Feature engineered embeddings for machine learning on molecular data

Claudio Jardim 17029008

WST895 Mini-dissertation: Mathematical statistics

Submitted in partial fulfillment of the degree MSc Advanced Data Analytics

Supervisor(s): Dr. A de Waal

Department of Statistics, University of Pretoria

28/11/2022

Abstract

The classification of molecules is of particular importance to the drug discovery process and several other use cases. Data in this domain can be partitioned into structural and sequence/text data. Several techniques such as deep learning are able to classify molecules and predict their functions using both types of data. Molecular structure and encoded chemical information are sufficient to classify a characteristic of a molecule. However, the use of a molecule's structural information typically requires large amounts of computational power with deep learning models that take a long time to train. In this study, we present a different approach to molecule classification that addresses the limitations of other techniques. This approach uses natural language processing techniques in the form of count vectorisation, term frequency-inverse document frequency, word2vec and latent Dirichlet allocation to feature engineer molecular text data. Through this approach we aim to make a robust and explainable embedding that is fast to implement and solely dependent on chemical (text) data such as the sequence of a protein. Further, we investigate the usefulness of these explainable embeddings for machine learning models, for representing a corpus of data in vector space and for protein-protein interaction prediction using embedding similarity. We apply the techniques on three different types of molecular text data: FASTA sequence data, Simplified Molecular Input Line Entry Specification data and Protein Data Bank data. We show that these embeddings provide excellent performance for classification and protein-protein bind prediction.

Acknowledgements

I have learnt a lot through this experience and I am grateful for the opportunity to write this dissertation. It has been difficult at times and enjoyable throughout. This challenging undertaking could not have been accomplished without the support I received along the way. I would like to thank:

- SilicoGenesis for allowing me to use their platform. The Silicogenesis team for their assistance and particularly Dr Dean Sherry for his expertise.
- My parent for their love, support and for providing me with a good education.
- My sister for her support and interest in this dissertation.
- I also want to express my deep gratitude to my supervisor: Dr Alta de Waal for her patient guidance, knowledge, support and endless ideas.

Ethics number: NAS124/2019

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The drug discovery process is very complicated and consists of many stages. Machine learning (ML) techniques have proven useful in several stages of the drug discovery and novel drug therapeutics creation process. Although ML cannot replace all of the classical methods in drug discovery, they can improve the process and assist in its completion. An example of this is molecular property prediction for drug screening. One can filter out molecules that do not contain the desired properties using molecular property prediction before further work is done on these molecules. This process can save time and money if it is performed accurately.

Some of the most commonly used techniques in this space are natural language processing (NLP) and deep learning (DL) [3, 24, 13, 41, 50, 56]. Current approaches use DL models and structural datasets that can result in increased computational cost, compute time and overall complexity. We do not aim to compete with these methods but rather to provide an explainable and computational effective alternative that is easily reproducible.

In this study, we predict the biophysical and physiological properties of general molecules expressed in the Simplified Molecular Input Line Entry Specification (SMILES) data type [55]. We then focus on proteins by predicting the classification of proteins using their residue sequences and attempt to find the best binding partner from a group of proteins given a query protein using extracted data from Protein Data Bank (PDB) files [5]. We only make use of the available text data with no structural information or added features. Any reference to text data will be the molecular text data obtained from the SMILES, FASTA [60] and PDB files. This molecular text data simply represents specific chemical and biological properties of a molecule, such as the amino acid residues that make up a protein molecule. We create embeddings of this molecular text data using count vectorisation (CVec), term frequency-inverse document frequency (TF-IDF), word2vec [35] and latent Dirichlet Allocation (LDA) [8] for both the molecular property prediction tasks and the protein tasks.

For the protein classification and molecular property prediction, we use three models in the form of a support vector machine classifier (SVM) [9], naïve Bayes classifier (NBC) and a NN (NN) classifier. We evaluate the usefulness and limitations of the embedding techniques and the ML classification techniques applied to the molecular text data. The focus of this study is the feature engineered embeddings and not the models. We assume default parameters [37] for all models unless otherwise stated. These models can be improved through hyperparameter tuning in the form of grid search and other techniques, but this is not the objective of this study. The objective of the study is to show how feature engineering can improve the baseline (or perform equally for much less complexity). The Moleculenet [56] collection of datasets is a benchmark created for evaluating ML techniques. We predict blood-brain barrier penetration (permeability) and qualitative binary binding results for a set of inhibitors of human β-secretase 1 (BACE-1). The molecules are represented as SMILES strings in these datasets.

We provide a special focus on large biomolecules consisting of chains and amino acid residues that are commonly known as proteins. These biomolecules are vital in many biological activities with each protein having a function. Proteins have properties that are uniquely determined by the chains of amino acids in the protein and its structural composition. Whilst some proteins function individually; the majority of proteins bind with other proteins in order to perform their biological functions. One way in which novel therapeutics can be created is through blockers. Blockers are drug candidates that can bind with a target protein and prevent that target protein from interacting with another protein. The prevention of these interactions can result in prevention of a disease. The process of finding a drug candidate that will bind to the target is called virtual screening. Due to the amount of data available it is possible to screen known drug candidates for a given target using ML techniques. We demonstrate the performance of our method on protein classification using FASTA data and dedicate a separate section to protein-protein interaction prediction using the PDB data.

DL and in particular geometric DL models have shown promising or state-of-the-art results for protein-protein interaction prediction [50, 25, 48]. However, there are limitations when applying this technique to protein-protein interaction prediction and structural data in general. These approaches are either applied to generated protein surface point clouds or directly to the atomic coordinate data obtained from the PDB files and other structural file formats. DL approaches can result in complex models that require large amounts of data and compute to train. If the right amount of data is not available these techniques can overfit and will then not generalise well to unseen data. These techniques are often implemented for point-level or atom-level prediction which usually results in a large class imbalance that can greatly alter metrics. Thus, point-level and atom-level prediction models require a solution such as sampling to correct the class imbalance. In general, there is a lack of molecule-level prediction techniques.

A molecule-level prediction would be sufficient for many classification tasks and even protein-protein

interaction prediction. We believe it is of practical benefit to implement a technique that can search a database of proteins in order to find the binding partner of a given query input protein. The atom-level binding site of the predicted binding partner protein or drug candidate can then be predicted through current DL techniques - if it is required. This proposed method is practically useful on its own but can also be enhanced by using atom-level or point-level techniques. Instead of replacing current techniques, we aim to provide a solution to combat their weaknesses and improve their capabilities.
We do this by means of applying feature engineered embeddings - resulting from techniques such as latent Dirichlet allocation (LDA) [8] - on the molecular text data. We believe that protein-protein interaction prediction can be achieved using the feature engineered embeddings and a histogram approach based on the Bhattacharyya distance. We believe that the information required to differentiate proteins and molecules is captured in the embeddings alone. This should greatly decrease the computational effort required to make molecular predictions and decrease the size of the files required for this task.

1.1 Research contributions

The dissertation contributes through the use of embeddings as a set of features for ML models, for representing a corpus of data in vector space (embedding similarity) and then, the use of semantic calculations on these representation to predict protein-protein interaction.
Four embedding techniques in the form of LDA, TF-IDF, CVec and word2vec were derived and defined for molecular text data. These embeddings reduce the dimensionality of the data, transforming molecular text data to a vector space, whilst keeping relative similarity between observations in their transformed state. These techniques may be applied to a broad range of applications that require a vector representation for data and show comparable results for the use cases we evaluated.

1.2 Outline of dissertation

Chapter 2 is a literature review specifically for techniques applied to biological data and molecular text data. Chapter 3 provides information on the datasets used and a comprehensive background on the types of data used. Chapter 4 describes the embedding techniques. A description of each technique is provided with a theoretical and mathematical formalisation. The details of the experiments conducted for evaluating the use of the embeddings as ML feature vectors are provided in Chapter 5. A description of the data preprocessing, models used, the training and evaluation setup, the experiments run and all of the results are provided. Chapter 6 provides details on the use of the embeddings for representing a corpus of data in vector space (embedding similarity) and predicting protein-protein interaction. Chapter 7 is the conclusion.

Chapter 2

Literature review

We focus on techniques applied specifically to biological data and molecular text data, in particular, for the literature review. We provide separate literature reviews for all of the techniques and methods used in later chapters. We have also provided a separate literature review for the data in Chapter 3 .
Current approaches mainly use DL and NLP [3, 24, 13, 41, 50, 56]. Most of these approaches use either structural data or text-based molecular data. Structural approaches use structural information and calculated features on the components of the molecule's structure. Usually, geometric deep learning models are used on the structural data [50, 17]. Molecular text approaches use data such as the SMILES representation of a molecule or its sequence. Since this data is represented using text, it must first be converted to a numerical format for ML models. Usually, this requires the molecular text data to be embedded as a numerical vector. There are overviews of embedding techniques for semantic text similarity measurement for classical NLP tasks [45]. Since the molecular text data we use is composed of text, it is logical to propose NLP techniques to embed molecular data. Text embeddings have proven useful for both SMILES data [56, 28, 23] and protein sequence data [22, 59]. Often the embedding of the data can drastically increase or decrease the model performance for a specific task [45]. We propose using four methods for embedding the atom and residue text information:

1. Count vectorisation (CVec)
2. Term frequency-inverse document frequency (TF-IDF)
3. LDA
4. word2vec

Recently self-supervised learning is being used to learn a semantic embedding of molecular text data [28, 23]. Many of these self-supervised techniques are examples of word2vec [35], a NN model that computes a continuous vector representation of words. The continuous vector of each word is chosen such

that the cosine similarity between vectors indicates the actual semantic similarity between the words. This approach based on text data has already been applied to molecular text data with a model trained on SMILES data to learn embeddings of molecular substructures [23]. Bio-vectors (Biovec) [4] are word vectors for n-grams specifically trained on biological sequences.

It is essential to note that LDA was initially proposed for population genetics [38] before its use in NLP. LDA characterises each topic by a distribution over words, and documents are represented as random mixtures over these latent topics [8]. LDA has also been used on general genetic data and sequence data [58]. These use cases have been extended to protein prediction tasks such as protein function prediction and protein folding [47]. LDA has been used in direct protein function prediction, and protein categorisation [57, 44]. There is a stark similarity between genetic sequence data (genetic data) and amino acid sequence data (protein data). More traditional approaches use LDA on biomedical text data in the form of research papers for protein-protein interaction prediction [2]. These traditional approaches are still very limited in that they require a large corpus of text and the availability of this data for the respective individual protein. It should be noted that substructures of the molecule can be used or thought of as words [53, 27, 23]. Using substructures of the molecules as words LDA has been applied to PDB data for protein structure comparison [46]. LDA applied to PDB data tries to efficiently represent protein structures and compare these representations based on similarity. Our approach may be helpful for other techniques and benefit the SMILES dataset embeddings. It has been shown that support vector machines work well in combination with LDA [10].

CVec, a technique that transforms text data into numerical data will also be used to feature engineer an embedding. Text data can be vectorised by counting the occurrence of words in the document. For a molecule x the count vector can be expressed as $[x_1, \ldots, x_r, \ldots, x_R]$, where R is the total number of residues or text items in the molecule (vocabulary size) and x_r is the number of times molecular text item or residue r appears in molecule x. This basic embedding is a good baseline with which to compare. Finally, we make use of the TF-IDF. The TF-IDF is a statistic measuring the importance of a word to a document that is in a corpus [15]. It is one of the most widely used weighting schemes for text data. The TF-IDF is the combination of two statistics. Term frequency is how often a term (residue or molecular text item) appears in the document or molecule. Inverse document frequency is the amount of information the term provides. This information measurement can be achieved by inspecting how common or rare a term is in a document. The inverse document frequency can be obtained by logarithmically scaling the total number of documents divided by the number of documents that contain the term.

Many other techniques use embeddings for classification tasks such as property prediction [28, 22, 56]. Property prediction makes use of a molecules structure to predict chemical attributes of the molecule. ML models ranging from classical linear models to modern deep learning techniques have been applied

to embeddings [56]. We, therefore, compare the embedding techniques using a SVM, NBC and a NN classifier.

Chapter 3

Data

In this chapter a detailed description of the data is given. A summary of the data is also provided in Table 3.2. The data preprocessing is described in Chapter 5.

3.1 SMILES data

The simplified molecular-input line-entry system (SMILES) was created in the 1980s and has evolved over the years. An open standard has since been created. SMILES uses short ASCII strings to represent the structure of molecules. A specific instance of a structure should be called a SMILES string. An example of a SMILES string with its skeletal formula is provided in Figure 3.1 for histidine. The SMILES datasets are obtained from [56]. These datasets are built on public databases and were created as a benchmark for ML techniques. Both of the SMILES datasets are property prediction tasks:

1. Blood-brain barrier penetration (BBBP) prediction.
2. Prediction of binding results for a set of inhibitors of human β-secretase 1 (BACE).

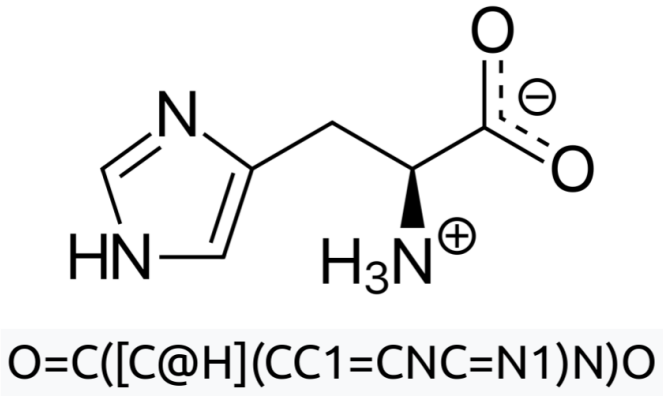

Figure 3.1: Skeletal formula and SMILES representation of the amino acid residue histidine [52]

We provide a short description on some of the symbols and their meaning.

1. Atoms- an atoms chemical element is abbreviated and placed in square brackets. Brackets are usually ommitted when they:
   (a) are part of the "organic subset" of B, Br, C, Cl, I, F, N, O, P or S.
   (b) do not have an assigned charge.
   (c) have the implicit hydrogen count.
   (d) are the normal isotopes.
   (e) are not chiral centers.
2. Bonds- ". - = # $: / \" are used to represent bonds.
   (a) Single bonds may be represented by "-", but usually the "-" is ommitted.
   (b) The characters: "=", "#", and "$" represent double, triple, and quadruple bonds.
   (c) A "non-bond" is indicated with a ".".
   (d) ":" indicates an aromatic or "one and a half" bond.
   (e) "/" or "\" indicate single bonds that are adjacent to double bonds.
3. Rings- acyclic structures are made by breaking rings up at certain points. Numerical ring closure labels are added to show connectivity between non-adjacent atoms.
4. Aromaticity- one of three forms can be used for aromatic rings:
   (a) Kekulé from
   (b) Using aromatic bond symbol
   (c) Representing the constituent B, C, N, O, P and S in lower-case b, c, n, o, p and s.
5. Branching- parenthesis are used to represent branches of the molecule.
6. Stereochemistry (the study of the structural arrangement of atoms in molecules)- SMILES does not require the specification of stereoisomers.
7. Isotopes- a number equal to the integer isotopic mass is placed before the atomic symbol to represent isomers.

3.2 FASTA data

The FASTA format is standard for protein sequence representation and is used almost universally [60]. The format is a representation of either amino acid (protein) sequences or nucleotide sequences in a text-based format. For protein FASTA files single letters represent each amino acid. The FASTA format also allows for comments and sequence names to precede the sequences themselves. FASTA is a simple and easily parsible format that most text processing tools and programming languages can manipulate. An example of an extracted protein sequence is provided for chains J and K of protein 5VTA in Figure 3.2. This protein sequence is similar to the FASTA format.

The FASTA dataset is used to make predictions on the classification of the protein and is retrieved from the Research Collaboratory for Structural Bioinformatics (RCSB) Protein Data Bank (PDB). The dataset is available on Kaggle.

```
QIVLSQSPAIKVTMTCRASSSVCNMHWYQQKPGSSPKPWLHGTSNLASGVPVRFSGSGSGTSFSLTIS
RVEATYFCQQWSNHPPTFGGGFQLQQSGPELVKPGASVKISCKASGYSFTNINWMKQSNGKSLEWIGV
VIPKYGTTNYNQKFQGKATLTVDQSSSTAYIQLNSLTSEDSAVYYCTRFRVFFDVWGTGTTVT
```

Figure 3.2: An example of the extracted sequence of protein 5VTA with chains J and K only

Each file consists of:

1. Description line - begins with a ">" or ";", has a unique identifier of the sequence. This identifier is sometimes accompanied by additional information.
2. Sequence- the actual sequence of the protein follows the description line.

