Jagiellonian University
Department of Theoretical Computer Science

Anna Maria Szymańska

Dimensionality reduction methods for Extended Connectivity Fingerprints

Bachelor Thesis
Supervisor: dr hab. Krzysztof Turowski

July 2025

I would like to express sincere gratitude to my Supervisor for the time devoted,
the valuable guidance, and the kindness shown towards the proposed topic,
as well as for the assistance in giving my thesis the right direction.

Contents

Notation

r - radius, parameter of the ECFP algorithm

N - length of the original encoding

n - length of the reduced encoding

$s(X,Y) = \frac{|X \cap Y|}{|X \cup Y|}$ - similarity coefficient of sets in the Jaccard metric [16]

$d(X,Y) = 1 - s(X,Y)$ - distance of sets in the Jaccard metric

$s(u,v) = \frac{\langle u,v \rangle}{\|u\|^2 + \|v\|^2 - \langle u,v \rangle}$ - similarity coefficient of vectors in the Jaccard metric [16]

$d(u,v) = 1 - s(u,v)$ - distance of vectors in the Jaccard metric

Chapter 1

Introduction

1.1 Numerical representation of chemical structures

Combinatorics for enumerating isomers of a compound, topology for the purpose of studying entanglements of polymer strands or graph theory for sampling a molecule from chemical space - those are examples of topics at the intersection of mathematics and computer science that have found significant applications in cheminformatics [5]. They provide fundamental methods for *in silico* analysis of chemical compounds [5]. With its development, the efficient digital representation of molecular structures is becoming crucial. It is particularly important as one of the key features of cheminformatics methods is their focus on calculations involving large collections of small molecules [15]. It is also relevant in the context of the vast size of the chemical space. The estimated number of potential compounds with drug-like properties is about 10^{60}, a number impossible to store physically as samples [15]. It is therefore necessary to represent and store a digital form of at least some of the chemical structures efficiently because the entire collection is still far from accessible.

Some of the largest databases of chemical structures are Enamine REAL containing 9.6 billion compounds, ZINC20 with 750 million compounds, or PubChem with 112 million compounds [12, 19, 25]. One of their main applications is virtual screening, a procedure for scoring, and filtering compounds based on similarity and substructure search or prediction of properties [15]. Virtual screening aims to rank a set of chemical structures, allowing, for example, to identify compounds similar to those that exhibit the desired properties or select those for synthesis and further analysis [3]. Digital databases offer the possibility of storing "virtual molecules", which are structures that have not yet been obtained, but can be synthesised with modern methods and are potentially of interest in screening [15].

A popular format for representing chemical compounds is the SMILES (Simplified Molecular Input Line Entry System) notation, decoding the structure of a compound into a sequence of letters (atomic symbols), numbers and special characters describing bond types, and brackets to indicate nested substructures [3, 21]. This notation allows compounds to be stored as strings of different lengths, each in a separate line. The key property is that it can be converted from and to a graph representation without loss of information about connectivity. Although SMILES is a convenient and commonly used method for storing data on a computer, it is typically converted back into a molecular graph for computational purposes [15].

A natural approach to representing molecules is therefore to consider them as graphs and then encode them in numerical form using graph methods. This corresponds well to one of the main questions in cheminformatics, namely whether two compounds are similar and whether a compound contains a given substructure, for example, a functional group responsible for certain property or a core structure of a class of compounds [10]. Determining the common substructures of molecular graphs reduces to the subgraph isomorphism problem, which is NP-complete [7]. Although heuristic algorithms such as Simulated Annealing and Genetic Algorithm have been developed for this problem, their application to ultra-large chemical databases remains inefficient [15, 17]. The procedure used in practice applies a two-step search, first eliminating non-matching molecules from further screening, and then evaluating only the potential candidates [15]. The method is based on performing comparisons and calculations on numerical representations instead of molecular graphs, thus approaching the difficult problem of graph isomorphism by simpler vector comparisons. It includes the heuristic assumption that such a simplified representation preserves sufficient information about mutual similarity and substructures present in the molecular graphs.

Different types of compound encodings can be classified by their type and the complexity of the information they capture. One-dimensional descriptors are single numerical values derived from a molecule's structure or properties [15]. Examples include polar surface area, molecular weight or the number of hydrogen bond acceptors and donors. Some of 1D descriptors can only be determined experimentally [3]. Two-dimensional descriptors, often referred to as fingerprints, are binary or numerical vectors derived from the molecular graph, representing the molecule's connectivity [15]. The most extensive, 3D descriptors are based on the full three-dimensional arrangement of a molecule - the coordinates of individual atoms or angles between bonds. They can be extended to include also physico-chemical properties of a compound [8]. The descriptors can capture features of the entire structure or specific functional groups. The choice of a suitable numerical representation is crucial, for example, for initial filtering of non-matching molecules in database queries. The selected descriptor should be adjusted to the analysed collection of compounds. A set of features may effectively distinguish a range of organic compounds, but it might be insufficient when differentiating molecules within a single family, such as hydrocarbons [15].

For similarity and substructure searches, 2D descriptors are most commonly used, as they offer a balance between the accuracy of information and the efficiency of storage and calculations. The main types of 2D chemical fingerprints are structural keys and hashed fingerprints [15]. Structural keys, such as MACCS (Molecular ACCess System), are generated based on a predefined dictionary of molecular fragments, where each bit represents the presence of a specific substructure [15]. The dictionary must be carefully adapted to the dataset, as adding new substructures would require updating all fingerprints stored in the database. Nevertheless, the advantage of this type of encoding is its ease of interpretation. The second type is hashed fingerprints, such as ECFP (Extended-Connectivity Fingerprint), consisting of hash values of molecule substructures projected onto a vector [15]. The hash of each substructure of a newly added compound is included in the encoding, without requiring changes to existing fingerprints.

Among various types of molecular representations, Extended-Connectivity Fingerprints have become prominent [10, 15]. As binary fingerprints, they are especially well-suited for substructure search, as a quick comparison of bits can immediately reveal the absence of the query substructure (when not all the corresponding bits are set) [15]. Another noteworthy practical advantage is that

fingerprints like ECFP can serve as a form of encryption for chemical structures. In collaboration between different companies, where structural data is exchanged, it is often desirable to keep the exact structures private. In such cases, one may only share the ECFP encodings which are sufficient for structure-activity relationship analysis but are not easily reversible to unambiguously reconstruct the original compounds [14]. The ECFP representation also has properties that make it particularly well-suited for use with the Jaccard similarity (also known as Tanimoto similarity), which is favoured due to its fast computation and natural interpretation for binary vectors [10].

The aim of this study is to analyse the properties of the ECFP algorithm and, on this basis, to evaluate and compare dimensionality reduction methods applied to ECFP fingerprints, in the context of efficient similarity search.

1.2 Extended Connectivity Fingerprints

One widely used solution for molecular representation is Extended Connectivity Fingerprints (ECFP), a class of topological fingerprints based on neighbourhood information of atoms [20]. They are intended to reflect the specific substructures present in the molecular graph. By adjusting which features are included in the generic algorithm, they can be adapted to encode specific properties of a compound. The results can be tuned by choosing what to consider when assigning the initial identifiers: atomic type and mass, bond multiplicities to non-hydrogen neighbours, etc.

1.2.1 The ECFP algorithm

The ECFP encoding is computed with the following algorithm [20]:

Algorithm 1 ECFP fingerprint generation

Input: Graph $G = (V, E)$
Output: Set of identifiers L

for $v_j \in V$ **do**
  $h_j^0 \leftarrow \mathrm{id}(v_j)$
end for
$L \leftarrow \{h_j^0 \colon j \in |V|\}$
for $i \in \{1, \ldots, r\}$ **do**
  $L_j^i \leftarrow \mathrm{sort}\big(\{(e_{jk}, h_k^{i-1}) : v_j v_k \in E\}\big)$, where e_{jk} is the multiplicity of bond (v_j, v_k)
  $h_j^i \leftarrow \mathrm{hash}\big(i,\ h_j^{i-1},\ L_j^i\big)$
  $L.\mathrm{insert}(h_j^i)$
end for
$L \leftarrow \mathrm{unique}(L)$
return L

As a standard approach, the generated list of identifiers, also called connectivity values, is transformed into a binary vector of a fixed length N [15]. The most commonly used parameter values are $N \in \{1024, 2048, 4096\}$ and $r \in \{1, 2, 3\}$ [13]. The final representation in the form of a binary vector can be interpreted as a dictionary indexed by encoded substructures.

As follows from the algorithm, the properties of ECFP encoding strongly depend on the choice of parameters, in particular the value of r. The parameter r determines the number of generated

identifiers, which is bounded from above by $(r+1)$ times the number of atoms in the molecule. The algorithm may return fewer identifiers if certain substructures are repeated or hash collisions occur. Thus, at some level, the ratio of the number of generated identifiers to the number of atoms can provide a measure of the complexity of the structure. If a molecule contains various substructures, it is sufficient to consider the close neighbourhood of atoms to distinguish it from a similar compound. The choice of r is crucial for determining similarity in the Jaccard metric. It is possible to give an example of molecules that need the value of parameter r linear in terms of their size to distinguish between them. As an example, two carbon chains of lengths k and $k+1$ for any r less than $\frac{k-1}{2}$ will be considered identical under the Jaccard metric (see Figure 1.1).

