www.uhasselt.be
Universiteit Hasselt
Campus Hasselt:
Martelarenlaan 42 | 3500 Hasselt
Campus Diepenbeek:
Agoralaan Gebouw D | 3590 Diepenbeek

Maastricht University

Faculty of Sciences
School for Information Technology
Master of Statistics and Data Science

Master's thesis

Applicability domain of chemical reaction modeling

Tim Van Eylen
Thesis presented in fulfillment of the requirements for the degree of Master of Statistics and Data Science, specialization Biostatistics

SUPERVISOR :
Prof. dr. Dirk VALKENBORG
SUPERVISOR :
Ms. Natalia DYUBANKOVA

2020
2021

Faculty of Sciences
School for Information Technology
Master of Statistics and Data Science

Master's thesis

Applicability domain of chemical reaction modeling

Tim Van Eylen
Thesis presented in fulfillment of the requirements for the degree of Master of Statistics and Data Science, specialization Biostatistics

SUPERVISOR :
Prof. dr. Dirk VALKENBORG
SUPERVISOR :
Ms. Natalia DYUBANKOVA

To Laura, for her unwavering support through many
days and nights spent studying to finish this work
and my studies.
Special thanks to Natalia, who showed relentless
enthusiasm and expertise throughout and was
always ready to assist, no matter the time of day.

Contents

Chapter 1

Introduction

1.1 Computer-assisted Synthetic Planning in Drug Discovery

In the field of organic chemistry, predicting the feasibility of a reaction pathway has always been a daunting task. Pharmaceutical research in particular is dependent on these organic pathways for small-molecule drug discoveries. Due to the estimated 10^{60} possible drug candidates, the over 135 million reported synthesised drug molecules constitute only a small fraction of the feasible molecule space [1]. Given that there is often more than one way of producing a certain molecule, the reaction space itself is even greater still. As potential drug molecules are often synthesised in a string of subsequent reactions, it is important to be able to predict each of these reactions correctly in order for the overall reaction to be a success. Even if it would be impossible to predict the outcome of each reaction with full accuracy, it can be helpful to determine a priori which of these reactions has the lowest probability of success in order to fail fast and reduce the economic impact of such failure. Moreover, the potential reaction space can be limited by the commercial availability of compounds and the perceived ease of synthesis.

As far back as 1967, researchers have attempted to develop software able to assist in designing these chemical pathways *in silico* [2]. These techniques have been coined computer-assisted synthetic planning (CASP) and have taken different routes to achieve their goal. Other goals than forward prediction (predicting the outcome of a reaction based on the input reactants) could be retrosynthesis (predicting the necessary reactants based on an input product molecule) or even

the elucidation of the mechanism behind the reaction [3].
The first such models were based on an extensive set of chemical rules. These rules were collected from the experience of human experts in chemical reaction modelling. Based on the collective set of rules, known as the template (or transform) library, reaction centers could be determined in input molecules and predictions were made based on the templates that were encoded in the model. The obvious disadvantage of these models is that they require manual encoding of the templates and building these libraries is very time consuming. Furthermore, deciding on the full extent of the reaction centers requires its own heuristics: Too many atoms included could make it difficult to find an appropriate template, while too few might oversimplify the reaction and produce incorrect results [4]. These constraints often make the models accurate only within a limited reaction space. Despite these drawbacks, 2013 saw the release of the most extensive of template-based models in *Chematica*, which uses a database of approximately 10 million compounds and more than 4,400 transforms. However, compiling this database of knowledge required over a decade of work [5]. In 2017, Merck KGaA acquired *Chematica* and rebranded it as *Synthia*. In its current form, the tool holds over 100,000 hand-coded reaction rules [6].
A second approach has been to rely on the physical and thermodynamic properties of the molecules. Machine learning techniques were applied to search for the most probable (lowest energy) transition states, which in turn provide an answer to which products are most likely to form [7, 8]. The major drawback to these methods is the high computational cost involved in the quantum mechanics that accompany these calculations of free energy.

Modern day developments have sparked a new interest in the possibilities of neural networks. With ever increasing hardware specifications, numerous architectures have been developed in the field of artificial intelligence (AI). In a well-publicised confrontation in 2016, DeepMind's reinforcement learning neural network AlphaGo defeated world champion Lee Sedol at Go, a strategic board game that had proven to be a great challenge for AI due to the sheer number of possible nodes at each step, making a brute-force approach intractable [9]. While neural networks have been around for many decades, this event set a landmark for what can be achieved with modern day hardware and spawned a renewed interest for many different research applications. This renaissance of neural networks has also been felt in the field of computational chemistry. They can be applied in combination with rule-based systems to select either the template reaction [10] or to filter reactive sites or select the correct molecular orbital interactions in mechanistic approaches [11, 12]. Coley et al. suggested a two-step approach to predict reaction outcomes. In the first step, the reactant graph is analyzed and the most likely bonds to change in the reaction are predicted by using a graph-convolutional neural network. In a second step, these changes are consolidated to form a set of possible products, from which the most likely to be formed is selected using a second neural network combined with the likelihood outcomes from the first step evaluated in each product candidate [13]. However, aside from these approaches, the use of neural networks has also proven to be a key contributor in the development of template-free prediction models, a third and final approach to reaction modelling. Borrowing from advances in natural language processing (NLP) [14], architectures with recurrent neural networks (RNN) for the encoder and decoder with a single attention layer in between have been proposed to cast the set of reactants and reagents onto a set of outcome molecules (forward prediction) or vice versa (retrosynthesis) [15, 16, 17]. These so-called sequence-to-sequence models treat the reaction prediction directly as a translation problem, where the implicit rules of organic pathways can be seen as the grammar of the language. However, these models make the distinction between reagents and reactants in the input, which requires atom-mapping to connect each atom in the product to a single atom in the reaction input, a process which is inherently linked to the use of templates and rule-based heuristics. In 2019, Schwaller et al. proposed a multi-head attention mechanism that is no longer based on the RNN component of the previous models and is able to be trained on data with unseparated reagents and reactants, therefore abolishing the need for atom-mapping [18, 14]. A single model top-1 accuracy of 88.6% was reported, which increased to 94.2% when the top-5 was considered. Building on this work, Su et al. developed a bilateral long-short-term memory model with self-attention to predict not the most likely product distribution, but to classify whether or not a reaction from a specific reaction family would be successful. To achieve this end, they conducted 700 reactions and used the yields as input in their model and an accuracy of over 80% was reported [19]. Tetko et al. proposed an NLP transformer model for retrosynthesis, scoring a 97% top-5 accuracy on single-step reactions [20]. A more comprehensive list of prediction models is available in literature reviews [3, 4, 5, 21].

1.2 Chemical Fingerprints and Datasets

No matter what approach or which model is selected, an appropriate format is required to read and output the data. This vocabulary needs to be unambiguous and complete. Throughout the last decades, several different formats have been proposed. One such formats is the IUPAC internal chemical identifier for reactions (RInChI) which aims at creating an unambiguous line notation for reactions by grouping them into reactants, products and catalysts after sorting

them within the group. The format allows for hashing into RInChI keys, shorter fingerprints of the original reaction, but does not allow any form of atom-mapping [22].
A different format is provided by the Chemical Markup Language (CML). This is the XML equivalent for chemical notation, which can be expanded to CMLReact. CML provides a large flexibility in the representation of a molecule and can hold several notation conventions at once. While this is the strength of the markup language, it is also a perceived weakness as there is not one unique way of representing any given molecule or reaction [23].
Finally, one of the oldest and most commonly used notations is the Simplified Molecular Input Line Entry System (SMILES) [24]. A clear set of rules was determined to project any organic compound onto a data string. In doing so, the notation is simplified wherever possible (e.g. removal of hydrogen atoms and single and aromatic covalent bonds). While some flexibility is allowed in the simplification, equivalent SMILES are interchangeable and can be read by most cheminformatics software packages available. Reactions can be represented by tokens separating reactants from products and molecules from one another. Extensions to reaction SMILES have been made (SMARTS, DeepSMILES, SELFIES) to include, among other, atom-mapping and reaction center information [25, 26].

