Neural Networks applied to protein-protein interaction abstract text mining

Codrin-Andrei Ripa(codrin.ripa87@e-uvt.ro), University Of West Timisoara
Prof. Carlos Manuel Jorge da Silva Pereira, PhD, Instituto Superior de Engenharia de Coimbra

The millions of biomedical articles available make it difficult for human researchers to extract relevant data. Our research focused on building a web application that allows dynamic, emergent indexing of Protein-Protein Interaction(PPI) scientific articles. Several approaches were considered, including the replication of a recent success in machine learning applied to the PPI problem. Two networks were created, one for protein recognition in text, and another for PPI classification. They were subsequently integrated in a web application. Success was been limited, but is sufficient as proof of concept for further use.

INTRODUCTION

A. The Problem

The collected genetic material of an organism is called a genome or genotype. The way it is expressed in an organism as it develops, especially considering external factors, is the phenotype. At a base level, the DNA is a backed up genetic code base with four key elements(nucleotides) in various arrangements. The double spiral splits into halves and unfolds into RNA, which bonds with the free organic molecules in the cell nucleus. Codons, units of three nucleotides, work to program aminoacid compounds. Multiple codons called create a chain of such amino acids which is known as a protein. In an organism, a protein serves multiple purposes, from chemical signals to support structures.[1]

Given the ubiquity and usefulness of proteins in organisms, scientists study the interactions between proteins and summarize them in scientific articles. Pubmed in particular featured 30 million citations on biomedical literature in May 2020[2], and even if protein-protein interaction articles comprise only a small subset of these, they still number in the tens of thousands, therefore being beyond the ability of human beings to filter and select. Applications in computer science therefore become a requirement and necessity in accessing relevant knowledge concerning protein-protein interactions.

B. Vision And Goals

With this project we proposed to create a web application that allows for emergent organization and accessing of protein-protein interactions published through Pubmed. Rather than build an indexer which iterate through all the Pubmed articles and extract relevant articles in a slow process, the web application would take requests from users and search for the relevant articles within a short time. These searches would then be stored in a database for rapid access in the future. This would allow for the construction of a database in real time, based on popular demand according to the latest research trends.

C. Development Methodology and Plan Schedule

The development of the web application has been SCRUM based. SCRUM is an agile software development method that emphasizes quick production of software products through cycle based iterations resulting in early delivery, evolutionary development and continuous improvement.[3] Our SCRUM approach was based on two week sprints, goals, and reports, concluding after 3 months with 7 cycles detailed below.

1. Study of biomedical concepts and natural language processing techniques.
2. Reviewing the computer science literature on the subject of protein-protein interactions
3. Attempts at implementing the Zhang et al neural network model.
4. Attempts at applying alternate neural network models.
5. Building protein recognition models and investigating Pubmed article fetching
6. Studying Django framework, graph representations and designing the database.
7. Integrating the various components in the final form of the web application.

D. Structure of the report

The report has three sections:

- State of the art discusses the previous attempts in machine learning applications to data mining protein-protein interactions.
- Web Platform displays the design for our web application.
- Results summarizes and discusses the training of the neural networks.

STATE OF THE ART

A. Learning algorithms

Given the vast field of natural language processing(NLP) as well as machine learning(ML) it is necessary to review some of the current literature and the latest methodologies.

Word vectors

Converting strings to numerical values is an essential component of any computer science application that deals in computation and analysis. In the machine learning fields it is common practice to convert words into vectors of a multidimensional vector space, permitting various comparisons and operations henceforth. These vectors are obtained by passing selected text corpora through neural networks which are trained to output specific vectors for words. The summary of such trainings are collected in matrices of (vocabulary, dimension) sizes. We have considered and used three types of word vector encodings called embeddings.[4]