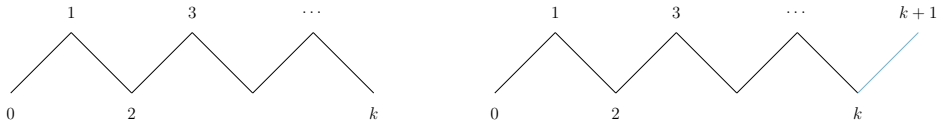


Figure 1.1: Carbon chains P_k and P_{k+1}

The ECFP algorithm is sensitive to local changes, but not to global ones. Replacing a certain substructure with another that was not previously present significantly affects the result returned by the algorithm, whereas swapping substructures within a compound does not. For instance, the structure of a carbon chain with substituents can be thought of as a tree, where the substituents are subtrees. If we swap two subtrees, the result is non-isomorphic to the original tree, but for small r the encoding will not change, meaning that in the Jaccard metric the trees will be identical. As an example, the trees shown in Figure 1.2 for $k = 10$ and $l = 7$ have a similarity of 1.0 when the fingerprints are generated using the ECFP algorithm with parameters $r = 2$ and $N \geq 2048$.

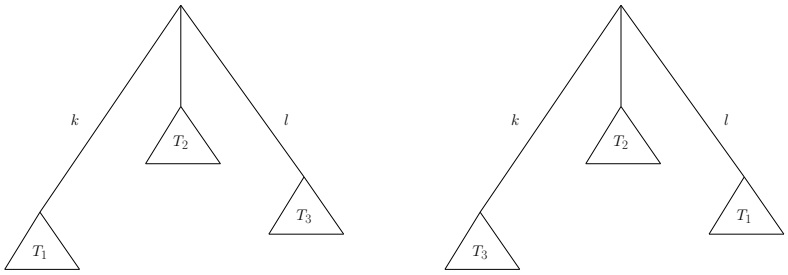


Figure 1.2: Non-isomorphic trees corresponding to hydrocarbons with swapped substituents

Overall, the choice of the parameters r and N affects both the density of fingerprints and the results of the similarity analysis of encoded structures.

1.3 Experimental properties of the ECFP encoding

Determining the expected number of generated identifiers or the number of collisions in the ECFP encoding for given parameters r and N requires making some assumptions about the encoded molecular structure, at least about the number of vertices and edges in the corresponding graph. Presumably, it would be possible to estimate these values if knowing an algorithm for sampling random compounds. One method that might be used for this uses context-sensitive grammar

to define the valid bonds between specific atoms and substructures [22]. Then, the result of a random walk on the production graph determines the correct structure. To ensure the validity of the conclusions, it would be needed to prove that, for a specific CSG, the described method generates compounds from a near-real distribution in terms of their size and the frequency of occurrence of specific substructures. It may thus be sufficient to experimentally determine the mean value of the investigated parameters using a large set of real-world compounds.

The dataset consisted of two subsets. The first subset consisted of 2.4 million compounds from the ChEMBL database classified as bioactive. It contained structures of different sizes with the largest made out of 681 atoms and an average of 31 atoms. The second subset, containing only hydrocarbons, was generated using the Wright, Richmond, Odlyzko, and McKay algorithm for enumeration of all non-isomorphic unrooted trees for a fixed number of nodes. The exact implementation was the one provided by Aric Hagberg. The hydrocarbon dataset was chosen to examine the properties of the ECFP encoding within the family of similar structures.

The ECFP encodings of all the compounds were computed using functions available in the `rdkit.Chem.rdMolDescriptors` module with parameters $r \in \{1,\ 2,\ 3\}$, $N \in \{1024,\ 2048,\ 4096, 8192\}$ as the most commonly used values [4, 14].

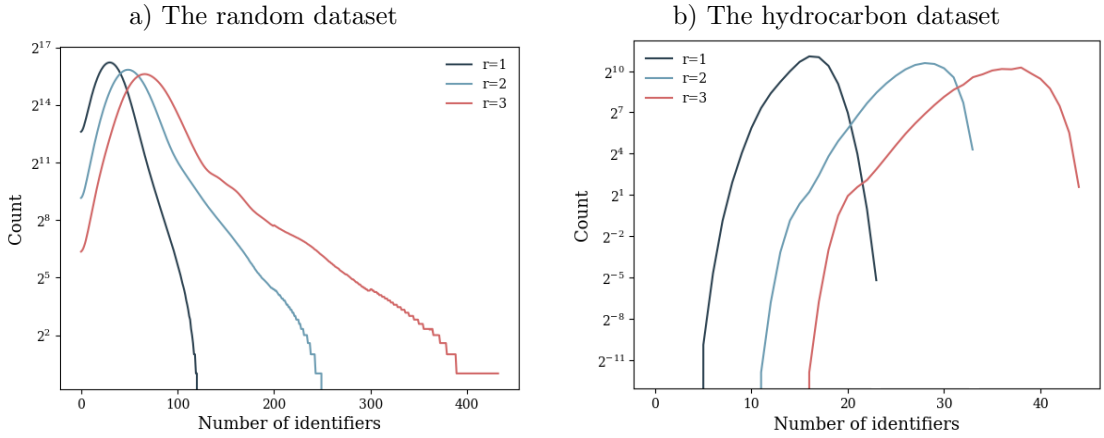


Figure 1.3: Number of identifiers generated in the ECFP procedure

The longest encoding for $r = 3$, in the random dataset, contained 432 identifiers. As expected, the encodings of the set of hydrocarbons had a distribution more concentrated around the mean, as the range of the number of identifiers was between 9 and 45. Comparing the results to the length of the encoding $N \approx 10^3$, the representation is sparse – even more so in the average case and for smaller r. Despite this, when projecting identifiers onto a binary vector of a fixed length some collisions may occur.

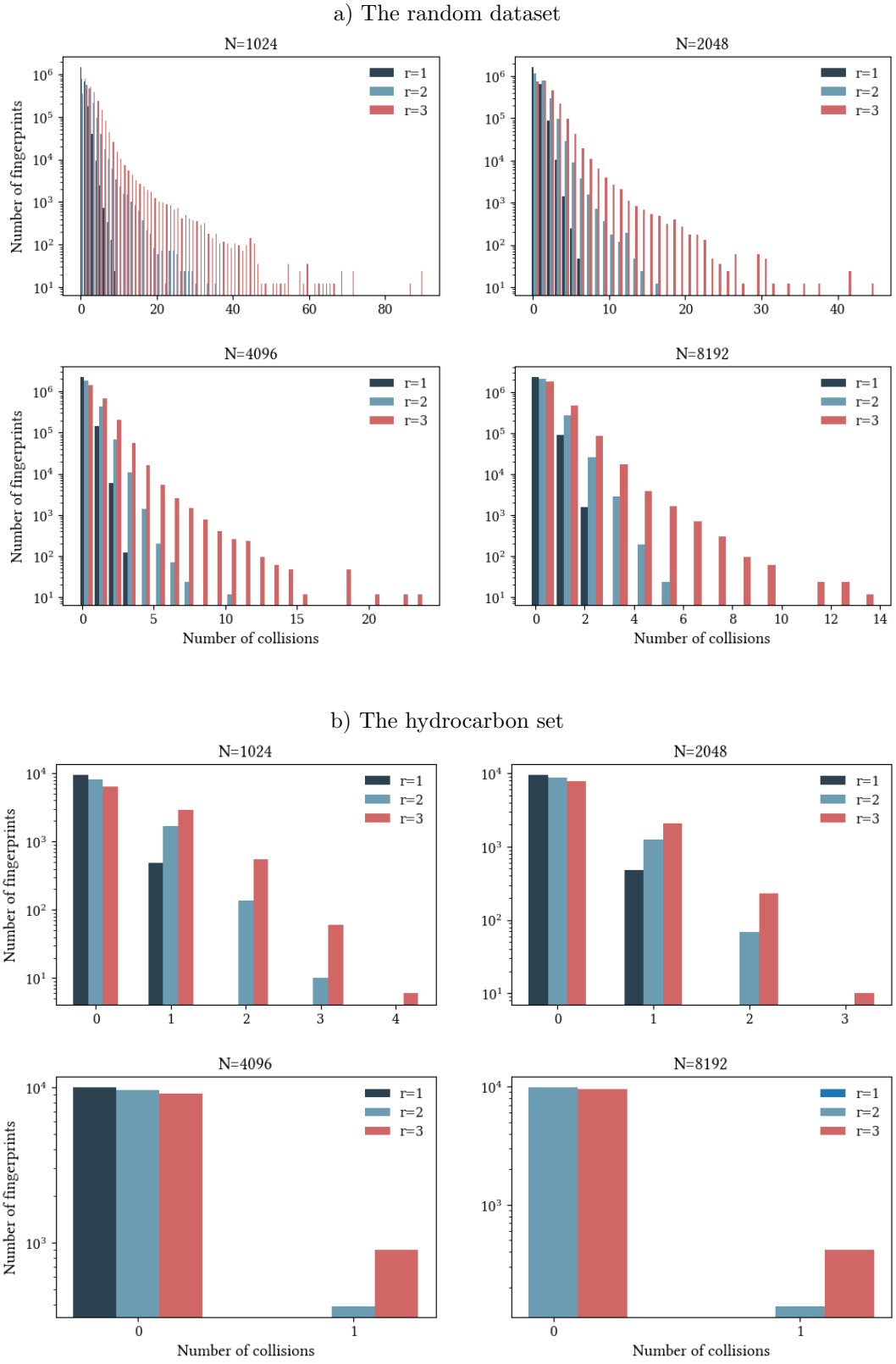


Figure 1.4: Number of collisions due to projection of identifiers on a vector depending on the parameters r, N for a) random set b) hydrocarbon family

The average number of collisions for all parameter configurations was less than 3, and for $r = 1$ or for $N \geq 4096$ it was less than 1. This raises the question of how much collisions affect the similarity coefficient.