Equally important as having a unified vocabulary to represent reaction data, is the availability of a large and reliable dataset. To this end, by far the most commonly used is the United States Patent and Trademark Office (USPTO) dataset. The USPTO provides a large, and, more importantly, freely available set of chemical reactions for which patents have been filed. Based on the abstracts, text-mining work has been done to extract the reactions from this dataset in the formats previously described [27]. Each entry in the USPTO was parsed into a CML file containing the reaction SMILES, the paragraph from which it was extracted and the reaction conditions in which the reaction took place. In this work, the full USPTO dataset consisted of a total of 1,387,365 unique canonicalized reactions, which were standardized according to the algorithm proposed by Madzhidov et al [28].
At various points in the lifecycle of this dataset, subsets were created by different research groups with different characteristics. The USPTO_MIT subset was created by Jin et al. in order to create a dataset with no stereochemical information. Furthermore, all exact duplicate reactions were removed as well as erroneous reactions. The resulting reaction dataset contained 480k reactions [29]. Using this subset as a starting point, Bradshaw et al. filtered the data even further and removed all reactions where no linear electron flow topology was

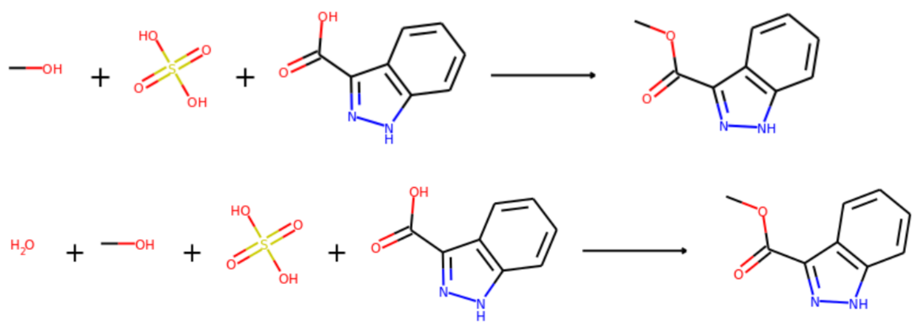


Figure 1.1: An example of two near-duplicate reactions in the USPTO dataset. The bottom reaction [ID 2187557] reported H_2O explicitly, where it was left out in the top reaction [ID 2874384].

present, with the intent of creating a prediction model (ELECTRO) based on electron flow [30]. Schwaller et al. on the other hand, created a subset of unique canonicalized reactions starting from an update of the original USPTO dataset that was published in 2016, keeping as much information as possible [17].
Asides from the publicly available USPTO dataset, reaction data for researchers is only sparsely available. While additional reaction databases exist, such as SciFinder, Reaxys, Pistachio or SPRESI, these are commercial packages and are not readily available to all research groups. Unfortunately, as is often the case with collecting data from published material, and patents in particular, the resulting datasets will suffer from publication bias. While the available reactions are plenty, they are but a fraction of the total attempted synthesis routes and may therefore lead to an overestimation of the success rate of certain pathways [1]. Raccuglia et al. showed the promise of using so-called 'dark', or unsuccessful, reactions in predicting the reaction success in the synthesis of inorganic materials [31].

1.3 Research Outline

As outlined in the previous section, the reaction SMILES generated during scraping of a large text-based dataset such as the USPTO, are dependent on the formulation used by the author of the journal or the patent. While reaction conditions are filtered out and stoichiometry and byproducts of the reaction are generally ignored, there is no consensus approach on reporting solvents, reagents or catalysts. This can potentially lead to duplicate reactions entering the data twice, but with slightly differing reaction SMILES. An example of such a near-duplicate reaction is given in figure 1.1. When duplicate reactions are present in the train-, validation- and test-splits, this will result in incorrect accuracy predictions due to data leakage. As reported by Schwaller et al., no consensus approach currently exists in the removal of near-duplicate reactions [17].

However, comparing reactions pairwise in order to find and remove near-duplicates can only be achieved in $O(n^2)$ time complexity, as the number of pairwise comparisons is given by $\frac{n(n-1)}{2}$. Due to the large scale of the datasets at hand, this is intractable even on modern computing hardware. To this end, the use of Locality-Sensitive Hashing (LSH), a family of hashing algorithms originally designed for the retrieval of near-duplicate web-pages [32], was explored on a combined dataset, consisting of the full USPTO, Reaxys and Pistachio reaction SMILES data.

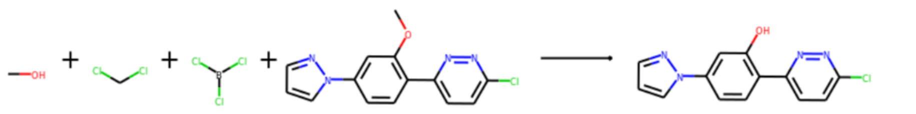
Chapter 2

Data Structure and Methodology

2.1 Data and Metrics

A combination of three different datasets was used. As described in the previous chapter, the USPTO dataset is freely available, built from a database of US patents, and the most common benchmark dataset for chemical computation research. Asides from the USPTO dataset, NextMove's Pistachio dataset was also used. This is a proprietary dataset consisting of a combination of US and EU patents, totalling over 4 million reactions. Finally, Elsevier's Reaxys dataset provides a combination of patents and journal entries. Just like Pistachio, the Reaxys dataset is proprietary and the most extensive of the three, providing information on over 15 million reactions. Due to the common sources of the three datasets, some overlap is present between them. The combination of the these datasets will from here on out be referred to as Reactlake. After removing the exact duplicates from Reactlake, it consisted of 18,727,618 remaining reactions.

All three datasets are available in the form of canonicalized reaction SMILES. Graphical representations of reactions throughout this work were created with RDKit (2020.03.3) in Python 3.6. These reaction SMILES were converted to individual molecules and one-hot encoded in reactant and product vectors. As such, each reaction SMILES can be represented by two reduced vectors containing the one-hot encoded positions of each individual molecule in the sparse matrix (see figure 2.1 for an example). In order to remove all near-duplicate reactions, a metric is required to formally capture the distance (or equivalently, the similarity)



```
OC.C(Cl)Cl.B(Cl)(Cl)Cl.c1(-c2ccc(Cl)nn2)c(OC)cc(-n2cccn2)cc1>>c1(-c2ccc(Cl)nn2)c(O)cc(-n2cccn2)cc1
[17513, 1277509, 4719637, 6688656] -> [2709986]
```

Figure 2.1: An example of a single reaction [ID 3365334] (top) with its corresponding reaction SMILES (middle) and reactant and product vectors (bottom).

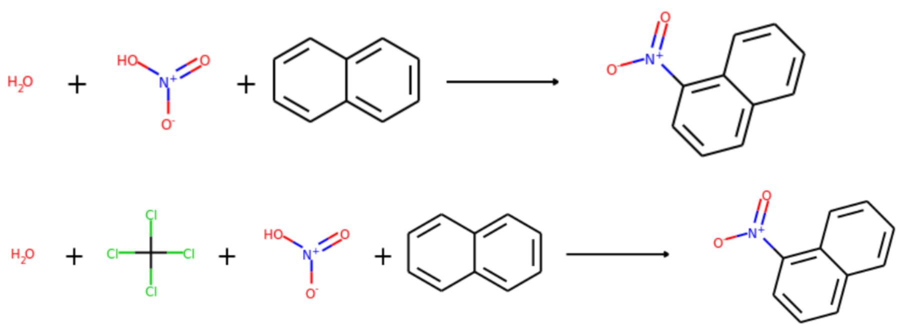


Figure 2.2: Two near-duplicate reactions differing only in the organic solvent CCl_4. Top [ID 127920]: reactant and product vectors are given by [301202, 6678879, 6679887] and [1818850], Bottom [ID 552473]: reactant and product vectors are [301202, 4719643, 6678879, 6679887] and [1818850].

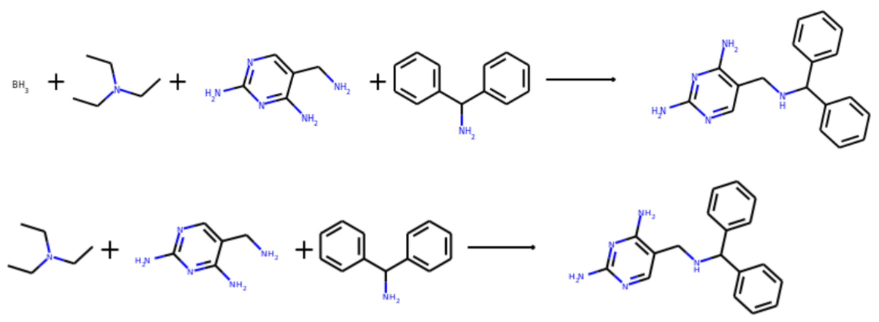
between two reactions. Next, a classification threshold should be set to determine wether or not the calculated distance is sufficient to classify the two reactions as near-duplicates.
Starting from the one-hot encoded vectors, several similarity indeces can be used, each providing slight nuances.

