Minhashing for Graph Similarity Computation

Can Güney Aksakalli[1] and Pascal Welke[2]

[1] RWTH Aachen University, Germany
can.aksakalli@rwth-aachen.de
[2] University of Bonn, Germany
welke@uni-bonn.de

Abstract

The similarity of graphs can be measured by the Jaccard distance based on shared subgraphs. Computing this distance measure for a large collection of graphs with large sets of substructures, however, takes too much time to be of practical relevance. In this work, we extend the work of Ralaivola Teixeira et al. [2012] and apply the Min-Hash technique introduced by Broder Broder [2000] to graph databases to approximate the Jaccard distance. We base our distance measure on paths and triangles. The implemented method is evaluated on a chemical dataset and the empirical evaluation shows promising results: Min-Hash based distance estimates can be computed faster and require less space than computing the exact Jaccard similarity.

1 Introduction

In the age of information explosion datasets become at the same time larger and more complex. This results in (at least) two challenges for practical algorithms: First, any approach with superlinear complexity in the number of objects in the dataset is not applicable. Second, the complexity for handling each object needs to be bounded. This renders even basic problems like distance computations between pairs of objects infeasible, when done naively.

Broder [2000] was one of the first scientists who tackled this problem in the history of the Internet, while he was working at AltaVista, an early search engine. In the web, there are some text documents with identical content but different URLs or intentionally copied down from another source. In order to rank pages correctly and filter out the search results with identical content, there was a need to filter out duplicated or near-duplicated content from crawled web pages. One approach of document distance relies on representing each document as the set of all its *w-shingles*. A w-shingle is a sequence of w consecutive characters in the document. The distance between two documents can then be defined as the Jaccard distance of their sets of shingles. Based on document size and the number of documents in the web, the size and the number of sets grow dramatically. It was not efficient to do this on all crawled documents in the web even during its early years. Eventually Broder et al. [1998] introduced an idea called *Min-Hash* that represents all documents as fixed size *sketches*. Instead of computing the intersection of two (possibly large) sets, the Jaccard distance can be *estimated* from these (small) sketches in a more scalable manner.

Broder [2000] proved that the proposed Min-Hash method works well for document deduplication. Teixeira et al. [2012] applied the Min-Hash idea to graph databases in order to find similar graphs. Graphs are a well suited model for many domains like chemoinformatics, the Web, linked data, and social networks. Finding similarities between graphs, however, is a highly demanding task. Even representing a graph as a set of all its substructures ("the shingling") becomes an intractable problem, as the number of non-isomorphic subgraphs of a given graph can be exponential. Once obtained, comparing the sets of these shingles for two graphs is difficult as well: the graph isomorphism problem is not known to be solvable in polynomial

time. Texeira et al. therefore use paths up to length 10, which can be extracted from graphs in polynomial time and efficiently be compared. They evaluate their method on seven small molecular datasets (containing up to 3400 molecules).

In this work, we implement the approach of Texeira et al. and extend it with a novel shingling method. We then evaluate the approach on a larger molecular dataset consisting of more than 32 000 molecules. We show that this method can be used in drug discovery to retrieve similar molecules for a given molecule, as well as for classification. Both tasks are important primitives in "virtual screening for drug discovery". Testing properties of molecules in a lab is expensive and takes a lot of time. Millions of chemical compounds are available and cannot all be tested for a certain chemical property. One would rather only test a few molecules in the lab and then compute a list of the most promising candidate compounds to be tested next.

2 Related Work

The similarity of pairs of graphs can be measured by *graph kernels*. The kernel method in machine learning refers to operating in high-dimensional space without mapping the input explicitly to a higher dimensional space. Likewise, graph kernels allow us to work on nonlinear structure of graphs by implicitly mapping them. There has been a lot of research carried out in this field, and over the years some graph kernels have been developed as a result of this research. Vishwanathan and Smola [2003] have studied *tree kernels* and used them to count shared subtrees between graphs. Horváth et al. [2004] proposed *cyclic pattern kernels*, which count common occurrences of cycles and trees between two graphs. However, these kernels miss important information of simple paths. We used *moleculer fingerprinting* from Ralaivola et al. [2005] which is a kind of spectral kernel derived from simple walks on graphs.