1.3.1 Effect of collisions on the similarity coefficient

It is possible to estimate the effect of a single collision on a value of the similarity coefficient.

Proposition 1. *Given two non-zero encodings $X \neq Y$, let x_1, x_2 correspond to identifiers from X that are projected onto the same bit. The absolute error for Jaccard similarity of X and Y is at most $\frac{1}{|X \cup Y|-1}$.*

Proof. The similarity coefficient is defined as $s(X,Y) = \frac{|X \cap Y|}{|X \cup Y|}$, where $|X \cup Y| \geq 2$, thus after the reduction at least one bit is set. Let $\tilde{s}(X,Y)$ denote the Jaccard similarity coefficient of the encodings of X and Y with a single collision of the bits $x_1,\ x_2$. There are three cases to consider:

- $x_1, x_2 \notin X \cap Y$:
  The value of $|X \cap Y|$ does not change because x_1, x_2 remain outside the intersection and the value of $|X \cup Y|$ decreases by 1.

  $$|s(X,Y) - \tilde{s}(X,Y)| = \left| \frac{|X \cap Y|}{|X \cup Y|} - \frac{|X \cap Y|}{|X \cup Y| - 1} \right| = \frac{|X \cap Y|}{|X \cup Y|(|X \cup Y| - 1)} \leq \frac{1}{|X \cup Y| - 1}$$

  This gives an absolute error of $\frac{1}{|X \cup Y|-1}$.

- $x_1 \in X \cap Y,\ x_2 \notin X \cap Y$:
  In this case, the value of $|X \cap Y|$ also doesn't change because initially only x_1 contribute to $|X \cap Y|$. The value of $|X \cup Y|$ again decreases by 1. The result is therefore the same as for the previous case.

- $x_1, x_2 \in X \cap Y$:
  Both $|X \cap Y|$ and $|X \cup Y|$ decrease.

  $$|s(X,Y) - \tilde{s}(X,Y)| = \left| \frac{|X \cap Y|}{|X \cup Y|} - \frac{|X \cap Y| - 1}{|X \cup Y| - 1} \right| = \frac{|X \cup Y| - |X \cap Y|}{|X \cup Y|(|X \cup Y| - 1)} < \frac{1}{|X \cup Y| - 1}$$

  This gives an absolute error of at most $\frac{1}{|X \cup Y|-1}$.

□

For single collisions, their effect on the value of the Jaccard similarity coefficient is rather small, although they may strongly influence the results of searching for structures at a fixed similarity cut-off. Minimising the inaccuracy by choosing a sufficiently large N increases the sparsity of the representation, the problem to be solved.

Chapter 2

Numerical representation

2.1 Variable-length representation

2.1.1 Data structure for representing fingerprints

The straightforward solution to the problem of sparsity is to store all generated identifiers in a structure of variable size. No information is lost, and the size of the structure is at most $(r+1)$ times the number of vertices, so in the average case it is smaller than the standard fingerprint length. This makes such a representation implementable in real-life cases.
The designed structure should allow:

- iteration over identifiers,
- checking whether the encoding contains a given identifier,
- fast calculation of the Jaccard similarity coefficient (i. e. the size of intersection and union of the sets of identifiers) for two structures of this type.

The main solution commonly used is to store an array of sorted identifiers [4]. Any specified value can be retrieved in time $\mathcal{O}(\log n)$ using binary search.

2.1.2 Time complexity of calculating similarity coefficient

The union and intersection for two sets of size n_1, n_2 stored as sorted arrays can be determined at time $\mathcal{O}(n_1 + n_2)$ using the two-pointer method. As the calculation of the similarity coefficient involves only the size of the union and the intersection, its value can be determined faster. For this purpose, an exponential search can be used combined with a binary search.

Algorithm 2 Computing Jaccard similarity

Input: $X \in \mathbb{R}^{n_1},\ Y \in \mathbb{R}^{n_2}$ sorted

Output: $s(X, Y)$

```
common ← 0, index ← 0
for x ∈ X do
    l ← index, r ← l + 1
    while r < n_2 and Y[r] < x do
        l ← r, r ← 2 · r
    end while
    new_index ← binSearch(Y, x, l, r)
    if new_index ≠ −1 then
        common ← common + 1, index ← new_index + 1
    else
        index ← r
    end if
end for
s ← common / (n_1 + n_2 − common)
return s
```

Proposition 2. *Given two ECFP fingerprints X, Y stored as described above, such that X, Y have sizes respectively n_1, n_2, where $n_1 \leq n_2$, then $s(X, Y)$ can be calculated in $\mathcal{O}(n_1 + n_1 \log \frac{n_2}{n_1})$ time complexity.*

Proof. Iterating over the values of X, we try to match elements in Y. We perform a binary search to find the index in Y to which we should jump. In total, this requires $\mathcal{O}(\sum_{i=1}^{n_1} \log x_i)$ comparisons, where x_i is the length of the i-th jump and $x_1 + \cdots + x_{n_1} = n_2$. From the inequality between the arithmetic and geometric mean, it follows that

$$\sum_{i=1}^{n_1} \log x_i = \log \left(\prod_{i=1}^{n_1} x_i \right) \leq \log \left(\left\lceil \frac{n_2}{n_1} \right\rceil^{n_1} \right) = n_1 \log \left\lceil \frac{n_2}{n_1} \right\rceil ,$$

which gives time complexity $\mathcal{O}\Big(n_1 + n_1 \log \frac{n_2}{n_1}\Big)$. □

Potential improvements to this complexity involve better use of memory or parallelization of calculations. In practice, however, the size of the arrays is at most the order of 10^3, so further optimisations are not needed.

2.2 Fixed-length representation

Fixed-length representations enable efficient storage of multidimensional numerical data. This property can simplify computations and is required for the input of certain classes of machine-learning models. However, reducing the dimensionality of the encodings inevitably causes some information loss, in particular a dictionary-like indication of the presence of a specific substructure. It may still be possible to retain enough information to preserve pairwise similarity between objects. Specifically for the case of ECFP encodings of a set of chemical compounds, the potential

solutions are the Johnson-Lindenstrauss and MinHash methods. They let project numerical data into a lower-dimensional space while preserving pairwise distances with high probability.

2.2.1 Johnson-Lindenstrauss reduction

Johnson-Lindenstrauss method is used to project a numerical or binary vectors of a fixed length into a smaller dimension while preserving the pairwise Euclidean distance. The position of points in space can thus reflect the similarity between them. The correctness of the method is based on the following lemma.

Lemma 3 (Johnson–Lindenstrauss, 1984 [9, 24])**.** *For a fixed $0 < \varepsilon < 1$ and a set of m points $\mathcal{X} \subset \mathbb{R}^N$ there exists linear mapping $f\colon \mathbb{R}^N \to \mathbb{R}^n$, where $n = \Omega(\log(m)\varepsilon^{-2})$, satisfying*

$$(1-\varepsilon)\left\|u-v\right\|^2 \leq \left\|f(u)-f(v)\right\|^2 \leq (1+\varepsilon)\left\|u-v\right\|^2$$

for all $u,\ v \in \mathcal{X}$.

Proof. Let $A \in \mathbb{R}^{n\times N}$ be a matrix such that $a_{ij} \sim \mathcal{N}(0,1)$. Defining a projection $\Pi = \frac{1}{\sqrt{n}}A$, we prove that Πx approximates the norm of x.

$$\left\|\Pi x\right\|^2 = \left\|\frac{1}{\sqrt{n}}Ax\right\|^2 = \frac{1}{n}\sum_{i=1}^{n}(a_i x)^2$$

The values a_{ij} are sampled at random from the distribution $\mathcal{N}(0,1)$, so $x_j a_{ij}$ is also a random variable.

$$\mathbb{E}\left[a_i x\right] = \mathbb{E}\left[\sum_{j=1}^{N} x_j a_{ij}\right] = \sum_{j=1}^{N} x_j \cdot \mathbb{E}\left[a_{ij}\right] = 0$$

$$\mathbb{E}\left[(a_i x)^2\right] = \operatorname{Var}\left[a_i x\right] = \sum_{j=1}^{N}\operatorname{Var}\left[a_{ij}x_j\right] = \sum_{j=1}^{N} x_j^2 \operatorname{Var}\left[a_{ij}\right] = \sum_{j=1}^{N} x_j^2 = \left\|x\right\|^2.$$

Hence, the expected norm of x after the projection is equal to the initial norm:

$$\mathbb{E}\left[\ \left\|\Pi x\right\|^2\ \right] = \mathbb{E}\left[\frac{1}{n}\sum_{i=1}^{n}(a_i x)^2\right] = \left\|x\right\|^2$$