The Hamming distance is an absolute distance metric that is defined as the number of different entries in two equal-length sets. It can be viewed as the number of changes required to map one vector onto the other. The Hamming distance of bit-vectors 010101 and 010111 is as such calculated as 1. For two molecule sets A and B of arbitrary sizes in the one-hot encoded vector, it can be generalized as the difference between the union and the intersection of the two reduced vectors or $|A \cup B| - |A \cap B|$. In the example of figure 2.2, the Hamming distance of the reactant vectors and product vectors is 1 and 0 respectively. While the Hamming distance provides an absolute measure of similarity between two equal-sized vectors, the Jaccard similarity can be seen as an extension of the Hamming distance, but relative to the size of the union of non-zero entries in the vectors. The Jaccard index or similarity coefficient (sometimes referred to

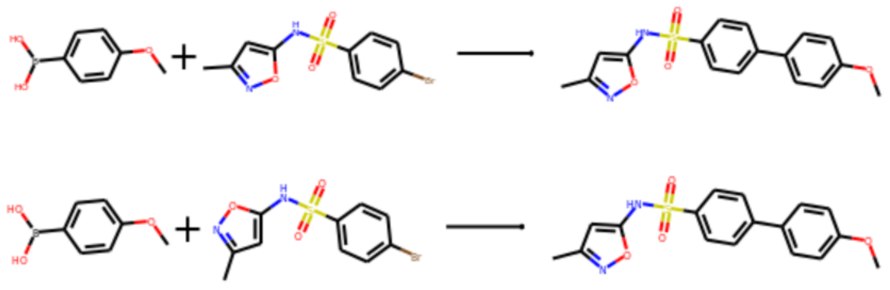
as the Tanimoto index) is defined as $\frac{|A \cap B|}{|A \cup B|}$ and is a measure of the ratio of equal entries to the total number of entries in the vectors. By design, the Jaccard index exists only in the interval [0,1]. The Jaccard distance is calculated as the complement of the Jaccard index: $1 - \frac{|A \cap B|}{|A \cup B|}$ or $\frac{|A \cup B| - |A \cap B|}{|A \cup B|}$. As such, it is the ratio of the Hamming distance to the union of entries in the two vectors. Applying this definition to the example of figure 2.2, the Jaccard distance of the reactant vectors is calculated as 0.25.
Other distance metrics, such as the Sørensen-Dice coeffient or the Tversky-distance are very similar to the Jaccard index and were not considered. Given that the distance metrics can be calculated on both the reactant and the product vectors of two reactions, this provides us four different metrics.

2.2 Benchmark Dataset

In order to find the thresholds to properly classify reactions as near-duplicates, a supervised dataset was generated. The dataset originated by the use of two different text mining algorithms that were used on the same subset of the USPTO_MIT subset of the USPTO dataset as described in the previous chapter. While the USPTO entries are the same, this produced small differences in the output reaction SMILES, which can be considered near-duplicate reactions. Two examples of such a simulated near-duplicate reactions are given in figure 2.3. A total of 37,372 reactions comparisons were created as such. To augment the dataset with reaction comparisons not considered to be (near-)duplicates, a sample of 100,000 reactions was taken from the full USPTO dataset and sliced in half. Next, these reactions were placed side-by-side to create 50,000 random reaction pairs. While these reaction comparisons may include near-duplicates by chance, it is reasonable to assume the majority of these comparisons are different reactions altogether and the distribution of the different metrics in this sample are considered a proper sample of the overall pairwise comparison metrics in the dataset. The combined 87,372 reactions were used to determine a proper set of classification metrics. Based on this dataset, a visual exploration was used to determine a starting point for the classification metrics. These metrics were then used to find near-duplicate reactions in samples of the USPTO dataset. Based on the evaluated outcome of these findings, the metrics thresholds were fine-tuned further until the algorithm provided satisfactory results.



(a)



(b)

Figure 2.3: Two examples of simulated near-duplicates originating from the USPTO_MIT subset of the USPTO dataset. (a) The bottom reaction is without the borane molecule included in the top reaction. (b) A different interpretation of the molecular orientation of one of the reactants leads to two different SMILES representations of one achiral molecule.

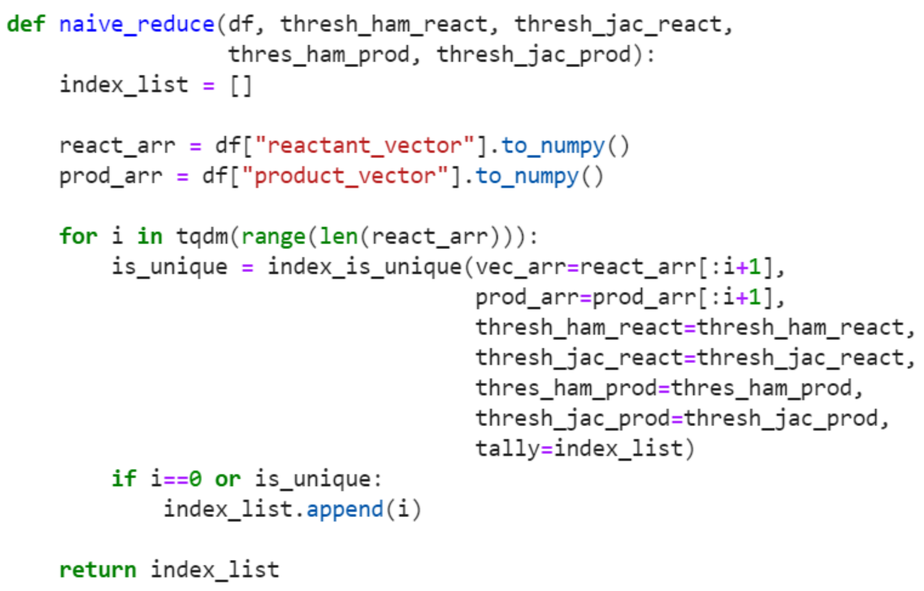
2.3 Reduction Algorithms

2.3.1 Brute-Force Calculations

When searching a dataset with ever increasing size, the use of a brute force pairwise comparison algorithm soon becomes intractable. A very simple algorithm is presented in figure 2.4. The number of pairwise comparisons is approximately equal to $\frac{n(n-1)}{2}$. In practice, the number is slightly less because near-duplicates that are identified as such will not be taken into consideration for future comparisons. Because all possible comparisons are performed, the results of the brute force algorithm are considered the ground truth in any accuracy benchmark.

2.3.2 Locality-Sensitive Hashing

The main idea behind Locality-Sensitive Hashing (LSH) is to provide an efficient technique to find nearest neighbours of the target vector. Any target candidate



```python
def naive_reduce(df, thresh_ham_react, thresh_jac_react,
                 thres_ham_prod, thresh_jac_prod):
    index_list = []

    react_arr = df["reactant_vector"].to_numpy()
    prod_arr = df["product_vector"].to_numpy()

    for i in tqdm(range(len(react_arr))):
        is_unique = index_is_unique(vec_arr=react_arr[:i+1],
                                    prod_arr=prod_arr[:i+1],
                                    thresh_ham_react=thresh_ham_react,
                                    thresh_jac_react=thresh_jac_react,
                                    thres_ham_prod=thres_ham_prod,
                                    thresh_jac_prod=thresh_jac_prod,
                                    tally=index_list)
        if i==0 or is_unique:
            index_list.append(i)

    return index_list
```

Figure 2.4: The brute force algorithm: The index_is_unique() function iterates over reactions in the index list and compares each entry against the new candidate. If a near-duplicate is not found, the candidate is added to the list.

can be compared against these nearest neighbours only, which results in O(n) time-complexity of the algorithm.

To achieve this goal, the LSH algorithm used is built upon a different algorithm, known as the MinHash algorithm. In this algorithm, a specific fingerprint for each vector is created in a very specific way. Each iteration, a permutation of the sparse vectors holding the entries is performed. The position of the first non-zero entry of this permutation is taken as the next bit in the fingerprint of a specific data vector. A schematic example of this process is given in figure 2.5. The most important property of this permutation algorithm is that the likelihood of a single bit in the fingerprints of two vectors colliding is equal to their Jaccard similarity. The fraction of equal bit entries between two MinHash fingerprints is therefore an unbiased estimator for the Jaccard similarity of their corresponding sets. However, creating permutations from the vectors would be a very memory-inefficient algorithm. The same Jaccard properties hold when each bit is calculated by substituting each permutation by a random hash function f. Let $e_1, e_2, ..., e_n$ be the n entries in the vector s. The j'th value in the MinHash fingerprint of s is then given by $min(f_j(e_1), f_j(e_2), ..., f_j(e_n))$. The latter implementation automatically clarifies the origin of the term MinHash. Each bit in the fingerprint will still be referred to as a permutation in this text.

$$\begin{array}{ccc} s_1 & s_2 & s_3 \\ \end{array} \qquad \begin{array}{ccc} s_1 & s_2 & s_3 \end{array} \qquad \begin{array}{ccc} s_1 & s_2 & s_3 \end{array}$$
$$\begin{bmatrix} 0 & 1 & 0 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 1 \end{bmatrix} \rightarrow \begin{bmatrix} 1 & 1 & 1 \\ 1 & 0 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 1 \end{bmatrix} \rightarrow \begin{bmatrix} 0 & 1 & 0 \\ 1 & 1 & 1 \\ 1 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 1 \\ 0 & 1 & 0 \end{bmatrix}$$

Figure 2.5: An example of two MinHash permutations. Every column represents a data vector. The 3-bit fingerprints from these permutations for the three data vectors (left to right) would be 201, 000 and 301. The estimated Jaccard similarities would be: $\hat{J}(s_1 - s_2) = 0.33$, $\hat{J}(s_2 - s_3) = 0.33$, $\hat{J}(s_1 - s_3) = 0.66$. The true Jaccard similarities are: $J(s_1 - s_2) = 0.125$, $J(s_2 - s_3) = 0.25$, $J(s_1 - s_3) = 0.75$.