Broder et al. [1998] introduced the MinHash method. Teixeira et al. [2012] combined MinHash method with graph kernels. However, they demonstrated and evaluated this approach on unweighted graph data in cheminformatics domain which unifies all types of chemical bounds. The type of the chemical bound has a significant role on the behavior of chemical components according to Rang et al. [2012]. We also tried to answer the question: How likely the type of chemical bounds would effect the molecular similarity. Thus we worked on both weighted and unweighted graphs.

3 Graph Minhashing

The flow of graph minhashing method is shown with a toy problem in Figure 1. The method has three steps. First, subpaths are extracted from two given graphs. Then these subpaths are mapped to a scalar value by fingerprinting. Now each graph is represented as a set of derived fingerprints. Thus the problem is reduced to set intersection. For the two graphs G_A and G_B, there exist sets of fingerprints S_A and S_B. The Jaccard similarity of these two sets can determine the similarity of graphs. *The resemblance* $r(A, B)$ is defined as

$$r(A, B) = \frac{|S_A \cap S_B|}{|S_A \cup S_B|} \tag{1}$$

However, due to the size of fingerprint sets (depending on size of distinct substructures in graphs, more elaborated in Experimental Results), it is not efficient to compute the actual similarity. Therefore minhashing method is applied to get sketches from the sets. *The sketch* (explained in Introduction section) $\overline{S}_A$ is defined as the result of hash function: $\overline{S}_A = h(S_A)$.