The first is the classic word2vec, which relies on contextual word vectors. A context window is used as a dimension for the desired word vectors, with the target word encoded as the occurrences of other words around it. These encodings are trained in a semisupervised manner in two ways: continuous bag-of-words (CBOW), in which the word is predicted based on the current context, and skip-gram, which predicts the context based on the target words. Both of these allow for words that are interchangeable to obtain similar word vectors (e.g. "child playing with a toy" vs "kid playing with a toy"), with the semisupervised training reducing the distance between similar words.[5]

The downside of word2vec is having to choose one of the two training approaches, limiting the predictive efficiency of one type or the other. By contrast GloVe embeddings instead consider a single co-occurence factor for words in a context, and its semisupervised learning process attempts to minimize the word vector based on a logarithm of this factor. In essence, the GloVe embedding training solves the word2vec duality by applying a single algorithm for both.[6]

fastText embeddings are trained with a different approach. Certain embeddings can be enriched by considering not only single words, but collections of words(e.g. bald eagle, blue whale) called n-grams. fastText uses a subset of n-grams called skipgrams, encoding each word as a collection of subwords (intuitively, the syllables of a word) in a context. Since such skipgrams are likely to be morphologically common(similar syllables occur in different words) it then becomes easier to construct word vectors for words not in the original corpora(assuming they include common syllables).[7]

Using pre-trained embeddings for encoding words into vectors gives an advantage, but it is important to remember that such embeddings are static weights which have a one-to-one correspondence to the words they encode. Embeddings are not predictive functions capable of outputting totally unknown words not originally found in the corpora. As a result, it is important to carefully select the correct embeddings for the task. Since ours is biomedical text mining, one useful embedding is the one provided by scispacy, trained on biomedical data with a ~100k vocabulary.

The inevitable result of using the above embeddings on a sentence is a matrix of all the words encoded as vectors. Rather than processing the words one at a time or generalizing the sentence to a single vector, we can use the whole resultant matrix, henceforth referred to as a sentence matrix, for study and processing.

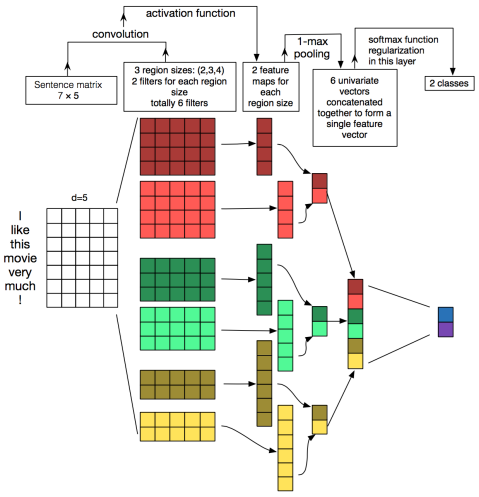
Neural Networks

Intuitively, the neural network approach to machine learning itself consists of adaptive functions. These neural networks are fed thousands of examples and are then evaluated by training functions which correct the parameters of the neural networks.

Mathematically, neural networks are compositions of complex linear algebra operators trained through a process called backpropagation. At each training iteration, these operators are evaluated based on averages of the difference between the desired outputs and the training outputs. Since each neural network is a composition of operators, the average effectiveness of our operator is calculated based on chain derivatives for each of the composing operators, allowing for fine tuning of individual parameters in individual operators. Training iterations are repeated until the desired effect is obtained, or until the researched modifies the training parameters, named hyperparameters.

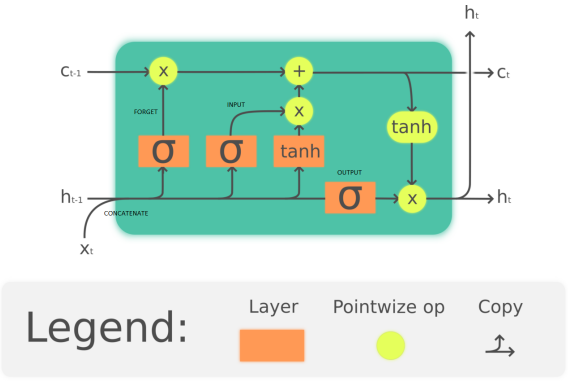
The first classic architecture we will consider is the multi layer perceptron network(MLP), a feedforward network composed of multiple layers of perceptrons with non-linear activation functions, using backpropagation techniques for training. To delve a bit into the details, perceptron layers can be conceptualized as a matrix multiplication between the parameter weights and the input of the previous layer, the first of these layers being the actual network input(training examples) and the final layer being the output. Feed forward signifies that there are no loops inside this network, the inputs passing from one layer to the other without being re-fed into the network, as would be the case with a recurrent neural network. MLPs have been successfully used in a wide variety of computer science fields, being especially useful in optimization problems.[8]

