Small molecule machine learning: All models are wrong, some may not even be useful

Fleming Kretschmer[1], Jan Seipp[2], Marcus Ludwig[1,3], Gunnar W. Klau[2], and Sebastian Böcker[1]

[1] Chair for Bioinformatics, Institute for Computer Science, Friedrich Schiller University Jena, Jena, Germany
[2] Algorithmic Bioinformatics, Institute for Computer Science, Heinrich Heine University Düsseldorf, Germany
[3] Currently at Bright Giant, Jena, Germany

Abstract. Small molecule machine learning tries to predict chemical, biochemical or biological properties from the structure of a molecule. Applications include prediction of toxicity, ligand binding or retention time. A recent trend is to develop end-to-end models that avoid the explicit integration of domain knowledge via inductive bias. A central assumption in doing so, is that there is no coverage bias in the training and evaluation data, meaning that these data are a representative subset of the true distribution we want to learn. Usually, the domain of applicability is neither considered nor analyzed for such large-scale end-to-end models.
Here, we investigate how well certain large-scale datasets from the field cover the space of all known biomolecular structures. Investigation of coverage requires a sensible distance measure between molecular structures. We use a well-known distance measure based on solving the Maximum Common Edge Subgraph (MCES) problem, which agrees well with the chemical intuition of similarity between compounds. Unfortunately, this computational problem is provably hard, severely restricting the use of the corresponding distance measure in large-scale studies. We introduce an exact approach that combines Integer Linear Programming and intricate heuristic bounds to ensure efficient computations and dependable results.
We find that several large-scale datasets frequently used in this domain of machine learning are far from a uniform coverage of known biomolecular structures. This severely confines the predictive power of models trained on this data. Next, we propose two further approaches to check if a training dataset differs substantially from the distribution of known biomolecular structures. On the positive side, our methods may allow creators of large-scale datasets to identify regions in molecular structure space where it is advisable to provide additional training data.

Introduction

Machine learning has been successfully used in biochemistry and chemistry for decades. We consider the task of predicting chemical, biochemical or biological properties of small molecules of biological interest from their molecular structure. A recent trend is to develop end-to-end models that avoid the explicit integration of domain knowledge via inductive bias[1]. Noteworthy examples are generative models for novel antibiotics[2] and highly toxic small molecules[3], or classifiers for antibiotic activity[4,5], olfactory perception[6] and enzyme-substrate prediction[7]. The MoleculeNet paper from 2018 presents 17 medium- to large-scale datasets for molecular property prediction[8]. These data are frequently used in machine learning to train and evaluate new models such as graph neural networks and graphormers; notably, the paper has received more than 2,000 citations in five years.

The fact that one should not use a model outside of its domain of applicability, has been well-known in the chemometrics community. The situation may be compared to "spatial bias", where one uses test (and training) data from a certain geographic location, but makes claims about a model's performance for other geographic location as well[9]. Yet, this problem is usually ignored when training large-scale end-to-end models for predicting biochemical properties. Recently, words of warnings have emerged that machine learning may result in a "reproducibility crisis" in science[9,10]. In particular, the datasets from MoleculeNet have been criticized[11,12]. Whereas it is comparatively simple to train a machine learning model that performs well in evaluations, it is much harder to derive a model that indeed contributes to solving the underlying question.

To ensure that a model is not used outside of its domain of applicability, we have to make sure that the data available for training *and evaluating* the model are a representative subset of the space of all molecules of interest. It is understood that a dataset that is "too small" will not allow us to learn all aspects of the problem approached. Yet, even for datasets that contain many thousands of samples, the choice of small molecules included in any such dataset is often far from random. For datasets that rely on experimental measurements, it is instead governed by availability of compounds and, hence, often

by monetary aspects. The availability of a compound depends on aspects such as difficulty of chemical synthesis, commercial availability of precursor compounds, and similar considerations from synthetic chemistry and biotechnology. The lower the availability of a compound, the higher the price, and the less likely this compound can be found in large-scale datasets. Clearly, this introduces bias into the training data. For datasets with experimental measurements, unavailability of certain compounds is not going to change in the near future.

To consider the training data distribution for small molecules necessitates some way of estimating the similarity or dissimilarity between molecular structures. Unfortunately, this is a highly intricate problem, and is currently being approached in two ways with individual shortcomings: Firstly, molecular fingerprints allow for a swift processing of large datasets[13]. Yet, measures based on molecular fingerprints are known to exhibit undesirable characteristics[14–22]. In particular, measured distances may differ substantially from chemical intuition. Second, methods based on computing the Maximum Common (Edge) Subgraph better capture the chemical intuition of structural similarity[23–25], see below, but unfortunately, require to solve computationally hard problems.

Results

Here, we show how to inspect a molecular structure dataset for its coverage of biomolecular structures. Our approach combines Uniform Manifold Approximation and Projection (UMAP) embeddings[26] and computation of structural distance via the Maximum Common Edge Subgraph (MCES). We introduce the *myopic MCES distance*, which is the informative exact MCES distance for closely related molecules or a good approximation thereof, in case an exact computation is not required and too costly. We demonstrate how we can compute this distance swiftly in practice using a combination of fast lower bounds and integer linear programming. We then show that the distribution of compound classes, as well as a measure for natural product-likeness[27,28] can give good indications on whether the distribution of molecular structures in a dataset differs substantially from that of biomolecular structures. Finally, we shortly discuss shortcomings of the well-known Tanimoto coefficient for performing this type of analysis.

Distribution of biomolecular structures

It is understood that we do not know the true "universe of small molecules of biological interest", as this includes small molecules yet to be discovered[29]. Here, we use a combination of 14 molecular structure databases (*biomolecular structures* for short) as a proxy of this space. These databases contain metabolites, drugs, toxins and other small molecules of biological interest. As used here, the union of databases contains 718,097 biomolecular structures, see the Methods section and Supplementary Table 2 for details. Clearly, this proxy is and will be incomplete; yet, restrictions on the domain of applicability may already be visible against this proxy.

Given a pair of molecular structures, we computed a distance using their Maximum Common Edge Subgraph. To speed up computations, we estimated (provably correct) lower bounds of all distances. We performed exact computations only if the distance bound is at most a chosen distance threshold, which we set to 10, unless stated otherwise. If the lower bound was above the threshold, we used this bound instead as a distance estimate. Similarly, if the exact distance was computed and above the threshold, we used the threshold instead. We used UMAP[26] to visualize the "universe" of biomolecular structures in a 2-dimensional plot. Clearly, any other method for projecting high-dimensional data given as distances (for instance, t-SNE[30], multidimensional scaling[31] or Minimum Spanning Trees) can also be applied (Supplementary Fig. 13).

To avoid both proliferating running times and cluttered plots, we subsampled 20,000 biomolecular structures (Supplementary Fig. 6). Total running time for MCES computations was about 15.5 days on a 40 core processor. To monitor the effect of subsampling, we uniformly subsubsampled nine times 10,000 molecular structures from this set. We present corresponding UMAP embeddings in Supplementary Fig. 7, to reveal variations. We observe that subsampling may indeed change the general layout of the UMAP embedding, but that the general layout is often surprisingly similar. It is well-known that these UMAP embeddings have to be interpreted with care[32], see also below.