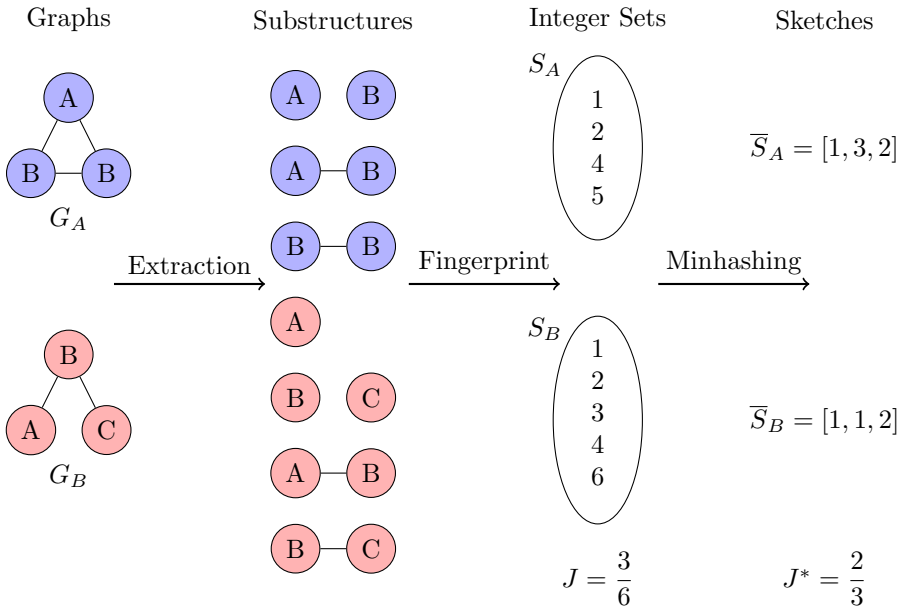

Figure 1: The flow of graph minhashing method

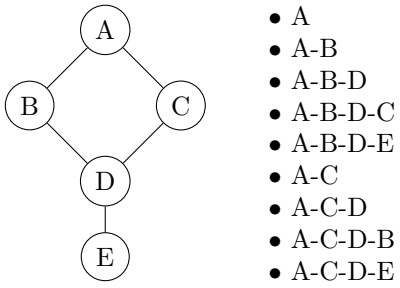

Figure 2: Example of DFS path extraction with depth $d = 4$

Now we end up with two sketches $\overline{S}_A$ and $\overline{S}_B$ as a result of minhashing. The number of similar components in these sketches can give an approximate result. In the toy problem in Figure 1, the actual Jaccard similarity of two sets J is slightly different from the approximated distance J^*. The details of the steps are explained in each regarding subsection.

3.1 Substructure Extraction

For document deduplication, the substructures are extracted by *w-shingling*. A contiguous subsequence of words in a text document are defined as *shingle* and size of these chunks as w. The document "A rose is a rose is a rose.", for example, results in the following set of 4-shingles:

$$\{(a, rose, is, a), (rose, is, a, rose), (is, a, rose, is)\} \tag{2}$$

Suppose we now have a graph that is a path. We could apply the same idea and represent

it as a set of its subpaths of a fixed length. If we consider the document above as a path with 8 vertices (the first vertex is labeled "A", the second vertex is labeled "rose", ...) and extract the subpaths of length 4, then we end up with an exactly same looking set of shingles (which are now paths, not substrings.) This approach can be generalized to graphs that are more complex than paths. We used *depth-first search* to extract all paths up to a certain length, as proposed by Ralaivola et al. [2005]. It is a slightly modified DFS algorithm which traverses all possible branches and extracts paths consisting of labeled vertices and edges. Figure 2 demonstrates how this method works on a toy example. Unlike the *w-shingling* method, it extracts vertex chains of all ranges of sizes up to a depth limit d from a graph (in practice for chemical molecules $d = 10$). In the recursive implementation of this search algorithm, the call stack will be terminated when it reaches depth limit d. We also tried fixed size sketches but it does not give any considerable results for molecule databases. The search is repeated starting from each vertex in order to extract all available paths up to size d.

As an alternative, we propose a new method of shingling for graphs based on triangles. A *triangle* is a subgraph on three vertices that contains at least two edges. Hence, there are two kinds of triangles: Open triangles with two edges, and closed triangles with three edges. All triangles in a graph can be computed efficiently: Given a vertex v and two of its neighbors x and y, we have to check if there is an edge from x to y to decide if the triangle is open or closed. It is easy to see that we find all triangles at least once if we iterate over all pairs of neighbors for each vertex. To be compatible with the further discussion, we can transform a triangle into a path in a canonical way (i.e. isomorphic triangles will be mapped to the same path).

3.2 Fingerprinting

The substructure extraction method returns paths. In the case of molecular graphs, these paths consist of labeled vertices and labeled edges. These paths need to be mapped to an integer value. We first transform the (possibly) alphanumeric labels of vertices and edges to integer label codes. For a path $P = (v_1, e_{12}, v_2, e_{23}, \ldots, e_{(c-1)c}, v_c)$, with label codes $L(v_i)$ for vertices, and $L(e_{ik})$ for edges, and the prime number P (in practice $P = 31$), the mapping function (for weighted graphs) is

$$integer([v_1, v_2 ... v_c]) = (\ldots(((P + L(v_1))P + L(e_{12}))P + L(v_2))P + \ldots)P + L(v_c) \quad (3)$$

It is apparent from the equation that the result can be overflowed by reaching the size limit of integer for the platform. However, it will always map these same substructures to the same scalar values. We used 32-bit integers for storing fingerprints. It can be expanded in order to reduce the probability of collision. As an alternative, we also experimented with a fingerprinting method that does not consider the edge codes (for unweighted graphs), i.e.

$$integer([v_1, v_2 ... v_c]) = (\ldots((P + L(v_1))P + L(v_2))P + \ldots)P + L(v_c) \quad (4)$$