A second architecture to consider are convolutional neural networks(CNN). While still feedforward networks, they instead make use of mathematical convolutions, which can be intuitively understood as specialized integrals. Convolutions have been studied since 1754[9], and have a well known algebra[10]. Convolutions have been previously used in the field of text analysis[11], especially 1 dimensional convolutions[12]. In contrast to MLPs, CNN operators are a collection of small windows that filter through the input, and therefore have fewer parameters, therefore making them more computationally desirable both in training and prediction, leading to faster training times and lower memory networks. These lightweight features allow for architectures of dozens of layers in what is called deep learning[13], a paradigm which has been applied in NLP.[14]



CNN in NLP

A final, newer approach to consider are recurrent neural networks(RNN). In contrast to feedforward networks, RNNs retain the previous input fed to them, expressing a temporal sequence and working towards a cumulative output. Using this memory, the input itself becomes variable, offering an advantage over the fixed dimension sizes of the input required by MLPs or CNNs. The cost, however, is increased computational demands, both in training and prediction. For a particular case study, Long short-term memory (LSTM) networks are actually units formed out of multiple neural networks, each with a different role. One is a forget gate, learning to filter out unnecessary input from previous iterations. Another is an input gate, pushing forward the relevant a copy of the current input to the next layer, and finally is an output gate, using the combined selected result for the output. LSTM are particularly useful for NLP as they allow for variable sentence size.[15][16]



LSTM unit

Previous attempts at data mining protein-protein interactions from the abstracts of scientific articles have of course been attempted. Miyao et al provide a meta-analysis of classic parsers applied to the problem. Accuracy, precision, recall and F-scores for such approaches revolve around 0.5[17]

Competitions related to protein-protein interactions have been set up, notably BioCreative. For the second edition, 26 teams competed with various methods, the top team consisting of Alex B et al reaching a precision of 0.37 with recall of 0.33 and F-score of 0.78[18]. Subsequent articles have noted the inconsistency and difficulty of the annotations in the BioCreative datasets[19]. The annotation of protein-protein interactions remains largely a problem, with teams such Jaerger et al reporting an 80% success rate[20]. Practical implementation are still non-existant.

Finally, we have settled on the deep convolutional model of Zhang et al[21]. Their article contains a meta-analysis of previous used methods as well, and concludes that RNNs and combinations with parsers offer some efficiency, but are computationally expensive. On the other hand, deep convolutional networks often fail in training because of the vanishing/exploding gradient problem. This is a well known problem in backpropagation that occurs because the limits and derivatives applied to the paramaters of a neural network reach outside of programming limits[22].

The Zhang et al model is based on 3 components: an embedding layer, a convolutional layer, and a classification layer. The classification layer is merely a pool for the network output. For the embedding layer, they use mark proteins by replacing them with the keywords PROTEIN1 and PROTEIN2 for the sought after protein-protein interaction, and PROTEIN0 for any other proteins.

Their convolutional layer merits more discussion. Their main unit is a residual convolutional block. This block consists of multiple sub-blocks containing three 1D convolutions of 64 window size, ending with a batch normalization. The input of this sub-block is added to the output through what is known as a skip connection. Skip connections are frequently used in deep convolutional neural networks applied to image processing.[23][24][25]

After testing, Zhang et al have settled on 6 residual convolutional blocks. They trained and tested their model on on five benchmark PPI corpora consisting of AIMed, BioInfer, HPRD50, IEPA, and LLL, collected, transformed and shared by Pyysalo et al.[26]. The end result has been precision, recall, and F1-score of in the range of 80%. Given such results, we had originally settled on their model. As the results section described, results were less than stellar.