The amino acid codes supported (22 amino acids and 3 special codes) and their respective amino acids are provided in Table 3.1.

Amino Acid 1-letter Code	Amino Acid
A	Alanine
B	Aspartic acid (D) or Asparagine (N)
C	Cysteine
D	Aspartic acid
E	Glutamic acid
F	Phenylalanine
G	Glycine
H	Histidine
I	Isoleucine
J	Leucine (L) or Isoleucine (I)
K	Lysine
L	Leucine
M	Methionine/Start codon
N	Asparagine
O	Pyrrolysine (rare)
P	Proline
Q	Glutamine
R	Arginine
S	Serine
T	Threonine
U	Selenocysteine (rare)
V	Valine
W	Tryptophan
Y	Tyrosine
Z	Glutamic acid (E) or Glutamine (Q)
X	any
*	translation stop
-	gap of indeterminate length

Table 3.1: Table of amino acids with 1-letter codes

3.3 PDB data

The PDB file format is a textual file format describing the three-dimensional structures of molecules held in the Protein Data Bank. The coordinates of each atom are determined by X-ray crystallography, NMR spectroscopy, and cryo-electron microscopy. Usually each PDB file will consist of some text that provides information on the molecule and structural information in the form of coordinates for each atom in the protein. This structural information is accompanied by information on the atom type, the residue the atom is part of, the chain the atom is a part of and the remoteness code of the atom (the remoteness code indicates which atoms bind or attach to each other). Within this PDB file one can break a protein down into its constituent components:

1. Chains- each protein will have one or more chains.

2. Residues- each chain will be made up of amino acid residues.

3. Atoms- each amino acid residue is made up of atoms.

An example of a PDB file for protein 5VTA is provided in Figure 3.4. A corresponding visualisation of the structure of 5VTA with the atoms of an asparagine residue highlighted in red are provided in Figure 3.3. The PDB dataset was obtained from Molecular Surface Interaction Fingerprints and is the same dataset used by two deep learning techniques [50, 17, 17]. This dataset will be used to "search" or find the binding partner of a given query protein.

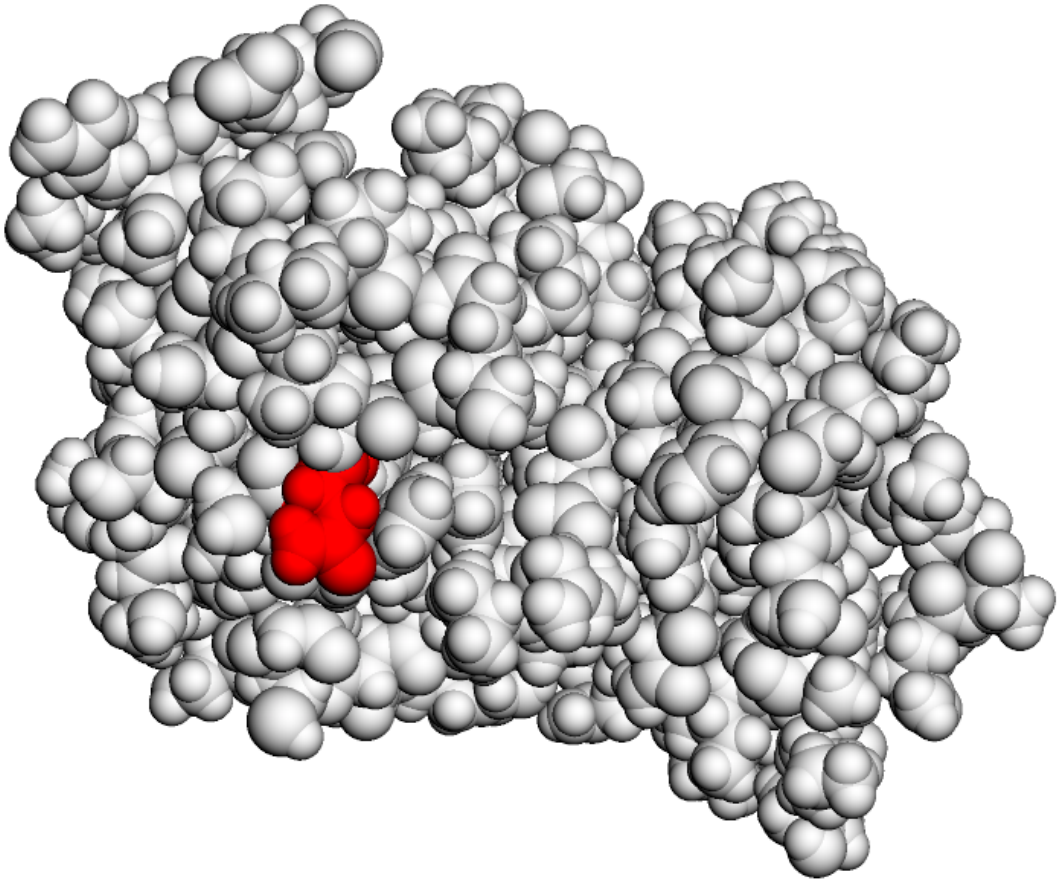

Figure 3.3: Visualisation of protein 5VTA with the atoms of an asparagine residue highlighted in red.

```
ATOM      1  N   ARG A  39     -83.823   2.256  42.737  1.00 46.52           N
ATOM      2  CA  ARG A  39     -82.406   1.761  42.842  1.00 46.80           C
ATOM      3  C   ARG A  39     -81.871   1.326  41.453  1.00 46.67           C
ATOM      4  O   ARG A  39     -82.299   0.289  40.907  1.00 49.46           O
ATOM      5  CB  ARG A  39     -82.327   0.599  43.852  1.00 47.21           C
ATOM      6  CG  ARG A  39     -81.134   0.632  44.799  1.00 46.64           C
ATOM      7  CD  ARG A  39     -79.809   0.674  44.068  1.00 47.31           C
ATOM      8  NE  ARG A  39     -78.686   0.287  44.924  1.00 49.23           N
ATOM      9  CZ  ARG A  39     -77.393   0.440  44.619  1.00 50.29           C
ATOM     10  NH1 ARG A  39     -77.014   0.999  43.462  1.00 49.55           N
```

Figure 3.4: The first 10 lines of the 5VTA proteins PDB file

In order to use this data we will filter out the structural information and only use the textual data on the residue type. The PDB data contains more information on the residue than the FASTA file format which also contains information on the amino acid sequence. It is important to note that after filtering, we will be left with the amino acid sequence. One could couple the sequence with the atom type, the residue the atom is part of and the remoteness code of the atom. We will leave this for future work.
Once the data has been filtered and cleaned we can transform the textual data to a vector representation. Embedding of molecular data is widely used for many applications such as the computation of similarity measures or for tasks such as sequence alignment [20]. In particular, there are several encoding methods for amino acid residues [30]. The best form of embedding will be investigated but some options can range from n-grams and count vector representation to Biovectors and protein-vectors (ProtVec) [4].

Summary of Datasets			
Data type	Dataset	Task	Number of proteins
SMILES [56]	BACE	Classification	1513
	BBBP	Classification	2039
FASTA [60]	Structural Protein Sequences (SPS)	Classification	16368
PDB [50, 17]	MASIF Dataset	Protein search	10962

Table 3.2: Table of summary details of the datasets

Chapter 4

Word embeddings

Molecules can have tremendous diversity which results in data with different dimensions. To address the variability in the lengths of the molecular text data, we create embedding vectors of equal dimension for every molecule. Word embeddings create a numerical vector of equal dimension that encodes the meaning of the data such that similar text data are closer to each other in the vector space. This means that every molecule will have a numerical vector representation of equal dimension with similar molecules having a distance between their vector representations that is indicative of their similarity. We make use of CVec, TF-IDF, LDA and word2vec to embed the molecular text data. Embedding techniques can be grouped into global and local embeddings which we will discuss next [32, 51, 26].

4.1 Global embeddings

Global embeddings featurise the molecular text data for the molecule as a whole [26]. Often these embeddings make use of counts or statistics to summarise the composition of a document or molecule at the document or molecule level.

4.1.1 Count vectorisation

CVec is a technique that transforms text data into numerical data [54, 61]. A method of doing this is by counting the occurrence of words in the document. For a molecule $\mathbf{x}$, the count vector can be expressed as $[x_1, \ldots, x_r, \ldots, x_R]$, where R is the total number of residues or text items in the molecule (vocabulary size) and x_r is the number of times molecular text item or residue r appears in molecule $\mathbf{x}$. An n-gram model can also be used for CVec [29]. An n-gram is a contiguous set of n words from a sample of text data. The use of n-grams on molecular text data may be very beneficial as it can introduce relations such as combining an atom with the amino acid residue it is a part of. Beyond this n-grams may also be

beneficial when applied to amino acid residues alone.

Another parameter that may be useful to set is stop words. Stop words are usually common words for the given language such as "a" and "the" in English. The specified stop words will be excluded and not counted as they may be irrelevant or provide no information gain. Stop words may be used to remove common elements that appear in all amino acid residues for the PDB dataset. An example of this is the central alpha carbon atom, amino group (NH2), carboxyl group (COOH) and hydrogen atom that are found in all amino acid residues [40]. It may be useful to specify the atoms making up these groups as stop words so that one can focus on the other atom or group of atoms bonded to the alpha carbon known as the variable group, side chain or R group. The variable group, side chain or R group refers to the part of the amino acid that varies between different amino acids [40]. The amino acid core structure is constant through all amino acid residues and remains the same. The R groups are different on all amino acids and this is where variation occurs between amino acid residues. It may then be beneficial to focus on the variation found in this variable region.

4.1.2 Term frequency-inverse document frequency

The TF-IDF is a statistic that measures the importance of a word or item to a document or file that is in a corpus [15, 43, 34]. It is one of the most widely used weighting schemes for text data. The TF-IDF is the combination of two statistics. These two statistics, term frequency and inverse document frequency, can be defined in various ways.

4.1.2.1 Term frequency

Term frequency $tf(term, molecule)$ is how many times a term (residue or molecular text item) appears in the document or molecule [34]. A formula for term frequency can be given by:

$$tf(term, molecule) = \frac{\text{Count of occurrences of } term \text{ in } molecule}{\text{Number of terms in the } molecule}.$$

A log normalised or logarithmically scaled version of the term frequency is given by:

$$logtf(term, molecule) = \frac{\log\left(1 + \text{Count of occurrences of } term \text{ in } molecule\right)}{\text{Number of terms in the } molecule}.$$

4.1.2.2 Inverse document frequency

Inverse document frequency is the amount of information the term provides [42]. This measurement of information can be achieved by inspecting how common or rare a term is in a document. The inverse

document frequency can be obtained by logarithmically scaling the total number of documents divided by the number of documents that contain the term.

$$idf(term, M) \quad = \quad log\left(\frac{|M|}{molecule \in M : term \in molecule}\right),$$

where:

- M is the corpus of molecules and
- $|M|$ is the total number of molecules in the corpus

4.1.2.3 Term frequency-inverse document frequency

Using the definitions and equations for term frequency and inverse document frequency one can obtain the formula of term frequency-inverse document frequency [15, 43, 34]:

$$TF\text{-}IDF(term, molecule, M) \quad = \quad tf(term, molecule) \cdot idf(term, M). \tag{4.1}$$

TF-IDF will filter out common terms since a lower document frequency of the term in the whole corpus and a higher term frequency in the document will result in a higher TF-IDF value or weight. The TF-IDF value can be used as a weighting factor that measures the importance of a word to a document or file. For molecular text data one can again use atom or residue text information as the words or terms of the document. Each molecule's file data will be a document and the full dataset of molecules files' will be the corpus.

4.1.2.4 Deriving term frequency-inverse document frequency as a Fisher kernel

It can be shown that TF-IDF can be related to the mathematical form of the Fisher kernel based on the Dirichlet compound multinomial (DCM) distribution [15]. To show this relationship we first need to understand the DCM distribution with regards to a "bag of words" representation of the proteins.
For each molecule $\mathbf{x}$, a vector containing the number of occurrences of each residue or molecular text item in the molecule is made $[x_1, \ldots, x_r, \ldots, x_R]$, where R is the total number of residues or text items in the molecule (vocabulary size) and x_r is the number of times molecular text item or residue r appears in molecule $\mathbf{x}$. The DCM distribution is given by:

$$p(\mathbf{x}) = \frac{n!}{\prod_{r=1}^{R} x_r!} \frac{\Gamma(s)}{\Gamma(s+n)} \prod_{r=1}^{R} \frac{\Gamma(x_r + \alpha_r)}{\Gamma(\alpha_r)}, \tag{4.2}$$

where the length of the molecules sequence or the length of the molecular text data is $n = \sum_r x_r$ and $s = \sum_r \alpha_r$ is the sum of the DCM parameters. Since a corpus will contain molecules of different lengths

and the DCM is a distribution over count vectors of the same length n; we need a family of DCMs with the same parameter values α_r to model a corpus. For a set of molecules M we have to consider the number of times a residue r appears for each molecule m. We add an m subscript to account for this so that $\mathbf{x_m}$ is a vector containing the number of occurrences of each residue or molecular text item for the molecule m. Thus, $\mathbf{x_m} = [x_{m1}, \ldots, x_{mr}, \ldots, x_{mR}]$, where R is the total number of residues or text items in the molecule m (vocabulary size) and x_{mr} is the number of times molecular text item or residue r appears in molecule $\mathbf{x_m}$, for every molecule m in set M. Thus, the maximum likelihood parameter α_r can be approximated by:

$$\alpha_r = \frac{\sum_{m \in M} I\left(x_{mr} \geq 1\right)}{\sum_{m \in M} \Psi\left(s + n_m\right) - \Psi\left(s\right)},$$

where, x_{mr} is the count of residue or molecular text item r in molecule m [15]. n_m is the length of molecule m and $\Psi\left(\right)$ is the digamma function. For a typical molecule $\Psi\left(s + n_m\right) - \Psi\left(s\right) \approx 1$ since s and n_m will be in the hundreds or low thousands. Thus,

$$\alpha_r \approx \frac{1}{|M|} \sum_{m \in M} I\left(x_{mr} \geq 1\right), \tag{4.3}$$

where, $|M|$ is the number of molecules in the collection of molecules M.
Thus, using equation (4.1) we can obtain a specific version of TF-IDF given by:

$$TF - IDF\left(x_r\right) = \log\left(x_r + 1\right) \cdot \log\left(\frac{|M|}{\sum_{m \in M} I\left(x_{mr} > 0\right)}\right).$$

Given the probability distribution in equation (4.2) the Fisher kernel measures the similarity of $\mathbf{x}$ and $\mathbf{y}$ with respect to this distribution.

$$k\left(\mathbf{x}, \mathbf{y}\right) = s\left(\mathbf{x}\right) \mathbf{H} s\left(\mathbf{y}\right),$$

where, $s\left(\mathbf{x}\right) = \frac{d}{d\alpha_r} \log\left(p\left(\mathbf{x}\right)\right)$ and $\mathbf{H}$ is the Hessian matrix of the second partial derivatives of $l\left(\mathbf{x}\right) = \log\left(p\left(\mathbf{x}\right)\right)$ with respect to the parameters α_r. The kernel function $k\left(\mathbf{x}, \mathbf{y}\right)$ is invariant to changes in the parameterisation of $p\left(\mathbf{x}\right)$. Since $\mathbf{H}$ is usually approximated by the identity matrix, the Fisher kernel will

differ for different parameterisations in this case. The log-likelihood of the DCM distribution is:

$$\begin{aligned} l(\mathbf{x}) &= \log(p(\mathbf{x})), \\ \log(p(\mathbf{x})) &= \log\left(\frac{n!}{\prod_{r=1}^{R} x_r!}\frac{\Gamma(s)}{\Gamma(s+n)}\prod_{r=1}^{R}\frac{\Gamma(x_r+\alpha_r)}{\Gamma(\alpha_r)}\right), \\ l(\mathbf{x}) &= \log(n!) - \log\left(\prod_{r=1}^{R} x_r!\right) + \log(\Gamma(s)) - \log(\Gamma(s+n)) + \log(\Gamma(x_r+\alpha_r)) - \log(\Gamma(\alpha_r)). \end{aligned}$$

The partial derivative of the log-likelihood function with respect to α_r is:

$$\begin{aligned} \frac{dl(\mathbf{x})}{d\alpha_r} &= \frac{d}{d\alpha_r}(n!) - \frac{d}{d\alpha_r}\left(\prod_{r=1}^{R} x_r!\right) + \frac{d}{d\alpha_r}(\Gamma(s)) - \frac{d}{d\alpha_r}(\Gamma(s+n)) \\ &\quad + \frac{d}{d\alpha_r}(\Gamma(x_r+\alpha_r)) - \frac{d}{d\alpha_r}(\Gamma(\alpha_r)), \qquad (4.4) \\ \frac{dl(x)}{d\alpha_r} &= \frac{d}{d\alpha_r}(\Gamma(s)) - \frac{d}{d\alpha_r}(\Gamma(s+n)) + \frac{d}{d\alpha_r}(\Gamma(x_r+\alpha_r)) - \frac{d}{d\alpha_r}(\Gamma(\alpha_r)), \\ \frac{dl(x)}{d\alpha_r} &= \Psi(s) - \Psi(s+n) + \Psi(x_r+\alpha_r) - \Psi(\alpha_r). \qquad (4.5) \end{aligned}$$

since, $\Psi(x) = \frac{d}{dx}(\Gamma(x))$ [15].