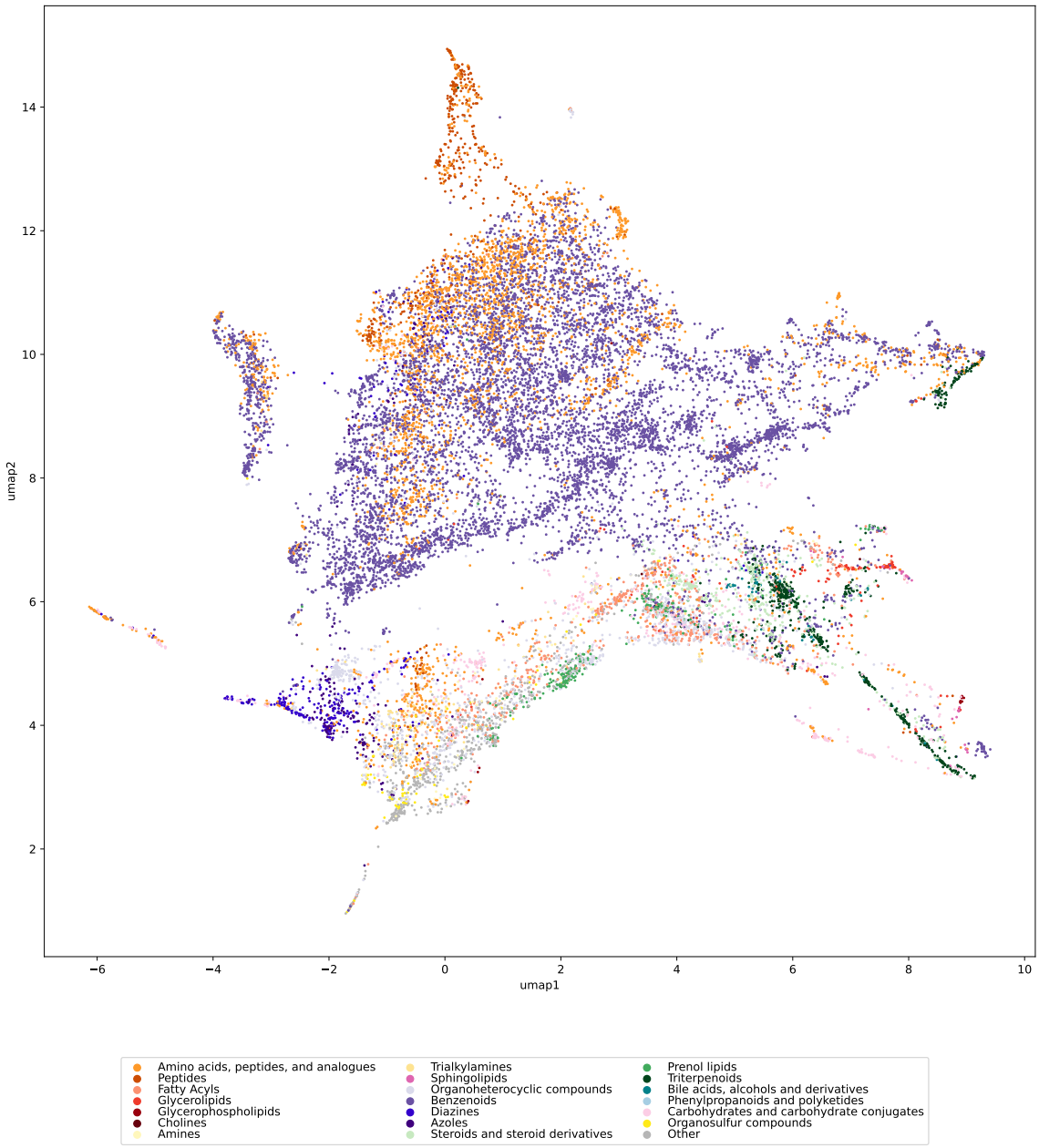

Fig. 1. Map of biomolecular structures with color-coded compound classes. Compound classes were chosen based on frequency in the biomolecular structures. For compounds belonging to more than one compound class, the class with the largest structural pattern is selected[33]. Certain compounds were excluded before computing the UMAP embedding and the plot, compare to Supplementary Fig. 6. The UMAP embedding was computed from the remaining 18,096 molecular structures. See Supplementary Fig. 8 for the corresponding plot where these compounds were reinserted.

Certain molecular structures and compound classes, in particular certain lipid classes, result in outlier clusters in the UMAP embedding (Supplementary Fig. 6). To avoid that these molecular structures dominate the UMAP embedding, we excluded them. This leaves us with 18,096 molecular structures, which will be considered in all following analyses. See Fig. 1 for the resulting map of biomolecular structures, where we have color-coded compound classes according to ClassyFire[33]. Excluded molecular structures can nevertheless be displayed in the UMAP embedding (Supplementary Fig. 8).

Fig. 2 shows the distribution of myopic MCES distances. As expected, most distances are large; yet, for every molecular structure, the smallest distance to another molecular structure is usually below 10 (Fig. 2b). In fact, we observe that distance 10 occurs much more often than what we would expect by chance. This is due to the threshold $T = 10$ used in our computations and, in particular, due to the

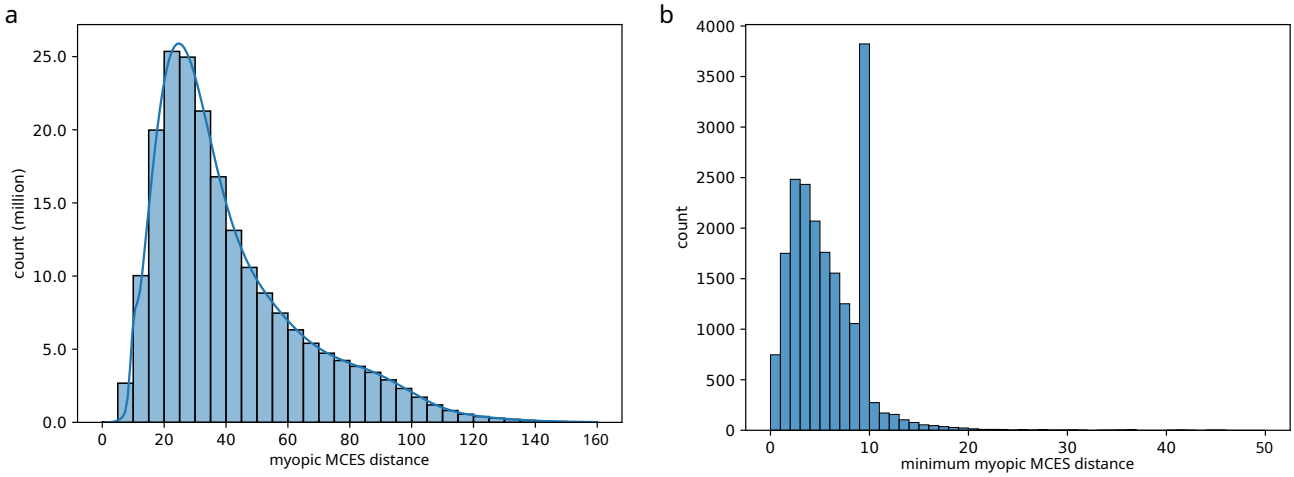

Fig. 2. Distribution of Maximum Common Edge Subgraph distances. Myopic MCES distances were computed with threshold $T = 10$. For the histogram plots, bin intervals are left-open and right-closed. For example, the tenth bin in the right plot covers distances x with $9 < x \leq 10$. (a) The distribution of all 199,870,021 computed distances from the subsampled biomolecular structures are shown as a histogram plot and kernel density estimate. The plot is truncated after distance 160. (b) Histogram of the *minimum* myopic MCES distance for each subsampled biomolecular structure against all other biomolecular structures, truncated after distance 50. The hump is due to the employed threshold $T = 10$, see text for details.

double thresholding: In case the exact MCES distance is computed but turns out to be larger then T, we instead use T as the myopic distance. Given that there is no such "hump" in the distribution of myopic MCES distances (Fig. 2a), we argue that this thresholding artifact has little effect on the computed UMAP embeddings.