WEB PLATFORM

A. Requirement Analysis.

The majority of machine learning platforms and tools are Python based. Notably, Keras is a user friendly, TensorFlow based open source machine learning library.[27] To augment To ease interaction, our web platform backend must also be Python based. For this purpose, we have chosen Django[28], a model view controller(MVC) web framework based on Python. Additionally, representing the protein-protein interactions in a visual manner involves a front end graph representation. For this purpose we used D3.js[29], a javascript library for interactive data visualizations in web browsers.

Separately, we have trained a protein recognition network using scispaCy[30], a Python package containing spaCy models for processing biomedical, scientific or clinical text. spaCy itself is a natural language processing open-source library depending on pretrained GloVe embeddings.[31]

B. Architecture

The researcher went with a simple but encompassing web application based on a search engine paradigm. The user is encouraged to enter a protein and select a level of depth for the interaction. If this protein exists in our database, based on the level of depth, a protein interaction network is displayed, expanding from the original protein.

By clicking on a node in the network, a new browser tab is opened and the user is taken to the corresponding scientific article on pubmed. This is achieved by are recursive search of the sql database based on the first protein. The result is compiled as a python dictionary which is then converted to a javascript object. We pass the result to a d3js function for generating an interactive SVG image which displays the protein interaction network.

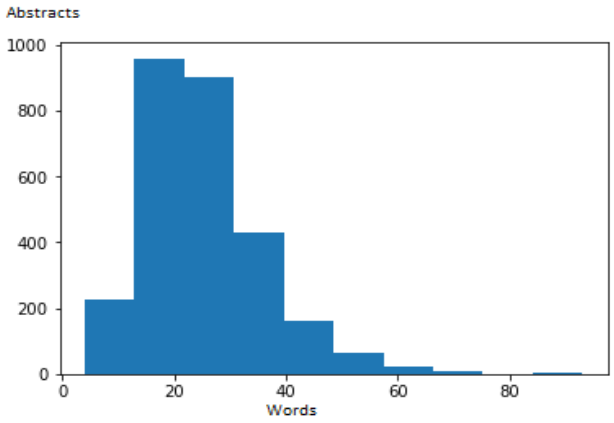
In case the user enters a protein that isn't in the database, the app checks if the entered word is indeed a protein, then adds it to a To Check list. This list is verified by a crawler thread every ten seconds. If the To Check list is not empty, the webcrawler compares the proteins in the To Check list against those already in the database. Ultimately this will be done by retrieving and checking pubmed articles. If an interaction is found, the protein is added to the database, along with a through table entry detailing the article specifying the interaction. The images in Annex A better explain the architecture.

RESULTS

A. Datasets

For the protein-protein interactionhe researcher has relied on five benchmark PPI corpora consisting of AIMed, BioInfer, HPRD50, IEPA, and LLL, collected, transformed and shared by Pyysalo et al.[26]. We briefly considered using the Biocreative II dataset, but abandoned it given their limited examples and annotation problems. Our preprocessing involved extracting the abstracts and proteins, converting the abstracts in lowercase, and replacing the suspected proteins with PROTEINA and PROTEINB. The rest of the proteins were left untouched.

After parsing and preprocessing, we obtained 11000 examples of protein-protein interactions. The average abstract was found to have 50 words, with a standard deviation of . As such, we curated our dataset to 10000 examples while zero padding the abstracts under 50 words, considering 50 the size of our input.



Additionally we made use of the STRING database[32] for obtaining protein preferred names in biomedical literature. 10000 examples of protein names were collected as strings. Both datasets were split into a 4:1 training:test ratio.