- For $\mathbf{x} \geq 1$: $\Psi(\mathbf{x})$ can be approximated as $(\mathbf{x} - 0.5)$, with the error tending to 0 as $\mathbf{x}$ tends to infinity.
- For $\mathbf{x} \ll 1$: $\Psi(\mathbf{x})$ can be approximated to $\frac{-1}{\mathbf{x}} + \Psi(1)$, with $\Psi(1) \approx -0.577$, with the error tending to 0 as x tends to 0 from above.

Equation (4.4) can be approximated as:

$$\frac{dl(\mathbf{x})}{d\alpha_r} = \Psi(s) - \Psi(s+n) + I(x_r \geq 1)[(x_r - 0.5) + \frac{1}{\alpha_r} - \Psi(1)].$$

Thus, we have derived TF-IDF as a Fisher kernel:

- The term $\Psi(s) - \Psi(s+n)$ is the same for all residues and molecular text data and has little influence on the score.
- The term $(x_r - 0.5)$ is the log normalised or logarithmically scaled term frequency.
- From equation (4.3) the term $\alpha_r \approx \frac{1}{|M|}\sum_{m \in M} I(x_{mr} \geq 1)$ which is the inverse document frequency.

4.1.3 Latent Dirichlet Allocation

We explain LDA for molecular text data. LDA is usually explained with NLP terminology, for example, a dataset is usually called a corpus [8]. Throughout this section, a molecule is equivalent to a document

and a residue or molecular text item is equivalent to a word. LDA for a corpus of molecules M can be described as a generative statistical topic model where;

- each molecule is a mixture of corpus-wide topics,
- each topic is a distribution over residues, and
- each residue is drawn from one of the topics

In order to discover the topics in a corpus it is easiest to reverse engineer the problem. First we need to make the molecules in the corpus. To do this, we can use a generative process for each molecule $\mathbf{x}$ in the corpus M with $|M|$ the number of molecules in the corpus.

- Let $\{1 \dots P\}$ be the vocabulary of molecular text items. A molecular text item is an item from the vocabulary of size P defined as the basic unit that makes up the data.
- A molecular text item will be a P-vectored x such that x^p has value 1 and x^u is 0 for all $p \neq u$.
- A molecule $\mathbf{x}$ is a combination of N molecular text items denoted by $\mathbf{x} = [x_1, \dots, x_N]$, where x_n is the nth molecular text item in the molecule.
- A corpus M is a collection of $|M|$ molecules denoted as $M = \{\mathbf{x_1}, \dots, \mathbf{x_{|M|}}\}$.

For each molecule x in a corpus M a generative process can be assumed [8, 38] to infer the latent variables. This process generates the latent variables from probability distributions with parameters equal to the known variables:

1. Draw a topic-molecule distribution θ from a Dirchlet distribution. $\theta \backsim Dir(\alpha)$ with α a vector of dimension equal to the number of topics K.
2. For each of the N molecular text items x_n:
   (a) Generate a topic $z_n \backsim multinomial(\theta)$.
   (b) Generate a molecular text item $x_n \backsim p(x_n|z_n, \beta)$ a multinomial probability conditioned on the topic z_n.

The probability density function for θ is given by:

$$p(\theta|\alpha) = \frac{\Gamma\left(\sum_{i=1}^{k} \alpha_i\right)}{\prod_{i=1}^{k} \Gamma(\alpha_i)} \theta_1^{\alpha_1 - 1} \dots \theta_k^{\alpha_k - 1},$$

where the parameter α is a K-vector with components $\alpha_i > 0$, and where $\Gamma(x)$ is the Gamma function. Thus the joint distribution of a topic mixture θ with a set of N topics $\mathbf{z} = [z_1, \dots, z_N]$ and a set of N

molecular text items is given by:

$$p(\theta, \mathbf{z}, \mathbf{x}|\alpha, \beta) = p(\theta|\alpha) \prod_{n=1}^{N} p(z_n|\theta)\, p(x_n|z_n, \beta)\,.$$

Integrating over θ:

$$p(\mathbf{z}, \mathbf{x}|\alpha, \beta) = \int p(\theta|\alpha) \prod_{n=1}^{N} p(z_n|\theta)\, p(x_n|z_n, \beta)\, d\theta.$$

Then summing over $\mathbf{z}$:

$$p(\mathbf{x}|\alpha, \beta) = \int p(\theta|\alpha) \sum_{\mathbf{z}} \prod_{n=1}^{N} p(z_n|\theta)\, p(x_n|z_n, \beta)\, d\theta. \tag{4.6}$$

Now applying the distributive law in reverse a sum of products in (4.5) can be changed to a product of sums:

$$\begin{aligned}
& \sum_{\mathbf{z}} \prod_{n=1}^{N} p(z_n|\theta)\, p(x_n|z_n, \beta)\,, \\
= \; & \sum_{z_1} \cdots \sum_{z_N} \left(p(z_n|\theta)\, p(x_n|z_n, \beta) \ldots p(z_n|\theta)\, p(x_n|z_n, \beta) \right), \\
= \; & \left(\sum_{z_1} p(z_n|\theta)\, p(x_n|z_n, \beta) \right) \ldots \left(\sum_{z_N} p(z_n|\theta)\, p(x_n|z_n, \beta) \right), \\
= \; & \prod_{n=1}^{N} \sum_{z_n} p(z_n|\theta)\, p(x_n|z_n, \beta)\,.
\end{aligned} \tag{4.7}$$

Substituting (4.6) into (4.5) gives the marginal distribution of a document:

$$p(\mathbf{x}|\alpha, \beta) = \int p(\theta|\alpha) \left(\prod_{n=1}^{N} \sum_{z_n} p(z_n|\theta)\, p(x_n|z_n, \beta) \right) d\theta.$$

Taking the product of the marginal probabilities for each molecule one obtains the probability of a corpus:

$$p(\mathbf{M}|\alpha, \beta) = \int p(\theta|\alpha) \left(\prod_{n=1}^{N} \sum_{z_n} p(z_n|\theta)\, p(x_n|z_n, \beta) \right) d\theta.$$

Using plate notation we provide an intuitive explanation of LDA [1]. Here the boxes are referred to as "plates". The outer plate represents molecules, while the inner plate represents the repeated positions of the molecular text items which are associated with a choice of topic and molecular text item.

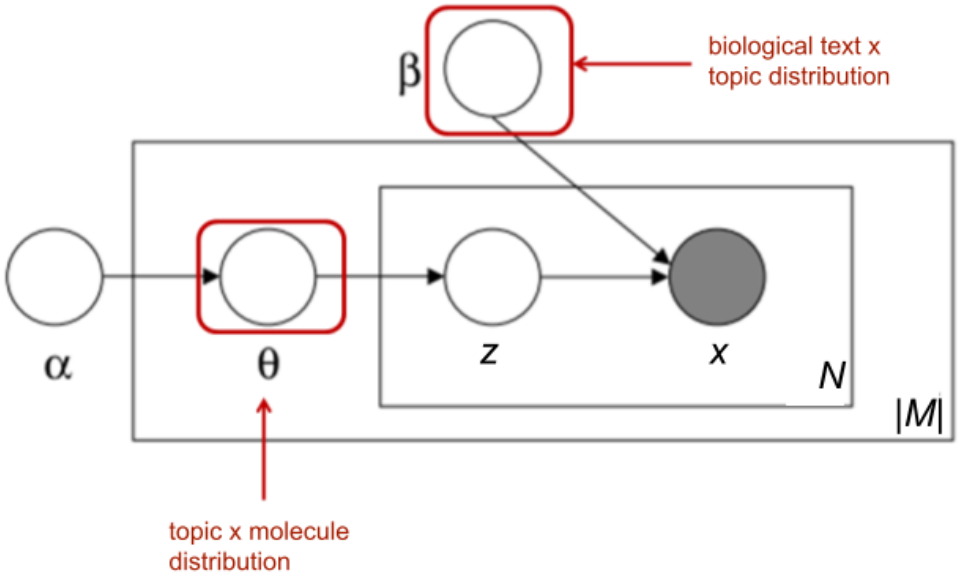

Figure 4.1: Plate notation for the LDA model

In the plate notation provided in Figure 4.1:

- $|M|$ is the number of molecules in the corpus M
- N is the number of molecular text items in a given molecule
- α is a corpus-level parameter of the Dirichlet prior on the per-molecule topic distributions
- β is a corpus-level parameter of the Dirichlet prior on the per-topic molecular text item distributions
- θ is the topic distribution for a given molecule
- z_{mn} is the topic for the n-th molecular text item in molecule m
- x_{mn} is the specific molecular text item

The only observed variable is x- hence it is coloured grey in the plate notation in Figure 4.1, Figure 4.2 and Figure 4.3. All of the other variables are latent variables.

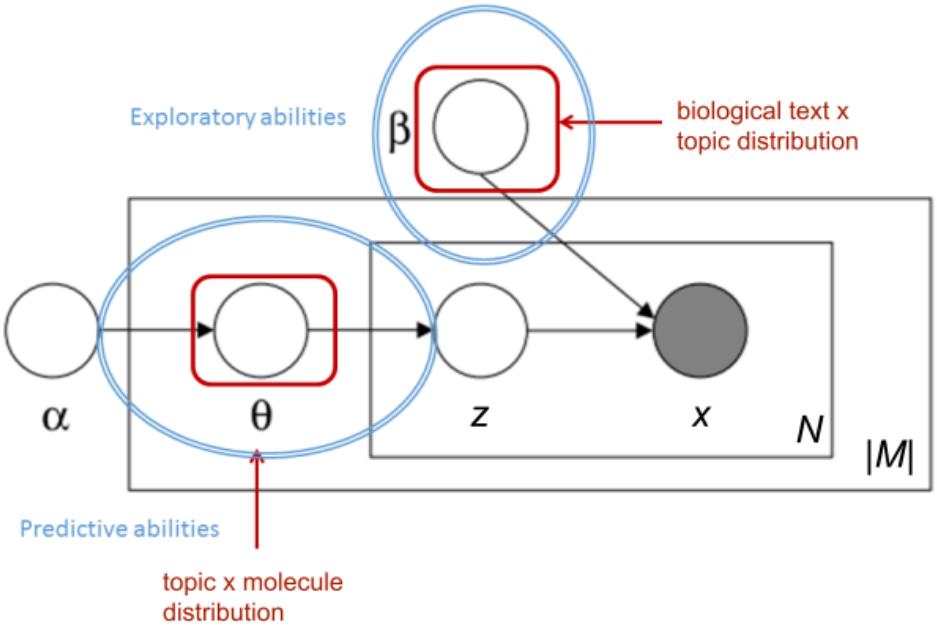

Figure 4.2: Plate notation for the LDA model illustrating the exploratory and predictive abilities of LDA

Using the view provided in Figure 4.2- the molecular text-topic distributions are a matrix with topics as rows and columns defined by molecular text items. The variable θ can be viewed as a matrix of topic-molecule distributions, where the rows would be individual molecules and the columns would be topics. Thus, each row of θ is a distribution over topics and each row of the molecular text-topic distributions is a distribution over molecular topics [8, 12] as can be seen in Figure 4.2. In this way the molecular text-topic distribution matrix and topic-molecule distribution matrix can be viewed as the decomposition of the original molecule-molecular text item matrix that represents the corpus of documents being modeled. Thus, LDA can also be thought of as a dimensionality reduction technique [8, 11] where the corpus is represented as an embedding in a lower dimensional form using the topic-molecule distribution matrix. The advantage of this approach is that the interpretable topics should create a semantic embedding of the molecules. Using the manifold hypothesis which states that: "High dimensional spaces tend to lie in the vicinity of underlying lower dimensional spaces (manifolds)" [16] it is clear that LDA can provide us with a manifold representation of the corpus. An manifold is a topological space

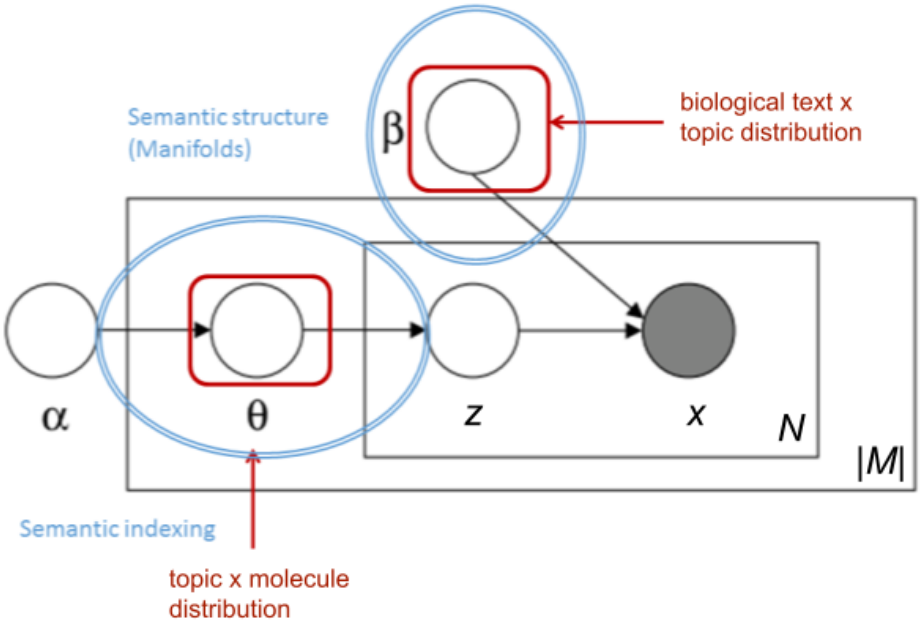

Figure 4.3: Plate notation for the LDA model illustrating the manifold hypothesis applied to LDA

Thus, one can perform a similarity measure on the semantic structure (molecular text-topic distribution matrix or manifolds) and index the query document using semantic indexing on the topic-molecule distribution matrix (Figure 4.3). Under the LDA model a molecule can be associated with more than one topic [8]. Thus, the topic probability vector for a molecule will be of dimension K where K is the number of topics. For a molecule x the topic probability vector can be represented as:

$$\theta_{\mathbf{x}} \quad = \quad [p\left(t_1\right), \ldots, p\left(t_K\right)],$$

where $p\left(t_k\right)$ is the probability of the molecule being associated with topic k. The LDA embedding will reduce the dimension of the molecular text items in a molecule to a vector with dimension equal to the number of topics chosen. This molecule-topic matrix will be a semantic embedding for the corpus of molecules.

Further information on the variables:

- α is a corpus-level parameter sampled once in the generation of the corpus
- β is a corpus-level parameter sampled once in the generation of the corpus

- θ_m is a molecule-level variable that is sampled once per molecule
- z_{mn} is a molecular text item-level variable and is sampled once for each molecular text item in each molecule
- x_{mn} is a molecular text item-level variable and is sampled once for each molecular text item in each molecule

Initial results saw a decrease in performance for more than 100 topics. For this reason we cap the number of topics to 100. We use 10, 20, 50 and 100 topics for our LDA embeddings.

4.2 Local embeddings

Local embeddings take the local features of a molecular text item into account [26]. Often local embeddings will be created with the use of a window that summarises the local neighborhood of a molecular text item. In this way a local descriptor is created at the molecular text item or word level instead of a global embedding that is often at a document or molecule level.