The random variable $a_i x \sim \mathcal{N}(0, \left\|x\right\|^2) = \left\|x\right\| \cdot \mathcal{N}(0,1)$ is concentrated around the mean, so the approximation does not deviate significantly from the expected value. For $y = \left\|\Pi x\right\|$, by applying the Chernoff inequality we obtain:

$$\Pr\left[|\mathbb{E}y - y| \geq \varepsilon \cdot \mathbb{E}y\right] \leq 2e^{-n\varepsilon^2/8}$$

For $n = \Omega(\log(1/\delta)\varepsilon^{-2})$, the inequality:

$$(1-\varepsilon)\left\|x\right\|^2 \leq \left\|\Pi x\right\|^2 \leq (1+\varepsilon)\left\|x\right\|^2$$

is satisfied with probability at least $(1-\delta)$.
Substituting x with $u-v$, we obtain the proof that the Johnson-Lindenstrauss reduction almost

preserves Euclidean distance between any pair of vectors from $\mathcal{X}$. Since there are $\binom{m}{2}$ pairs of points, from the linearity of expectation, all distances are preserved within error ε with probability at least $(1-\delta)$ if we assume $n = \Omega(\log(m/\delta)\varepsilon^{-2})$. □

The proof is constructive and the defined projection can be easily implemented according to the following steps:

Algorithm 3 Johnson-Lindenstrauss reduction

Input: $n \in \mathbb{N}$, $x \in \mathbb{R}^N$
Output: $x' \in \mathbb{R}^n$

```
Initialize A ∈ R^{n×N}
for i ∈ {0, ..., n − 1} do
    for j ∈ {0, ..., N − 1} do
        A[i][j] ← N(0, 1)
    end for
end for
Π ← (1/√n) A
return Πx
```

The transformation can be performed in time $\mathcal{O}(m \cdot Nn) = \mathcal{O}(m \cdot N \log(m/\delta)\varepsilon^{-2})$ for a set of m vectors, that is, polynomial in terms of size of $\mathcal{X}$ and the initial length of encoding, which in this case is a constant of order 10^3. Hence, for large sets of compounds it holds that $m \gg N$. The time complexity of multiplication of matrix A with a vector u can be improved to $\Theta(k \cdot \|u\|_0) = \mathcal{O}(n \cdot N)$, where k denotes the maximum support of columns of matrix A [6]. Different variants of the transformation f have been developed that satisfy the Johnson-Lindenstrauss lemma and improve time complexity. For example, the Fast Johnson-Lindenstrauss Transform, which uses a sparse projection matrix, achieves $\mathcal{O}(N \log N)$ time complexity in the general case, while Ailon and Liberty reduced it to $\mathcal{O}(N \log n)$ when $n = \mathcal{O}(N^{1/2-\gamma})$ for a fixed constant $\gamma > 0$ [6].

It remains to show that the Johnson-Lindenstrauss lemma can be generalised to another distance metric.

Proposition 4. *The Johnson-Lindenstrauss lemma is true for Jaccard similarity metric.*

Proof. The projection Π does not change the expected value of the scalar product, because matrix $A^T A$ is usually close to the unit matrix.

$$\mathbb{E}\left[(A^T A)_{ij}\right] = \sum_{k=1}^{n} \mathbb{E}\left[a_{ki} \cdot a_{kj}\right] = \sum_{k=1}^{n} \mathbb{1}[i=j] = n \cdot \mathbb{1}[i=j]$$

$$\mathbb{E}\left[\langle \Pi u, \Pi v \rangle\right] = \mathbb{E}\left[\left\langle \frac{1}{\sqrt{n}} Au, \frac{1}{\sqrt{n}} Av \right\rangle\right] = \frac{1}{n}\mathbb{E}\left[u^T A^T A v\right] = \frac{1}{n} u^T \cdot \mathbb{E}\left[A^T A\right] \cdot v = u^T v,$$

The inequality

$$(1-\varepsilon)\left\|u-v\right\|^2 \leq \left\|f(u)-f(v)\right\|^2 \leq (1+\varepsilon)\left\|u-v\right\|^2,$$

follows from the Lemma 3. Now, it is needed to prove that the inequality also holds for the scalar product. The useful identities are:

Parallelogram law:

$$\|x+y\|^2 + \|x-y\|^2 = 2\|x\|^2 + 2\|y\|^2$$

Polarisation identity:

$$\langle x, y\rangle = \frac{1}{4}\left(\|x+y\|^2 - \|x-y\|^2\right)$$

It follows that:

$$\langle f(u), f(v)\rangle = \frac{1}{4}\left(\|f(u)+f(v)\|^2 - \|f(u)-f(v)\|^2\right) \geq \frac{1}{4}\left((1-\varepsilon)\|u+v\|^2 - (1+\varepsilon)\|u-v\|^2\right)$$

$$= \frac{1}{4}\left(\|u+v\|^2 - \|u-v\|^2\right) - \frac{1}{4}\varepsilon\left(\|u+v\|^2 + \|u-v\|^2\right) = \langle u, v\rangle - \frac{1}{2}\varepsilon\left(\|u\|^2 + \|v\|^2\right)$$

To complete the proof, an additional observation is needed:

Lemma 5. *The Jaccard similarity is not affected by positive scalar multiplication.*

Proof. Let $c \in \mathbb{R}^+$, then:

$$s(cu, cv) = \frac{\langle cu, cv\rangle}{\|cu\|^2 + \|cv\|^2 - \langle cu, cv\rangle} = \frac{c^2\langle u, v\rangle}{c^2\left(\|u\|^2 + \|v\|^2\right) - c^2\langle u, v\rangle} = \frac{\langle u, v\rangle}{\|u\|^2 + \|v\|^2 - \langle u, v\rangle} = s(u, v)$$

□

We can now assume that the binary fingerprints of length N have norm at most 1, since they can be multiplied by $\frac{1}{\sqrt{N}}$, and then the dot product is bounded by:

$$\langle f(u), f(v)\rangle \geq \langle u, v\rangle - \frac{1}{2}\varepsilon\left(\|u\|^2 + \|v\|^2\right) \geq \langle u, v\rangle - \varepsilon$$

This proves the inequality:

$$(1-\varepsilon)\langle u, v\rangle \leq \langle f(u), f(v)\rangle \leq (1+\varepsilon)\langle u, v\rangle$$

The formula for Jaccard similarity is:

$$s(f(u), f(v)) = \frac{\langle f(u), f(v)\rangle}{\|f(u)\| + \|f(v)\| - \langle f(u), f(v)\rangle} = \frac{\langle f(u), f(v)\rangle}{\|f(u) - f(v)\| + \langle f(u), f(v)\rangle},$$

which can be derived using the parallelogram law and the polarisation identity:

$$\|f(u)\| + \|f(v)\| - \langle f(u), f(v)\rangle = \frac{1}{2}\left(\|f(u)+f(v)\|^2 + \|f(u)-f(v)\|^2\right) - \langle f(u), f(v)\rangle$$

$$= \frac{1}{2}\left(4\langle f(u), f(v)\rangle + 2\|f(u)-f(v)\|^2\right) - \langle f(u), f(v)\rangle = \langle f(u), f(v)\rangle + \|f(u)-f(v)\|^2$$

Plugging the above results for $0 < \varepsilon \leq \frac{1}{2}$ into the formula for Jaccard similarity gives:

$$s(f(u), f(v)) \leq \frac{\langle u, v\rangle \cdot (1+\varepsilon)}{\left(\|u-v\| + \langle u, v\rangle\right) \cdot (1-\varepsilon)} = s(u, v) \cdot \left(1 + \frac{2\varepsilon}{1-\varepsilon}\right) \leq s(u, v) \cdot (1+4\varepsilon)$$

$$s(f(u), f(v)) \geq \frac{\langle u, v\rangle \cdot (1-\varepsilon)}{\Big(\big\|u-v\big\| + \langle u, v\rangle\Big) \cdot (1+\varepsilon)} = s(u,v) \cdot \left(1 - \frac{2\varepsilon}{1+\varepsilon}\right) \leq s(u,v) \cdot (1-2\varepsilon)$$

It shows that for $n \in \Omega(\log(m/\delta)\varepsilon^{-2})$ the Jaccard similarity satisfies

$$(1-\varepsilon) \cdot s(u,v) \leq s(f(u), f(v)) \leq (1+\varepsilon) \cdot s(u,v)$$

□

2.2.2 MinHash reduction

Unlike the previous method, MinHash can be applied to both fixed and variable length vectors. The input vectors can be numerical or binary, but the latter requires transforming them into a set of non-zero indices. The advantage of the MinHash method is that it directly preserves the Jaccard similarity. Thus, the MinHash method is well suited to convert the list of identifiers returned by the ECFP algorithm directly into a fixed-length encoding without previously projecting it into a vector.

To present the MinHash reduction, first, a few definitions are needed [2, 18]. For a fixed $k \in \mathbb{N}$ and a set $X \subseteq \Omega$ with given ordering corresponding to a random permutation of Ω, define:

$$\min_k(X) = \{x_i \in X : 1 \leq i \leq \min\{k, |X|\}\}$$

Then, the similarity of sets $X, Y \subseteq \Omega$ can be expressed as:

$$s_k(X,Y) = \frac{|\min_k(X \cup Y) \cap \min_k(X) \cap \min_k(Y)|}{|\min_k(X \cup Y)|}$$

With the above definitions in hand, it is possible to prove the lemma that is fundamental for the MinHash method.