3.3 Minhashing

After fingerprinting, the problem is reduced to set intersection. In order to approximate the Jaccard similarity of two sets it is sufficient to get relatively small, fixed size *sketches* for each set. *The sketches* can be computed fairly quickly in linear time. As proved by Broder et al. [1998], let π be chosen to represent a random permutation function which permutes the 32-bit integers. Then

$$Pr(min\{\pi(S_A)\} = min\{\pi(S_B)\}) = \frac{|S_A \cap S_B|}{|S_A \cup S_B|} = r(A, B) \tag{5}$$

Since π is chosen uniformly, any integer value of the range has the same chance to be the minimum after permutation. Intersection of two sets increases the probability of getting the same element as a result of MinHash. Thus we can choose a set t random permutations $\pi_1, ..., \pi_t$ and store a sketch value for each sets, a list of size t

$$\overline{S}_A = (min\{\pi_1(S_A)\}, min\{\pi_2(S_A)\}, ..., min\{\pi_t(S_A)\}) \tag{6}$$

If we consider the toy problem in Figure 1, the sketches are derived from π_1, π_2, π_3 in Table 1. π permutes values of two sets and minimum value is picked (as the box is filled with color of the graph). The resemblance is estimated by comparing results of hash functions. The approximate resemblance of A and B is the rate of corresponding equal elements in $\overline{S}_A$ and $\overline{S}_B$. Since this is probabilistic approximation of intersection, using more permutation functions tends to give a better approximation. In experimental results, we tried different values for size t and determine a compact sketch size that yields a good approximation quality for our data.

In practice, it is impossible to choose an uniform permutation function π for 32-bit integers. So we used a smaller set of permutation functions. First, a pseudorandom integer list (*hashFunctions* variable in the Java method) is generated when the application starts. MinHash method gets minimum of XOR (exclusive or) of given set and respected permutation integer (the criteria of min in the implemented method). Thus each bit of the fingerprints is implicitly permuted by respected pseudorandom integer. However, this permutation is not uniform function over all integer values because there exist some permutations which can not be achieved by binary XOR operation of an integer. Since all fingerprints have the same probability to be selected as minimum after this permutation, it still satisfies *min-wise independence condition*. Hence the Java method for minhashing which returns the sketch:

```
public List<Integer> minhash(Set<Integer> fingerprintSet) {
  return hashFunctions.stream()
          .map(h -> fingerprintSet.stream()
            .min(Comparator.comparing(i -> i ^ h)).get()
          )
          .collect(Collectors.toList());
}
```

After computing sketches, we have compact representation of graphs and distance metric between graphs. It is possible to find closest items by querying an input graph. The implementation is completely working on the RAM and searching all elements in linear time.

4 Experimental Results

In this section, we evaluate the quality of min-hash based distance estimates on a large dataset of chemical molecules. We first investigate the error that we incur by using a sketch based similarity instead of the actual Jaccard similarity. In the following two steps, we investigate the feasibility of sketch based distances to answer two common tasks in computational chemistry: First, we are interested in an information retrieval task, namely fetching the most similar

		1	2	3	4	5	6	7
h_1	π_1	1	2	3	4	5	6	7
	S_A	1	1	0	1	1	0	0
	S_B	1	1	1	1	0	1	0
h_2	π_2	3	7	1	6	2	5	4
	S_A	0	0	1	0	1	1	1
	S_B	1	0	1	1	1	0	1
h_3	π_3	7	4	3	6	1	2	5
	S_A	0	1	0	0	1	1	1
	S_B	0	1	1	1	1	1	0

Table 1: Example of minhashing for the toy example in Figure 1.

molecules for a given molecule in the database. Second, we try to classify the molecules into two classes using a k-Nearest Neighbor classifier.