4.2.1 Word2vec

Word2vec uses a group of related NN models to compute a continuous vector representation of words [35]. The continuous vector of each word is chosen such that the cosine similarity between vectors is an indicator of the true semantic similarity between the words. In this way a corpus of documents can be converted into a vector space with each word in the corpus having a unique embedding vector in this feature space. The NN models are shallow networks with only two layers. Word2vec can either make use of continuous bag-of-words (CBOW) or continuous skip-gram model architectures [35]. The CBOW model takes into account the local window of words for the current word to create its embedding. In this way word2vec takes local features into account to create a local feature embedding vector for a word instead of a global one. The skip-gram model does the opposite of the CBOW model. It takes the current word and uses it to predict the local window of words. The word2vec model can be applied and trained on a corpus of molecules that are comprised of molecular text items.

It is often helpful to subsample words that have a frequency above a certain threshold. Subsampling frequent words such as "a" and "the" allows the model to focus on words that occur less frequently and have more information. The dimension of the embedding vector is directly related to the quality of the embedding. Thus it is beneficial to increase the dimension of the embedding vector until the performance metrics stop improving or the cost of increasing the dimension is too high [35]. Lastly, the size of the context window can also be adjusted to improve the quality of the embeddings [35]. The context window is the number of words around the target word that are used by the model to make predictions for

the target words representation [35]. Initial results saw a decrease in performance for embeddings with dimension less than 100 or greater than 300. Other word2vec models also use embeddings of similar dimension [23]. For these reason we use word2vec embeddings of size 100, 200 and 300 for the FASTA datasets. For the SMILES datasets we use the pre-trained Mol2vec model [23].

4.3 Probabilistic distributional semantic methods for protein-protein interaction

Once a vector embedding is learnt for a corpus of proteins we can perform semantic calculations on the vector in Euclidean space. We can find the semantic similarity of two protein embedding vectors with the use of information retrieval (IR). IR can be defined as the task of finding a protein from a corpus that best matches a query [19]. It is clear that the process of IR is easy to apply to the domain of protein-protein interaction prediction. For protein-protein interaction the IR task can be defined as finding a protein from a corpus that is the best binding partner for the query protein. To do this we need to make use of a similarity measure in the form of the Bhattacharyya distance for the embedding vectors.

4.3.0.1 Bhattacharyya distance

The Bhattacharyya distance is a measure of the similarity between pairs of features in the vector space model [7]. It makes use of the Bhattacharyya coefficient which can be used to quantify the amount of overlap between two proabability distributions or two normalised vectors. We use the Bhattacharyya distance between two embedding vectors $\mathbf{u}$ and $\mathbf{v}$ of dimension N can be defined as:

$$bd\left(u,v\right) \quad = \quad -\ln\left(bc\left(u,v\right)\right).$$

where, $bc\left(u,v\right)$ is the Bhattacharyya coefficient which is defined as:

$$bc\left(u,v\right) \quad = \quad \sum_{i=1}^{N}\sqrt{u_i v_i}.$$

$$\begin{aligned} 0 \quad &\leq bc \leq \quad 1, \text{ since } u_i \leq 1 \text{ and } v_i \leq 1 \; \forall i \in 1,\ldots,N, \\ \therefore 0 \quad &\leq bd \leq \quad \infty. \end{aligned}$$

The variables u_i and v_i are less than or equal to 1 since they are components of normalised vectors u and v. These vectors must be normalised since the input vectors for the Bhattacharyya coefficient need

to represent probabilities. It should be noted that the Bhattacharyya coefficient is not a metric since it does not satisfy the triangle inequality [7].

Since a molecule can be expressed as text it makes sense to apply embedding techniques to the data. This allows us to compare molecules of all shapes and sizes by using an embedding vector of equal dimension. These embedding techniques capture the components of the molecule such that similar molecules will have vector representations that are closer to one another in the vector space. This means that machine learning models will be able to use these embedding vectors as features. If similar molecules have similar embedding vectors machine learning models will be able to classify a given property of these molecules using the embedding vector.

Chapter 5

Experiments

We run 11 experiments to compare the combination of embeddings (as features) with NN, naïve Bayes classifier (NBC) and support vector machine classifier (SVM) models. First, we determine the optimal feature dimensions, for the LDA embeddings, for each model individually in experiments 1-3. In experiments 4-6 we find the optimal dimension for the word2vec embeddings, for each model individually on the FASTA sequences. We then find the best embedding technique for each dataset for the SVM in experiment 7. We find the best embedding technique for each dataset for the NBC in experiment 8. We do the same for each dataset for the NN in experiment 9. We compare all of the models in experiment 10 by using the best embedding for each dataset regarding each model to compare the models. Lastly, we compare our approach to the Moleculenet [56] benchmark in experiment 11 using a "scaffold" split[1]. Unless stated otherwise, we use 80% of the data to train the models and 20% to test them. We set the random state to 0 for the training and test splits for all of the experiments. Due to the large combination of models and datasets we did not perform hyper-parameter tuning. We tried to use the simplest architecture with default parameters for each model. LDA embeddings of dimension 10, 20, 50 and 100 are considered. Embedding dimensions of 100, 200 and 300 are considered for word2vec embeddings. Higher dimensions are not considered due to initial experiments showing a a decrease in performance for larger dimensions and compute limitations for both LDA and word2vec embeddings. Future work should be conducted on hyper-parameter tuning and larger embedding dimensions. All experiments and coding has been done in Python (Python 3.9). The experiments were each run 50 times on a Lenovo machine using Ubuntu 20.04 with an Intel i7-8565U processor, 2GB NVIDIA GeForce MX230 and 12 GB of RAM. All of the code is available *here*[2]. In this chapter we provide:

1. An overview of the data preprocessing

[1] A scaffold split attempts to separate molecules according to their structure such that structurally different molecules will be in different subsets [6].

[2] `https://github.com/Claudmj/Feature-engineered-embeddings-for-machine-learning-on-molecular-data`

2. An overview of SVMs

3. An overview of NBCs

4. The NN architecture

5. Tables summarising each experiment with its respective results

5.1 Data preprocessing

We provide an overview of the data preprocessing for each data type. A full description of all of the data is given in Chapter 3. For every dataset we first:

- Remove any duplicates
- Remove null values

5.1.1 SMILES data

For the SMILES datasets we make use of the *rdkit*[3][31] and *mol2vec*[4][23] Python packages. Using these two packages we can divide each molecule into substructures using a specified fixed radius. Each substructure is usually the group of atoms, within the specified radius, closest to a given heavy atom. Each substructure is then encoded using a Morgan fingerprint [14, 18]. A Morgan fingerprint is a bit vector representation of a molecule [14, 18]. The presence or absence of a substructure is represented by each bit. In this way the substructures will be words represented by their unique Morgan fingerprint that compose a molecular sentence. Morgan fingerprints can be used to measure the chemical similarity between molecules[18] and have properties that benefit our approach. Each molecular sentence will be a molecule and these sentences can now be fed into the embedding techniques to get a vector representation of the molecule.

5.1.2 FASTA (SPS) data

The SPS dataset is dataset of extracted FASTA sequences. For this dataset we first join the labels to the protein sequences using the raw datasets obtained from Kaggle[5]. Once this is completed we filter the molecules for proteins only. We then filter the resulting set for sequences that are classified as one of the top two categories with the most samples. We do this to produce a binary classification problem in order to simplify our model architectures. We insert a space between every amino acid residue (letter) in the sequence and remove the letter "X" since it can be any amino acid residue.

[3] `https://www.rdkit.org/`
[4] `https://github.com/samoturk/mol2vec`
[5] `https://www.kaggle.com/datasets/shahir/protein-data-set?resource=download&select=pdb_data_seq.csv`

5.1.3 PDB data

For the PDB dataset we filter out only the text data. The amino acid residues are taken out of each PDB file and saved in a text file.

5.2 Support vector machine

Support vector machines (SVM) are classification techniques which [9] map training data points such that the distance between classes is maximised. The SVM classifier then finds a hyperplane that distinctly separates the training data points for optimal classification. New data points are mapped into this space for prediction. The new data points get the predicted class of the side onto which they are mapped. SVMs can handle non-linear classification with the use of the kernel trick. They are memory efficient, versatile and effective in high-dimensional spaces. We make use of the *scikit-learn*[6][37] implementation of an SVM classifier with a radial basis function. For all other parameters the default *scikit-learn* parameters were used.

5.3 Naïve Bayes classifier

An NBC is a family of probabilistic classifiers based on Bayes' theorem. They are examples of a simple Bayesian network. All features or embeddings are assumed to be independent and are thought to contribute equally to the outcome. We use the *scikit-learn MultinomialNB*[7][37] classifier for the CVec embeddings and the *GaussianNB* for the other embeddings. We use a multinomial NBC for the CVec embeddings since these are the frequency with which terms occur. The other embeddings are continuous; thus, a Gaussian NBC is used with the assumption that the embeddings of each class are normally distributed.

5.4 Neural network architecture

We train a simple NN classifier for each embedding type on each dataset using the PyTorch Python library [36]. If the embedding has dimension N then the classifier has an input layer of dimension N, a hidden layer of dimension $N/2$ and an output layer with the final predictions. An illustration of the architecture is provided in Figure 5.1. We use batch normalisation and rectified linear unit (ReLU) activation between the input layer and hidden layer. We use sigmoid activation for the predictions to ensure the outputs are between 0 and 1 for the binary classification tasks. Each model is trained for 50 epochs with a batch size of 200. The epoch with the smallest loss is used for the evaluation metrics.

[6] `https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html`
[7] `https://scikit-learn.org/stable/modules/naive_bayes.html`

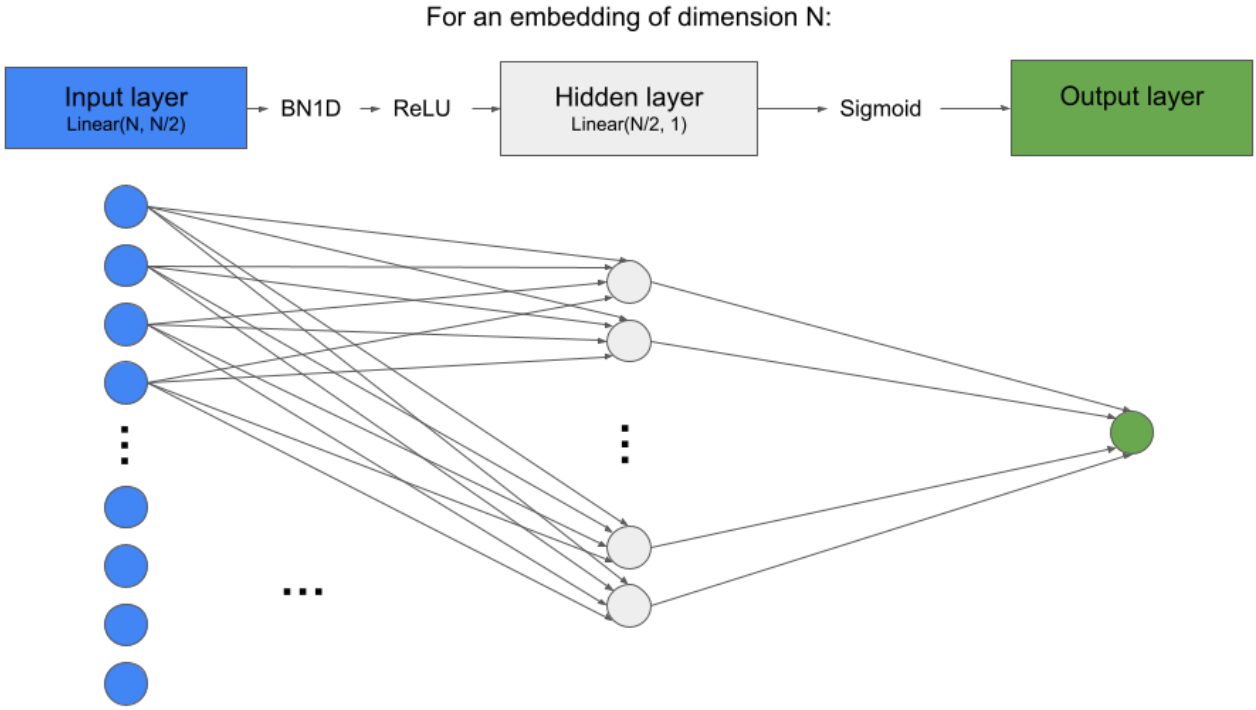

Figure 5.1: NN classifier architecture

In the NN architecture provided in Figure 5.1:

- *BN1D* represents a batch normalisation layer.
- *ReLU* shows that rectified linear unit activation is applied.
- *Sigmoid* shows that the element wise sigmoid function is applied.
- N represents the dimension or size of the input. This will usually be the dimension of the embedding.

5.5 Experiment results

Each experiment was run 50 times. The mean and standard deviation of the accuracy, area under the receiver operating characteristic curve (AUC) and F1 score are calculated for each experiment across the 50 runs.

5.5.1 Experiment 1: Optimal LDA embedding dimension for SVM

We need to find the optimal feature dimension for the LDA embeddings for the SVM. We compare feature dimensions of 10, 20, 50 and 100 for the LDA embeddings and take the dimension that provides the best result in terms of accuracy, area under the receiver operating characteristic curve (AUC) and F1 score.

Table 5.1: Experiment 1: Optimal LDA embedding dimension for SVM

Data type	Dataset	n	Accuracy ± (SD)%	AUC ± (SD)%	F1 ± (SD)%
Smiles	BACE	10	63.99 ± 3.16	63.20 ± 3.34	54.68 ± 8.09
		20	66.33 ± 3.31	65.80 ± 3.33	60.47 ± 5.50
		50	69.83 ± 3.18	69.42 ± 3.20	65.64 ± 4.40
		100	**72.13 ± 3.09**	**71.82 ± 3.19**	**68.94 ± 4.52**
	BBBP	10	80.95 ± 1.65	64.59 ± 4.33	88.48 ± 0.98
		20	82.59 ± 1.70	67.95 ± 3.71	89.39 ± 1.06
		50	84.24 ± 1.41	70.87 ± 2.49	90.36 ± 0.88
		100	**85.60 ± 1.32**	**73.35 ± 2.35**	**91.16 ± 0.81**
FASTA	SPS	10	69.44 ± 0.70	69.39 ± 0.78	66.01 ± 1.08
		20	71.74 ± 0.71	71.63 ± 0.74	68.32 ± 0.90
		50	**72.86 ± 0.55**	**72.59 ± 0.57**	**69.14 ± 0.69**
		100	72.53 ± 0.62	72.19 ± 0.63	68.60 ± 0.75

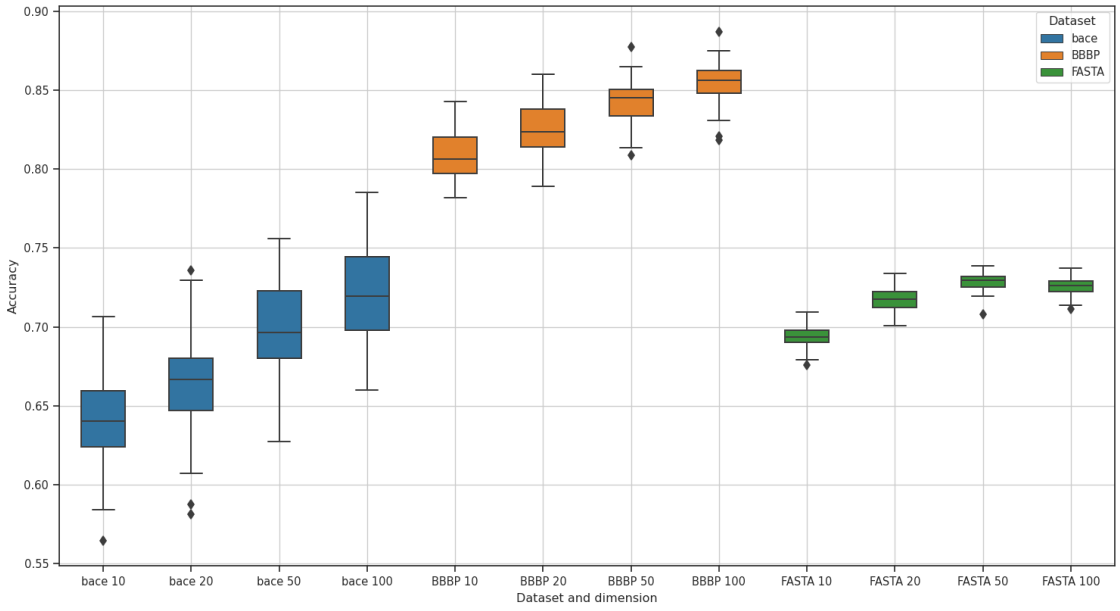

Figure 5.2: Experiment 1: Boxplots comparing the accuracy of different LDA embedding dimensions on the different datasets for the SVM

It is clear from Table 5.1 and Figure 5.2 that the LDA embeddings of lower dimension lose too much information across all of the datasets for the SVM. For the SVM model:

- The best LDA embedding dimension for the BACE dataset was 100 with an average accuracy of 72.13%, an average AUC of 71.82% and an average F1 score of 68.94%.
- The best LDA embedding dimension for the BBBP dataset was 100 with an average accuracy of 85.60%, an average AUC of 73.35% and an average F1 score of 91.16%.