Chari *et al.*[32] showed that one must be highly cautious deducing structure of the data from a 2-dimensional UMAP embedding. We stress that this is not a restriction for what we are doing: We *already know* that our data are structured, in the sense that molecular structures can be (dis)similar, or belong or not belong to the same compound classes. What we are investigating is to what extent a subset of the data is a uniform subsample: If we are able to spot non-uniformness in the 2-dimensional UMAP embedding, then it is presumably not a uniform subsample in higher dimensions, either. It is nevertheless indisputable that the UMAP embedding is far from perfect, meaning that a small/large distance in the plot does not necessarily imply a small/large MCES distance (Supplementary Fig. 9). UMAP shares these restrictions with any method that projects a high-dimensional space into the plane. Clearly, there is also some arbitrariness in the layout of the UMAP plot, see [34] and Supplementary Fig. 7. Finally, when inserting new samples into an existing UMAP embedding, those new samples tend to be inserted into the existing "structure of the plot", rather than generating novel distant clusters or singletons. Compare Supplementary Fig. 6 and 8: Whereas the lipid classes form distant clusters in the original UMAP embedding, they are integrated into the existing plot structure when reinserted.

Distribution of molecular structures in public datasets

We consider ten public molecular structure datasets frequently used to train machine learning models[35–47] See Supplementary Table 3 and the Methods section for details. All of these datasets have in common that molecular structures are labeled by experimentally determined measures. Also, all of these datasets have repeatedly been used to train and evaluate machine learning models[8,48,49]. Not all molecular structures in a dataset are biomolecules: See the discussion of natural product-likeness score distributions below. Even less so, all molecular structures in a dataset are contained in the 20,000 subsampled biomolecular structures, see again Supplementary Table 3 for the statistics.

We investigated to what extent the molecular structures in each dataset are a uniform subset of biomolecular structures. For each dataset, we computed myopic MCES distances for all molecular structures, combining the training dataset and the 18,096 biomolecular structures. Nine resulting plots can be found in Fig. 3 and one in Supplementary Fig. 10. To ensure comparability of plots, we used the

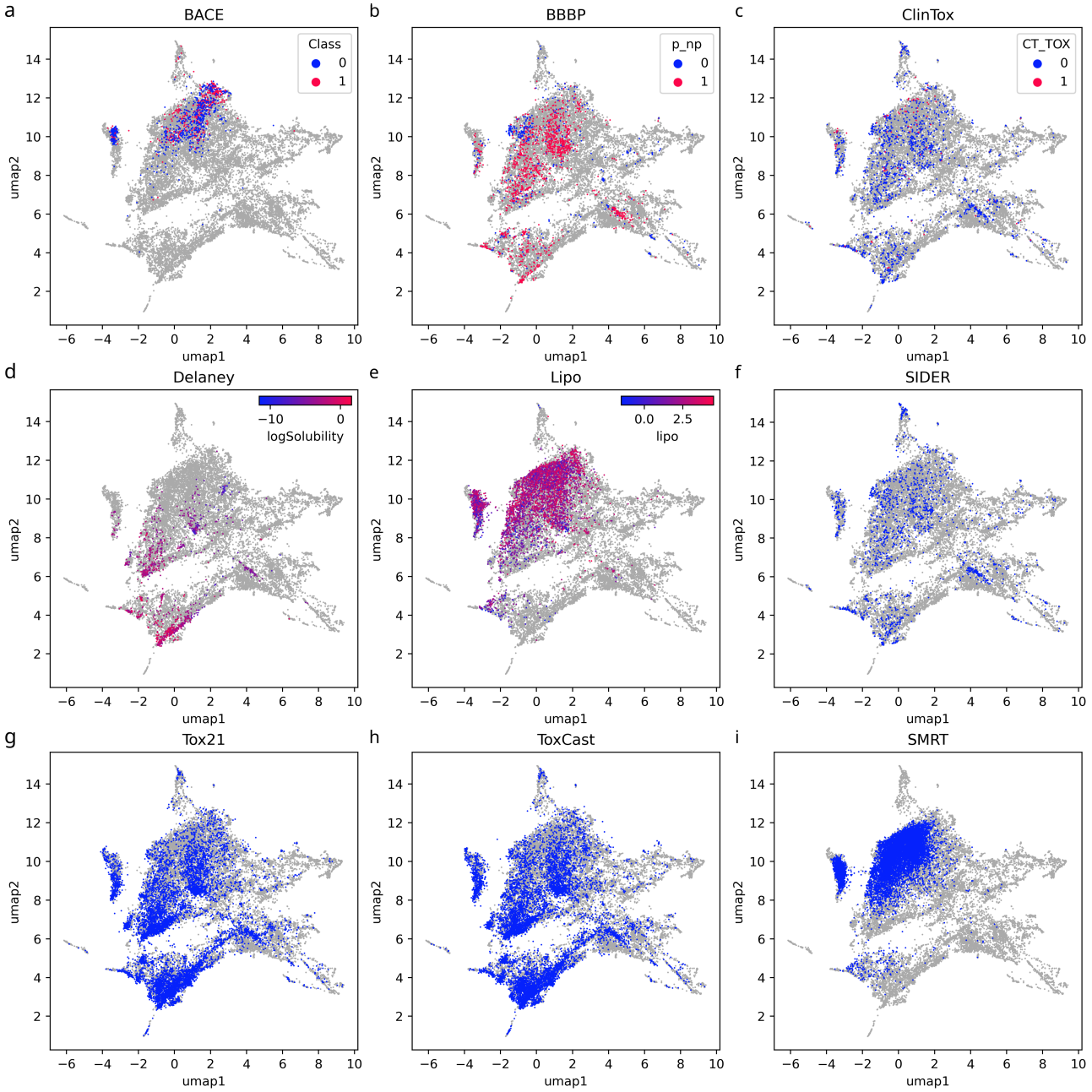

Fig. 3. Maps of nine public molecular structure training datasets. (a–i) UMAP embedding of nine public datasets that are frequently used to train, evaluate and compare machine learning models. Shown are datasets BACE (a), BBBP (b), ClinTox (c), Delaney (d), Lipo (e), SIDER (f), Tox21 (g), ToxCast (h), and SMRT (i). We use the same UMAP embedding as in Fig. 1. No molecular structures fall outside of this plot. Number of samples differ for each plot, as each plot combines the 18,096 biomolecular structures and the structures from the dataset. See Supplementary Table 3 for details. biomolecular structures are shown in light gray. In case there are multiple classes or values for regression, we color-code those in the plots.

same UMAP embedding as in Fig. 1. We observe that the subset of molecular structures available in public datasets is usually far from uniform. We stress that we can make this observation based on the available UMAP embeddings, without the need that these plots represent some chemical or biochemical "truth". We argue that most of the public datasets are also not representative, meaning that large areas of the biomolecular structures are completely missing in the datasets. In fact, some datasets are concentrated in one or few areas in the plot.

Recall that additional samples tend to get inserted in the existing structure of the plot even if they are different from all existing samples (Supplementary Fig. 6 and 8). Hence, even molecular structures highly different from all biomolecular structures will not result in novel outlier clusters. Arguably the best coverage is observed for the toxicity datasets and the MS/MS dataset. Given the known limitations

of the UMAP plot, this *does not imply* that these datasets contain a uniform or representative subset of molecular structures. Instead, these plots can only warn us that a dataset is *not* representative.