Lemma 6 ([2])**.** *The value of $s_k(X,Y)$ approximates the Jaccard similarity $s(X,Y)$.*

Proof. Since the sets X and Y are ordered according to a random permutation, each element of the sets has an equal probability of being the i-th smallest. For a certain $x \in \min_k(X \cup Y)$:

$$P(x \in \min_k(X) \cap \min_k(Y)) = P(x \in X \cap Y) = \frac{|X \cap Y|}{|X \cup Y|} = s(X,Y)$$

□

By setting $k = 1$, the above similarity estimator can be applied to the sets of ECFP identifiers, to derive an implementable algorithm.

Algorithm 4 MinHash reduction

Input: $n \in \mathbb{N},\ X \in \mathbb{R}^N$

Output: $\tilde{X} \in \mathbb{R}^n$

1 define uniform hashing functions $h_1, \ldots, h_n : \mathbb{R} \to [0,1]$

2 $f(X)_i \leftarrow \min_{x \in X}\ h_i(x)$

3 **return** $[\, f(X)_1, \ldots, f(X)_n \,]$

The algorithm has time complexity $\mathcal{O}(Nn)$. It can be proven to preserve the Jaccard similarity with high probability.

Proposition 7 ([2, 18])**.** *For $0 < \varepsilon < 1$ and $n = \mathcal{O}(\log(m/\delta)\varepsilon^{-2})$, the Jaccard similarity of X, Y transformed using the MinHash method $s(f(X), f(Y)) = \frac{1}{n}\sum_{i=1}^{n} \mathbb{1}[f(X)_i = f(Y)_i]$ satisfies*

$$s(X,Y) - \varepsilon \leq s(f(X), f(Y)) \leq s(X,Y) + \varepsilon$$

for any $X \in \mathbb{R}^{N_1}$, $Y \in \mathbb{R}^{N_2}$.

Proof. The estimator $s(f(X), f(Y))$ is unbiased, because $\mathbb{E}\,[\,\mathbb{1}[f(X)_i = f(Y)_i]\,] = s(X,Y)$. The inaccuracy of approximation can be bounded by applying Hoeffding inequality:

$$P\left(|s(f(X), f(Y)) - s(X,Y)| \geq \varepsilon\right) \leq 2e^{-2n\varepsilon^2}$$

It simply follows that for some X and Y, and $n = \mathcal{O}(\log(1/\delta)\varepsilon^{-2})$ with probability at least $(1-\delta)$ it is true that

$$s(X,Y) - \varepsilon \leq s(f(X), f(Y)) \leq s(X,Y) + \varepsilon$$

Therefore, from the linearity of expectation, the similarity is preserved for all possible pairs of vectors when $n = \Omega(\log(m/\delta)\varepsilon^{-2})$. □

Chapter 3

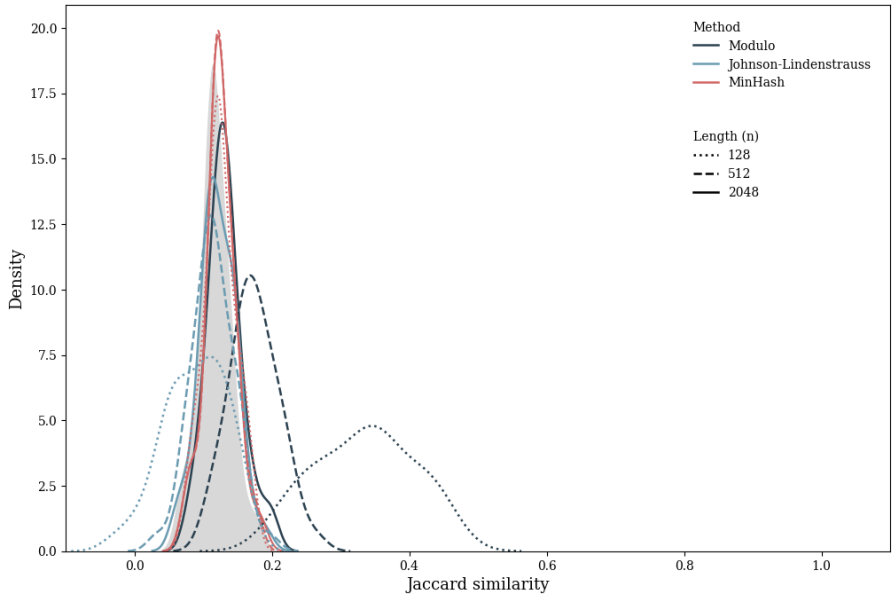
Applications

3.1 Effect of dimension reduction on pairwise similarity

The Johnson-Lindenstrauss and MinHash methods have been implemented (see Appendix) and tested on a set of 1000 compounds randomly selected from the original datasets using the bash terminal `shuf` command. The parameters used for the analysis were $r = 3$ and $N \in \{128, 512, 2048\}$, so $N \sim \log(m)\epsilon^{-2}$ for $m = 1000$ and $0.05 \leq \epsilon \leq 0.25$.

The goal of the experiment was to compare the approaches to dimensionality reduction of the encodings, by evaluating how well each method preserved the pairwise similarity between the compounds. The Johnson-Lindenstrauss and MinHash methods were compared to manipulating the length of the fingerprints modulo the value of N. The Figure 3.1 shows how the distribution of distances in the Jaccard metric has changed for the studied methods.

a) The random dataset



b) The hydrocarbon dataset

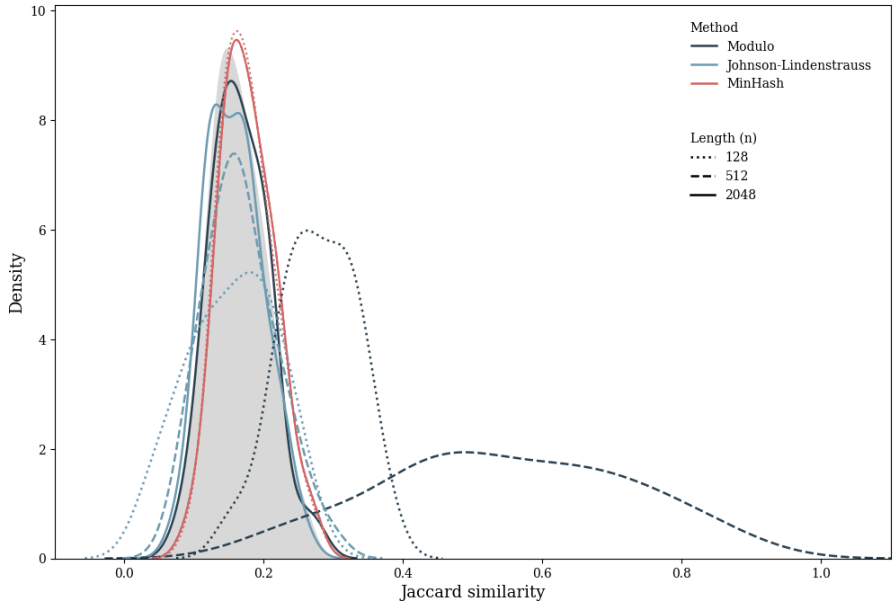


Figure 3.1: Distribution of distance in Jaccard metric for the encodings reduced with different methods

For the original non-reduced fingerprints from the random dataset, Jaccard distance ranged from 0.871 to 0.985 with a mean of 0.933. For $N = 8192$, the distributions of distances for reduced encodings did not differ much from the original distributions. For smaller values of N, however, the Johnson-Lindenstrauss and MinHash methods outperformed the naive method. Reducing the fingerprint length modulo N lead to visibly broadened distribution. For $N = 128$, the mean distance was equal to 0.715 with the most similar pair of compounds having distance of 0.587. For both other methods, the absolute difference in pairwise distance compared to the original one in no case exceeded 0.10 for the Johnson-Lindenstrauss and 0.03 for the MinHash reduction.

3.2 Effect of dimension reduction on KNN-search accuracy

Cluster analysis and similarity search play a crucial role in identifying patterns in molecular structures and predicting their properties. Consequently, they are among the most commonly used methods of *in silico* studies exploring the structure-activity relationship of chemical compounds. In such context, under the assumption that the similarity between compounds is proportional to the similarity of their numerical representation, the similarity search can be effectively performed as a KNN-search in the feature space [23]. This process is computationally expensive, particularly when operating on high-dimensional representations such as molecular fingerprints. The nearest neighbour similarity search, therefore, serves as an apt validation test for the practical applicability of dimensionality reduction techniques like Johnson-Lindenstrauss and MinHash, which aim to reduce computational costs without sacrificing accuracy.

The goal is to reduce the likelihood of a false negative, i.e. rejecting a potential neighbour that would be selected based on the original, non-reduced encoding. An error of this kind is more costly than a false positive because it excludes from further study a compound that could

potentially be a good candidate for the desired active ingredient. The assumption is that the nearest-neighbour search method itself does not use an approximation algorithm and always returns the correct result. In addition, in the context of this problem, instead of similarity, the Jaccard distance (see Notation) will be used, which is directly related to the similarity and follows the convention used for the KNN search. This analysis refers to the MinHash reduction, which has proven to be the best experimentally and has the desired theoretical properties.