- The best LDA embedding dimension for the SPS dataset was 50 with an average accuracy of 72.86%, an average AUC of 72.59% and an average F1 score of 69.14%.

5.5.2 Experiment 2: Optimal LDA embedding dimension for NBC

We need to find the optimal feature dimension for the LDA embeddings for the NBC. We compare feature dimensions of 10, 20, 50 and 100 for the LDA embeddings and take the dimension that provides the best result in terms of accuracy, AUC and F1 score.

Table 5.2: Experiment 2: Optimal LDA embedding dimension for NBC

Data type	Dataset	n	Accuracy ± (SD)%	AUC ± (SD)%	F1 ± (SD)%
Smiles	BACE	10	56.14 ± 5.28	56.37 ± 4.78	51.32 ± 20.40
		20	56.21 ± 4.79	57.24 ± 4.17	59.74 ± 14.27
		50	59.08 ± 4.44	60.28 ± 4.19	64.31 ± 12.61
		100	**59.85 ± 3.25**	**61.35 ± 3.13**	**68.07 ± 5.65**
	BBBP	**10**	**55.88 ± 19.69**	**62.99 ± 7.12**	**57.94 ± 27.87**
		20	47.78 ± 14.96	62.44 ± 5.70	48.28 ± 20.56
		50	40.56 ± 5.77	60.61 ± 3.53	38.78 ± 9.26
		100	44.76 ± 5.09	62.52 ± 3.38	46.09 ± 7.52
FASTA	SPS	**10**	**64.09 ± 1.05**	66.09 ± 1.07	65.81 ± 1.35
		20	63.88 ± 1.18	**66.33 ± 0.98**	66.64 ± 1.06
		50	62.17 ± 1.69	65.43 ± 1.29	**66.90 ± 0.72**
		100	63.34 ± 1.33	65.94 ± 1.03	66.51 ± 0.81

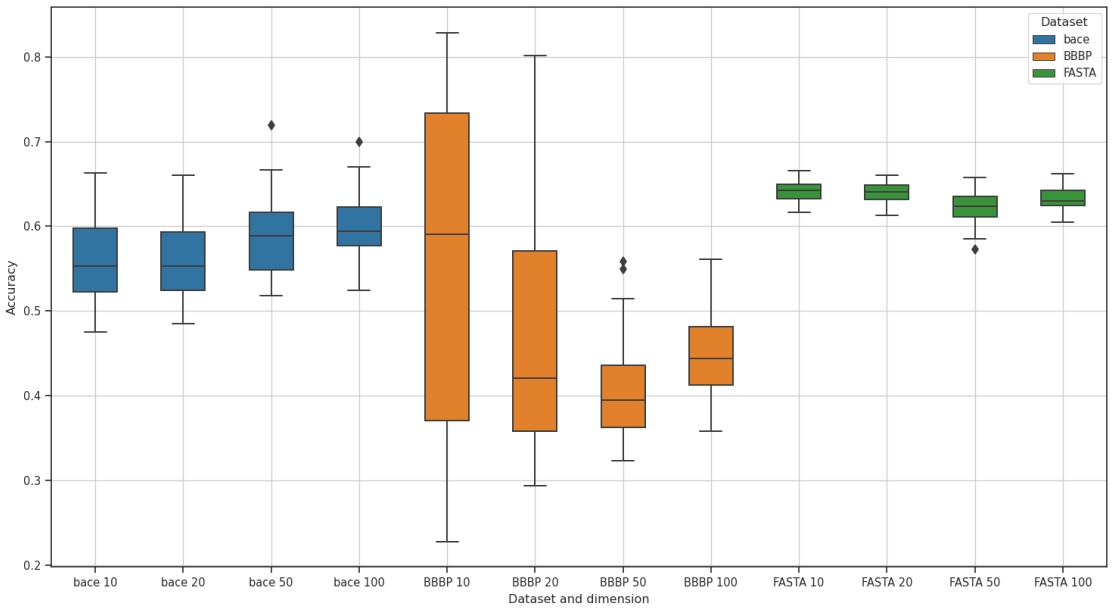

Figure 5.3: Experiment 2: Boxplots comparing the accuracy of different LDA embedding dimensions on the different datasets for the NBC

It is clear from Table 5.2 and Figure 5.3 that the LDA embeddings of lower dimension lose too much information for the BACE dataset for the NBC. The lower dimensional embeddings seem to capture the most information for the BBBP and SPS datasets. Furthermore, the NBC model performance is much lower than that of the SVM in Table 5.1. For the NBC model:

- The best LDA embedding dimension for the BACE dataset was 100 with an average accuracy of 59.85%, an average AUC of 61.35% and an average F1 score of 68.07%.
- The best LDA embedding dimension for the BBBP dataset was 10 with an average accuracy of 55.88%, an average AUC of 62.9% and an average F1 score of 57.94%.
- The best LDA embedding dimension for the SPS dataset was 10 with an average accuracy of 64.09%, an average AUC of 66.09% and an average F1 score of 65.81%.

5.5.3 Experiment 3: Optimal LDA embedding dimension for NN

We need to find the optimal feature dimension for the LDA embeddings for the NN. We compare feature dimensions of 10, 20, 50 and 100 for the LDA embeddings and take the dimension that provides the best result in terms of accuracy, AUC and F1 score.

Table 5.3: Experiment 3: Optimal LDA embedding dimension for NN

Data type	Dataset	n	Accuracy ± (SD)%	AUC ± (SD)%	F1 ± (SD)%
Smiles	BACE	10	58.01 ± 4.91	61.60 ± 5.79	42.61 ± 17.54
		20	63.52 ± 4.93	67.56 ± 5.60	54.31 ± 11.28
		50	68.62 ± 3.48	75.63 ± 3.07	64.64 ± 4.72
		100	**72.96 ± 2.91**	**80.38 ± 2.87**	**69.98 ± 3.64**
	BBBP	10	68.05 ± 15.22	64.56 ± 16.79	74.93 ± 15.67
		20	71.86 ± 15.26	71.64 ± 16.07	77.72 ± 15.71
		50	73.98 ± 16.24	74.53 ± 16.62	78.54 ± 16.39
		100	**79.39 ± 15.54**	**80.90 ± 15.76**	**83.35 ± 15.48**
FASTA	SPS	10	66.45 ± 1.00	73.10 ± 0.94	61.28 ± 2.18
		20	69.06 ± 0.72	76.34 ± 0.58	64.76 ± 1.28
		50	71.05 ± 0.57	78.64 ± 0.49	66.57 ± 0.97
		100	**71.37 ± 0.65**	**78.98 ± 0.56**	**66.81 ± 0.98**

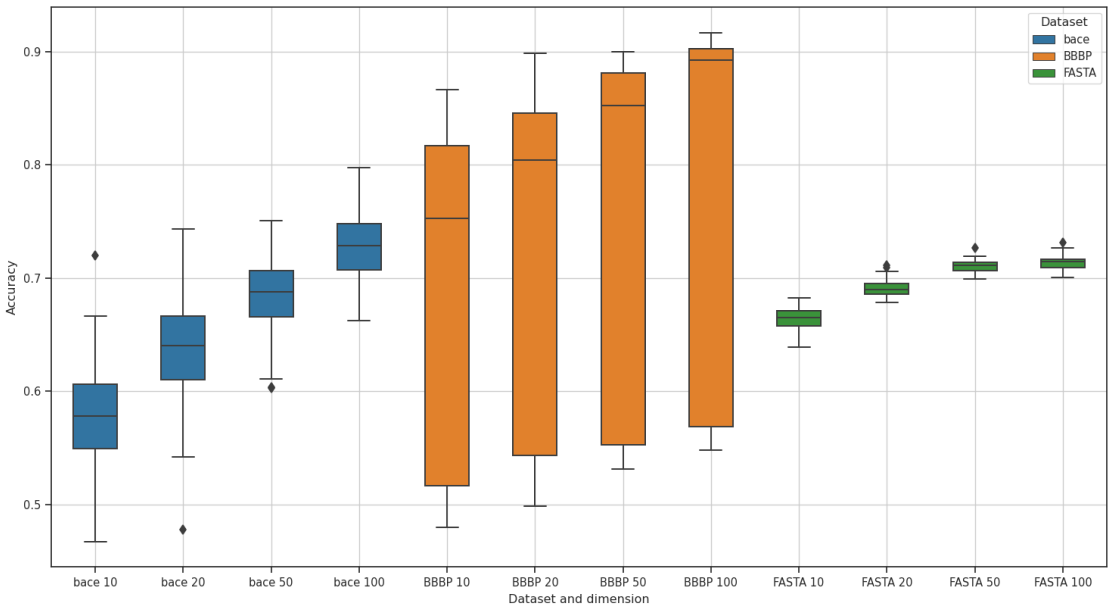

Figure 5.4: Experiment 3: Boxplots comparing the accuracy of different LDA embedding dimensions on the different datasets for the NN

It is clear from Table 5.3 and Figure 5.4 that the LDA embeddings of lower dimension lose too much information across all of the datasets for the NN. For the NN model:

- The best LDA embedding dimension for the BACE dataset was 100 with an average accuracy of 72.96%, an average AUC of 80.38% and an average F1 score of 69.98%.
- The best LDA embedding dimension for the BBBP dataset was 100 with an average accuracy of 79.39%, an average AUC of 80.90% and an average F1 score of 83.35%.
- The best LDA embedding dimension for the SPS dataset was 50 with an average accuracy of 71.37%, an average AUC of 78.98% and an average F1 score of 66.81%.

5.5.4 Experiment 4: Optimal word2vec embedding dimension for SVM

We need to find the optimal feature dimension for the word2vec embeddings for the SVM. We compare feature dimensions of 100, 200 and 300 for the word2vec embeddings and take the dimension that provides the best metric for the FASTA dataset.

Table 5.4: Experiment 4: Optimal word2vec embedding dimension for SVM

Data type	Dataset	n	Accuracy ± (SD)%	AUC ± (SD)%	F1 ± (SD)%
FASTA	SPS	**100**	**70.92 ± 0.15**	**69.86 ± 0.17**	**64.81 ± 0.24**
		200	70.91 ± 0.14	69.85 ± 0.14	64.80 ± 0.19
		300	70.90 ± 0.17	69.84 ± 0.17	64.81 ± 0.23

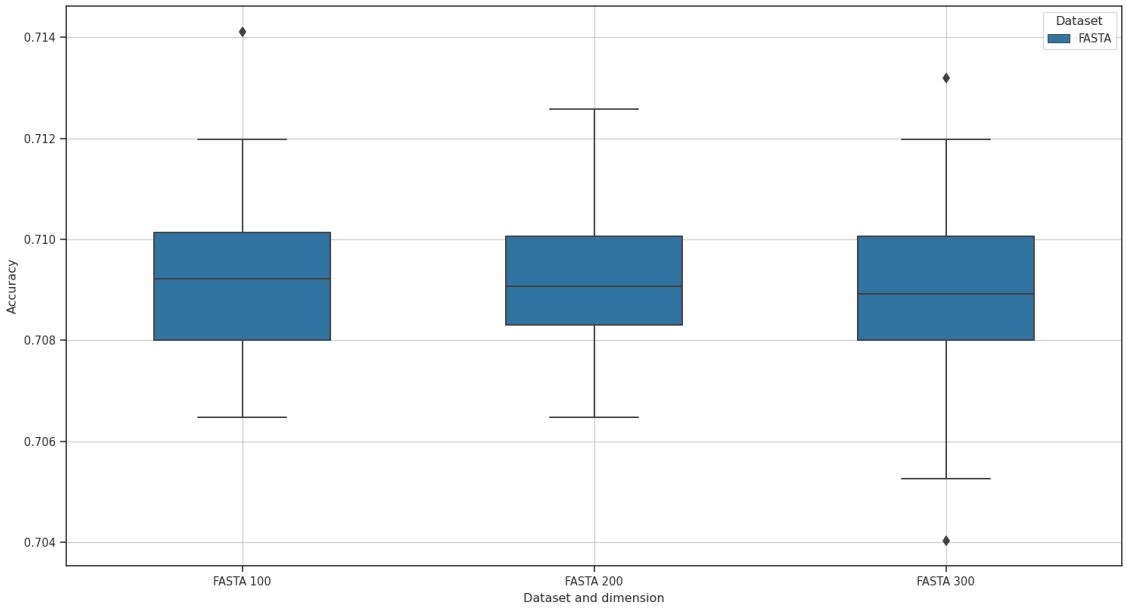

Figure 5.5: Experiment 4: Boxplots comparing the accuracy of different word2vec embedding dimensions on the SPS Fasta dataset for the SVM

The word2vec embeddings of lower dimension contain enough information and perform slightly better than those of higher dimension for the SPS dataset and the SVM model Table 5.4 and Figure 5.5. The best word2vec embedding dimension for the SPS dataset and SVM model was 100 with an average accuracy of 70.92%, an average AUC of 69.86% and an average F1 score of 64.81%.

5.5.5 Experiment 5: Optimal word2vec embedding dimension for NBC

We need to find the optimal feature dimension for the word2vec embeddings for the NBC. We compare feature dimensions of 100, 200 and 300 for the word2vec embeddings and take the dimension that provides the best metric for the FASTA dataset.

Table 5.5: Experiment 5: Optimal word2vec embedding dimension for NBC

Data type	Dataset	n	Accuracy ± (SD)%	AUC ± (SD)%	F1 ± (SD)%
FASTA	SPS	**100**	**65.49 ± 1.58**	65.12 ± 1.76	60.85 ± 2.42
		200	65.40 ± 1.22	65.20 ± 1.40	61.27 ± 2.16
		300	65.37 ± 1.23	**65.35 ± 1.42**	**61.80 ± 1.97**

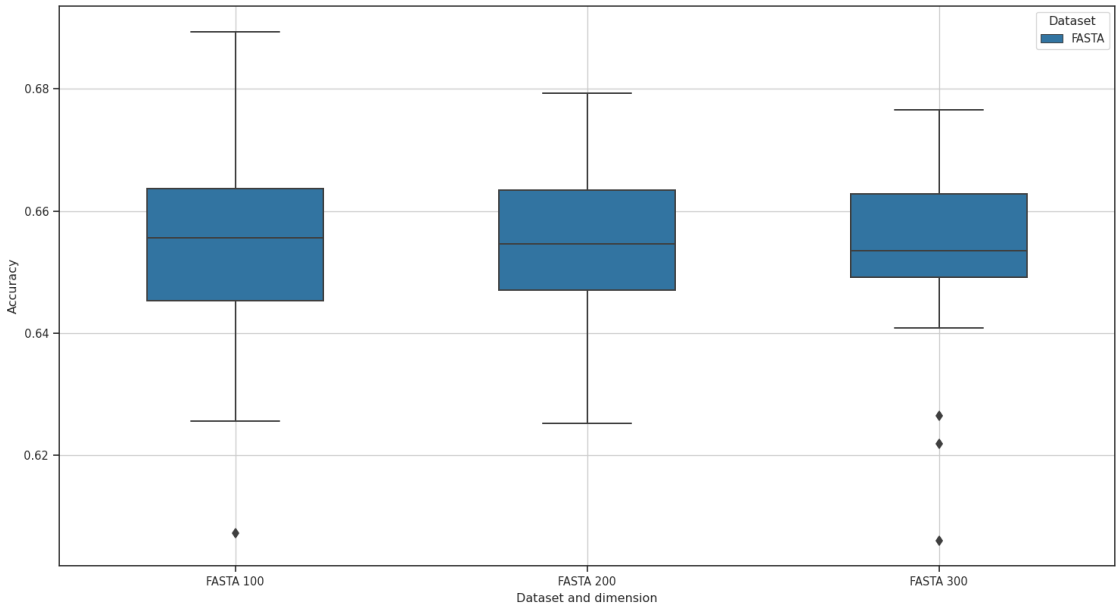

Figure 5.6: Experiment 5: Boxplots comparing the accuracy of different word2vec embedding dimensions on the SPS Fasta dataset for the NBC

The word2vec embeddings of lower dimension contain enough information and perform slightly better (in terms of accuracy) than those of higher dimension for the SPS dataset and the NBC Table 5.5 and Figure 5.6. The best word2vec embedding dimension for the SPS dataset and NBC was 100 with an average accuracy of 65.49%, an average AUC of 65.12% and an average F1 score of 60.85%.