We implemented the min-hash scheme and Jaccard similarity computation in Java. Our experiments were carried out under JVM 1.8 installed on a desktop PC with an Intel Core i5-3230M CPU (2.60GHz), and 8 GB RAM, running Ubuntu 14.04. The fingerprints, as well as min-hash sketch values are stored as integer data type of Java which is 32-bit.

We used maximal path length $d = 10$ for the DFS path extraction as suggested by Ralaivola et al. [2005]. The average size of extracted fingerprint sets is 613.14 for unweighted paths and 1534.31 for weighted paths. Using weighted paths (i.e., where edge labels are considered) generates larger fingerprint sets, as was to be expected. We experimented a bit with increased path lengths, but it made almost no difference in our quality measures while increasing the runtime dramatically. Preprocessing runtime of sketches size $t = 64$ for 42,687 molecules is T00:06:12.603 for unweighted and T00:50:43.983 for weighted. Since the weighted edge extraction operates on path chains with edges and more number of fingerprints, it takes more time.

Nevertheless, it takes more time than the expected runtime regarding to linear complexity. The reason is that, the routine of altering vertex chains into a chain with edges takes more time due to the graph data structure which we use[1]. The library iterates though all edges to get the edge between two given vertices.

We evaluated our implementation on the AIDS dataset provided by the National Cancer Institute [2]. It contains screening results of 42,687 molecules against HIV, 422 of them are labeled as "active", 1,081 are "moderately active", while 41,184 were "inactive". For our experiments, we considered the "moderately active" molecules to be inactive, as well. The average number of atoms (vertices) per molecule is 45.70 and the average number of bounds (edges) is 47.71.

4.1 Jaccard Similarity vs MinHash Similarity

In theory, graph minhashing method allows us to compute a compact sketch for finding similar graphs while still being able to compute similarities between pairs of graphs efficiently. In practice, however, transforming the sets of substructures that are the input of the Jaccard similarity function first to sets of fingerprints and then to min-hash sketches might result in a large error, while saving some space. Hence, we look at the mean squared error (MSE) of a sample of random 1000 pairs for MinHash estimation (it is really expensive to experiment

[1] JGraphT - a free Java Graph Library `http://jgrapht.org/`

[2] `http://cactus.nci.nih.gov`

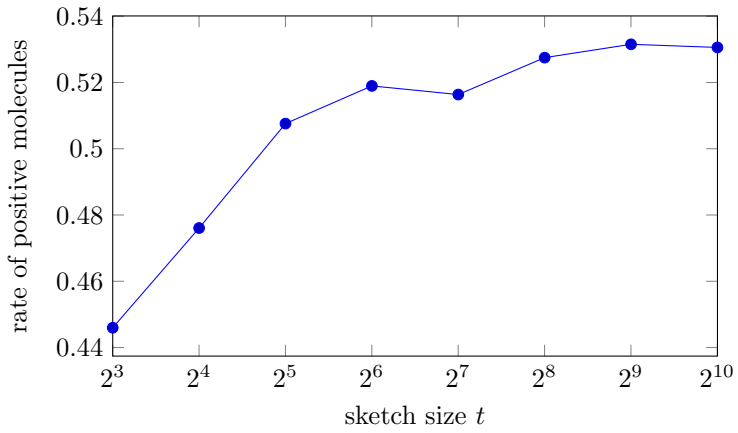

Figure 3: Precision at k=10 for different sketch sizes t (unweighted graph fingerprinting)

on all pairs due to time complexity of computing Jaccard Similarity of large sets). The mean of actual similarity between pairs computed by Jaccard similarity is 0.12461. For sketch size $t = 64$ MSE is 0.01073 and for $t = 128$ it is 0.01051. We see that the error is smaller for larger sketch sizes.

Using MinHash sketches allows us to save on storage. For instance, we would store only 11 MB of sketches ($t = 64$) instead of 105 MB of (unweighted) fingerprints.