Maximum Common Edge Subgraph computations

The Maximum Common Edge Subgraph problem is well-suited to capture high similarities between molecular structures[23–25,50]. The *MCES distance* between two molecular structures equals the number of edges in the first molecular graph plus the number of edges in the second molecular graph, minus twice the number of edges in the MCES. This distance agrees well with chemical intuition: It is the minimum number of edges we have to remove from both graphs so that the resulting graphs are isomorphic, ignoring singleton nodes. From a chemical standpoint, we can think of it as the "number of chemical reactions" required to transform one molecule into another. Molecular graphs are labeled graphs where nodes are associated with atom types and edges with bond orders. When comparing two such graphs we have to take into account these labels and how to compare them, such as "double vs. aromatic bond", see the Methods section for details. We do not consider hydrogen atoms in our computations.

Unfortunately, computing the MCES is provably hard, and it is considered highly unlikely that an efficient algorithm for this problem exists.[4] To compute distances for our UMAP embedding in Fig. 1 required to swiftly solve more than 160 million instances of the NP-hard MCES problem. We present an efficient implementation based on computing lower distance bounds to quickly recognize dissimilar structure pairs, with an additional step for potentially similar pairs for which we compute provably exact distances with an Integer Linear Programming (ILP) formulation. Using an ILP to compute the MCES has two key advantages over previous methods based on enumerating cliques in a product graph: (i) Contrary to the clique-based methods, the ILP tends to be fast when the input is similar, which, as we argue, are exactly the interesting cases and (ii) the ILP avoids the cumbersome treatment of the so-called ΔY exchange that occurs when modeling the problem via line graphs. Our method is the first to use an ILP for the comparison of chemical structures.

In practice, we decide for a distance threshold (say, $T = 10$ edge modifications) whether our bounds guarantee that the true distance is at least T; in these cases, we use the bound as an approximation of the true distance. Only if no bound can guarantee that the distance is at least T, we execute the exact algorithm and report its result, this time using T as an upper bound. We call the resulting distance the *myopic MCES distance*. The two-step procedure has two advantages: (a) The ILP is usually fast if the MCES distance is small, whereas running times can get very large for larger distances. Our two-step approach thus explicitly and efficiently excludes most running time-intensive exact computations; (b) Considering a pair of molecular structures, it is obviously of high interest to know whether the MCES distance between those two structures is 2 or 8. Yet, we argue that it is *mostly irrelevant to know whether the MCES distance is 42 or 48*: In both cases, the two molecular structures are highly dissimilar. Both numbers can serve as reasonable approximations for the true distance. In particular, divergence in the larger distance will not result in substantial changes of the UMAP embedding, by formulation of the UMAP optimization problem[26].

We performed an in-depth evaluation of our methods using a subset containing 20,000 uniformly subsampled instances, where an *instance* is a pair of biomolecular structures (Fig. 4). Running times were measured on a 40-core processor running 80 threads in parallel; we report running times per thread. For the ILP, 24 of 20,000 instances did not finish within four days of wall clock time. For those instances, we use the time at which computations were stopped as a running time proxy. Doing so, total running time of the ILP equals 234.2 days for the 20,000 instances (average 16.9 min per instance).

[4] Precisely speaking, the MCES problem is NP-hard, as it generalizes subgraph isomorphism which, in turn, generalizes the clique problem. As MCES is clearly in NP, it is NP-complete. Next, an "efficient algorithm" is an algorithm with running time polynomial in the size of the instance, that is, the number of edges or nodes of the two graphs for the MCES problem. It is known that there cannot be an algorithm with polynomial time for an NP-hard problem, unless P = NP. The field of theoretical computer science usually assumes that $P \neq NP$ holds, and there exists a Millennium prize to prove or disprove this. But even if P = NP, it is assumed that the exponent in the running time of any polynomial time algorithm will be prohibitively high, and the resulting algorithms will be of no practical use, similar to the polynomial-time algorithm for PRIMES[51].