5.5.6 Experiment 6: Optimal word2vec embedding dimension for NN

We need to find the optimal feature dimension for the word2vec embeddings for the NN. We compare feature dimensions of 100, 200 and 300 for the word2vec embeddings and take the dimension that provides the best metric for the FASTA dataset.

Table 5.6: Experiment 6: Optimal word2vec embedding dimension for NN

Data type	Dataset	n	Accuracy ± (SD)%	AUC ± (SD)%	F1 ± (SD)%
FASTA	SPS	100	71.08 ± 0.52	78.82 ± 0.37	65.78 ± 1.58
		200	**71.40 ± 0.44**	**79.17 ± 0.39**	65.70 ± 1.37
		300	71.27 ± 0.49	79.13 ± 0.45	**66.20 ± 1.66**

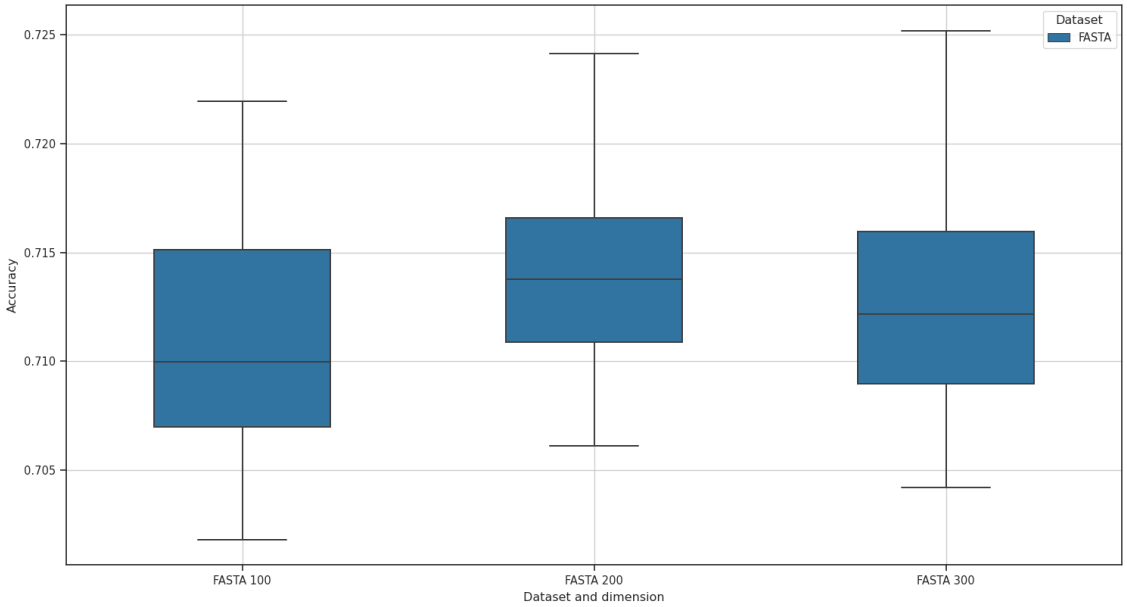

Figure 5.7: Experiment 6: Boxplots comparing the accuracy of different word2vec embedding dimensions on the SPS Fasta dataset for the NN

The word2vec embeddings of dimension 200 contain enough information and perform slightly better than those of dimension 100 or 200 for the SPS dataset and the NN Table 5.4 and Figure 5.7. The best word2vec embedding dimension for the SPS dataset and NN model was 200 with an average accuracy of 71.40%, an average AUC of 79.17% and an average F1 score of 65.70%.

5.5.7 Experiment 7: Finding the best embedding for the SVM

We compare the different embeddings in terms of accuracy, AUC and F1 score for the SVM. We use the optimal embedding dimension for the word2vec and LDA embeddings that achieved the best results in previous experiments. We use LDA embeddings of dimension 50 for the SPS dataset and dimension 100 for the BACE and BBBP dataset. We use word2vec dimensions of 100 for the SPS dataset and for the BACE and BBBP datasets we use a pre-trained model [23] that outputs word2vec embeddings of dimension 300.

Table 5.7: Experiment 7: Finding the best embedding for the SVM

Data type	Dataset	Technique	Accuracy ± (SD)%	AUC ± (SD)%	F1 ± (SD)%
Smiles	BACE	CVec	81.85 ± 1.12×10^{-14}	81.62 ± 0.00	80.14 ± 2.24×10^{-14}
		TF-IDF	**85.48 ± 1.12**$\times10^{-14}$	**85.41 ± 1.12**$\times10^{-14}$	**84.62 ± 1.12**$\times10^{-14}$
		LDA	72.13 ± 3.09	71.82 ± 3.19	68.94 ± 4.52
		word2vec	66.67 ± 1.12×10^{-14}	66.01 ± 1.12×10^{-14}	60.08 ± 1.12×10^{-14}
	BBBP	CVec	87.75 ± 2.24×10^{-14}	76.21 ± 1.12×10^{-14}	92.49 ± 1.12×10^{-14}
		TF-IDF	**91.18 ± 1.12**$\times10^{-14}$	**82.39 ± 0.00**	**94.55 ± 0.00**
		LDA	85.60 ± 1.32	73.35 ± 2.35	91.16 ± 0.81
		word2vec	80.64 ± 0.00	56.91 ± 0.00	88.89 ± 0.00
FASTA	SPS	**CVec**	**74.01 ± 1.12**$\times10^{-14}$	**73.05 ± 0.00**	68.66 ± 0.00
		TF-IDF	73.18 ± 0.00	72.68± 1.12×10^{-14}	68.91 ± 2.24×10^{-14}
		LDA	72.86 ± 0.55	72.59 ± 0.57	**69.14 ± 0.69**
		word2vec	70.92 ± 0.15	69.86 ± 0.17	64.81 ± 0.24

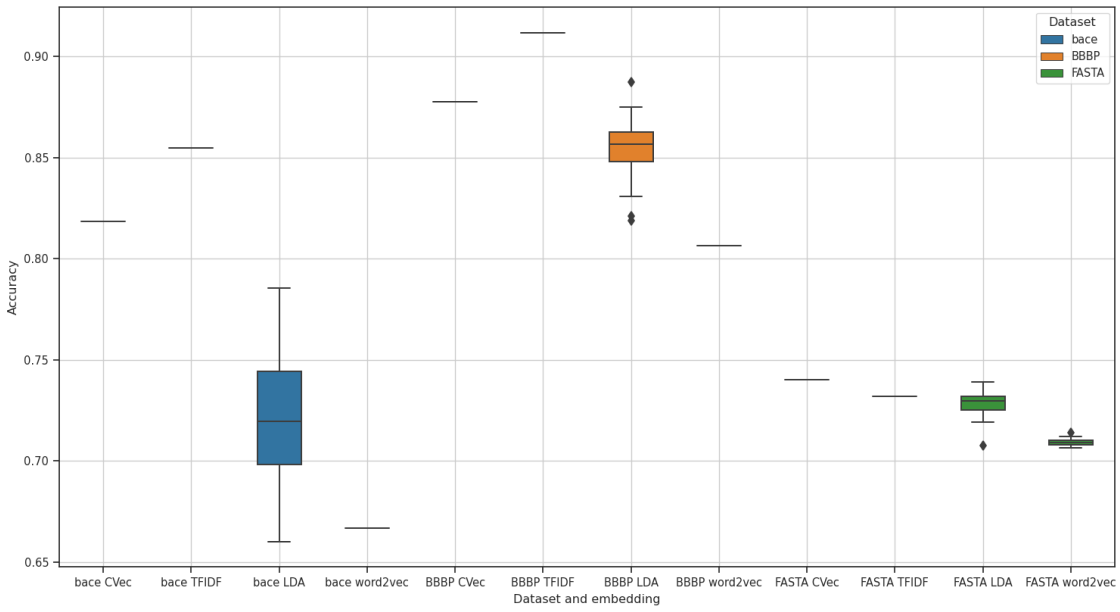

Figure 5.8: Experiment 7: Boxplots comparing the accuracy of different embedding techniques on the different datasets for the SVM

The word2vec and LDA embeddings are not the best embedding technique for any dataset for the SVM according to Table 5.7 and Figure 5.8. It appears that the best embeddings for the SVM are TF-IDF for the BACE and BBBP datasets (85.48% and 91.18% average accuracy) and CVec for the SPS dataset (74.01% average accuracy). It is interesting to note that TF-IDF seems to perform better for SMILES datasets whilst CVec performs better for the FASTA datasets for the SVM.

5.5.8 Experiment 8: Finding the best embedding for the NBC

We compare the different embeddings in terms of accuracy, AUC and F1 score for the NBC. We use the optimal embedding dimension for the word2vec and LDA embeddings, and compare it with count vectorisation (CVec) and TF-IDF. We use LDA embeddings of dimension 100 for the BACE dataset, dimension 10 for the SPS dataset and BBBP dataset. We use word2vec dimensions of 100 for the SPS dataset and for the BACE and BBBP datasets we use a pre-trained model [23] that outputs word2vec embeddings of dimension 300.

Table 5.8: Experiment 8: Finding the best embedding for the NBC

Data type	Dataset	Technique	Accuracy ± (SD)%	AUC ± (SD)%	F1 ± (SD)%
Smiles	BACE	**CVec**	**76.24 ± 1.12**$\times 10^{-14}$	**76.11 ± 1.12**$\times 10^{-14}$	**74.65 ± 1.12**$\times 10^{-14}$
		TF-IDF	67.66 ± 0.00	68.85 ± 1.12$\times 10^{-14}$	73.22 ± 1.12$\times 10^{-14}$
		LDA	59.85 ± 3.25	61.35 ± 3.13	68.07 ± 5.65
		word2vec	68.65 ± 1.12$\times 10^{-14}$	68.39 ± 1.12$\times 10^{-14}$	65.70 ± 1.12$\times 10^{-14}$
	BBBP	**CVec**	**82.84 ± 0.00**	**75.85 ± 0.00**	**88.92 ± 1.12**$\times 10^{-14}$
		TF-IDF	65.44 ± 0.00	71.06 ± 1.12$\times 10^{-14}$	73.35 ± 1.12$\times 10^{-14}$
		LDA	55.88 ± 19.69	62.99 ± 7.12	57.94 ± 27.87
		word2vec	75.25 ± 1.12$\times 10^{-14}$	61.42 ± 0.00	84.44 ± 1.12$\times 10^{-14}$
FASTA	SPS	**CVec**	**66.16± 1.12**$\times 10^{-14}$	**66.99 ± 0.00**	64.98 ± 0.00
		TF-IDF	45.57 ± 0.00	51.94± 1.12$\times 10^{-14}$	60.70 ± 0.00
		LDA	64.09 ± 1.05	66.09 ± 1.07	**65.81 ± 1.35**
		word2vec	65.49 ± 1.58	65.12 ± 1.76	60.85 ± 2.42

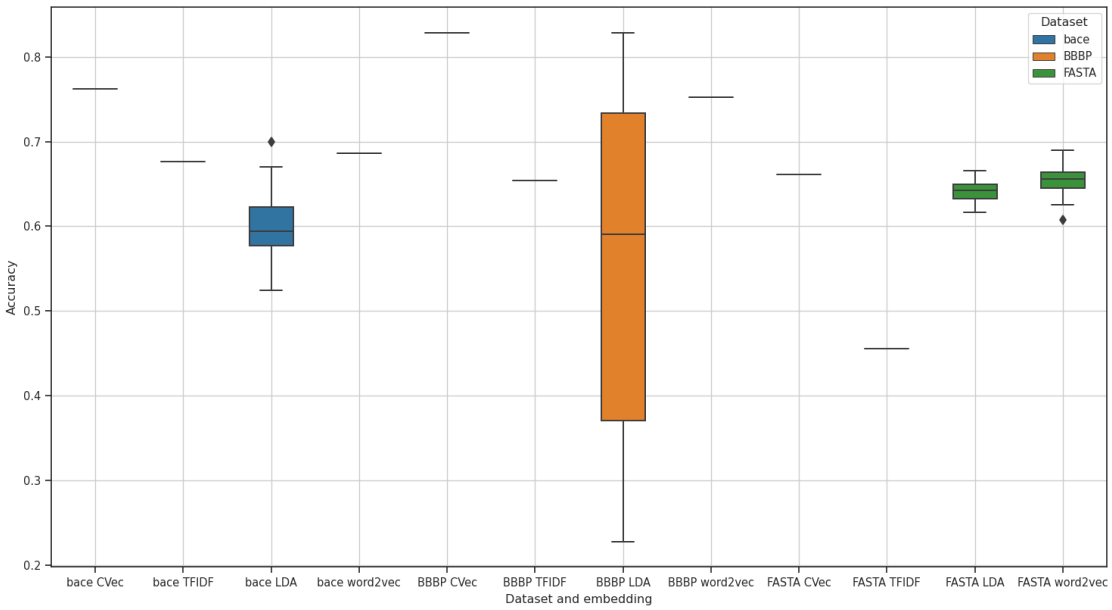

Figure 5.9: Experiment 8: Boxplots comparing the accuracy of different embedding techniques on the different datasets for the NBC

CVec is the best embedding technique for the NBC across all datasets as seen in Table 5.8 and Figure 5.9. CVec embeddings achieve 76.24% average accuracy on the BACE dataset, 82.84% average accuracy on the BBBP dataset and 66.16% average accuracy on the SPS dataset.

5.5.9 Experiment 9: Finding the best embedding for the NN

We compare the different embeddings in terms of accuracy, AUC and F1 score for the NN. We use the optimal embedding dimension for the word2vec and LDA embeddings. We use LDA embeddings of dimension 100 for the BACE dataset, BBBP dataset and SPS dataset. We use word2vec dimensions of 200 for the SPS dataset and for the BACE and BBBP datasets we use a pre-trained model [23] that outputs word2vec embeddings of dimension 300.