For the analysis, the Jaccard distance distribution for the random set and hydrocarbons can be modelled with a normal distribution.

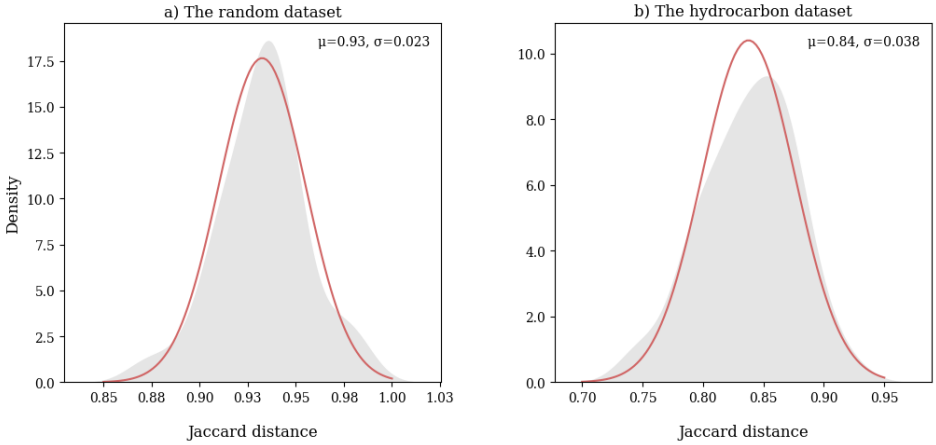


Figure 3.2: Fit of normal distribution to distribution of Jaccard distance

The problem has two variants: searching for exactly k nearest neighbours or searching for all neighbours with a distance of at most k from the input structure. Let E be an event that a vector v is included in the results of the search based on the original encoding, and let $\tilde{E}$ denote the event that $\tilde{v} = f(v)$ is included in the results of the search based on the reduced encoding. Let u be the encoding of a given input structure. Suppose we search in a set of m structures with N being the maximum number of identifiers generated by the ECFP algorithm. The reduced encoding has length $n = \mathcal{O}(\log(m)\varepsilon^{-2})$, as in the Lemma 7.

3.2.1 Search with fixed distance threshold

We are given the initial structure u and the distance threshold $0 < k < 1$. The probability of observing a false negative is:

$$P(\neg\tilde{E} \mid E) = P(d(\tilde{u}, \tilde{v}) > k \mid d(u, v) \leq k)$$

Since the size of the set of ECFP identifiers for each structure is limited linearly from the size of the structure, the distances between structures take on values from a specific discrete range $D \subseteq \left\{ \frac{|u \cap v|}{|u \cup v|} : 0 \leq |u \cap v| \leq N,\ 0 < |u \cup v| \leq 2N \right\}$. Hence, the above probability can be bounded by iterating over all possible distances between u and v that are not greater than k:

$$P(\neg\tilde{E} \mid E) = \sum_{d_i \in D,\ d_i \leq k} P(d(\tilde{u}, \tilde{v}) > d(u, v) + (k - d_i) \mid d(u, v) = d_i) \cdot P(d(u, v) = d_i)$$

For the MinHash method, we can bound the probability using the Proposition 7 and the fact that the distribution of Jaccard distance can be approximated using a normal distribution $\mathcal{N}(\mu, \sigma^2)$ fitted for the used datasets, to obtain:

$$P(\neg\tilde{E} \mid E) \leq \sum_{d_i \in D} 2e^{-n(k-d_i)^2} \cdot \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(d_i-\mu)^2}{2\sigma^2}} = \sum_{d_i \in D} 2e^{-n(k-d_i)^2} \cdot \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(d_i-\mu)^2}{2\sigma^2}}$$

$$= \sqrt{\frac{2}{\pi\sigma^2}} \cdot \sum_{d_i \in D} e^{-n(k-d_i)^2 - \frac{(d_i-\mu)^2}{2\sigma^2}}$$

The results obtained experimentally and averaged over 1000 trials are presented in Table 3.1.

(a) The random dataset

k	$n = 128$	$n = 512$	$n = 2048$
0.5	0.002	< 0.001	< 0.001
0.6	0.010	0.002	< 0.001
0.7	0.033	0.001	< 0.001
0.8	0.177	0.008	< 0.001
0.9	0.203	0.008	< 0.001

(b) The hydrocarbon dataset

k	$n = 128$	$n = 512$	$n = 2048$
0.5	0.008	< 0.001	< 0.001
0.6	0.057	< 0.001	< 0.001
0.7	0.069	< 0.001	< 0.001
0.8	0.034	< 0.001	< 0.001
0.9	0.001	< 0.001	< 0.001

Table 3.1: Experimental false positive rate in KNN depending on the distance threshold k

3.2.2 Search with fixed number of results

We are given the initial structure u and the searched number of nearest neighbours k. Let E_j represent the probability that v is the j-th nearest neighbour of u. The probability of observing a false negative is:

$$P(\neg\tilde{E} \mid E) = \frac{P(\neg\tilde{E} \wedge E)}{P(E)} = \frac{1}{P(E)} \sum_{j \in [k]} P(\neg\tilde{E} \wedge E_j)$$

This can be approximated by the probability that there are at least $k - j$ vectors that are further from u than v when non-reduced but closer after the transformation. For a vector v_i satisfying the above condition, let d_v, d_{v_i} be respectively the distances of v and v_i from u. Let d_{uv} denote the distance by which v moves relative to u, that is $d_{uv} = |d(u, v) - d(\tilde{u}, \tilde{v})|$. The following scenarios are possible:

- If $d(u, v) \leq d(\tilde{u}, \tilde{v})$ and $d(u, v_i) \leq d(\tilde{u}, \tilde{v_i})$, the distance of v_i from u must decrease by at least $d_{v_i} - d_v + d_{uv}$:

$$P(|d(u, v_i) - d(\tilde{u}, \tilde{v_i})| \geq d_{v_i} - d_v + d_{uv}) \leq 2e^{-n(d_{v_i} - d_v + d_{uv})^2}$$

- If $d(u, v) > d(\tilde{u}, \tilde{v})$ and $d(u, v_i) \geq d(\tilde{u}, \tilde{v_i})$, the distance of v_i from u must decrease by at least $d_{v_i} - d_v - d_{uv}$:

$$P(|d(u, v_i) - d(\tilde{u}, \tilde{v_i})| \geq d_{v_i} - d_v - d_{uv}) \leq 2e^{-n(d_{v_i} - d_v - d_{uv})^2}$$

- If $d(u,v) > d(\tilde{u},\tilde{v})$ and $d(u,v_i) > d(\tilde{u},\tilde{v_i})$, the distance of v_i from u must decrease by at most $d_{v_i} - d_v + d_{uv}$:

$$P(|d(u,v_i) - d(\tilde{u},\tilde{v_i})|) \leq d_{v_i} - d_v + d_{uv}) \leq 1 - 2e^{-n(d_{v_i}-d_v+d_{uv})^2}$$

Combining the above, and given d_v, d_{v_i} and d_{uv}, the probability that the transformation introduced an inversion between v and v_i with respect to the distance to u can be bounded by:

$$2e^{-n(d_{v_i}-d_v+d_{uv})^2} + 2e^{-n(d_{v_i}-d_v-d_{uv})^2} + 1 - 2e^{-n(d_{v_i}-d_v+d_{uv})^2} = 2e^{-n(d_{v_i}-d_v-d_{uv})^2} + 1$$

Summing over the possible values of d_v, d_{v_i} and d_{uv}, we obtain the bound:

$$P(d(u,v) < d(u,v_i) \wedge d(\tilde{u},\tilde{v}) > d(\tilde{u},\tilde{v_i})) \leq$$

$$\leq \sum_{d_v \leq d_{v_i} \in D} \left(P(d(u,v) = d_v,\ d(u,v_i) = d_{v_i}) \cdot \sum_{d_v \in D} P(|d(u,v) - d(\tilde{u},\tilde{v})| = d_{uv}) \cdot \left(2e^{-n(d_{v_i}-d_v-d_{uv})^2} + 1\right)\right)$$

$$\leq \sum_{d_v \leq d_{v_i} \in D} \left(\frac{1}{2\pi\sigma^2} e^{-\frac{(d_v-\mu)^2+(d_{v_i}-\mu)^2}{2\sigma^2}} \cdot \sum_{d_{uv} \in D} \left(2e^{-nd_{uv}^2} \cdot \left(2e^{-n(d_{v_i}-d_v-d_{uv})^2} + 1\right)\right)\right)$$

$$= \frac{1}{\pi\sigma^2} \cdot \sum_{d_v \leq d_{v_i} \in D} \left(e^{-\frac{(d_v-\mu)^2+(d_{v_i}-\mu)^2}{2\sigma^2}} \cdot \sum_{d_{uv} \in D} e^{-nd_{uv}^2} \cdot \left(2e^{-n(d_{v_i}-d_v-d_{uv})^2} + 1\right)\right)$$

To simplify the inner sum, the product can be split into two series of the form e^{-x^2}, and then approximated by a Gaussian integral. Let $c = d_{v_i} - d_v$ for clarity.