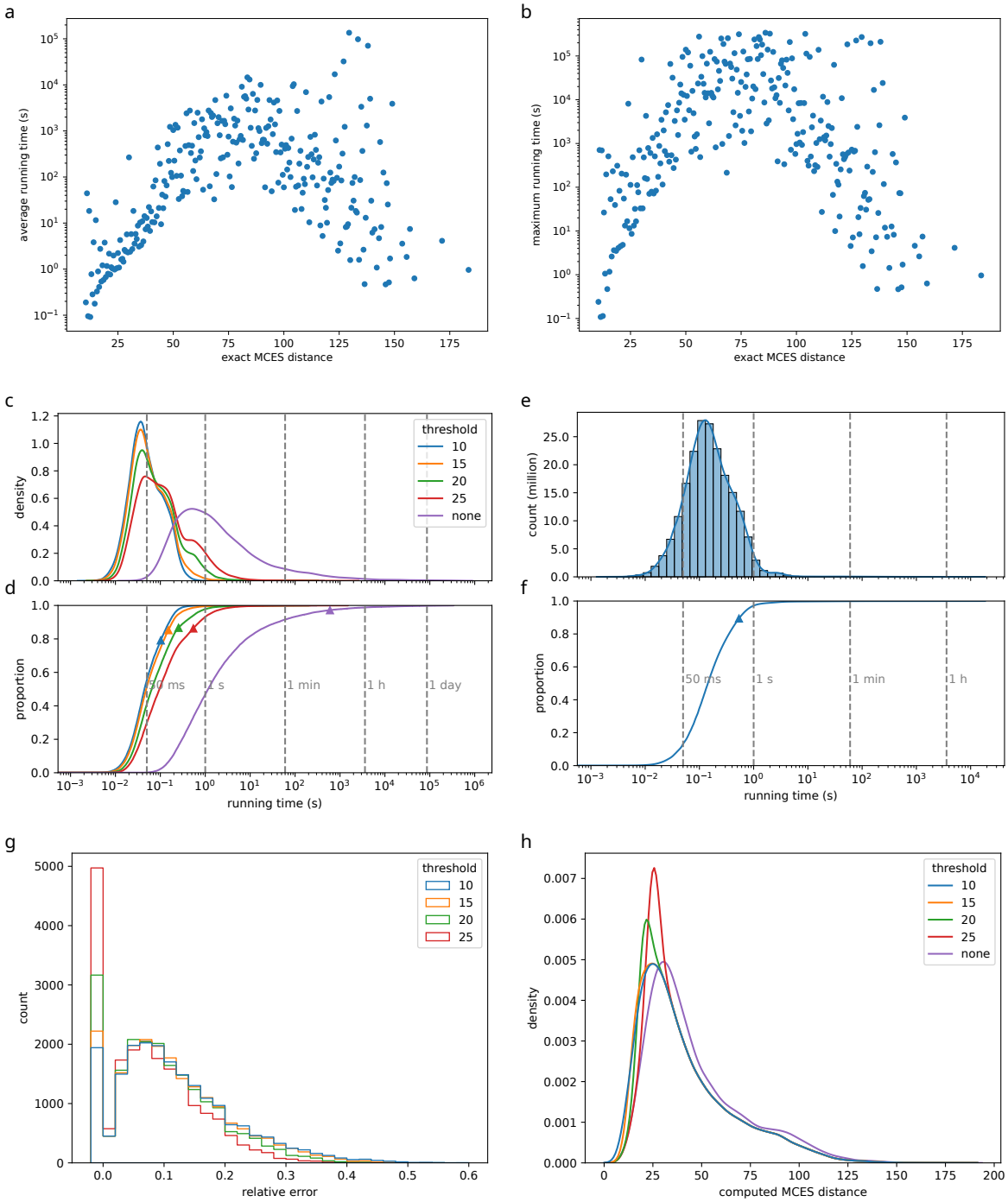

Fig. 4. Computing the Maximum Common Edge Subgraph. Each instance is a pair of molecular structures from the biomolecular structures. For (d,e) we use all 199,870,021 instances from our subsampled biomolecular structures, for all other plots a uniform subsample of 20,000 instances. (a,b) Running time of the ILP vs. exact MCES distance of the instance. Shown are (a) average and (b) maximum running times for each MCES distance bin. Note the logarithmic y-axis. (c,d) Distribution of running times using a MCES distance thresholds $T = 10, 15, 20, 25$ and no threshold. Note the logarithmic x-axis. (c) Kernel density estimates and (d) total running times (empirical cumulative distributions). Average running times shown as triangles. (e,f) Distribution of running times for $T = 10$ using all instances. Note the logarithmic x-axis. The distribution contains outliers not visible in the plot: 213 instances (0.0001 %) have a running time longer than one hour. (e) Histogram and kernel density estimate, (f) total running time (empirical cumulative distribution). Average running time shown as a triangle. (g) Using bounds instead of exact computations results in an error of distance estimates. If we increase threshold T, then more instances are computed exactly. Shown is the histogram of relative errors for different thresholds, using bin width 0.02 and truncated at 0.6. Intervals are left-open; the first bin constitutes only instances with a (relative) error of exactly zero. (h) Kernel density estimates of myopic and exact MCES distance distributions ("computed MCES distance"). Compare to Fig. 2.

Sorting instances by ILP running time, we observe that 1.04 % of the instances are responsible for more than 95 % of the total running time.

We first consider the dependence of ILP running times on the exact MCES distance (Fig. 4 ab). ILP instances that did not finish are excluded from this plot. For distances up to 75, we observe a clear correlation between distance and running time. For example, 29.5 % of the instances have MCES distance up to 30, but contribute only 0.91 % to the total running time. For larger distances up to 100, this correlation becomes less clear. Apparently, both the MCES distance and the actual structure of the MCES instance are affecting running times. Beyond distance 100, results must be interpreted with care, as there are few such instances corresponding to few outlier structures.

Second, we evaluated how the combination of bounds and exact computations results in favorable running times (Fig. 4 cd). We concentrate on thresholds $10 \leq T \leq 25$ reasonable for measuring molecular structure similarity. Increasing the threshold T means that more instances have to be computed exactly, since fewer instances result in a lower bound above T. Compared to the exact method, we observe a massive running time improvement: Even for the largest threshold $T = 25$, total and average running times decrease 1101-fold. Further lowering the threshold also further reduces running times: Using $T = 10$ instead of $T = 25$ decreases the total and average running times 5.3-fold. For the myopic MCES distances, we observe "right humps" in the distributions of running times (Fig. 4 c). We attribute these humps to instances that have to be computed exactly.

To exclude bias through subsampling, we repeated the above analysis using all pairs from the 19,994 biomolecular structures. To avoid proliferating running times — here and throughout the rest of the paper — we concentrated on threshold $T = 10$, see Fig. 4 ef. Total running time for the complete set was 1240 days, corresponding to 15.5 days of wall clock time on a 40 core processor. We observe basically the same pattern as for the subsampled instances, but the average running time per instance is 5.2-fold larger than for the subsampled set (536 ms vs. 102 ms). Here, 0.2 % of the instances are responsible for half of the total running time. The difference in average running time for the same threshold can likely be attributed to a difference in ILP solver settings, see the Methods section for details. We did not use a running time limit for the ILP solver, see Methods section.

Finally, we analyzed the error of computing bounds instead of exact distances: The myopic MCES distance may differ from the exact distance; the *absolute error* is the difference between the exact distance value and the myopic MCES distance. Recall that the myopic MCES distance is a lower bound, so the absolute error is never negative. Deviations from the true value are relevant mostly in comparison to the true value. To this end, we investigate the *relative error*, dividing the absolute error by the exact distance value (Fig. 4 g). As expected, more distances estimates have a non-zero error if we lower threshold T: In detail, 22.5 % of instances have zero error for $T = 25$; this decreases to 9.5 % for $T = 10$. Beyond this, we observe that varying the threshold T between 10 and 25 does not substantially change the shape of the relative error distribution. We observe that the distribution of myopic MCES distances (compare to Fig. 2) is very similar for different thresholds and for exact computations, only shifted (Fig. 4 h).

Recently, an implementation of RASCAL[52] has become available in RDKit (RDKit: Open-source cheminformatics, `https://www.rdkit.org`). RASCAL solves the MCES problem, computing the exact MCES when a predefined similarity threshold is exceeded. Similar to our implementation, the results of computed bounds can be used to approximate similarity when no exact calculations are performed. In contrast to the (absolute) myopic MCES distance, a relative measure of similarity introduced by Johnson[53] is used. To compare our method against RDKit's RASCAL implementation, we used the same subset of 20,000 instances as above. Using the default similarity threshold of 0.7, only 280 of these instances (1.40 %) were calculated exactly. Notably, 100 of those 280 instances (35.7 %) failed to compute, either because an internal limit on the size of the product graph ("maxNumberMatchingBonds") was exceeded (98 instances), or because the time-out of one hour was reached (two instances). Disabling the internal size limit on the product graph, 68 of those 98 instances were computed exactly in less than one hour. In 29 cases, the running time threshold was reached and in one case, the available memory of 256 GB was exceeded. For all 279 exactly calculated instances where the memory was not exceeded, the average wall-clock per instance was 7.92 min, using one hour as proxy for instances with time-out.

Next, to compute more instances exactly, we lowered the similarity threshold to 0.5. Doing so, 2899 (14.5 %) of the instances were calculated exactly. Of these, 423 (14.6 %) of these calculations did not finish due to the internal size limit (408 instances) or time-outs (15 instances). When disabling the

internal size limit, 186 of 408 instances exceeded the time limit of one hour; for five instances, the memory was exceeded. Here, the average wall-clock time for the exactly calculated 2894 instances was 4.47 min, disregarding the instances were the program crashed because memory usage was exceeded. Again, for instances with time-out one hour was used as proxy.

Notably, the internal size limit can only be changed by modifying and re-compiling RDKit's C++ source code; the strict size limit was a deliberate choice of the developers to prevent excessive memory usage. Also, the high memory usage of this approach makes it basically impossible to parallelize computations. Finally, computing *approximate* results via RDKit's RASCAL is orders of magnitude faster than computing MCES bounds, with a mean wall-clock time of 1.36 ms for the former. Yet, comparing running times between RDKit and our MCES code is misleading, as it is dominated by running time differences between compiled C++ and interpreted Python code.

Numerous variants of the MCES problem exist, such as finding a Maximum Common Subgraph (MCS), a connected MCES, or restricted variants of the problem[25]. It is extremely challenging to compare the quality of results from different such variants, so we refrain from an in-depth evaluation. Seipp[54] performed running time evaluations of the myopic MCES distance against MCS computations using RDKit ("rdFMCS" module as opposed to "rdRascalMCES", which contains the RASCAL implementation) and SMARTScompare[55], and found that the myopic MCES approach is several orders of magnitude faster than exact MCS methods implemented there, and sometimes even on par with heuristic methods[50]. This is noteworthy as the MCES problem is usually assumed to be "much harder" than the MCS problem.

For numerous computational tasks, it is required that a distance measure is a metric. We can easily show that both the bounds as well as the exact MCES distances are (pseudo)metrics. For the myopic MCES distances, the important fact is that we apply double thresholding: On the one hand, the exact distance is only computed in case the bounds are below threshold T. On the other hand, if the exact distance is above T, we instead report T as the myopic MCES distance. This implies that whenever we execute exact computations, the reported distance is smaller or equal to T; whereas if we only execute bound computations, the reported distance is greater or equal to T. Doing so speeds up ILP computations via constraint (8). Beyond that, it allows us to prove that the myopic MCES distance is indeed a metric, see the Methods section. If we do not use the second threshold (that is, we return the exact MCES distance even if it is larger than T) then this does not only increase running times of the ILP. In addition, the resulting distance measure is no longer a metric: See Supplementary Fig. 14 for an example where the triangle inequality is violated. Instead of double thresholding, we may use the Floyd-Warshall algorithm[56] for finding all shortest paths in a complete graph, to enforce the triangle inequality.