Table 5.9: Experiment 9: Finding the best embedding for the NN

Data type	Dataset	Technique	Accuracy ± (SD)%	AUC ± (SD)%	F1 ± (SD)%
Smiles	BACE	**CVec**	**88.25 ± 1.23**	**93.61 ± 0.48**	**87.61 ± 1.51**
		TF-IDF	83.38 ± 1.30	91.02 ± 0.73	82.31 ± 1.40
		LDA	72.96 ± 2.91	80.38 ± 2.87	69.98 ± 3.64
		word2vec	69.57 ± 0.91	76.20 ± 0.73	67.98 ± 2.09
	BBBP	**CVec**	**88.57 ± 11.63**	**90.67 ± 11.63**	**91.02 ± 11.65**
		TF-IDF	84.48 ± 13.76	86.20 ± 13.85	87.37 ± 13.81
		LDA	79.39 ± 15.54	80.90 ± 15.76	83.35 ± 15.48
		word2vec	59.57 ± 13.44	55.78 ± 13.42	65.42 ± 13.43
FASTA	SPS	**CVec**	**73.45 ± 0.58**	**81.03 ± 0.36**	**68.50 ± 0.99**
		TF-IDF	72.32 ± 0.49	79.75 ± 0.27	67.41 ± 1.03
		LDA	71.37 ± 0.65	78.98 ± 0.56	66.81 ± 0.98
		word2vec	71.40 ± 0.44	79.17 ± 0.39	65.70 ± 1.37

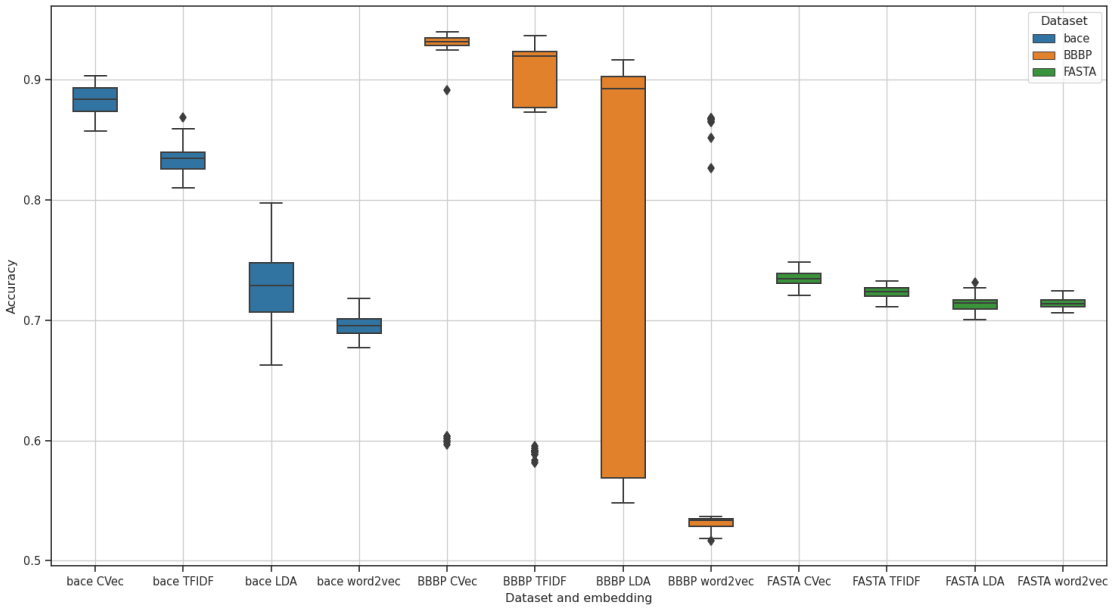

Figure 5.10: Experiment 9: Boxplots comparing the accuracy of different embedding techniques on the different datasets for the NN

CVec is the best embedding technique for the NN across all datasets as seen in Table 5.9 and Figure 5.10. CVec embeddings achieve 88.25% average accuracy on the BACE dataset, 88.57% average accuracy on the BBBP dataset and 73.45% average accuracy on the SPS dataset.

5.5.10 Experiment 10: Finding the best embedding and model combination for each dataset

We compare the best embedding and model combination for each dataset in terms of accuracy, AUC and F1 score for the NN. On the BACE dataset we use:

- TF-IDF embeddings with the SVM model (85.48% average accuracy Table 5.7).
- CVec embeddings with the NBC model (76.24% average accuracy Table 5.8).
- CVec embeddings with the NN model (88.25% average accuracy Table 5.9).

On the BBBP dataset we use:

- TF-IDF embeddings with the SVM model (91.18% average accuracy Table 5.7).
- CVec embeddings with the NBC model (82.84.24% average accuracy Table 5.8).
- CVec embeddings with the NN model (88.57% average accuracy Table 5.9).

On the SPS dataset we use:

- CVec embeddings with the SVM model (74.01% average accuracy Table 5.7).
- CVec embeddings with the NBC model (66.16% average accuracy Table 5.8).
- CVec embeddings with the NN model (73.45% average accuracy Table 5.9).

Table 5.10: Experiment 10: Finding the best embedding and model combination for each dataset

Dataset	Embedding	Model	Accuracy ± (SD)%	AUC ± (SD)%	F1 ± (SD)%
BACE	TF-IDF	SVM	$85.48 \pm 1.12\times10^{-14}$	$85.41 \pm 1.12\times10^{-14}$	$84.62 \pm 1.12\times10^{-14}$
	CVec	NBC	$76.24 \pm 1.12\times10^{-14}$	$76.11 \pm 1.12\times10^{-14}$	$74.65 \pm 1.12\times10^{-14}$
	CVec	**NN**	**88.25 ± 1.23**	**93.61 ± 0.48**	**87.61 ± 1.51**
BBBP	**TF-IDF**	**SVM**	**91.18 ± 1.12**$\times10^{-14}$	82.39 ± 0.00	**94.55 ± 0.00**
	CVec	NBC	82.84 ± 0.00	75.85 ± 0.00	$88.92 \pm 1.12\times10^{-14}$
	CVec	NN	88.57 ± 11.63	**90.67 ± 11.63**	91.02 ± 11.65
SPS	**CVec**	**SVM**	**74.01 ± 1.12**$\times10^{-14}$	73.05 ± 0.00	**68.66 ± 0.00**
	CVec	NBC	$66.16 \pm 1.12\times10^{-14}$	66.99 ± 0.00	64.98 ± 0.00
	CVec	NN	73.45 ± 0.58	**81.03 ± 0.36**	68.50 ± 0.99

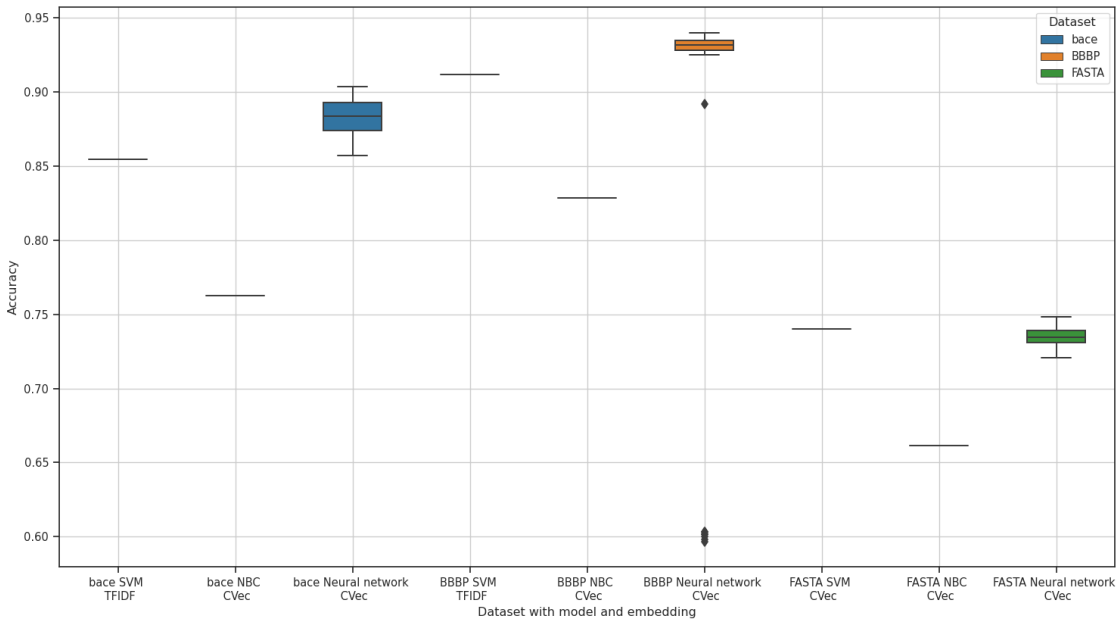

Figure 5.11: Experiment 10: Boxplots comparing the accuracy of the best model and embedding combination on each of the different datasets

CVec and the NN model were the best combination for the BACE dataset (average accuracy 88.25% Table 5.10), TF-IDF and a SVM were the best for the BBBP dataset (average accuracy 91.18% Table 5.10) and CVec and a SVM were the best for the SPS dataset (average accuracy 74.01% Table 5.10). These results are visualised in Figure 5.11).

5.5.11 Experiment 11: Comparison with Moleculenet

We compare the best embedding and model combination for the BACE and BBBP datasets, in terms of AUC, to the best model presented by Moleculenet [56]. Our best model and embedding combination for both of these datasets is CVec with a NN model. We make use of a "scaffold" split from the *deepchem* [8] [39] package to compare to the Moleculenet results. A scaffold split is more "difficult" than a random split as it splits the molecules according to the the core structure or scaffold of a molecule resulting in structurally different training and test sets [39].

[8] `https://deepchem.readthedocs.io/en/latest/api_reference/moleculenet.html`

Table 5.11: Experiment 11: Comparison with Moleculenet

Dataset	Method	AUC %
BBBP	Moleculenet	72.90
	Ours (CVec and NN)	**82.23**
BACE	**Moleculenet**	**86.7**0
	Ours (CVec and NN)	82.75

Our method scores a higher AUC (82.23% compared to 72.90% Table 5.11) for the BBBP dataset and a slightly lower AUC (82.75% compared to 86.70% Table 5.11) for the BACE dataset Table 5.11. We only use the SMILES strings, yet our method scores higher or only slightly lower than alternative approaches that use more features.

5.5.12 Conclusion

It should be noted that there is little to no variation between the 50 runs for CVec and TF-IDF embeddings Table 5.8 and Table 5.7. This is because these embeddings will be the same between runs due to the deterministic nature of CVec and TF-IDF. There is little variation in the classification model as well since we use the same random state for the data split between runs. There is greater variation between the runs that use LDA and word2vec embeddings since these embeddings require the training of a new LDA or word2vec model for each run. This new embedding model will differ slightly between runs and thus result in variation of the embeddings between runs. This coupled with the variation in the trained classifiers results in greater variation than for the CVec and TF-IDF embedding runs Table 5.8 and Table 5.7. In particular, there is greater variation for the LDA embeddings and the NBC and NN model for the BBBP dataset as can be seen in Table 5.2 and Table 5.3. We hypothesise that this is due to the large vocabulary size of 2115 which is almost equal to the number of data points for the BBBP dataset and the general sensitivity of NNs to the initialisation of weights. It seems that this dataset is particularly difficult to separate between the two classes when using LDA embeddings. Future work is required to investigate the variability between runs.

CVec and TF-IDF were the best embedding techniques - that we evaluated Table 5.10. CVec was the overall best embedding for all of the models evaluated on all of the datasets. The normalisation in the TF-IDF embeddings lost some of the required information and thus achieved slightly lower metrics than CVec. All of the global embedding techniques performed better than the local embedding techniques, such as word2vec. Overall our approach achieved better metrics for the BBBP dataset and achieved a slightly lower AUC for the BACE dataset from the Moleculenet benchmark [56].

Chapter 6

Word embeddings applied to protein-protein interaction prediction (Embeddings similarity)

In this chapter, we investigate the use of word embeddings for protein-protein interaction prediction, which we believe to be a novel approach. We make use of the LDA embeddings on the extracted protein sequences from the PDB dataset. We can run a hypothesis test since we have the ground truth for known protein-protein interactions. Before we make protein-protein interaction predictions we test if the average Bhattacharyya distance between the embeddings of proteins involved in a known protein-protein interaction are less than the average distances between those that are not. To do this we make use of an urn model [21].

6.1 Hypothesis test

In this section we describe our research approach and then summarise our results. We test the following null and alternative hypothesis:

- Null hypothesis: the average Bhattacharyya distance between the embeddings of proteins involved in a known protein-protein interaction are greater or equal to those that are not.
- Alternative hypothesis: the average Bhattacharyya distance between the embeddings of proteins involved in a known protein-protein interaction are less than those that are not.

6.1.1 Research approach

In order to test the null hypothesis:

1. Extract the protein sequences from the 10962 PDB files.
2. Create LDA embeddings for each protein sequence.
3. Group the sequences into ligands[1] and receptors [2].
4. Calculate the Bhattacharyya distance betweeen every receptor protein sequence embedding and every ligand protein sequence embedding (5481×5481 distances).
5. Calculate the mean distance between all of the embeddings that are part of a known protein-protein interaction. This is the hypothesised value.
6. We randomly sample 5481 distances from all of the distances 1000000 times and calculate the mean of each sample.
7. Finally, we calculate a p-value by counting the number of sample means that are less than the average distance between the embeddings that are part of known protein-protein interaction.

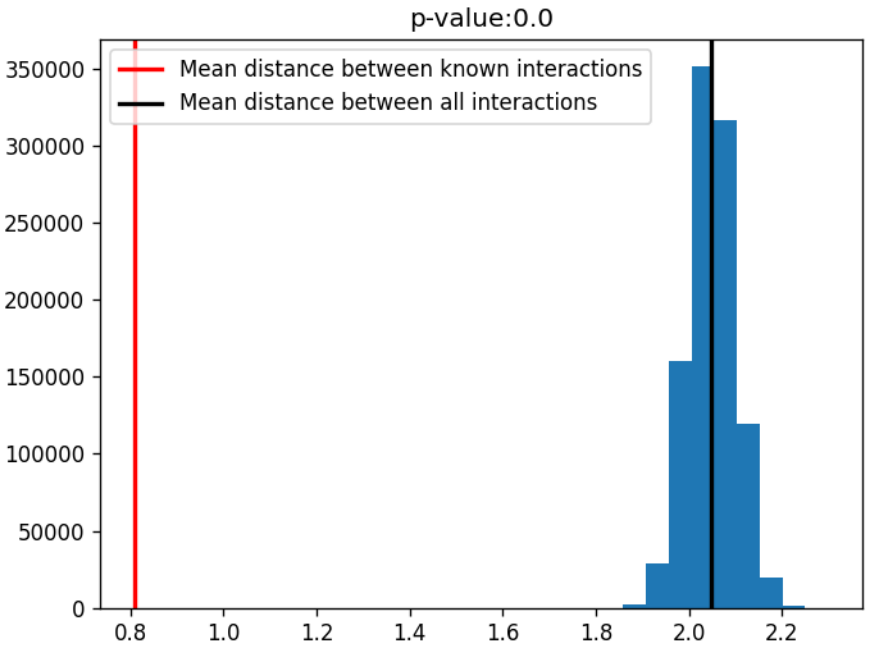

Figure 6.1: Histogram illustrating the sample mean distances with a red line indicating the known protein-protein interaction mean distance and a black line indicating the unknown protein-protein interaction mean distance

[1] A molecule that forms a complex with a receiving molecule or receptor.
[2] A special type of protein that functions by forming a complex with a ligand

6.1.2 Results

Since the p-value is 0 we reject the null hypothesis that the average difference between the embeddings of proteins involved in a known protein-protein interaction are greater or equal to those that are not. The mean distance between known interactions and all interactions is visualised in Figure 6.1. It is clear that the distance between embeddings of proteins that are in a known protein-protein interaction are less than those that are not. This means that the LDA embeddings are able to capture the interactions between proteins as semantic similarity using the protein sequences alone. We must highlight the relationship between p-values and large sample sizes. A larger sample size results in a smaller p-value [33]. The larger the sample size, the greater the accuracy of the statistical test, and the more likely it is to find a significant difference [49]. This is because statistical significance is dependent on both the effect size and the sample size. However, Cohen's d is 0.63 for this test indicating that the distances between the known interactions differs by 0.63 standard deviations to the distance between the unknown interactions. Cohen's d measures the standardised difference between the two sample means which is 0.63 standard deviations in Figure 6.1. Thus, we can differentiate between the two groups and make predictions on protein-protein interaction using the LDA embeddings.

6.2 Protein-protein interaction prediction

Since, we can differentiate between the known protein-protein interactions and unknown protein-protein interactions we can make predictions on protein-protein interaction using the LDA embeddings. We define the method we use for protein-protein interaction prediction and summarise the results.

6.2.1 Heuristic method

The Bhattacharyya distance between protein embeddings can also be used to predict the interactions in an inventive way. To predict whether or not two proteins interact we:

1. Calculate the distance between every receptor protein sequence embedding and every ligand protein sequence embedding.
2. For a given receptor:
   (a) Count how many ligands have a larger distance than the query ligand.
   (b) Take this count and divide it by the total number of ligands to get an interaction probability or $P(\textit{given receptor interacting with query ligand})$.

We only label known protein-protein interactions as interacting and label everything else as non-interacting. We must state that this labelling may not be strictly correct. In reality, some receptors may interact

with a ligand other than the known interacting ligand, but we use this labelling strategy to compare our results to other methods that use the same dataset [50]. Our method gives a probability of interaction for which a threshold can be chosen for binary prediction. The method is a simple heuristic and, as such, will not be influenced by incorrect labels as a trained model would. We evaluate the method by making a prediction on the known interacting ligand for each receptor as well as an unknown (dummy) ligand. Our method manages to achieve 83% AUC which is slightly better than the dMaSIF approach with and AUC of 82% [50].

6.2.2 Results

A simple heuristic is able to differentiate between known protein-protein interactions and unknown protein-protein interactions using protein sequences only Table 6.1. We achieve a high probability of interaction between certain receptors and several ligands. We note that this is more realistic than a simple binary classification which may not be completely correct. This means that this probabilistic prediction can be used to discover potential interactions that might not be known. We leave this for future work.