Assuming each vector set has been mapped to a MinHash fingerprint of m permutations, these fingerprints can now be used in the LSH algorithm. To do so, the m permutations are divided into p blocks of bits and each of these blocks will be hashed together into hash buckets. In the final step of the algorithm, two vectors will only be considered as a comparison pair if at least one of their p blocks ends up in the same hash bucket. An example of this is given in figure 2.6. Due to the intrinsic speed and efficiency of hashing, this greatly

s_1: 43592 35743 29452 29504
s_2: 43592 43654 23145 23914
s_3: 34211 11032 20134 23914

Figure 2.6: Three hypothetical fingerprints of 20 permutations are separated into bit buckets of length 5. In this example, the first and second vector sets will be considered for comparison, as well as the second and third.

reduces the overall computation time compared to the very inefficient process of performing pairwise comparisons. In particular, when hashing the fingerprint sequences in hash buckets, they are stored in memory and do not require any form of processing when introducing new candidates for comparison. This in contrast to the brute force comparisons, where the pairwise distance has to be calculated for each comparison in each new iteration. While this strongly benefits the computation time, it can be taxing on memory. Depending on available hardware, this could force a limit on dataset size and number of permutations per fingerprint allowed.

However, two types of errors are introduced due to the fact that the permutations in the fingerprint are not injective and the collision of a permutation in two sets is therefore a stochastic event, depending on the random hashing function. As

described earlier, the probability of the collision of a single permutation in two sets is given by their Jaccard similarity. These errors can incur a cost towards accuracy and computation time.
On the one hand, there is the false positive event that two bit blocks collide, even though they do not originate from near-duplicate reactions. This false positive event will decrease performance, as a fruitless comparison is triggered, but will not affect accuracy of the algorithm as it is only the final comparison, taking into consideration the four metrics defined earlier, that will determine whether or not two vector sets are considered to be near-duplicates. In this application, the LSH approach only acts as a preliminary filter to decrease computation time.
On the other hand, there is a possibility that two near-duplicate vector sets do not share a single bit block and will not be taken into consideration as near-duplicates, effectively harming the accuracy of LSH as compared to the ground truth. Assuming a comparison is warranted if the Jaccard similarity exceeds a certain threshold x, the ideal probability of collision as a function of t, the Jaccard similarity, is a stepwise function given by:

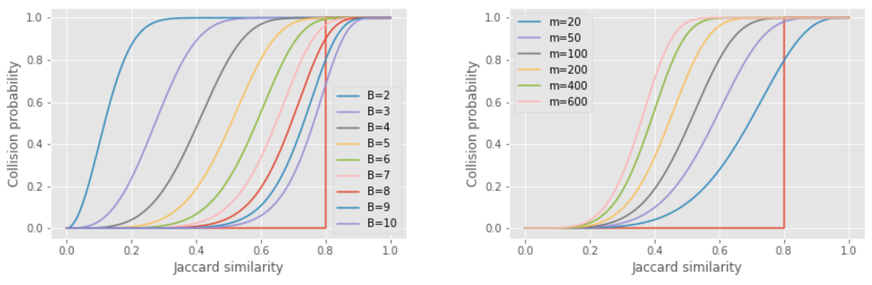
$$P(t) = \begin{cases} 0 & t < x \\ 1 & t \geq x \end{cases}$$

Because the probability of collision in the LSH algorithm is known for a single permutation, the probability of a collision of at least one block in two fingerprints (P(t)), assuming m permutations and p blocks of size $B = m/p$, is construed as:

$$\begin{aligned}
P(t) &= 1 - P(\text{No collision}) \\
&= 1 - P(\text{Single block not colliding})^{m/B} \\
P(\text{Single block not colliding}) &= 1 - P(\text{Collision of single block}) \\
&= 1 - P(\text{Collision of all permutations in single block}) \\
&= 1 - t^{(B)} \\
P(t) &= 1 - (1 - t^{B})^{m/B}
\end{aligned}$$

An overview of P(t) for several choices of B for fixed size of m and several choices of m for fixed size B and a threshold of 0.8 is given in figure 2.7. As can be seen in this example, a trade-off between the two types of errors is always present. As discussed earlier, permutation size may be limited due to memory constraints. Not only does it play a factor in determining P(t), a larger permutation size also decreases the variance of the estimated Jaccard similarity.
Since LSH acts as a preliminary filter for selecting comparisons, it is important to note that the Jaccard similarity chosen for the threshold is not necessarily

the same Jaccard similarity as chosen in the comparison metrics. A logical requirement is that the threshold selected in LSH would be smaller or equal to the threshold of the Jaccard similarity selected in the near-duplicate reduction metric for these vector sets. Implementation was done via the MinHash and MinHash



(a) (b)

Figure 2.7: Collision probabilities for: (a) fixed permutation size $m = 100$ and various block sizes (b) fixed block size $B = 5$ and various permutation sizes.

LSH classes from the datasketch (1.5.3) module in Python 3.6. An overview of the implementation of the algorithm is given in figure 2.8. The datasketch

```
def minhash_final(react_arr, prod_arr, perms, lshjac, thresh_ham_react, thresh_jac_react,
                  thresh_ham_prod, thresh_jac_prod, lshmetric='product'):

    lsh = MinHashLSH(threshold=lshjac, num_perm=perms)
    index_list = []

    if lshmetric == 'reactant':
        for i in tqdm(range(len(react_arr))):
            m = MinHash(num_perm=perms)
            for mol in react_arr[i]:
                m.update(str(mol).encode('utf8'))

            similars = lsh.query(m)
            is_unique = True

            if len(similars) != 0:
                for j in similars:
                    if (ham_distance(prod_arr[i], prod_arr[j]) <= thresh_ham_prod) and \
                    (ham_distance(react_arr[i], react_arr[j]) <= thresh_ham_react) and \
                    (jaccard_distance(react_arr[i], react_arr[j]) <= thresh_jac_react) and \
                    (jaccard_distance(prod_arr[i], prod_arr[j]) <= thresh_jac_prod):
                        is_unique = False
                        break

            if i==0 or is_unique:
                lsh.insert(i, m)
                index_list.append(i)
```

Figure 2.8: An overview of the LSH algorithm.

implementation only requires the user to set the number of permutations and the LSH query threshold. The block size is automatically optimized for these two parameters.

Each new candidate reaction is mapped to a fingerprint in the form of a MinHash object. For this purpose, both the reactant vector or the product vector can be selected. Next, near-duplicate candidates are queried from the already selected reactions that are considered to be unique reactions. The datasketch query will search through all candidates with at least one collision and will only return these reaction vectors that have an estimated Jaccard equivalence higher or equal to the threshold selected. The estimated Jaccard equivalence is calculated via the overlap in the MinHash fingerprints. This process is considered to be the preliminary LSH filter. The candidate is then compared to this selected set of vectors based on the previously selected metrics. If no match is found, the MinHash object is added to the LSH object and to the index list of unique reactions, otherwise it is discarded.

2.4 Forward Prediction Modelling

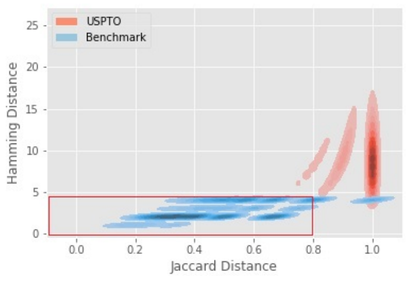
In conclusion, the Reactlake data is used in the previously described IBM Molecular Transformer. In doing so, three different models were trained. First, the USPTO_MIT data [17] was used without augmentation and without separating reactants from reagents in the input. The single-model top-3 accuracies were calculated and compared to the accuracies described in the original research paper to confirm the proper use of the Molecular Transformer.
Next, the model was trained on two versions of the Reactlake data set. One version contained the Reactlake data set after removal of the near-duplicates using the previously described LSH algorithm. The other data set contained a random sample of the full Reactlake dataset, but equal in size to the reduced Reactlake dataset. The size was kept equal in order to properly compare results between the two Reactlake datasets. Both Reactlake models were trained and assessed on three separate random splits.