4.2 Positive Instance Retrieval - Precision at k

A common task in chemoinformatics is retrieving similar molecules for a given molecule. The reasoning behind this is that "similar" molecules tend to behave similarly. As noted before, our dataset is highly imbalanced. Less than 1% of all molecules are labeled as "active". Hence, a random subset of molecules from that data is expected to contain just 1% of active molecules.

We computed sketches from unweighted paths and retrieved closest items from all molecules by querying 422 positive molecules (the element being queried is not excluded). The count of positive elements in first 10 closest results is averaged over all positive molecules (precision at k=10) for different sketch sizes t. The results are shown in Figure 3. It can be seen directly from the figure that, the fraction of retrieved positive molecules is significantly higher than the expected 1% for all sketch sizes.

Since it is a probabilistic approximation, a higher number of sketches would not necessarily give better approximation, but decreases the probability of error. For instance, sketch size 2^6 gives better result than 2^7 in this experiment. As Broder et al. [1998] proved, the approximation quality does not increase after a certain sketch size. Thus we conduct the rest of the experiments with fixed sketch size $t = 2^6 = 64$.

Besides average of 10 results, we also investigated the precision at $k = 1$ to 100 closest items of each positive molecule. The results can be seen in Figure 4. Again the molecule being queried is not excluded from the dataset. The average for the first retrieved element is not equal to one. That demonstrates that, at least 8% of active molecules are collided with a passive molecule.

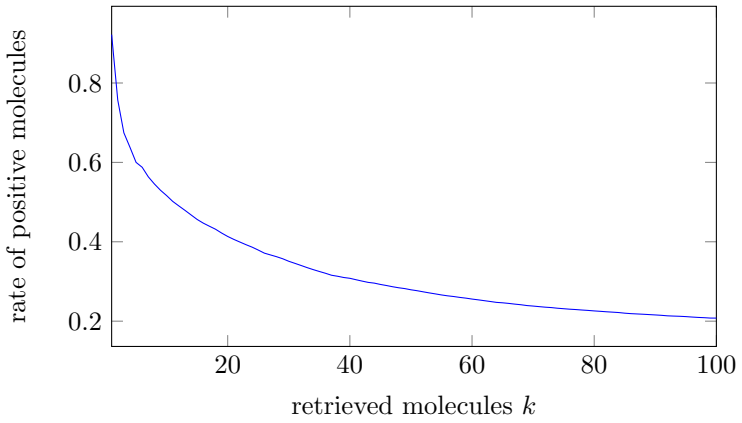

Figure 4: Precision at k from 1 to 100. (sketch sizes $t = 64$, unweighted graph fingerprinting)

		Actual	
		Positive	Negative
Predicted	Positive	216	149
	Negative	206	42116
	ACC= 0.991	TPR= 0.511	TNR= 0.995

Table 2: The confusion matrix for k-NN classifier, $k = 3$, sketch size $t = 64$, unweighted

4.3 Classification Accuracy

We used k-Nearest Neighbor classifier to evaluate the quality difference between weighted and unweighted fingerprints for $k = 3$. Leave one out cross validation is performed for all molecules with k-NN classifier. We present the confusion matrix in Table 2 (unweighted) and Table 3 (weighted). Since the dataset is imbalanced, accuracy (ACC) might be misleading. However, the results for true positive rate (TPR) are still promising. It also shows that, taking weighted edges into account is not significantly effecting the end result for this settings. However, it might be the case for another domain. Furthermore, according to Yvonne C. Martin and Traphagen [2002], structurally similar molecules do not necessarily behave similarly.

We also experimented with the proposed triangle based extraction method. We present the confusion matrix in Table 4. Although it extracts only tree vertices as substructure unlike other large number of paths, it still gives promising results.