Compound class distribution

Besides a uniform subsample, there is another feature a molecular structure dataset should exhibit so that it represents the full space of biomolecular structures: Namely, all compound classes that biomolecules belong to should also be present in the training data. If a dataset completely misses molecular structures from a particular compound class, then a machine learning model trained on the data might show poor predictions for this compound class. The same holds true in case very few samples exist for a particular compound class. This will not be noticeable in evaluations: If we split the dataset into test and training data, the compound class is still absent from the test data. Similarly, if only few examples from a compound class are present, then those will have little or no influence on evaluation statistics. Consequently, we may overestimate the power of the resulting machine learning models for real-world data.

Historically, compound classes were defined based on, say, biochemical precursors. Unfortunately, class annotations were available only for a small fraction of molecular structures. Recently, certain compound class ontologies were defined purely based on molecular structures[33,57]. Here, we concentrate on the ClassyFire ChemOnt ontology, which contains 4825 classes[33].

Throughout this paper, we concentrated on machine learning models for biomolecular structures. Consequently, we want to ignore compound classes that contain basically no biomolecular structures:

	biomol.	BACE	BBBP	ClinTox	Delaney	Lipo	SIDER	Tox21	ToxCast	SMRT	MS/MS
Fatty acid esters	14.7%	1	23	54	36	24	44	287	291	2,706	1,377
Carboxylic acids	12.3%	25	189	265	0	448	313	911	1,126	1,362	4,148
Polyols	12.0%	1	65	73	23	49	89	221	253	88	2,867
Prenol lipids	11.3%	4	13	36	23	18	31	245	256	223	1,833
Acetals	10.7%	14	108	80	18	47	86	255	273	2,060	2,410
Carbohydrates and carbohydrate conjugates	10.3%	0	64	78	19	34	91	207	235	440	2,447
Lactones	9.7%	2	40	57	15	38	47	184	197	355	2,162
Pyrans	9.4%	1	12	17	10	71	19	85	90	462	2,297
Benzopyrans	9.2%	47	21	20	11	79	16	102	113	571	2,339
Trialkylamines	9.0%	51	550	291	1	995	281	838	900	16,206	1,428
Aralkylamines	8.9%	595	371	257	6	767	270	636	693	5,814	1,604
Cyclic alcohols and derivatives	8.8%	1	178	139	39	33	150	264	288	229	2,002
1-benzopyrans	8.8%	47	21	20	11	78	16	100	110	548	2,255
Glycosyl compounds	8.4%	0	51	55	11	26	62	141	155	77	2,164
Dicarboxylic acids and derivatives	8.0%	6	82	114	11	23	118	422	508	256	1,670
Tertiary alcohols	7.5%	5	170	118	42	92	117	299	318	379	1,568
Pyranones and derivatives	7.2%	0	10	12	9	70	9	70	78	379	1,946
1-hydroxy-4-unsubstituted benzenoids	7.2%	3	44	45	25	83	43	251	264	275	1,852
Aryl ketones	7.1%	11	91	58	11	139	49	294	317	1,227	1,546
Monosaccharides	7.0%	0	49	58	9	23	66	145	170	425	1,586
Organic phosphoric acids and derivatives	6.9%	0	3	19	6	2	22	92	99	13	461
O-glycosyl compounds	6.8%	0	44	42	8	1	43	102	107	27	1,883
Phosphate esters	6.8%	0	3	16	6	2	17	84	86	8	439
Alkyl phosphates	6.6%	0	3	13	5	2	14	67	71	3	422
Pyrroles	6.6%	129	66	63	5	378	85	170	186	9,364	1,312
Cyclic ketones	6.3%	2	168	121	50	51	125	285	312	589	1,051
Vinylogous acids	6.0%	0	51	41	10	57	37	179	203	330	1,465
Piperidines	5.9%	48	241	133	7	718	129	391	422	20,264	1,171
Indoles and derivatives	5.6%	79	64	60	5	256	81	173	182	4,420	1,376
Glycerophospholipids	5.5%	0	0	2	0	0	1	5	6	0	217
Dialkyl phosphates	5.5%	0	0	3	2	0	5	28	30	0	250
Dialkylamines	5.4%	530	139	138	1	404	147	342	385	4,459	831
Tricarboxylic acids and derivatives	5.3%	2	7	12	1	1	18	73	74	13	368
Alpha,beta-unsaturated carboxylic esters	5.2%	7	12	27	7	11	24	191	198	2,572	1,075
Enoate esters	5.2%	7	12	27	7	11	24	189	196	2,572	1,074
Pyrrolidines	5.0%	92	113	85	7	193	121	215	247	10,646	867

Table 1. Compound class distribution. Compound classes are not represented equally in the datasets. Here, all ClassyFire compound classes occurring in at least 5 % of the biomolecular structures are investigated. The occurrences for structures of the datasets BACE, BBBP, ClinTox, Delaney, Lipo, SIDER, Tox21, ToxCast, SMRT and MS/MS are shown. Some molecular structures could not be classified by ClassyFire and were discarded: This applies to seven structures from BBBP, one from SIDER, four from ToxCast, and 1275 biomolecular structures. Compound classes with at least 15 occurrences in all datasets are omitted; this applies to 65 compound classes. See Supplementary Table 4 for a larger subset of ClassyFire classes at 1 % cutoff.

If no or few biomolecules are part of a certain compound class, it is not surprising that a molecular structure dataset will also contain no or few molecular structures for that compound class. To this end, we only consider those classes where at least 5 % or 1 % of biomolecular structures are part of the class, respectively. See Table 1 and Supplementary Table 4 for the corresponding statistics. There, we have chosen a somewhat arbitrary threshold of 15 molecular structures ought to be present for any compound class. We argue that trends will stay the same, independent of the chosen threshold. We only display those compound classes in the two figures for which at least one dataset contains less than 15 examples. For 65 compound classes (threshold 5 %) and 95 compound classes (threshold 1 %), respectively, none of the datasets contains less than 15 examples. Yet, recall that ClassyFire ChemOnt is an ontology, and that a molecular structure may belong to several compound classes simultaneously. Compound classes on higher levels of the ontology can be huge (for example, virtually all molecular structures belong the the class *organic compounds*) and consequently, somewhat uninformative. We provide the full statistics in Supplementary Table 5.

Considering the compound class distribution in a dataset overlaps to a certain extent with MCES UMAP embeddings, compare to Fig. 1. Yet, both approaches also complement each other: The UMAP embedding introduces a certain amount of arbitrariness, and may show structure even if there is none[32] (Supplementary Fig. 9). In contrast, compound classes have a biochemical meaning, and we only superimpose this known structure onto the dataset. On the other hand, this restricts our analysis: Compound classes represent true structure in the space of molecules, but by no means, they represent *all* structure in this space. Hence, compound class analysis cannot detect all shortcomings of our training data. In contrast, the UMAP embedding allows to spot issues without the corset of compound classes.