Chapter 3

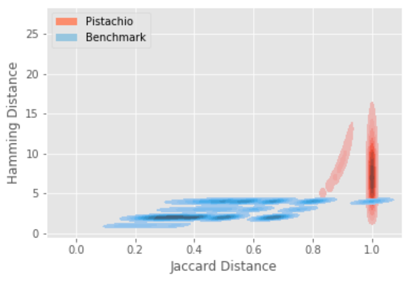
Results

3.1 Reduction Metrics

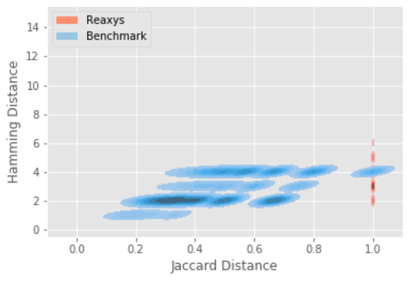
A visual exploration of the joint kernel distribution estimate (KDE) of the Hamming distances and Jaccard distances of the reactant vectors of the benchmark dataset is given in figure 3.1. Similar benchmark datasets were created from the Reaxys and Pistachio datasets to show the general applicability of these classification metrics. The benchmark dataset also contains reactions with a Jaccard distance of 1. These reactions can be regarded as miss-classified as a side-effect of the pairing algorithm used to join the reactions, as two reactions with a Jaccard distance of 1 have no molecules in common in the reactant vector. Based on this exploration, one could suggest to define a near-duplicate reaction as a reaction with a Hamming distance lower or equal to 4 and a Jaccard distance lower or equal to 0.8. However, as can be seen from the examples in figure 3.2, using only the reactant vectors to calculate distance returns many false positives when applied to the USPTO dataset. This can in large part be explained by the presence of common reagent molecules across reactions, effectively diluting the Jaccard distance. When the reactant vectors are relatively small, the Hamming distance will be small as well, keeping in mind that the Hamming distance can never be larger than the union of the two reactant vectors.
It is of importance to notice the sensitivity of the algorithm to false positive reactions. While the benchmark dataset in figure 3.1 shows a sample of 50,000 reaction comparisons from the USPTO dataset, a reaction candidate in the Reactlake dataset will be compared against a set of reactions of over 9,360,000 in size on average. While the vast majority of non-duplicate reactions will be classified correctly, a single false positive among the comparisons will lead to an



(a)



(b)

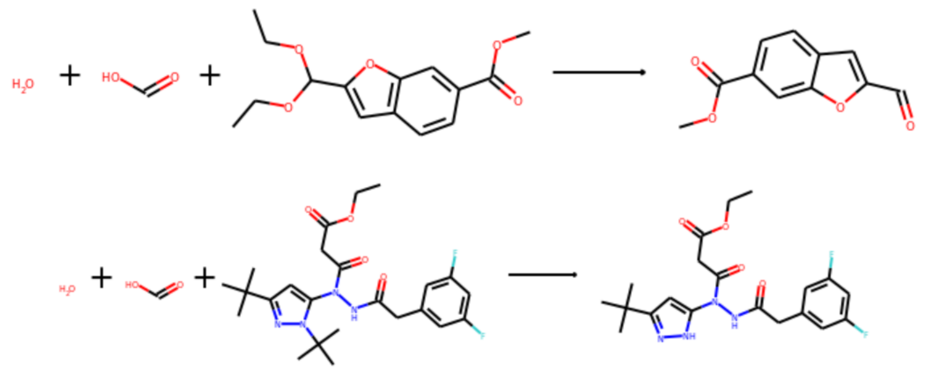


(c)

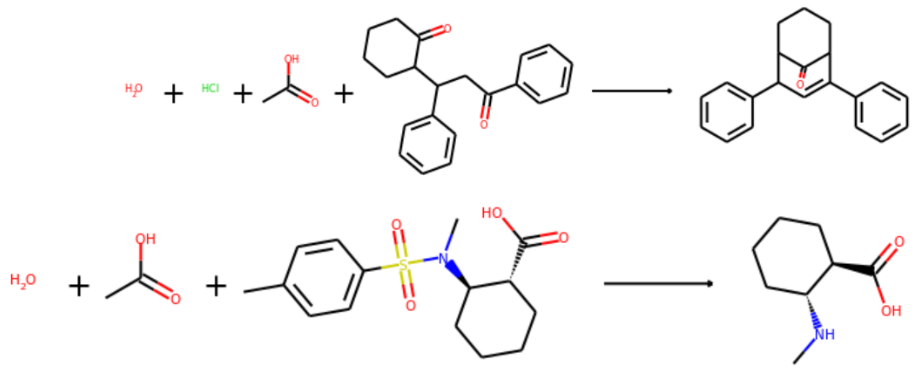
Figure 3.1: Kernel density estimate plots of the benchmark datasets for the joint distribution of the Jaccard and Hamming distances. (a) The KDE plot for the MIT400k and USPTO benchmark dataset. The red rectangle shows the area for a Jaccard distance < 0.8 and a Hamming distance < 4. (b) and (c) show similar plots for the Pistachio and Reaxys datasets respectively.

incorrect removal of the reaction from the dataset.

In order to further enhance the classification metrics, the Jaccard distances and Hamming distances for the product vectors were included as well. Due to the tendency of products being reported on more parsimoniously, with reagents, solvents and byproducts often being removed all together, the resolution of the Jaccard and Hamming distances is larger in the product vectors, allowing for easier separation. The Jaccard distance of 99.5% of the benchmark product vectors was exactly 0, while 99.98% of the Jaccard distances in the USPTO sample were exactly 1. At the same time, the Hamming distance of 93.6% of the USPTO sample product vector comparisons was 2 or less, while the Hamming distance of 99.5% of the benchmark reactions was exactly 0 and no Hamming distances exceeded 1. Based on these comparisons, the following metrics were selected for all further near-duplicate reaction searches: Jaccard distance of reactant vectors ≤ 0.8, Hamming distance of reactant vectors ≤ 4, Jaccard distance of reactant vectors ≤ 0.4 and Hamming distance of product vectors ≤ 1.



(a)



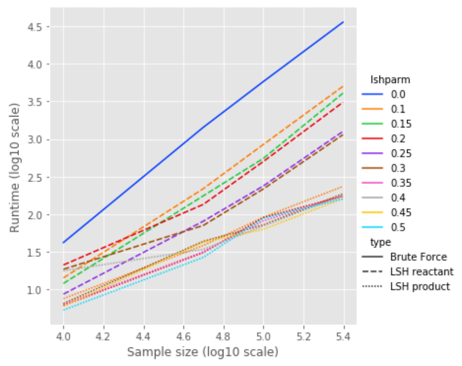
(b)

Figure 3.2: Two examples of false positive duplicate reactions in the USPTO dataset from a sample of 100,000 reactions. (a) Due to the presence of water and formic acid in both reactions [ID 683384, ID 2297334], the Jaccard distance drops to 0.5, while the Hamming distance is 2. (b) A similar false positive duplicate reaction pair [ID 575959, ID 3641143] with a Jaccard distance of 0.6 and a Hamming distance of 3.

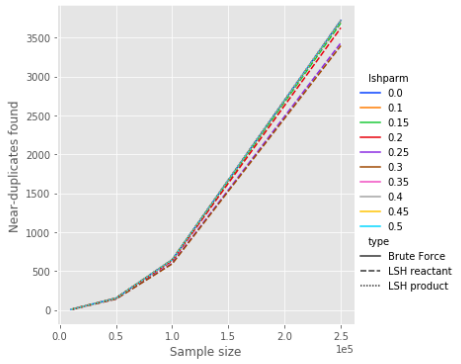
3.2 Reduction Efficiency & Accuracy

Using the datasketch implementation for the LSH query, a total of three parameters are required as input.

Because each reaction is split into a reaction and a product vector, the first choice is which vector is to be used in the LSH query. As can be seen in figure 3.3, the algorithm works best on the product vector, both for speed and accuracy. This is to be expected as the Jaccard similarities of the product vector are more polarized towards 0 or 1. This reduces the chance of both false positives and false negatives. It is of note that the runtimes reported here are hardware dependent and should only be considered relative to one another. Second, the threshold in the query has to be decided on. As outlined previously, the estimated Jaccard similarity is calculated as the ratio of matching bits in two MinHash fingerprints.