B. Classifier Results

	acc train	acc test	f1	precision	recall
Recognition	0.9887744188308716	0.9882808327674866	0.9882807695341203	0.988279960267048	0.9882818790910916

Fig. 1 Recognition Network

Accuracy	word2vec train	word2vec test	glove train	glove test	fasttext train	fasttext test
Dense	0.7407158017158508	0.707619845867157	0.7455262541770935	0.7277397513389587	0.7578410506248474	0.6917808055877686
CNN	0.8091206550598145	0.7315924763679504	0.794111967086792	0.704623281955719	0.7107946872711182	0.7324486374855042
LSTM	0.8508754968643188	0.7140411138534546	0.864537239074707	0.7260273694992065	0.8509716987609863	0.6930650472640991

Fig. 2A LR=1

Accuracy	word2vec train	word2vec test	glove train	glove test	fasttext train	fasttext test
Dense	0.7107946872711182	0.7324486374855042	0.7107946872711182	0.7324486374855042	0.7107946872711182	0.7324486374855042
CNN	0.7107946872711182	0.7324486374855042	0.7107946872711182	0.7324486374855042	0.7107946872711182	0.7324486374855042
LSTM	0.8144121766090393	0.6720890402793884	0.8173946738243103	0.616866409778595	0.813450038433075	0.6750856041908264

Fig. 2B LR=0.01

Accuracy	word2vec train	word2vec test	glove train	glove test	fasttext train	fasttext test
Dense	0.7107946872711182	0.7324486374855042	0.7107946872711182	0.7324486374855042	0.7107946872711182	0.7324486374855042
CNN	0.7107946872711182	0.7324486374855042	0.7107946872711182	0.7324486374855042	0.7107946872711182	0.7324486374855042
LSTM	0.7107946872711182	0.7324486374855042	0.7126226425170898	0.7311643958091736	0.7107946872711182	0.7324486374855042

Fig. 3 SGD

Accuracy	word2vec train	word2vec test	word2vecMed train	word2vecMed test	glove train	glove test	fasttext train	fasttext test
Dense	0.7753511667251587	0.628852725028917	0.7500481009483337	0.639126718044281	0.7565903663635254	0.7033390402793884	0.7710217237472534	0.6699486374855042
CNN	0.7107946872711182	0.7324486374855042	0.7107946872711182	0.7324486374855042	0.7107946872711182	0.7324486374855042	0.7107946872711182	0.7324486374855042
LSTM	0.9914373755455017	0.6605308055877686	0.987877607345581	0.6635273694992065	0.9931691288948059	0.6797945499420166	0.9882624745368958	0.5984588861465454

Fig. 4 Additional Embedding

Accuracy	word2vecMed train	word2vecMed test	glove train	glove test	fasttext train	fasttext test
Dense	0.8155666589736938	0.6613869667053223	0.8225899338722229	0.6626712083816528	0.8018087148666382	0.6207191944122314
Really Dense	0.7107946872711182	0.7324486374855042	0.7107946872711182	0.7324486374855042	0.7107946872711182	0.7324486374855042
LSTM	1.0	0.6258561611175537	1.0	0.6879280805587769	1.0	0.65625
Dropout LSTM	1.0	0.6571061611175537	1.0	0.6459760069847107	0.9996151328086853	0.6515411138534546

Fig. 5 Dropping the CNN

Accuracy	word2vecMed train	word2vecMed test	glove train	glove test
Dense	0.7932460904121399	0.6618150472640991	0.7107946872711182	0.7324486374855042
Really Dense	0.7107946872711182	0.7324486374855042	0.7107946872711182	0.7324486374855042

Fig.6 MLP

	acc train	acc test	f1	precision	recall
Zhang	0.9976909756660461	0.6776540875434875	0.5509085826111266	0.5602643041207727	0.5494241963763881
MLP	0.7107946872711182	0.7324486374855042	0.422782307882382	0.3662243150684932	0.5

Fig7. Zhang vs MLP

C. Discussion

Our protein recongition network is a simple n layer MLP that worked as a binary classifier. Protein strings were vectorized using scispaCy for speed and convenience. The task was not difficult, which is why a 98% success rate across all metrics was obtained(Fig. 1). However, the strength of this network depends entirely on the list of proteins used for training. The researcher found the list to not be well curated, containing examples such as "beta-glucosidase/6-phospho-beta-glucosidase/beta-galactosidase" and a large number of chemical shorthands[https://www.anaspec.com/html/chemical_abbreviations.html] such as "cyp716a1". Given this dataset, some practical errors in the network are to be expected.