		Actual	
		Positive	Negative
Predicted	Positive	213	160
	Negative	209	42105
	ACC= 0.991	TPR= 0.504	TNR= 0.996

Table 3: The confusion matrix for k-NN classifier, $k = 3$, sketch size $t = 64$, weighted

		Actual	
		Positive	Negative
Predicted	Positive	118	120
	Negative	304	42143
		ACC= 0.990	TPR= 0.279	TNR= 0.997

Table 4: The confusion matrix for k-NN classifier, $k = 3$, sketch size $t = 64$, triangle extraction

5 Conclusion

We applied the idea of minhashing that was originally introduced for text documents to graph databases. Using this technique, we were able to efficiently approximate Jaccard similarities based on sets of substructures of graphs. The substructures that we extracted include paths and triangles of vertices. This allowed us to solve two common tasks in computational chemistry, namely similar molecule retrieval and classification. The empirical evaluation on a chemical dataset shows promising results.

We implemented our code in Java and released it under MIT license on GitHub[3]. We invite interested users to contribute to this project or use it for their own benefit. Future work might include a database backend and further substructure extraction methods. The molecule information and preprocessed sketch could then be stored into a DBMS and the retrieval process can be improved for large graph datasets.

References

Andrei Z. Broder. Identifying and filtering near-duplicate documents. In *Proceedings of the 11th Annual Symposium on Combinatorial Pattern Matching*, COM '00, pages 1–10, London, UK, UK, 2000. Springer-Verlag. ISBN 3-540-67633-3. URL `http://dl.acm.org/citation.cfm?id=647819.736184`.

Andrei Z. Broder, Moses Charikar, Alan M. Frieze, and Michael Mitzenmacher. Min-wise independent permutations (extended abstract). In *Proceedings of the Thirtieth Annual ACM Symposium on Theory of Computing*, STOC '98, pages 327–336, New York, NY, USA, 1998. ACM. ISBN 0-89791-962-9. doi: 10.1145/276698.276781. URL `http://doi.acm.org/10.1145/276698.276781`.

Tamás Horváth, Thomas Gärtner, and Stefan Wrobel. Cyclic pattern kernels for predictive graph mining. In *Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '04, pages 158–167, New York, NY, USA, 2004. ACM. ISBN 1-58113-888-1. doi: 10.1145/1014052.1014072. URL `http://doi.acm.org/10.1145/1014052.1014072`.

Liva Ralaivola, Sanjay J. Swamidass, Hiroto Saigo, and Pierre Baldi. Graph kernels for chemical informatics. *Neural Networks*, 18(8):1093 – 1110, 2005. ISSN 0893-6080. doi: http://dx.doi.org/10.1016/j.neunet.2005.07.009. URL `http://www.sciencedirect.com/science/article/pii/S0893608005001693`. Neural Networks and Kernel Methods for Structured Domains.

[3]Graph Min-Hash `https://github.com/aksakalli/graph-min-hash`

H. P. Rang, M. M. Dale, J. M. Ritter, R. J. Flower, and G. Henderson. *Rang and Dale's Pharmacology*, chapter 3. How drugs act: molecular aspects. Edinburgh ; New York : Churchill Livingstone, 7th ed edition, 2012.

Carlos H. C. Teixeira, Arlei Silva, and Wagner Meira Jr. Min-hash fingerprints for graph kernels: A trade-off among accuracy, efficiency, and compression. *Journal of Information and Data Management*, 3(3):227–242, 2012. URL `http://seer.lcc.ufmg.br/index.php/jidm/article/view/199`.

S. V. N. Vishwanathan and Alex Smola. Fast Kernels for String and Tree Matching. *Advances in Neural Information Processing Systems*, 15, 2003. URL `http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.2.7163`.

James L. Kofron Yvonne C. Martin and Linda M. Traphagen. Do structurally similar molecules have similar biological activity? *Journal of Medicinal Chemistry*, 45(19):4350–4358, 2002. doi: 10.1021/jm020155c. URL `http://dx.doi.org/10.1021/jm020155c`. PMID: 12213076.