Natural product-likeness score distributions

As a third approach to test whether a molecular structure dataset reproduces the universe of biomolecules, we propose to study the distribution of natural product-likeness scores. These scores

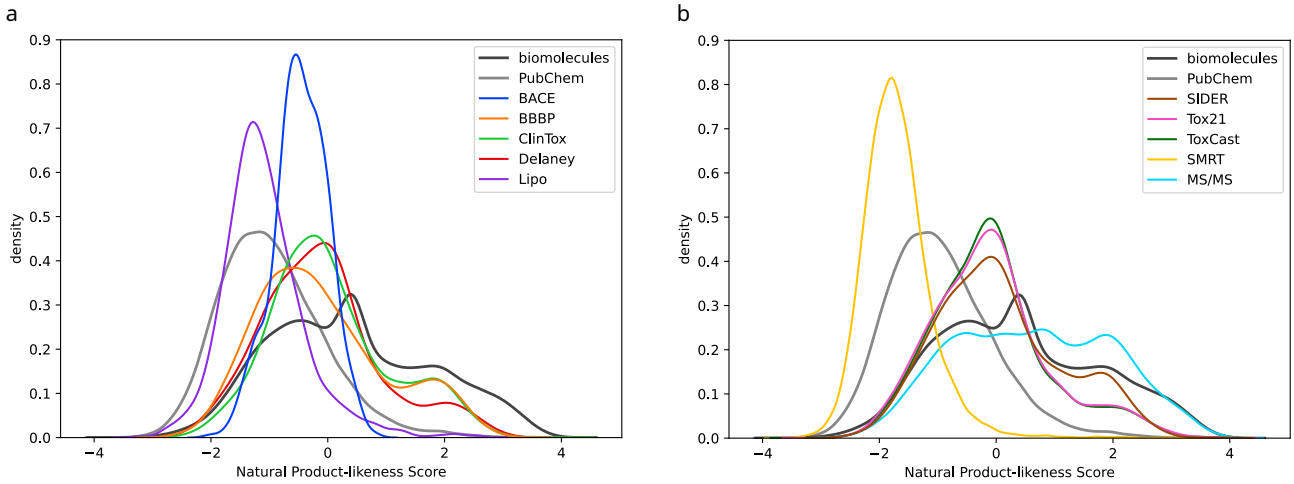

Fig. 5. Natural product-likeness score distributions. Large scores indicate that a molecular structure is regarded similar to a natural product. Shown are kernel density estimates (a) for datasets BACE, BBBP, ClinTox, Delaney, Lipo and (b) for datasets SIDER, Tox21, ToxCast, SMRT and MS/MS. For comparison, we show kernel density estimates for the 19,994 biomolecular structures and PubChem (20k uniformly subsampled molecular structures).

provide a measure of how molecules are similar to the structural space covered by natural products[27]. Natural product, in turn, are chemical entities produced by living organisms. Hence, natural product-likeness scores allow us to differentiate between biomolecules and "synthetic molecules" presumably not of biological interest. Here, we use the score of Ertl *et al.*[27] as implemented in RDKit.

Our reason to study natural product-likeness scores is linked to the intrinsic problem of providing a molecular structure dataset with experimental data: As discussed above, the choice of small molecules included in a dataset is usually governed by monetary aspects. Next, the prize of a compound depends on the difficulty of chemical synthesis etc, not its importance for building a machine learning model. Certain molecules may be regarded as highly interesting for a certain dataset, but if those molecules are too expensive, they cannot be measured.

We found that natural product-likeness scores allow us to get an impression how far a particular dataset deviates from the universe of biomolecules. In particular, we can compare the score distribution to that of PubChem[58]: On the one hand, PubChem contains the vast majority of biomolecular structures used here. On the other hand, PubChem additionally contains many million molecular structures that are presumably "molecules of no particular biological interest". Hence, a distribution somewhat similar to that of biomolecular structures hints at a dataset where indeed, molecules of biological interest are present in sufficient number. In contrast, a distribution similar to or beyond that of PubChem, indicates that the dataset contains only a small fraction of biomolecules. If we apply a model trained on this data to real-world data that, according to our assumptions, was measured from a biological source, then the model may again perform much worse than what we would expect from evaluation results. See Fig. 5 for natural product-likeness score distributions of the ten datasets. We observe that for three datasets (BACE, Lipo, SMRT) the distribution of natural product-likeness scores is substantially different from that of biomolecules. For the other seven datasets, we do not observe a particularly noteworthy distribution pattern.

Tanimoto coefficients

Tanimoto coefficients[59] are arguably the most-commonly employed measure to estimate the (dis)similarity of two molecular structures, and are the "working horse" for numerous cheminformatics applications including virtual screening. To compute a Tanimoto coefficient, we independently transform each molecular structure into a bit string, where each position encodes for the presence or absence of a particular subgroup. Then, we compare the two bit strings using the Jaccard index or the Jaccard distance for similarities and dissimilarities, respectively. One of the main advantages of Tanimoto coefficients is that after transforming each molecular structure into a bit string, subsequent

operations can be executed extremely fast. Consequently, it is most likely that Tanimoto coefficients will remain in use for any application that requires swift computations.

Unfortunately, Tanimoto coefficients also have a number of issues, *because* we are transforming a molecular structure to a bit string. It is inevitable that substantial information is lost in this transformation. Citing Bajusz *et al.*[22], "despite the generally positive findings about the applicability of the Tanimoto coefficient several of its weaknesses have also been reported from as early as in a 1998 study." Numerous resulting issues were first discussed by Flower[15], but see also [14, 16–22]. For example, similarity values when comparing large structures, for which many bits of the bit vector are Ones, behave very differently from similarity values when comparing small structures, for which almost all bits are Zeros[14,16,20]. Also, Tanimoto coefficients tend to be close to 1/3 even for very distant molecules[17]. These issues hold for *any* type of molecular fingerprint, be it a predefined list of molecular properties (say, MACCS[60] or PubChem CACTVS[61]) or combinatorial fingerprints (Morgan[13], Extended Connectivity[62]). Notably, these issues have nothing to do with hashing Extended Connectivity fingerprints to, say, 2048 bits[62]. Finally, similar issues hold for *any* similarity measure based on molecular fingerprints (say, the Sørensen-Dice coefficient), as the encoding into a binary vector is responsible for the issues[15].

In application, Tanimoto coefficients often result in counterintuitive values both for highly similar and highly dissimilar molecular structures: See Supplementary Fig. 11 for a collection of molecular structure pairs that demonstrate this problem. The MAP4 fingerprint (MinHashed atom-pair fingerprint up to a diameter of four bonds)[63] seems to be less susceptible to the issue of high values for intuitively dissimilar structures, while still exhibiting the same questionable behavior with lower values. Unlike machine learning task such as structure-activity or structure-property prediction[64,65], concatenation of molecular fingerprints will not improve (dis)similarity estimation but only "dilute" results.

For comparison, we have also computed UMAP embeddings using distances computed from Tanimoto coefficients, see Supplementary Fig. 12. Tanimoto-based plots may also be helpful for spotting inhomogeneous training data; yet, given the known limitations and issues of the fingerprint-based distance measures discussed above, we suggest to treat the resulting UMAP plot with even more care than the MCES-based plot.

Methods

In this paper, we consider only "two-dimensional" *molecular structures*: Precisely speaking, we consider the identity and connectivity (with bond types including aromatic bonds) of the atoms but ignore the stereo-configuration for asymmetric centers and double bonds. All molecular structures were standardized using the PubChem standardization web service[58]. Few structures that failed standardization were excluded. Tanimoto coefficients in Supplementary Fig. 11 and 12 were calculated using Molecular ACCess System (MACCS)[60], PubChem (CACTVS)[61], and Extended-Connectivity (ECFP)[62] fingerprints. MACCS and ECFP fingerprints were computed in RDKit, using the "MACCSkeys.GenMACCSKeys" and "rdMolDescriptors.GetMorganFingerprintAsBitVect" (radius 2, 2048 bits) functions, respectively; CACTVS fingerprints were retrieved from PubChem. MAP4 fingerprints (MinHashed atom-pair fingerprint up to a diameter of four bonds)[63] were computed with the official Python package (`https://github.com/reymond-group/map4`), setting the number of bits to 4096 and using 1 minus the value from the "Minhash.get_distance"-function to obtain Tanimoto coefficients.

