the following confusion matrices:



After concatenating the 'yes' outputs from each network, I obtained a new dataset that looks like so:

Output from 'Aww' Network	Output from 'Politics' Network	Output from 'Showerthoughts' Network	label
$P(x_1$=Aww)	$P(x_1$=Politics)	$P(x_1$=Showerthoughts)	Aww
$P(x_2$=Aww)	$P(x_2$=Politics)	$P(x_2$=Showerthoughts)	Aww
⋮	⋮	⋮	⋮
$P(x_{2826}$=Aww)	$P(x_{2826}$=Politics)	$P(x_{2826}$=Showerthoughts)	Politics

These three columns are run through another linear SVM model, and my final neural network, obtaining the following results:

Toy Data

Method	Validation results
SVM	93.5%
Neural Network	94.3%
Feed SVM	94.3%
Feed Neural Network	95.2%

Notice that both SVM and the neural network saw improvements in their accuracies after the data was treated with the feed networks.

B. Full Data

While the toy data had 3 unique labels, the full data has 7, so we will have 7 columns produced from our feed networks instead of 3. What is more, there is a good degree of arbitrariness as to which label gets assigned to which question. For example, "Do giraffes get struck by lightening more often than other animals?" is labeled as a physics question, when there could be a strong argument that it should be labeled as a biology question instead. As a result, I did not expect to see the same level of accuracy that I observed in the toy data set.

Our baseline SVM and neural network produced the following results:

Full Data

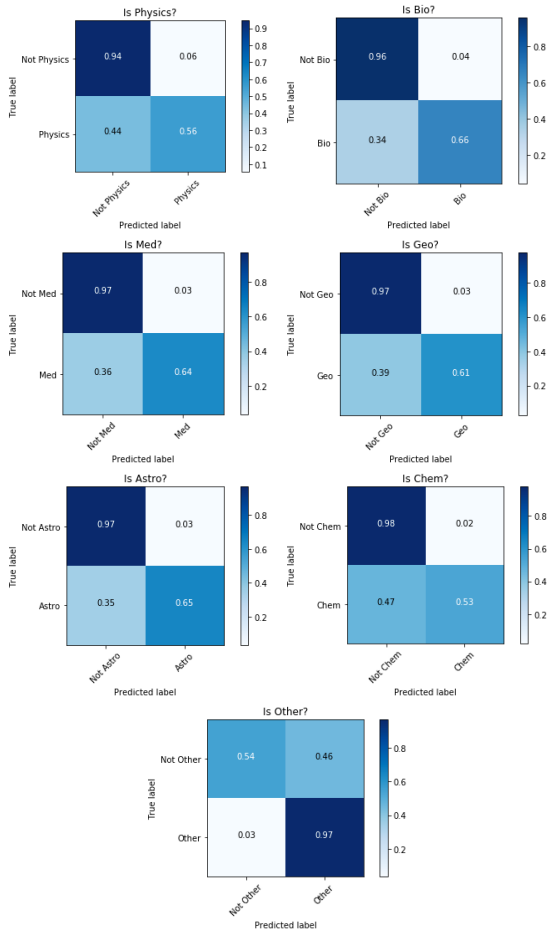
Method	Validation results
SVM	65.4%
Neural Network	65.9%

After feeding the data through the 7 binary networks (or feed networks) I obtain the following validation results:

Validation Results

Binary Label	Accuracy
Physics	86.1%
Bio	89.9%
Medicine	91.9%
Geology	92.8%
Chemistry	94.2%
Astronomy	94.1%
Other	89.9%

Corresponding to the following confusion matrices:



It is clear that there is a lot of confusion taking place in terms of the 'yes' category for each model.

After taking the output that corresponds to 'yes' from each network and row-binding them into a new matrix (for both the test and training data) I obtained a new dataset that looks like so:

output from 'Physics'	output from 'Bio'	...	output from 'Other'	Tag
$P(x_1$=Physics)	$P(x_1$=Bio)	...	$P(x_1$=Other)	Physics
$P(x_2$=Physics)	$P(x_2$=Bio)	...	$P(x_2$=Other)	Physics
⋮	⋮		⋮	⋮
$P(x_{4305}$=Physics)	$P(x_{4305}$=Bio)	...	$P(x_{4305}$=Other)	Physics

Taking this new data set and running it through the final neural network and SVM model I obtain the following results:

Full Data

Method	Validation results
SVM	65.4%
Neural Network	65.9%
Feed SVM	67.1%
Feed Neural Network	65.1%

So despite a large variable reduction, the feed networks did comparably well or better than their original counterparts. Specifically, SVM with the new reduced data set did the best by almost a 2% margine.

IV. Discussion

This variable reduction technique using binary neural networks (or as I like to call it, using feed networks) shows promise, but it still needs to be compared to more established techniques. Future work will include comparing it against other variable reduction techniques such as PCA, LDA, and Random Forest feature selection.