Although the reaseacher had high hopes for the Zhang et al model combined with the scispaCy vectorization, after training in accordance to their hyperparameters the results were less than stellar(Fig. 7). We cannot comment on why these results occurred, only that we were unable to replicate their results on the same dataset. Since Zhang et al have not provided a practical implementation, the researcher could not continue in this direction.

In an attempt to salvage the project, other alternatives were tried, namely three types of pretrained embeddings of 300 dimension(word2vec, GloVe, and fastText trained on wikipedia and google corpora) along with three types of neural network architectures:

- MLP - A 5 layer funnel architecture: 2000, 1000, 500, 200, 50, 1D GlobalMaxpool, sigmoid activations

- CNNs -A 3 Layer :
  - kernel sizes:(3,3),(4,4),(4,8))
  - strides: 3,4,8
  - filters: 100, 200, 300
  - relu activations
  - GlobalMaxPooling2D at the end

- LSTM: A one layer network, with 200 units, a tanh activation and sigmoid reccurent activations.

The researcher implemented a grid search consisting of the three embeddings and networks, focusing on accuracy as a quick shorthand for measuring performance. We tried various measures, from tweaking hyperparameters (Fig 2A and 2B), changing the optimizer to Stochastic Gradient Descent(Fig 3), introducing an embedding trained on biomedical text(Fig 4). Clearly the MLP outpaced the other approaches(Fig 5), and was studied in more detail by introducing an additional layer(Fig 6), with more advanced classifier metrics implemented.

We then took a more concrete look at the metrics between the trained MLP network and the trained Zhang model. The first is more accurate, but produces more false positives. Both produce the same rate of true positives(about a coin toss). In the end, out of two lesser evils, we decided for the MLP to export and use on the web platform.

CONCLUSION AND FUTURE WORK

Given the amount of work required(parsing, neural network training, front end, back end, database design), short development time(3 months) and the inexperience of the researcher, the result can only be summed up as a proof of concept. While a working web application has been produced and put into use, it's weakness is the neural networks.