(a)



(b)

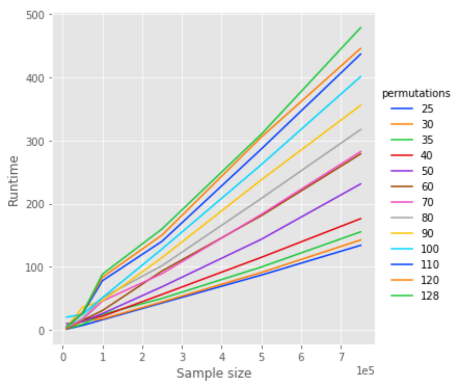
Figure 3.3: Results for different LSH thresholds (lshparm) for a fixed number of permutations (128) on a sample size of 250,000 reactions: (a) The wall times on a log-log scale (b) The number of duplicates removed from the sample slowly digressed from the ground truth (3720) for LSH on the reactant vectors for increasing thresholds, while it remained constant for LSH on the product vectors.

After a new fingerprint is hashed into its corresponding bit buckets, the LSH query searches among the reactions with at least one bit block collision and out of these reactions, returns those with an estimated Jaccard similarity exceeding the threshold. As stated earlier, a logical choice of threshold is one lower than the Jaccard similarity threshold of the corresponding vector in the reduction metrics. Table 3.1 shows the results of an increasing threshold on the accuracy and wall times for LSH performed on the reactant or product vectors at the endpoint of figure 3.3. Not only is LSH on the product vectors faster, it maintained an accuracy of 100% versus the ground truth (3720 near-duplicates removed), even with increasing thresholds. The final hyperparameter is the number of permutations or the length of the MinHash fingerprint. A larger fingerprint will increase the estimation precision of the estimated Jaccard similarity. At the same time, the number of permutations can not directly be linked to the collision probability of two vectors. Additional permutations can either be used to increase the number of bit blocks, which increases the probability of collision, or to increase the block length, which has the opposite effect. As figure 3.4a shows, there is a computational prize in increasing the number of permutations due to the increased number of hashing functions that have to be calculated for each individual candidate. Due to the optimization of the LSH algorithm in the datasketch implementation, the effect on the accuracy is minimal. This can be seen in figure 3.4b and table 3.2.

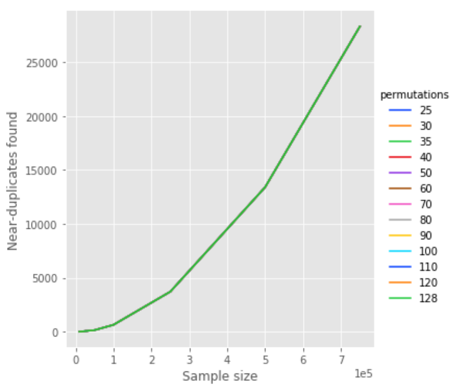
However, the effect of the permutation number on the performance of the

Table 3.1: Wall times and near-duplicate results for the LSH reduction on reactant and product vectors for a sample of 250,000 reactions of the USPTO dataset.

	Threshold	Wall Time (s)	Near-duplicates
	0.10	5020	3702
	0.15	4069	3,688
Reactant	0.20	3080	3,626
	0.25	1263	3,432
	0.30	1142	3,396
	0.10	234	
	0.20	174	
	0.30	187	
Product	0.35	174	3,720
	0.40	177	
	0.45	162	
	0.50	160	



(a)



(b)

Figure 3.4: Influence of the number of permutations on the results on a sample size of 750,000 reactions and a fixed threshold of 0.5 on the product vector: (a) The wall times showing the linear time complexity of LSH (b) The effect of the number of permutations is negligible on the number of near-duplicates found in the sample.

algorithm is not only apparent on the runtime. Perhaps of more importance is the strain it puts on memory. Each candidate fingerprint that has been assessed as unique is introduced into the LSH object, causing memory usage to grow almost linearly with the sample size of the dataset in scope. An example is given in figure 3.5 for a sample size of 1,000,000 reactions. The initial sharp increase in memory usage is due to the dataset being loaded. The next incline which lasts for most of the runtime is attributed to the ever increasing size of the LSH object. With an average of 8.7kb per insertion of a MinHash element of 128 permutations, one gigabyte of memory can store an LSH object with

Table 3.2: Wall times and near-duplicate results for the LSH reduction on product vectors for a sample of 250,000 reactions of the USPTO dataset and different numbers of permutations for a threshold of 0.5.

Permutations	Wall Time (s)	Near-duplicates
30	44	3,715
40	56	3,717
60	94	3,719
80	102	3,719
100	128	3,719
128	160	3,720

approximately 115,000 entries. This makes scaling the algorithm to sample sizes of several millions only tractable either on machines with copious amounts of memory or with a decreased amount of permutations. The full Reactlake

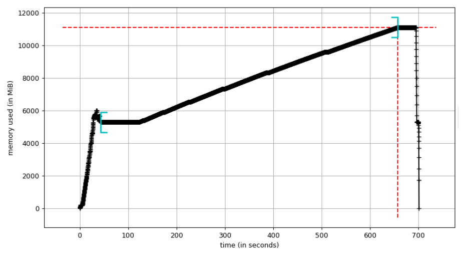


Figure 3.5: Memory consumption during the LSH reduction of 1 million reactions. The incline is due to the rapid growth of the LSH object.

dataset of 18,727,618 unique reactions was reduced with both 30 permutations and 128 permutations, results of which are given in table 3.3. From here on

Table 3.3: Results and wall times of the Reactlake LSH reduction for both 30 and 128 permutations, together with the specifications of the machines.

Permutations	30	128
Cores	16	32
Clock Speed	2.50 GHz	2.30 GHz
Memory	61.3 GB	956.25 GB
Wall time	02h52m43s	19h23m04s
Near-duplicates	2,546,311	2,556,514

out, the reduced Reactlake dataset refers to the 16,171,104 reactions remaining after performing the MinHash LSH algorithm on the product vectors with 128 permutations, a threshold of 0.5 and the metrics as described in section 3.1. An overview of the distributions of the parent databases for the original and reduced Reactlake sets are given in table 3.4. However, it must be noted that the order in which the datasets are concatenated prior to reduction plays a role in the

Table 3.4: Size of the original Reactlake subsets before and after MinHash LSH near-duplicate reduction.

Subset	Original Entries	Reduced Entries
USPTO	1,387,365	1,302,985 (93.9%)
Reaxys	15,524,697	13,634,854 (87.8%)
Pistachio	1,815,556	1,233,265 (67.9%)
Total	18,727,618	16,171,104 (86.3%)

reduction percentage of each individual dataset. Each candidate is held against the dynamic list of retained reactions. A reaction at the end of the dataset is therefore compared against millions of others and will have a higher chance of colliding with a near-duplicate than that same reaction at the start of the dataset. This effect is also visible when reducing the initial Reactlake data by filtering out the exact unique values within and between datasets. This is why only less than 2 million reactions were considered from the Pistachio dataset, out of an original size of more than 4 million. Table 3.5 provides a decomposition of the near-duplicate collisions per individual dataset. With USPTO the first dataset in Reactlake, its observations are only compared internally, while Pistachio's observations are compared against both retained USPTO and Reaxys data.

Table 3.5: Decomposition of the near-duplicate reduction results per individual dataset. Each cell provides the number of reactions that were removed from the dataset designated by the row label during the comparison against the retained reaction observations from the dataset designated by the column label.

	USPTO	Reaxys	Pistachio	Total
USPTO	84,380			84,380
Reaxys	572,876	1,316,967		1,889,843
Pistachio	435,146	7,722	139,423	582,291

3.3 Molecular Transformer

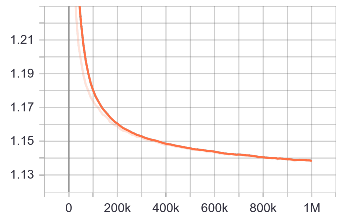
An initial attempt was made to reproduce the results from Schwaller et al. [17] on the USPTO_MIT subset. In the original research paper, results were reported on several incarnations of the model, both on the original and augmented SMILES data. The baseline model on the original data was performed in 500,000 learning steps with a batch size of 4,096 and a validation batch size of 32. The model was further enhanced by averaging the last 20 checkpoints, with each checkpoint generated at 10,000 learning steps. Finally, several ensemble models were tested, one of which with averaged checkpoints. In this work, we recreated the baseline single model on the non-augmented MIT subset data. The same data and the same splits were used as in the original work. For this model, 300k learning steps were used and the original train/test/validation splits (409,035/40,000/30,000) were retained. The model was trained on a single GPU (NVIDIA Tesla V100-SXM2 16GB) with the batch size set to 13,000 and the validation batch size left at the default of 32. The final five checkpoints were averaged and top-1 to top-3 scores were calculated on a single run. The results are presented in table 3.6. While the training length was reduced and a different checkpoint average was

Table 3.6: Reproduction of the single baseline model trained on the MIT subset.