Table 6.1: Protein-protein interaction prediction

Method	AUC %
Ours	**83**
dMaSIF	82

Chapter 7

Conclusion

In this dissertation, we derived latent embeddings for molecular text data and evaluated the use of these embeddings as features and for representing a corpus of data in vector space. CVec, TF-IDF, LDA and word2vec embeddings provide a useful transformation of molecular text data to a vector space. These embeddings reduce the dimensionality of the data whilst keeping relative similarity between observations in their transformed state. In this way, a small amount of data can be used to understand a dataset. We evaluated the use of these embeddings as a set of features for ML models, for representing a corpus of data in vector space (embedding similarity) and lastly, we used semantic calculations on this representation to predict protein-protein interaction.

7.1 Use as an ML feature vector

For use as feature vectors for ML models, CVec and TF-IDF were the best embedding techniques - that we evaluated. CVec was the overall best embedding for all of the models evaluated on all of the datasets. The normalisation in the TF-IDF embeddings lost some of the required information and thus achieved slightly lower metrics than CVec. It should be noted that there is little to no variation between the 50 runs for CVec and TF-IDF embeddings. This is because these embeddings will be the same between runs due to the deterministic nature of CVec and TF-IDF. There is little variation in the classification model as well since we use the same random state for the data split between runs. There is greater variation between the runs that use LDA and word2vec embeddings since these embeddings require the training of a new LDA or word2vec model for each run. This new embedding model will differ slightly between runs and thus result in variation of the embeddings between runs. This coupled with the variation in the trained classifiers results in greater variation than for the CVec and TF-IDF embedding runs. In particular, there is greater variation for the LDA embeddings and the NBC and NN model for the BBBP dataset as can be seen in Table 5.2 and Table 5.3. We hypothesise that this is due to the large vocabulary

size of 2115 which is almost equal to the number of data points for the BBBP dataset and the general sensitivity of NNs to the initialisation of weights. It seems that this dataset is particularly difficult to separate between the two classes when using LDA embeddings. Future work is required to investigate the variability between runs. The LDA embeddings achieved lower metrics than the TF-IDF embeddings. The topic-molecule distribution of the LDA embeddings did not capture enough relative information to be used as a feature vector for ML models. The embedding of the local features used by word2vec did not benefit classification performance. All of the global embedding techniques performed better than the local embedding techniques, such as word2vec, for use as an ML feature vector. Overall our approach achieved better metrics for the BBBP dataset and achieved a slightly lower AUC for the BACE dataset from the Moleculenet benchmark [56]. We must note that our methods achieve similar or better results whilst being computationally efficient[1].

7.2 Representing a corpus of data in vector space

LDA embeddings proved useful as a representation of the corpus in vector space. We rejected the null hypothesis that the average distance between the embeddings of proteins involved in a known protein-protein interaction is greater or equal to those that are not in a known protein-protein interaction. This means that LDA embeddings between known interacting proteins were more similar, in vector, space than those that were not in a known interaction. However, this approach has a limitation due to the relationship between p-values and the large sample size. Despite this limitation, the distances between the known interactions differ by 0.63 standard deviations from the distance between the unknown interactions.

7.3 Word embeddings applied to protein-protein interaction prediction (Embeddings similarity

We showed that a heuristic can be used on the distances between the embeddings to make predictions on protein-protein interaction. This simple technique was able to achieve a higher AUC value than current DL approaches on the given dataset. Our heuristic is computationally efficient and requires much less compute than other approaches. The method does not require any training data and, as such, will not be influenced by incorrect labels or a class imbalance as a trained model would. This is particularly important when in reality, some receptors may interact with a ligand other than the known interacting ligand. Thus it is better to provide a probabilistic prediction instead of a binary one. Further, a threshold can be chosen to get binary predictions for a specific task. However, this method is limited because LDA

[1]It took less than 10 minutes to train and run predictions for the longest single run on the largest dataset on the machine described in Chapter 5.

is a topic model and several proteins may have topics in common with many other proteins. This will hinder the results and these proteins may be predicted to bind with extremely low probability. A heuristic that takes the size of topics into account may provide better results for these cases.

7.4 Future work

The use of a learnt embedding for machine learning tasks warrants further investigation. A learnt embedding can be integrated into the architecture of the classifier or regressor and may improve prediction results by providing an embedding that is learnt for the specific task. We believe that further work should be done, particularly on global embeddings. Since global embeddings outperformed local embeddings for all tasks. We believe that the LDA embedding could have potential use for explainability due to its effectiveness in corpus representation. Heuristics that take into account the size of topics should also be investigated for protein-protein interaction prediction. Overall, these embeddings have proven useful for the specific tasks evaluated and may apply to other problems not evaluated here.

Bibliography

[1] David Anastasiu, Andrea Tagarelli, and George Karypis. *Document Clustering: The Next Frontier*, pages 305–338. Chapman and Hall/CRC, 01 2013.

[2] David Andrzejewski. Modeling protein–protein interactions in biomedical abstracts with latent dirichlet allocation. Technical report, University of Wisconsin-Madison, 2006.

[3] Alya A Arabi. Artificial intelligence in drug design: algorithms, applications, challenges and ethics. *Future Drug Discovery*, 3(2):FDD59, 2021.

[4] Ehsaneddin Asgari and Mohammad RK Mofrad. Continuous distributed representation of biological sequences for deep proteomics and genomics. *PLOS ONE*, 10(11):e0141287, 2015.

[5] RCSB Protein Data Bank. Homepage.

[6] Guy W Bemis and Mark A Murcko. The properties of known drugs. 1. molecular frameworks. *Journal of Medicinal Chemistry*, 39(15):2887–2893, 1996.

[7] Anil Bhattacharyya. On a measure of divergence between two multinomial populations. In *Sankhyā: the Indian Journal of Statistics*, pages 401–406. JSTOR, 1946.

[8] David M Blei, Andrew Y Ng, and Michael I Jordan. Latent dirichlet allocation. *Journal of Machine Learning Research*, 3(Jan):993–1022, 2003.

[9] Bernhard E Boser, Isabelle M Guyon, and Vladimir N Vapnik. A training algorithm for optimal margin classifiers. In *Proceedings of the Fifth Annual Workshop on Computational Learning Theory*, pages 144–152, 1992.

[10] Yaw-Huei Chen and Shu-Fong Li. Using latent Dirichlet allocation to improve text classification performance of support vector machine. In *2016 IEEE Congress on Evolutionary Computation (CEC)*, pages 1280–1286. IEEE, 2016.

[11] Steven P Crain, Ke Zhou, Shuang-Hong Yang, and Hongyuan Zha. Dimensionality reduction and topic modeling: From latent semantic indexing to latent dirichlet allocation and beyond. In *Mining Text Data*, pages 129–161. Springer, 2012.

[12] Alta De Waal and Etienne Barnard. Evaluating topic models with stability. In *Nineteenth Annual Symposium of the Pattern Recognition Association of South Africa (PRASA 2008)*, volume 5221, pages 79–84, 2008.

[13] Jianyuan Deng, Zhibo Yang, Iwao Ojima, Dimitris Samaras, and Fusheng Wang. Artificial intelligence in drug discovery: applications and techniques. *Briefings in Bioinformatics*, 23(1):bbab430, 2022.

[14] Yi Ding, Minchun Chen, Chao Guo, Peng Zhang, and Jingwen Wang. Molecular fingerprint-based machine learning assisted QSAR model development for prediction of ionic liquid properties. *Journal of Molecular Liquids*, 326:115212, 2021.

[15] Charles Elkan. Deriving TF-IDF as a Fisher kernel. In *International Symposium on String Processing and Information Retrieval*, pages 295–300. Springer, 2005.

[16] Charles Fefferman, Sanjoy Mitter, and Hariharan Narayanan. Testing the manifold hypothesis. *Journal of the American Mathematical Society*, 29(4):983–1049, 2016.

[17] Pablo Gainza, Freyr Sverrisson, Frederico Monti, Emanuele Rodola, D Boscaini, MM Bronstein, and BE Correia. Deciphering interaction fingerprints from protein molecular surfaces using geometric deep learning. *Nature Methods*, 17(2):184–192, 2020.

[18] Robert Glem, Andreas Bender, Catrin Hasselgren, Lars Carlsson, Scott Boyer, and James Smith. Circular fingerprints: Flexible molecular descriptors with applications from physical chemistry to adme. *IDrugs : the investigational drugs journal*, 9:199–204, 04 2006.

[19] Tristan Harris. Probabilistic distributional semantic methods for small unlabelled text. In *SACAIR 2020 Proceedings: Applications of AI*, page 133. SACAIR, 2020.

[20] Steven Henikoff and Jorja G Henikoff. Amino acid substitution matrices from protein blocks. *Proceedings of the National Academy of Sciences*, 89(22):10915–10919, 1992.

[21] Fred M Hoppe. Pólya-like urns and the Ewens' sampling formula. *Journal of Mathematical Biology*, 20(1):91–94, 1984.

[22] Nabil Ibtehaz and Daisuke Kihara. Application of sequence embedding in protein sequence-based predictions. *arXiv preprint arXiv:2110.07609*, 2021.

[23] Sabrina Jaeger, Simone Fulle, and Samo Turk. Mol2vec: unsupervised machine learning approach with chemical intuition. *Journal of chemical information and modeling*, 58(1):27–35, 2018.

[24] Tamer N Jarada, Jon G Rokne, and Reda Alhajj. A review of computational drug repositioning: strategies, approaches, opportunities, challenges, and directions. *Journal of Cheminformatics*, 12(1):1–23, 2020.

[25] Xianfeng Jia, Bowen Dai, Zhaoxian Zhu, Jitong Wang, Wenming Qiao, Donghui Long, and Licheng Ling. Strong and machinable carbon aerogel monoliths with low thermal conductivity prepared via ambient pressure drying. *Carbon*, 108:551–560, 2016.

[26] Dan Jurafsky and James H. Martin. *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition.* Pearson Prentice Hall, 2021.

[27] Ichio Kikuchi and Akihito Kikuchi. Latent dirichlet allocation and objective functions to explore chemical space. *IRCQE*, 2021.

[28] Hyunseob Kim, Jeongcheol Lee, Sunil Ahn, and Jongsuk Ruth Lee. A merged molecular representation learning for molecular properties prediction with a web-based service. *Scientific Reports*, 11(1):1–9, 2021.

[29] Grzegorz Kondrak. N-gram similarity and distance. In *International Symposium on String Processing and Information Retrieval*, pages 115–126. Springer, 2005.

[30] Carolin Kosiol and Nick Goldman. Different versions of the dayhoff rate matrix. *Molecular Biology and Evolution*, 22(2):193–199, 2005.

[31] Greg Landrum et al. Rdkit: Open-source cheminformatics software. 2016. *URL http://www. rdkit. org/, https://github. com/rdkit/rdkit*, 149(150):650, 2016.

[32] Wenxin Liang, Ran Feng, Xinyue Liu, Yuangang Li, and Xianchao Zhang. Gltm: A global and local word embedding-based topic model for short texts. *IEEE Access*, 6:43612–43621, 2018.

[33] Mingfeng Lin, Henry Lucas, and Galit Shmueli. Too big to fail: Large samples and the p-value problem. *Information Systems Research*, 24:906–917, 12 2013.

[34] Hans Peter Luhn. A statistical approach to mechanized encoding and searching of literary information. *IBM Journal of Research and Development*, 1(4):309–317, 1957.

[35] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.

[36] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019.

[37] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[38] Jonathan K Pritchard, Matthew Stephens, and Peter Donnelly. Inference of population structure using multilocus genotype data. *Genetics*, 155(2):945–959, 2000.

[39] Bharath Ramsundar, Peter Eastman, Patrick Walters, Vijay Pande, Karl Leswing, and Zhenqin Wu. *Deep Learning for the Life Sciences*. O'Reilly Media, 2019. `https://www.amazon.com/Deep-Learning-Life-Sciences-Microscopy/dp/1492039837`.

[40] Michael K. Reddy. amino acid, Nov 2022.

[41] Ahmet Sureyya Rifaioglu, Heval Atas, Maria Jesus Martin, Rengul Cetin-Atalay, Volkan Atalay, and Tunca Doğan. Recent applications of deep learning and machine intelligence on in silico drug discovery: methods, tools and databases. *Briefings in Bioinformatics*, 20(5):1878–1912, 2019.

[42] Stephen Robertson. Understanding inverse document frequency: on theoretical arguments for IDF. In *Journal of Documentation*, volume 60, pages 503–520. Emerald Group Publishing Limited, 2004.

[43] Gerard Salton and Christopher Buckley. Term-weighting approaches in automatic text retrieval. *Information Processing & Management*, 24(5):513–523, 1988.

[44] Nadine Schneider, Nikolas Fechner, Gregory A Landrum, and Nikolaus Stiefl. Chemical topic modeling: Exploring molecular data sets using a common text-mining approach. *Journal of Chemical Information and Modeling*, 57(8):1816–1831, 2017.

[45] Omid Shahmirzadi, Adam Lugowski, and Kenneth Younge. Text similarity in vector space models: a comparative study. In *2019 18th IEEE international Conference on Machine Learning and Applications (ICMLA)*, pages 659–666. IEEE, 2019.

[46] S Shivashankar, S Srivathsan, Balaraman Ravindran, and Ashish V Tendulkar. Multi-view methods

for protein structure comparison using latent dirichlet allocation. *Bioinformatics*, 27(13):i61–i68, 2011.

[47] Lavneet Singh, Girija Chetty, and Dharmendra Sharma. A novel approach to protein structure prediction using PCA or LDA based extreme learning machines. In *International Conference on Neural Information Processing*, pages 492–499. Springer, 2012.

[48] Hannes Stärk, Octavian-Eugen Ganea, Lagnajit Pattanaik, Regina Barzilay, and Tommi Jaakkola. Equibind: Geometric deep learning for drug binding structure prediction. *arXiv preprint arXiv:2202.05146*, 2022.

[49] Gail M Sullivan and Richard Feinn. Using effect size- or why the p value is not enough. *Journal of Graduate Medical Education*, 4(3):279–282, 2012.

[50] Freyr Sverrisson, Jean Feydy, Bruno E Correia, and Michael M Bronstein. Fast end-to-end learning on protein surfaces. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 15272–15281, 2021.

[51] Turing. Word embeddings in nlp: A complete guide. `https://www.turing.com/kb/guide-on-word-embeddings-in-nlp`. Accessed: 2022-03-30.

[52] Vaccinationist. Physiologically relevant skeletal formula of lhistidine. created with chemdoodle 7.0.2 and adobe illustrator cc 2015, 2016.

[53] Justin Johan Jozias van Der Hooft, Joe Wandy, Michael P Barrett, Karl EV Burgess, and Simon Rogers. Topic modeling for untargeted substructure exploration in metabolomics. *Proceedings of the National Academy of Sciences*, 113(48):13738–13743, 2016.

[54] Hanna M Wallach. Topic modeling: beyond bag-of-words. In *Proceedings of the 23rd International Conference on Machine Learning*, pages 977–984, 2006.

[55] David Weininger. Smiles, a chemical language and information system. 1. introduction to methodology and encoding rules. *Journal of Chemical Information and Computer Sciences*, 28(1):31–36, 1988.

[56] Zhenqin Wu, Bharath Ramsundar, Evan N Feinberg, Joseph Gomes, Caleb Geniesse, Aneesh S Pappu, Karl Leswing, and Vijay Pande. Moleculenet: a benchmark for molecular machine learning. *Chemical Science*, 9(2):513–530, 2018.

[57] Cao Xiao, Ping Zhang, W Chaovalitwongse, Jianying Hu, and Fei Wang. Adverse drug reaction prediction with symbolic latent Dirichlet allocation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 31(1), 2017.

[58] Hima Bindu Yalamanchili, Soon Jye Kho, and Michael L Raymer. Latent dirichlet allocation for classification using gene expression data. In *2017 IEEE 17th International Conference on Bioinformatics and Bioengineering (BIBE)*, pages 39–44. IEEE, 2017.

[59] Kevin K Yang, Zachary Wu, Claire N Bedbrook, and Frances H Arnold. Learned protein embeddings for machine learning. *Bioinformatics*, 34(15):2642–2648, 2018.

[60] Yang Zhang. What is fasta format?

[61] Yin Zhang, Rong Jin, and Zhi-Hua Zhou. Understanding bag-of-words model: a statistical framework. *International Journal of Machine Learning and Cybernetics*, 1(1):43–52, 2010.