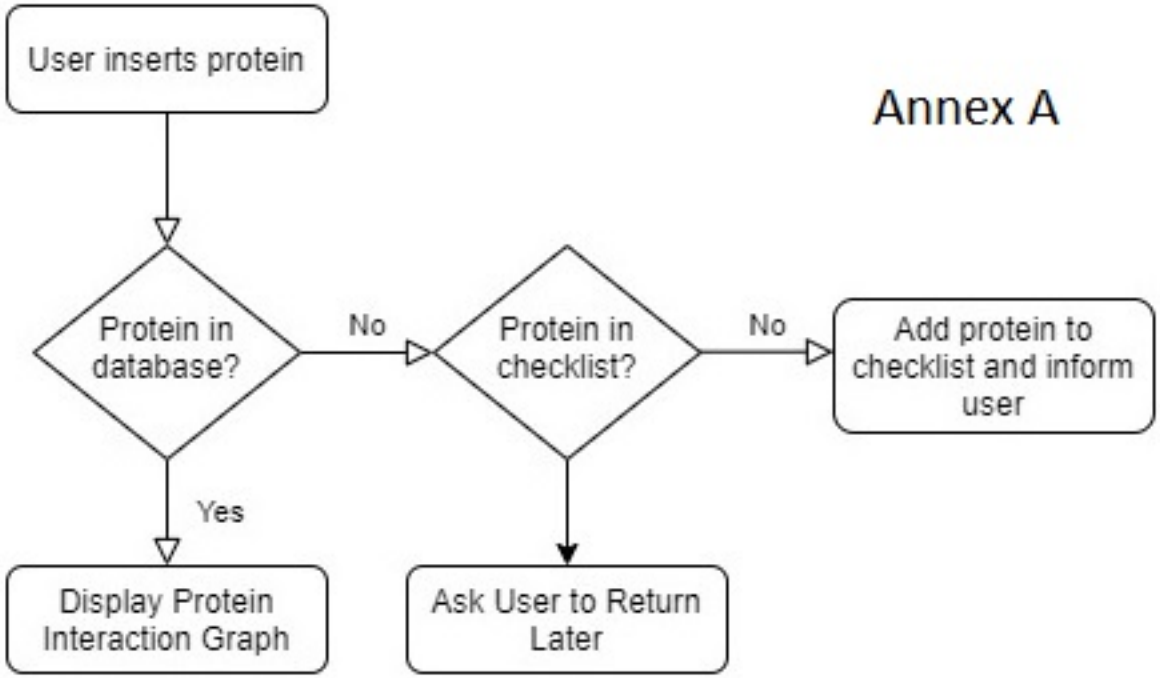
Future efforts in this direction need to start with better, larger curated datasets that more accurately represent the problem. Many Pubmed abstracts are over 100 words in length, and the practical implementation has been forced to split these and average out the results of the parts.
More advanced neural networks are also worth investigating, along with more fine tuning of hyperparameters.

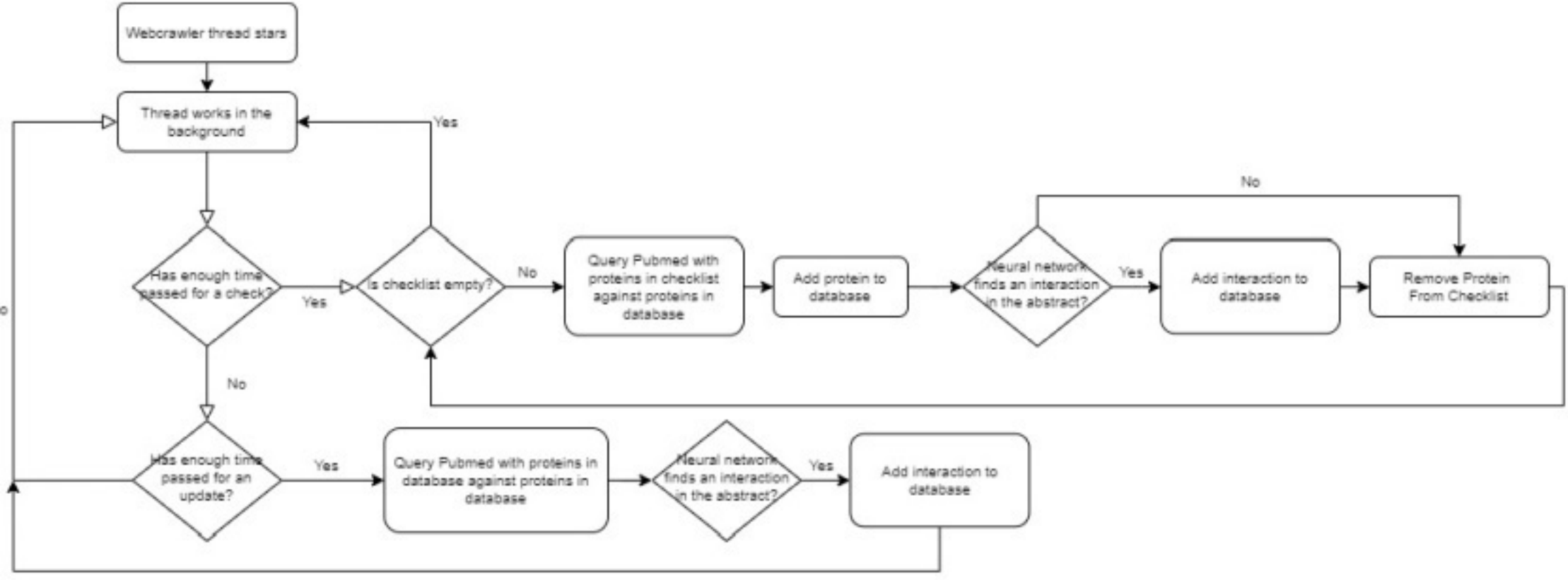
On the web platform side, more robustness is needed. Not enough failsafes are in place for the various errors and exceptions that are likely to occur.