In terms of textual analysis, this project would greatly benefit from an examination into the optimal method of vectorization. I currently use Spacy because it is efficient and easy to use, but other methods such as GLoVe, Elmo, and bag of words might return a more accurate word vector representation.

This project was also limited by time and computational power. Given more of both, I would like to spend a more rigorous effort fine tuning the parameters of my networks and SVM models. This will be a project for the future.

Finally, once there are significant enough results for use, I would like to connect this method to its bot. That will require creating, automating and testing the bot. This is no small task, and will be a part of my future endeavours.

V. Conclusion

While there is still much work to do, I am very satisfied with the technique of neural network dimensionality reduction, or as I like to call it, feed network reduction. It guarantees that your new dimensions are the same as the number of unique labels in your data, and simulations show that it can produce results comparable to, or even better than, that of the original data.

From examining this data, it is safe to say that the r/askreddit data set is tricky. The labels are chosen by 10 different humans, all with their own opinions of what should belong where. As such, a 67% accuracy is something that I am quite happy with. Especially when one observes the confusion matrix associated with that result. (figure 2) My best model to date uses feed network reduction for pre-treatment, and then an SVM model for full classification.

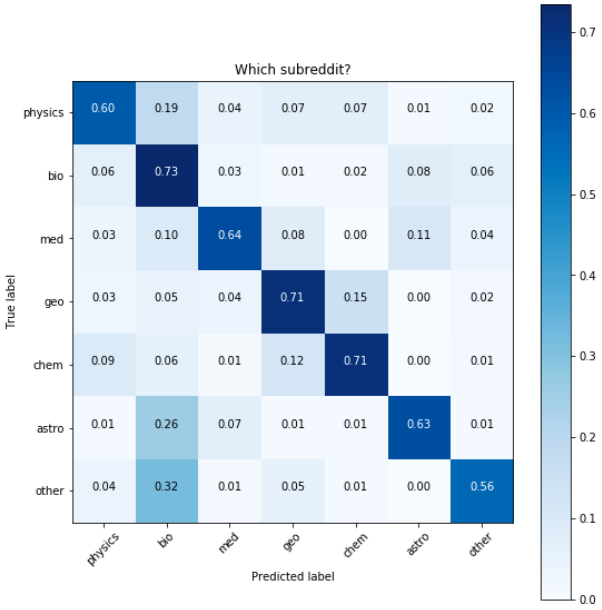


Fig. 2. Note the hot areas, where Other and Astro get confused often for Bio. Without them, there is a strong patter of correctly classified points

In terms of the goal of creating a redditbot, I would like stronger results before attempting to implement it in the actual website. What is more, the current moderators of r/askscience are not allowing outide bots to run on their subreddit. While I am appealing their decision, I will not create a bot against their will. You can track its progress at my website: www.wbbpredictions.com.

References

[1] similar web. "Reddit.com Traffic Statistics." *SimilarWeb*, 2018, www.similarweb.com/website/reddit.com#overview.

[2] r/askscience, 'Reddit', 12/9/2018, www.reddit.com/r/askscience

[3] Boe B. PRAW: The Python Reddit API Wrapper. 2012-, https://github.com/praw-dev/praw/ [Online; accessed 2017-09-29].

[4] Diederik P. Kingma and Jimmy Ba. Adam: *A Method for Stochastic Optimization*.arXiv e-prints, page arXiv:1412.6980, December 2014.

[5] Chen G. LEC 8: Artificial Neural Networks (ANN). 2018.

[6] 4.5.6, and 9.3. An Introduction to Statistical Learning: with Applications in R. *An Introduction to Statistical Learning: with Applications in R*, by Gareth James et al., Springer, 2017.

[7] Shanelynn. Word Embeddings in Python with Spacy and Gensim. Shane Lynn, 26 Mar. 2018, www.shanelynn.ie/word-embeddings-in-python-with-spacy-and-gensim/.

[8] Hinton, G. E., and R. R. Salakhutdinov. Reducing the Dimensionality of Data with Neural Networks. Science, American Association for the Advancement of Science, 28 July 2006, science.sciencemag.org/content/313/5786/504.full.

[9] Mikolov, Tomas, et al. "Efficient estimation of word representations in vector space." arXiv preprint arXiv:1301.3781 (2013).

[10] Turian, Joseph, Lev Ratinov, and Yoshua Bengio. "Word representations: a simple and general method for semi-supervised learning." Proceedings of the 48th annual meeting of the association for computational linguistics. Association for Computational Linguistics, 2010.

[11] Levy, Omer, and Yoav Goldberg. "Neural word embedding as implicit matrix factorization." Advances in neural information processing systems. 2014.

[12] Goldberg, Yoav, and Omer Levy. "word2vec Explained: deriving Mikolov et al.'s negative-sampling word-embedding method." arXiv preprint arXiv:1402.3722 (2014).

[13] Drozdzal, Michal, et al. ”The importance of skip connections in biomedical image segmentation.” Deep Learning and Data Labeling for Medical Applications. Springer, Cham, 2016. 179-187.