$$\sum_{d_{uv} \in D} e^{-nd_{uv}^2} \cdot \left(2e^{-n(c-d_{uv})^2} + 1\right) = \sum_{d_{uv} \in D} e^{-n(c-d_{uv})^2 + d_{uv}^2} + \sum_{d_{uv} \in D} e^{-nd_{uv}^2}$$

The second sum can be directly approximated by the integral, since for sufficiently large N the possible values of d_{uv} are arranged densely.

$$\sum_{d_{uv} \in D} e^{-nd_{uv}^2} \approx \int_D e^{-nx^2} dx \leq \int_0^{+\infty} e^{-nx^2} dx = \frac{1}{2}\sqrt{\frac{\pi}{n}}$$

The first sum requires simplification, before a similar approximation can be applied. Because $c \in [0,1]$, it follows that:

$$\sum_{d_{uv} \in D} e^{-nc^2} \cdot e^{2n\left(d_{uv}c - d_{uv}^2\right)} = \sum_{d_{uv} \in D} e^{-nc^2} \cdot e^{n\frac{c^2}{2}} \cdot e^{-2n\left(d_{uv}-\frac{c}{2}\right)^2} = e^{-n\frac{c^2}{2}} \cdot \sum_{d_{uv} \in D} e^{-2n\left(d_{uv}-\frac{c}{2}\right)^2}$$

$$\approx e^{-n\frac{c^2}{2}} \cdot \int_D e^{-n\left(x-\frac{c}{2}\right)^2} dx \ \leq\ e^{-n\frac{c^2}{2}} \cdot \int_0^{+\infty} e^{-n\left(x-\frac{c}{2}\right)^2} dx \ \leq\ \frac{1}{2} e^{-n\frac{c^2}{2}} \cdot \sqrt{\frac{\pi}{n}}$$

Applying the above derivation, the consequent result is:

$$P(\neg\tilde{E} \wedge E_j) \leq \binom{m-j}{k-j} \left(\frac{1}{\pi\sigma^2} \cdot \sum_{d_v \leq d_{v_i} \in D} \left(e^{-\frac{(d_v-\mu)^2+(d_{v_i}-\mu)^2}{2\sigma^2}} \cdot \sqrt{\frac{\pi}{n}} \cdot e^{-n\frac{(d_{v_i}-d_v)^2}{2}} \right) \right)^{k-j}$$

$$= \binom{m-j}{k-j} \left(\frac{1}{\sqrt{n\pi\sigma^2}} \cdot \sum_{d_v \leq d_{v_i} \in D} \left(e^{-\frac{(d_v-\mu)^2+(d_{v_i}-\mu)^2}{2\sigma^2}} \cdot e^{-n\frac{(d_{v_i}-d_v)^2}{2}} \right) \right)^{k-j}$$

The results obtained experimentally over 1000 trials and different threshold values are presented in Table 3.2.

(a) The random dataset

k	$n = 128$	$n = 512$	$n = 2048$
10	0.319	0.005	0.025
20	0.311	0.046	0.024
30	0.305	0.046	0.024
40	0.297	0.044	0.023
50	0.290	0.043	0.022

(b) The hydrocarbon dataset

k	$n = 128$	$n = 512$	$n = 2048$
10	0.152	0.020	0.020
20	0.139	0.019	0.019
30	0.135	0.022	0.021
40	0.129	0.023	0.023
50	0.126	0.022	0.022

Table 3.2: Experimental false positive rate in KNN depending on the count threshold k

3.3 Effect of dimension reduction on KNN-search time

The KNN algorithm in a basic implementation has a time complexity of $\mathcal{O}(m \cdot N)$, where m is the size of the dataset, N is the number of dimensions of the data-space. Reducing the dimensionality is therefore a straightforward optimisation leading to improvement in both memory and time complexity by reducing the size of the vectors and making comparisons faster. Although in sequential KNN procedure, the complexity is only improved by a certain constant, the reduction is much more relevant for advanced search methods.

Many of the currently adopted KNN search techniques are based on constructing a multidimensional data structure to perform clustering on the data [23]. A standard approach used among others in K-D-B-tree, quadtree, R-tree or SR-tree methods is to partition the data space and to search only in the relevant classes [23]. Such approaches reduce the number of necessary comparisons and are fundamental to faster KNN queries. However, the performance of this approach deteriorates rapidly with an increasing number of dimensions, as drastically as this phenomenon occurs in the literature as a "dimensional curse" [23]. Some frameworks, such as R-tree or SR-tree, attempt to deal with this problem by rejecting the assumption of uniform data distribution in space and restricting the search to the most significant dimensions [11]. Nevertheless, as demonstrated by Weber et al., for each partition-based algorithm, there is a number of dimensions N beyond which it converges to time complexity $\mathcal{O}(N)$, thus a linear search of the collection is equally or more efficient [23]. In practice the boundary value is $N \approx 600$ [23].

When the partitions are performed on every dimension, as in the case of K-D-tree and quadtree, the number of possible partitions is of order 2^N. For $N = 1024$, as in the case of ECFP vectors, the number of classes is thus close to 10^{308}, and such structures are not applicable for large N. Moreover, in sets containing about a million vectors, the number significantly exceeds the size of the dataset, making most partition classes empty, especially for sparse representations such as ECFP. Far worse, despite a sparsely populated index structure, some classes may contain several points that need to be searched sequentially. Real datasets, such as random collections of chemical

compounds, often contain clusters of similar objects [1]. The compounds in one cluster may not even be distinguishable by ECFP fingerprints. Some dimensions in the data space may also be meaningless because, for example, the bit corresponding to -CH3 or -CH2- groups is present in the vast majority of fingerprints.

Dimensionality reduction is a versatile partial solution to the above problems. It does not solve them entirely, as sometimes the size of the resulting data space is still too big, nevertheless, it is advised for real-life cases whenever possible [1]. By transforming high-dimensional data with methods such as Johnson-Lindenstrauss or MinHash reductions, it is possible to decrease the number of dimensions from N to the order of $\log m$, thus making the number of partitions comparable to the size of the dataset. Consequently, in expectation, the data space is filled more evenly, and the index structure is less sparse. Thereby, reducing the number of dimensions may not only improve the complexity for sequential KNN-search but also enable the use of advanced index structures to perform queries in $\mathcal{O}(m \cdot \log N)$ time complexity.

3.4 Summary

The study has shown that both Johnson–Lindenstrauss and MinHash reduction methods provide an effective means of compressing ECFP binary fingerprints while preserving pairwise similarity. In addition to their solid theoretical foundations, the methods offer several practical benefits. Experiments on real-world chemical compound datasets confirmed that the false positive rate in KNN-search using encodings reduced with MinHash does not exceed 5% at a fixed accuracy threshold of 0.1. This suggests that the evaluated reduction methods may also perform well in clustering applications, for instance, in chemical space partitioning. It would also be valuable to assess how well these approaches perform in predictive modelling tasks, such as QSAR or QSPR.

A remarkable advantage of the MinHash method is the possibility to compress ECFP fingerprints with a larger radius (a higher value of parameter r) without increasing collision rates. This allows more structural information to be captured while maintaining compression efficiency and accuracy in distinguishing similar molecules.

From an implementation perspective, both the Johnson–Lindenstrauss and MinHash methods are highly parallelizable, as each coordinate of the reduced fingerprint is computed independently. Given that the original fingerprint length is on the order of 10^3 and the reduced length is 10^2, it is possible to transform a single vector using GPU in nearly constant time. However, a technical drawback is that the resulting vectors have non-integer values. In the case of Johnson-Lindenstrauss projection, this problem can be solved by sampling the projection matrix from a discrete distribution with suitable mean and variance, such as the Rademacher distribution over $\{-1, 1\}$, and for the MinHash method, the hash functions would require adaptation.

Building on these promising results, several questions remain to be explored. The main challenge lies in improving the selection of parameters to best preserve similarity information between encoded molecular structures. By studying how the size of molecules and diversity of functional-groups influence the rate of collisions of identifiers and the minimal value of parameter r required to distinguish molecules, it may be possible to optimize the encoding method and the fingerprint length. Adaptive reduction method of encodings based on an entire dataset is a particularly interesting topic that was not deeply investigated in this work but holds potential for improving practical results in applications such as KNN-search or clustering. As indicated by differences

in results for a random set of small molecules and a family of hydrocarbons, adjusting the encoding strategy to a specific dataset is important. Rather than applying uniform compression to all fingerprint bits, it may be beneficial to identify and merge highly correlated identifiers, or discard bits that are present in the vast majority of compounds and thus are not selective.

The studied methods, while already effective, leave room for further refinement, adapting them to dataset characteristics. As such, Johnson-Lindenstrauss and MinHash reductions represent a starting point for exploration in efficient molecular representation.

Appendix

The user API for the implementations is available in the repository:
github.com/a-szymanska/dimensionality-reduction.