References

1. Genetics for Dummies®, 2nd Edition, Tara Rodden Robinson
2. https://pubmed.ncbi.nlm.nih.gov/
3. https://www.scrumguides.org/docs/scrumguide/v2017/2017-Scrum-Guide-US.pdf
4. https://www.ijcai.org/Proceedings/15/Papers/513.pdf
5. https://arxiv.org/abs/1301.3781
6. https://www.aclweb.org/anthology/D14-1162/
7. https://arxiv.org/abs/1607.04606
8. "Machine Learning textbook". www.cs.cmu.edu. Retrieved 2020-05-28.
9. Dominguez-Torres, Alejandro (Nov 2, 2010). "Origin and history of convolution". 41 pgs.
10. http://www.slideshare.net/Alexdfar/origin-adn-history-of-convolution. Cranfield, Bedford MK43 OAL, UK. Retrieved Mar 13, 2013.
11. von zur Gathen, J.; Gerhard, J. (2003), Modern Computer Algebra, Cambridge University Press, ISBN 0-521-82646-2.
12. https://ieeexplore.ieee.org/abstract/document/8244338
13. https://www.scitepress.org/Papers/2018/67305/67305.pdf
14. https://www.semanticscholar.org/paper/Deep-learning-in-neural-networks%3A-An-overview-Schmidhuber/193edd20cae92c6759c18ce93eeea96afd9528eb
15. https://arxiv.org/abs/1703.03091
16. https://www.aclweb.org/anthology/P16-2037.pdf
17. https://www.aclweb.org/anthology/S17-2126.pdf
18. https://academic.oup.com/bioinformatics/article/25/3/394/244043
19. https://link.springer.com/article/10.1186/gb-2008-9-s2-s4
20. https://genomebiology.biomedcentral.com/articles/10.1186/gb-2008-9-s2-s1
21. https://link.springer.com/article/10.1186/1471-2105-9-S8-S2
22. https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=8756171
23. S. Hochreiter. Untersuchungen zu dynamischen neuronalen Netzen. Diploma thesis, Institut f. Informatik, Technische Univ. Munich, 1991.
24. https://arxiv.org/abs/1606.08921
25. http://papers.nips.cc/paper/6172-image-restoration-using-very-deep-convolutional-encoder-decoder-networks-with-symmetric-skip-connections
26. https://link.springer.com/chapter/10.1007/978-3-319-46976-8_19
27. http://mars.cs.utu.fi/PPICorpora
28. https://keras.io/
29. https://www.djangoproject.com/
30. https://d3js.org/
31. https://allenai.github.io/scispacy/
32. https://spacy.io/
33. https://string-db.org/cgi/about.pl

Annex A


User inserts protein
Protein in database?
No
Protein in checklist?
No
Add protein to checklist and inform user
Yes
Display Protein Interaction Graph
Ask User to Return Later


Webcrawler thread stars
Thread works in the background
Has enough time passed for a check?
Yes
Is checklist empty?
Yes
No
Query Pubmed with proteins in checklist against proteins in database
Add protein to database
Neural network finds an interaction in the abstract?
No
Yes
Add interaction to database
Remove Protein From Checklist
No
Has enough time passed for an update?
Yes
Query Pubmed with proteins in database against proteins in database
Neural network finds an interaction in the abstract?
Yes
Add interaction to database