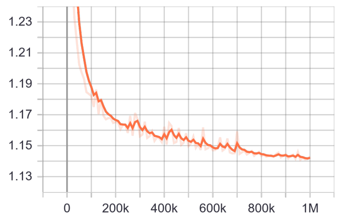
	Top-1 %	Top-2 %	Top-3 %
IBM	88.6	92.4	93.5
Reproduced	88.4	91.4	92.2

used, the results are fairly similar and confirm the reported results. Only one training run was used, so some variability in the model is present as well, due to the random initialization of the model weights.

The same approach was used for the full and reduced Reactlake datasets. Due to the sheer size of these datasets, splits of 40,000 were selected both for the test and validation sets. The remainder of the data was used in training. The model was trained on a single GPU. The training batch size was reduced to 2,000 and the validation batch size to 8 to accommodate GPU memory requirements. All other model parameters were kept the same. To reduce variability due to random training weight initiation and test selection, three different runs were performed on different splits, both for the full and reduced Reactlake datasets. The validation loss was monitored in TensorBoard. The perplexity of two training runs, one for the full and one for the reduced Reactlake set, is given in figure 3.6, both with a smoothing factor of 0.6. Based on these plots, it can be argued that



(a)



(b)

Figure 3.6: The perplexity of the validation steps plotted against the learning steps for a single training run for (a) the full and (b) the reduced model. Perplexity is a metric for the average validation loss.

the loss functions have not yet reached their minimums and additional training runs may still provide increased model performance. However, since 1 million learning steps took more than 64 hours to complete on a single GPU, it was decided not to increase the learning steps any further and the current model output can be considered representative for the conclusions to be made in this work. The full results of the test scores are given in table 3.7. Because no specific subset of Reactlake was used, the results are compared against the single model on the mixed stereochemistry included USPTO subset in the research paper by Schwaller et al.

Table 3.7: Test set accuracies for the three model training runs for both the full and reduced Reactlake data, compared against the single model accuracy of the Molecular Transformer on the mixed stereochemistry USPTO subset.

		Top-1 %	Top-2 %	Top-3 %
Full	Run 1	45.1	49.2	52.0
	Run 2	45.0	49.1	51.9
	Run 3	44.4	48.5	51.2
	Average	**44.8**	**48.9**	**51.7**
Reduced	Run 1	42.7	46.9	49.7
	Run 2	42.6	46.8	49.6
	Run 3	43.2	47.5	50.4
	Average	**42.8**	**47.1**	**49.9**
	IBM	76.2	82.4	84.3

As is clear from these results, the Molecular Transformer model, with this set of parameters, is not capable of providing the same accuracy on the Reactlake data as on the USPTO data by itself. A breakdown of the origin of the different reactions in the test set is given in table 3.8.

Table 3.8: A breakdown of the average distribution of the origin of the reactions in the test set and their respective translation accuracies.

		Amount	Top-1 %	Top-2 %	Top-3 %
Full	Pistachio	3,906.7 (9.8%)	53.1	56.1	58.2
	USPTO	2,951.0 (7.4%)	67.0	70.5	72.7
	Reaxys	33,142.3 (82.9%)	41.9	46.2	49.1
Reduced	Pistachio	3,087.7 (7.7%)	45.0	48.3	50.5
	USPTO	3,188.0 (8.0%)	62.7	66.7	69.2
	Reaxys	33,724.3 (84.3%)	40.8	45.1	48.0
IBM	USPTO	50,258	76.2	82.4	84.3

This clearly shows that the largest part of the test reaction set originates from Reaxys, which is in line with the overall distribution in Reactlake. Because of this, the overall accuracy results are strongly dominated by the translation accuracy on the Reaxys samples in the test set, which has the lowest of all three dataset accuracies. The conditional accuracy on the USPTO dataset is by far the largest of the three and is closer to the reported result by Schwaller et al. Deviations might in part be due to underfitting given that the loss function had not yet reached a minimum at 1 million learning steps. Second, these results could be a product of differences in grammar between the reactions in the three individual datasets, hence the varying conditional accuracy results in the test set. The difference in grammar could be due to the reporting hueristics used for the three different sets. A simple example is given when looking at the reaction product distribution. In the USPTO set, only 3.3% of the reactions have more than a single reported product. This increases to 5.4% for Pistachio and 16.5% for Reaxys, signaling an increasing translation complexity, corresponding to the lower prediction efficiency. With Reaxys being the dominant data source during training, it could be that the resulting model is less performant on the USPTO data and, in general, that combining data from different sources could lead to lower conditional accuracies than training them separately.

A result that can be generalized among both the marginal and conditional results is the lower accuracy in the reduced Reactlake dataset. Overall, a reduction of 2% was obtained in the top-1 accuracy, and a drop of 1.8% in the top-2 and top-3 categories. Again, this marginal result is dominated by the score of the reactions in the Reaxys data, while the effect is larger for the Pistachio (8.1% top-1) and the USPTO (4.3%) data. However, looking back at table 3.4, it is important to note that the Pistachio dataset was reduced the most with only 67.9% of the original reactions retained, and therefore, the weight of Pistachio in the reduced Reactlake set will be smaller. Building on the prior assumption that the reaction grammar differs from one individual dataset to another, the decrease in the fraction of reactions from Pistachio in the reduced Reactlake set during training could be in part responsible for the drop in accuracy. The opposite effect can be in play for the Reaxys dataset, which can work to offset the accuracy reduction from removing near-duplicates from its data. Although the conditional results should not be viewed independently from the overall composition of the dataset, the decrease in test accuracy is common among all results and can therefore be generally attributed to the effect of removing near-duplicates from the overall sample a priori.

Chapter 4

Discussion and Outlook

In this work, the use of MinHash LSH was explored as an efficient mechanism to remove near-duplicate reactions from very large reaction datasets. To this end, data was combined from three different sources: The publicly available USPTO dataset and the proprietary sets Pistachio and Reaxys.

At first, an effort was made to determine a set of distance metrics to define near-duplicates among reactions. A random set of reaction comparisons within the USPTO dataset was generated and labeled as unique reactions. This set of reactions was augmented by artificial near-duplicate reactions, created by two different text-mining algorithms from the same USPTO subset. In an iterative process, a combination of Jaccard and Hamming distances on both the reactant and product vectors were selected and fine-tuned, until no false positive reactions were retrieved from the pairwise comparisons within a sample of the USPTO dataset. While this approach generated a set of rules considered to be acceptable for the outline of this work, future research could consider improvements via a more formal approach. Tree based machine learning methods could be trained on top of a supervised dataset to decide on the classification of near-duplicate and unique reactions. However, in the incarnation of the supervised reaction set as construed for this work, misclassified reactions led to an underwhelming performance of such classification techniques. It would prove useful to recreate a reaction set with both unique and near-duplicate reactions, annotated by experts in organic chemistry to improve these efforts.

To achieve a tractable runtime, MinHash LSH was succesfully implemented as a preliminary filter to reduce the number of comparison candidates when searching for and eliminating near-duplicate reactions in large datasets. Applying this algorithm to the product vector provided the best results in both processing time and accuracy versus the ground truth. Due to the intrinsic nature of MinHash LSH, large processing times required in a brute force comparison algorithm are traded for the requirement of sufficient memory to store the hashed reaction fingerprints. Although the memory requirements can be kept in check by reducing the number of permutations in the creation of the MinHash fingerprint, this negatively impacts both the accuracy of the estimated Jaccard similarity and the available flexibility for selecting the best combination of block size and amount. Using 128 permutations and an LSH query threshold of 0.5 on the product vector, a total of 2,556,514 reactions were removed from the initial 18,727,618 reactions with unique reaction SMILES present in the Reactlake set. While the exact processing time is dependent on the hardware available, the algorithm is shown to run in linear time complexity, providing proper scalability for increasingly large datasets. In combination with the proper metrics to measure distance between two reactions, this provides an elegant approach to cleaning reaction datasets and removing near-duplicate reactions.

Finally, the applicability of the Molecular Transformer, developed by Schwaller et al., was expanded from the USPTO data to the full Reactlake dataset. After having validated the outcome of the original research paper, the same model was trained on the concatenated USPTO, Pistachio and Reaxys reaction SMILES. In parallel, the model was also trained on the Reactlake subset where all near-duplicates where removed using the aforementioned approach. Accuracies of both models were tested on a 40,000 sample test set, split from the training data. Both the marginal accuracies and the accuracies conditional on the original reaction set were monitored. Due to the sheer size of the training data, minimization of the loss function was not entirely achieved after 1,000,000 learning steps. It was concluded from the results that the combination of the three different reaction datasets resulted in an inferior overall translation accuracy compared to using only the USPTO reaction set. While the reactions from the USPTO set were predicted with a higher precision than those from Pistachio and especially those from Reaxys, by far the largest of the three, the accuracies were not as high as those initially reported by Schwaller et al. It is hypothesized that the difference in the performance of the Molecular Transformer between the three reaction sets can be ascribed to differences in the composition of each of these sets. Further research can be aimed at elucidating whether this is due to intrinsic differences of the reaction families most common in the datasets or different reporting hueristics used. An example was given by exploring the distribution of the products across the reaction sets. It was shown that reactions from Reaxys were much more likely to contain more than a single reaction product as compared to

Pistachio and especially USPTO. Such differences in the constitution of reactions across a mixture of datasets will automatically lead to dilution of what can be considered the common grammar or chemical reaction heuristics and reduce the translation accuracy. Of interest for future research would be to train the Molecular Transformer model on each of Pistachio and Reaxys separately and compare the results to the conditional results from the model trained on the Reactlake dataset. It could also be explored if the parameters in the Molecular Transformer model can be further optimized specifically for the reaction set under analysis.