A.1 Implementation of the Johnson-Lindenstrauss method

```
class JohnsonLindenstrauss
{
    int N;                                  // original dimension
    int n;                                  // reduced dimension
    std::vector<std::vector<float>> Pi;     // projection matrix

public:
    JohnsonLindenstrauss(int N, int n): N(N), n(n) {
        std::random_device rd;
        std::mt19937 gen(rd());
        std::normal_distribution<float> norm_dist(0.0, 1.0);

        Pi.resize(n, std::vector<float>(N));
        for (int i = 0; i < n; i++) {
            for (int j = 0; j < N; j++) {
                Pi[i][j] = norm_dist(gen);
            }
        }
    }

    std::vector<float> reduce(const std::vector<unsigned char> &v) {
        std::vector<float> v_proj(n, 0.0);
        for (int i = 0; i < n; i++) {
            float dot_i = 0.0;
            for (int j = 0; j < N; j++) {
                dot_i += Pi[i][j] * static_cast<float>(v[j]);
            }
            v_proj[i] = dot_i / std::sqrt(n);
        }
        return v_proj;
    }
};
```

A.2 Implementation of the MinHash method

```
const uint32_t p = 1364310083u;
const uint32_t mod = 4294967029u;

class MinHash
{
    int N;                                              // original dimension
    int n;                                              // reduced dimension
    std::vector<std::function<float(uint32_t)>> H;   // hash functions
    static float get_hash(uint32_t x, uint32_t seed) {
        uint32_t hashValue = ((x ^ seed) * p) % mod;
        return (float)hashValue / std::numeric_limits<uint32_t>::max();
    }

public:
    MinHash(int N, int n): N(N), n(n) {
        std::random_device rd;
        std::mt19937 gen(rd());
        std::uniform_int_distribution<uint32_t> uni_dist(1, mod);
        for (int i = 0; i < n; i++) {
            uint32_t seed = uni_dist(gen);
            H.push_back([seed](uint32_t x) { return get_hash(x, seed); });
        }
    }

    std::vector<float> reduce(const std::vector<unsigned char> &v) {
        std::vector<int> v_nonzero;
        for (int j = 0; j < N; j++) {
            if (v[j] != 0) v_nonzero.push_back(j);
        }
        std::vector<float> v_proj(n, 1.0f);
        for (int i = 0; i < n; i++) {
            auto min_hash = H[i](v[0]);
            int min_x = 0;
            for (int j : v_nonzero) {
                auto hash_j = H[i](j);
                if (hash_j < min_hash) {
                    min_hash = hash_j;
                    min_x = j;
                }
            }
            v_proj[i] = (float) min_x / N;
        }
        return v_proj;
    }
};
```

Bibliography

[1] Stefan Berchtold, Daniel Keim, and Hans-Peter Kriegel. "The X-tree: an index structure for high-dimensional data". In: *Readings in Multimedia Computing and Networking*. Morgan Kaufmann Publishers Inc., 2001, pp. 451–462. ISBN: 1558606513. DOI: 10.5555/383802.383913. URL: https://dl.acm.org/doi/10.5555/383802.383913 (visited on 04/25/2025).

[2] Andrei Broder. "On the resemblance and containment of documents". In: *Proceedings. Compression and Complexity of SEQUENCES 1997 (Cat. No.97TB100171)* (1997), pp. 21–29. URL: https://api.semanticscholar.org/CorpusID:11748509 (visited on 01/14/2025).

[3] Nathan Brown. *In Silico Medicinal Chemistry: Computational Methods to Support Drug Design*. Ed. by Jonathan Hirst. 8. The Royal Society of Chemistry, 2016, pp. 26–40. ISBN: 2041-3181. DOI: 10.1039/9781782622604. URL: http://ndl.ethernet.edu.et/handle/123456789/7933 (visited on 04/26/2025).

[4] Chemaxon. *Extended Connectivity Fingerprint (ECFP)*. URL: https://dl.chemaxon.com/docs/HTML/docs169190/Extended_Connectivity_Fingerprint_(ECFP).html (visited on 01/04/2025).

[5] National Research Council. *Mathematical Challenges from Theoretical/Computational Chemistry*. National Academy Press, 1995. URL: https://apps.dtic.mil/sti/tr/pdf/ADA307664.pdf (visited on 04/26/2025).

[6] Casper Benjamin Freksen. "An Introduction to Johnson-Lindenstrauss Transforms". 2021. URL: https://api.semanticscholar.org/CorpusID:232075679 (visited on 01/15/2025).

[7] Michael Garey and David Johnson. *Computers and Intractability; A Guide to the Theory of NP-Completeness*. W. H. Freeman, 1990. ISBN: 0716710455. DOI: 10.5555/574848. URL: https://perso.limos.fr/~palafour/PAPERS/PDF/Garey-Johnson79.pdf (visited on 05/11/2025).

[8] Johann Gasteiger and Thomas Engel. *Chemoinformatics*. Wiley-VCH, 2003. DOI: 10.1002/3527601643. URL: https://www.fkkt.um.si/ukemat/Gasteiger-Chemoinfo.pdf (visited on 04/26/2025).

[9] Benyamin Ghojogh, Ali Ghodsi, Fakhri Karray, et al. "Johnson-Lindenstrauss Lemma, Linear and Nonlinear Random Projections, Random Fourier Features, and Random Kitchen Sinks: Tutorial and Survey". In: *ArXiv* abs/2108.04172 (2021). URL: https://api.semanticscholar.org/CorpusID:236956607 (visited on 12/20/2024).

[10] Muthukumarasamy Karthikeyan and Renu Vyas. *Practical Chemoinformatics*. Springer, 2014. DOI: 10.1007/978-81-322-1780-0. URL: https://link.springer.com/book/10.1007/978-81-322-1780-0 (visited on 04/26/2025).

[11] Norio Katayama and Shin'ichi Satoh. "The SR-tree: an index structure for high-dimensional nearest neighbor queries". In: *ACM SIGMOD Record*. Vol. 26. Association for Computing Machinery, 1997, pp. 369–380. DOI: 10.1145/253262.253347. URL: https://dl.acm.org/doi/10.1145/253262.253347 (visited on 04/25/2025).

[12] Sunghwan Kim, Jie Chen, Tiejun Cheng, et al. "PubChem 2023 update". In: *Nucleic Acids Research* 51 (2023). DOI: 10.1093/nar/gkac956. URL: https://pmc.ncbi.nlm.nih.gov/articles/PMC9825602/ (visited on 04/26/2025).

[13] Gregory Landrum. *Fingerprints in the RDKit*. 2012. URL: https://www.rdkit.org/UGM/2012/Landrum_RDKit_UGM.Fingerprints.Final.pptx.pdf (visited on 03/24/2025).

[14] Tuan Le, Robin Winter, Frank Noé, et al. *Neuraldecipher – reverse-engineering extended-connectivity fingerprints (ECFPs) to their molecular structures*. Vol. 11. The Royal Society of Chemistry, 2020, pp. 10378–10389. DOI: 10.1039/D0SC03115A. URL: http://dx.doi.org/10.1039/D0SC03115A (visited on 03/24/2025).

[15] Andrew Leach and Valerie Gillet. *Introduction to Chemoinformatics*. Springer, 2007. URL: https://link.springer.com/book/10.1007/978-1-4020-6291-9 (visited on 04/26/2025).

[16] Avivit Levy, Riva Shalom, and Michal Chalamish. "A Guide to Similarity Measures". 2024. URL: https://api.semanticscholar.org/CorpusID:271874662 (visited on 01/13/2025).

[17] Zuqing Li, Bernard Chen, and Dongsheng Che. "Solving the Subgraph Isomorphism Problem Using Simulated Annealing and Evolutionary Algorithms". In: *International Conference on Artificial Intelligence* (2016), pp. 293–299. URL: https://worldcomp-proceedings.com/proc/p2016/ICA3229.pdf (visited on 05/11/2025).

[18] Christopher Musco. *CS-GY 9223 D: Lecture 2 Supplemental Finish MinHash, Exponential Tail Bounds*. 2020. URL: https://www.chrismusco.com/amlds2020/lectures/lec2_supp.pdf (visited on 01/04/2025).

[19] *REAL Compounds*. URL: https://enamine.net/compound-collections/real-compounds (visited on 04/26/2025).

[20] David Rogers and Mathew Hahn. "Extended-Connectivity Fingerprints". In: *Journal of chemical information and modeling* 50 (2010), pp. 742–754. DOI: 10.1021/ci100050t. URL: https://api.semanticscholar.org/CorpusID:5132461 (visited on 01/04/2025).

[21] *SMILES*. Chemaxon. URL: https://docs.chemaxon.com/display/docs/formats_smiles.md (visited on 04/26/2025).

[22] Michael Sun, Minghao Guo, Weize Yuan, et al. "Representing Molecules as Random Walks Over Interpretable Grammars". 2024. URL: https://api.semanticscholar.org/CorpusID:268379689 (visited on 01/11/2025).

[23] Roger Weber, Hans-Jörg Schek, and Stephen Blott. "A Quantitative Analysis and Performance Study for Similarity-Search Methods in High-Dimensional Spaces". In: *Very Large Data Bases Conference*. Morgan Kaufmann Publishers Inc., 1998, pp. 194–205. DOI: 10.5555/645924.671192. URL: https://api.semanticscholar.org/CorpusID:8109758 (visited on 04/25/2025).

[24] Matt Weinberg. *Dimensionality Reduction and the Johnson-Lindenstrauss Lemma*. 2019. URL: https://www.cs.princeton.edu/~smattw/Teaching/Fa19Lectures/lec9/lec9.pdf (visited on 01/04/2025).

[25] *ZINC20*. URL: https://zinc.docking.orgs (visited on 04/26/2025).