While the accuracy of the forward predictions could not be generalized over the three different subsets in the Reactlake data, a common result is the lower prediction average of the set where the near-duplicates where removed. Overall, a performance reduction of 2% on the top-1 accuracy was achieved. Importantly, the same effect was noticeable, but to different extents, on each of the subsets in Reactlake. The largest decrease in performance was noticed in the Pistachio subset, which was also the subset with the largest size reduction after near-duplicate removal. This could be an effect of the high number of near-duplicates present in the Pistachio subset, inflating the original accuracy numbers, but it could also be a result of the smaller weight of this subset in the training set, making it harder for the model to correctly predict test set reactions from this subset if differences between the subsets do indeed exist. However, even for USPTO and Reaxys, whose weights in the train, validation, and test sets increased compared to the original Reactlake set, a reduction in accuracy can be seen, showing the overall effect on prediction performance after removal of the near-duplicates from the reaction set.

This shows that the removal of near-duplicate reactions is an important first step in the preparation of any reaction dataset prior to model training, validation and testing. As noted before, future efforts could entail training of the Molecular Transformer model on the individual reaction subsets, with and without near-duplicates to measure the effect over the different reaction sets, and measure the correlation between the drop in predictive power and the number of near-duplicates within each set individually.

In conclusion, the results in this work bring to the attention the importance of a standardized approach of reporting chemical reactions for the success of future efforts in reaction prediction modelling.

Bibliography

[1] Boström, J., Brown, D., Young, R. et al. (2018) Expanding the medicinal chemistry synthetic toolbox. *Nat Rev Drug Discov* **17**, 709–727.

[2] Corey, E.J. (1967) General methods for the construction of complex molecules. *Pure Appl. Chem.* **14(1)**, 19-38.

[3] Engkvist, O., Norrby, P., Selmi, N., et al. (2018) Computational prediction of chemical reactions: current status and outlook. *Drug Discovery Today* **23(6)**, 1203-1218.

[4] Coley, C.W., Green, W.H., Jensen, K.F. (2018) Machine Learning in Computer-Aided Synthesis Planning. *Acc. Chem. Res.* **51**, 1281-1289.

[5] Szymkuć, S., Gajewska, E. P., Grzybowski, B. A, et al. (2016) Computer-Assisted Synthetic Planning: The End of the Beginning. *Angew. Chem. Int. Ed.* **55**, 5904-5937.

[6] Merck KGaA (2020) Synthia: One AI, That Can Save Years of Research [Online] Available at: https://www.merckgroup.com/en/research/science-space/envisioning-tomorrow/future-of-scientific-work/synthia.html (Accessed: 06 June 2021).

[7] Wang, L.P., McGibbon, R.T., Pande, V. et al. (2016) Automated Discovery and Refinement of Reactive Molecular Dynamics Pathways. *J. Chem. Theory Comput.* **12(2)**, 638-649

[8] Chaffey-Millar, H., Nikodem, A., Matveev, A.V. et al. (2012) Improving Upon String Methods for Transition State Discovery. *J. Chem. Theory Comput.* **8(2)**, 777-786.

[9] Hassabis, D., Silver, D., Schrittwieser, J et al. (2017) Mastering the game of Go without human knowledge. *Nature* **550** 354-359.

[10] Segler, M.H.S., Waller, M.P. (2017) Neural-Symbolic Machine Learning for Retrosynthesis and Reaction Prediction. *Chem. Eur. J.* **23(25)**, 5966-5971.

[11] Kayala, M.A., Azencott, C., Chen, J.H. et al. (2011) Learning to Predict Chemical Reactions. *J. Chem. Inf. Model.* **51(9)**, 2209-2222.

[12] Kayala, M.A., Baldi, P. (2012) ReactionPredictor: Prediction of Complex Chemical Reactions at the Mechanistic Level Using Machine Learning. *J. Chem. Inf. Model.* **52(10)**, 2526-2540.

[13] Coley, C.W., Jin, W., Rogers, L. et al. (2019) A graph-convolutional neural network model for the prediction of chemical reactivity. *Chem. Sci.* **10** 370-377.

[14] Vaswani, A., Shazeer, N., Parmar, N. et al. (2017) *Advances in Neural Information Processing Systems*, 5998-6008.

[15] Nam, J., Kim, J. (2016) Linking the Neural Machine Translation and the Prediction of Organic Chemistry Reactions. *arXiv* 1612.09529.

[16] Liu, B., Ramsundar, B., Kawthekar, P. et al. (2017) Retrosynthetic Reaction Prediction Using Neural Sequence-to-Sequence Models. *ACS Cent. Sci.* **3** 1103-1113.

[17] Schwaller, P., Gaudin, T., Lányi, D. et al. (2018) "Found in Translation": predicting outcomes of complex organic chemistry reactions using neural sequence-to-sequence models. *Chemical Science* **28**.

[18] Schwaller, P., Laino, T., Gaudin, T. et al. (2019) Molecular Transformer: A Model for Uncertainty-Calibrated Chemical Reaction Prediction. *ACS Cent. Sci.* **5(9)** 1572–1583.

[19] Su, S., Yang, Y., Gan, H. et al. (2020) Predicting the Feasibility of Copper(I)-Catalyzed Alkyne-Azide Cycloaddition Reactions Using a Recurrent Neural Network with a Self-Attention Mechanism. *J. Chem. Inf. Model.* **60** 1165-1174.

[20] Tetko, I.V., Karpov, V., Van Deursen, R. et al. (2020) State-of-the-art augmented NLP transformer models for direct and single-step retrosynthesis. *Nat Commun* **11** 5575-5585.

[21] Schwaller, P., Laino, T. (2019) Data-Driven Learning Systems for Chemical Reaction Prediction: An Analysis of Recent Approaches. *Machine Learning in Chemistry: Data-Driven Algorithms, Learning Systems, and Predictions.* ACS Publications, Washington, 61–79.

[22] Grethe, G., Goodman, J.M., Allen, C.H.G. (2013) International chemical identifier for reactions (RInChI). *Journal of Cheminformatics* **5(45)**

[23] Holliday, G., Murray-Rust, P., Rzepa, H.S. (2006) Chemical Markup, XML, and the World Wide Web. 6. CMLReact, an XML Vocabulary for Chemical Reactions. *J. Chem. Inf. Model.* **46** 145-157.

[24] Weininger, D. (1988) SMILES, a chemical language and information system. 1. Introduction to methodology and encoding rules. *J. Chem. Inf. Comput. Sci.* **28(1)** 31-36.

[25] O'Boyle, N., Dalke, A. (2018) DeepSMILES: An Adaptation of SMILES for Use in Machine-Learning of Chemical Structures. *ChemRxiv* https://doi.org/10.26434/chemrxiv.7097960.v1.

[26] Krenn, M., Häse, F., Nigam, A.K. et al. (2019) SELFIES: A Robust Representation of Semantically Constrained Graphs with an Example Application in Chemistry. *arXiv* 1905.13741.

[27] Lowe, D.M. (2012) Extraction of chemical structures and reactions from the literature. Ph.D. thesis. *University of Cambridge*

[28] Madzhidov, T., Lin, A.I., Nugmanov, R., et al. (2020) Atom-to-Atom Mapping: A Benchmarking Study of Popular Mapping Algorithms and Consensus Strategies. *ChemRxiv* Preprint. https://doi.org/10.26434/chemrxiv.13012679.v1

[29] Jin, W., Coley, C., Barzilay, R. et al. (2017) Predicting Organic Reaction Outcomes with Weisfeiler-Lehman Network. *arXiv* 1709.04555.

[30] Bradshaw, J., Kusner, M.J., Paige, B. et al. (2019) A Generative Model for Electron Paths. *arXiv: Chemical Physics* 1805.10970.

[31] Raccuglia, P., Elbert, K.C., Adler, P.D.F. et al. (2016) Machine-learning-assisted materials discovery using failed experiments. *Nature* **533** 73-76.

[32] Broder, A.Z. (1997) On the resemblance and containment of documents. *Proceedings. Compression and Complexity of SEQUENCES* 21-29.