Biomolecular structures

As a proxy of the universe of biomolecules, we use a union of several molecular structure databases that contain such molecules[58,66–79]. Recall that we interpret the term "biomolecules" as molecules of biological interest; this includes drugs, toxins and common contaminants. As *biomolecular structures*, we combine all molecular structures from databases KEGG (Kyoto Encyclopedia of Genes and Genomes), ChEBI (Chemical Entities of Biological Interest), HMDB (Human Metabolome Database), YMDB (Yeast Metabolome Database), PlantCyc, MetaCyc, KNApSAcK, UNPD (Universal Natural

Products Database), HSDB (Hazardous Substances Data Bank), Super Natural II, COCONUT (COlleCtion of Open Natural prodUcTs), and NORMAN (Network of reference laboratories, research centres and related organisations for monitoring of emerging environmental substances). See Supplementary Table 2 for details. Notably, we did not include the ChEMBL structure database[80]. We found that the distribution of molecular structures of its 1.9 million small molecules differed substantially from the structure databases mentioned above. For example, 81.9 % of the molecular structures in ChEMBL have a natural product-likeness score below zero, compare to Fig. 5.

HMDB and YMDB cover molecules that can be found in specific organisms. KEGG, ChEBI, PlantCyc, MetaCyc and KNApSAcK allow to link molecules to species and/or pathway information. Hazardous compounds can be found in HSDB. The NORMAN suspect list focuses on compounds that are expected to occur in the environment, including industrial chemicals, pesticides, pharmaceuticals and food additives. General collections of natural products are UNPD, Super Natural II and COCONUT. COCONUT is a large collection of natural products from 53 different data sources, including ChEBI natural products, KNApSAcK, UNPD and Super Natural II, but also FooDB, Marine Natural Products, GNPS (Global Natural Products Social) molecular structures, and a subset of ZINC (ZINC Is Not Commercial) with natural products.

In addition, we use subsets of large molecular structure databases that are flagged as being of biological relevance: These are compounds from PubChem which are either MeSH-annotated or part of particular classification schemes based on Schymanski *et al.*[81] . The PubChem classification "bio and metabolites" comprises structures from the PubChem Compound TOC categories 'Biomolecular Interactions and Pathways', 'Bionecessity', 'Metabolite Pathways', 'Metabolism and Metabolites', 'Plant Concentrations' and 'Metabolite References'; PubChem "drug" comprises structures from 'Drug and Medication Information' and 'Pharmacology'; PubChem "food" are structures of category 'Food Additives and Ingredients'; and PubChem "safety and toxic" comprises structures of 'Toxicity', 'Chemical Safety', 'Safety and Hazards' and 'Agrochemical Information'.

The combined dataset contains 718,097 unique molecular structures of metabolites and other molecules that can be expected in biological samples, see again Supplementary Table 2. The used combination of databases is a highly comprehensive list of *molecular structures of biological interest*. There are clearly larger databases such as PubChem, but those databases also contain molecular structures not of biological interest. We are not aware of a more comprehensive list of biomolecular structures.

Molecular structure datasets

We consider the following datasets frequently used in machine learning:

- The *BACE* dataset comprises 1513 synthetic human BACE-1 inhibitors reported in the scientific literature[35]. It provides inhibitory concentrations (IC_{50}) and additionally transforms these into binary labels by applying a concentration threshold.
- The *BBBP* dataset consists of 2039 molecules curated from the scientific literature discussing blood-brain barrier penetration[36]. The authors acknowledge that such a dataset curated from the literature is generally biased over-representing positive examples. Modeling blood-brain barrier penetration is of interest since the barrier is impenetrable to most drugs.
- The *ClinTox* dataset compares FDA-approved drugs from SWEETLEAD database[37] with drugs that failed clinical trials for toxicity reasons retrieved from the Aggregate Analysis of ClinicalTrials.gov (AACT) database (`http://www.ctti-clinicaltrials.org/aact-database`). The dataset contains 1478 molecular structures.
- The *Delaney* dataset contains 1128 compounds with water solubility data[38].
- The *Lipo* dataset provides lipophilicity data for 4200 compounds[39] and is part of the ChEMBL database[82] which contains bioactive molecules. For each compound octanol-water partition coefficients were experimentally measured (log D at pH 7.4).
- The Side Effect Resource (SIDER)[40] is a database of drugs and adverse drug reactions. The *SIDER* dataset contains 1427 molecular structures assembled in the DeepChem library[41] with side effects grouped in 27 system organ classes according to MedDRA classifications (Medical Dictionary for Regulatory Activities, http://www.meddra.org/).

- The *SMRT* (Small Molecule Retention Time) dataset consists of molecules and their experimentally acquired reverse-phase chromatography retention times[42]. Pure standard materials of 80,038 small molecules including metabolites, natural products and drug-like small molecules have been measured. This dataset is much larger than the other datasets. To avoid proliferating running times and cluttered plots, we uniformly subsampled 10,000 standardized molecular structures.
- *ToxCast*[43] is an extended dataset from the Tox21 program that includes toxicology data based on *in vitro* high-throughput screening of 8576 compounds.
- The *Tox21* dataset consists of samples from 12 toxicological experiments of *in vitro* bioassays and a total of 7831 molecular structures[44] used in the 2014 Tox21 Data Challenge[45]. This dataset has a relative large overlap in molecular structures with the ToxCast dataset: 6311 molecular structures are found in both datasets.
- The *MS/MS* dataset contains compounds that have at least one reference tandem mass spectrum in a spectral library. It comprises compounds from GNPS[46], MassBank[47] and NIST 17 (commercial, National Institute for Standards and Technology, Tandem Mass Spectral Library, 2017 release). Note that NIST17 is not public but very frequently used to train machine learning models. The dataset underwent several rounds of manual validation, discarding numerous molecular structures in the process[83,84]. In total, the dataset contains 18,848 unique molecular structures. It is different from the other datasets in that it is not one public dataset but rather, a union of different spectral libraries. Furthermore, for each compound one or more tandem mass spectra (MS/MS) are recorded, each being not a single value but rather complex structured data.

All datasets except *SMRT* and *MS/MS* were retrieved from the molecular property prediction framework CHEMPROP[48] at `https://github.com/chemprop/chemprop`. The *SMRT* dataset was downloaded from `https://doi.org/10.6084/m9.figshare.8038913`. Structures for *MS/MS* are based on the CSI:FingerID training dataset[83,84]. It was downloaded from `https://gnps.ucsd.edu/` (GNPS) and `https://massbank.eu/` (MassBank), but also contains the commercial NIST 17 MS/MS library. See Supplementary Tab. 3 for further statistics on the datasets. To the best of our knowledge, these are among the largest public datasets containing molecular structures based on *experimental data*. Many of these datasets are "living datasets", meaning that more compounds have been added at a later stage. Apparently, Tox21 and ToxCast are now provided as one dataset (`https://www.epa.gov/chemical-research/exploring-toxcast-data`).

Subsampling molecular structures

The biomolecular structure set contains 718,097 molecular structures. From this set, we uniformly sampled a subset of cardinality 20,000. We later noticed that six of these 20,000 molecular structures are single ions, for example, a single iron ion. These molecular structures resulted in execution errors when computing the MCES, but are also of no interest for our analysis. To this end, we discarded these six structures before computing distances, embeddings and running times. Consequently, our subsampled set contains 19,994 uniformly sampled molecular structures. The number of nodes n in the subsampled molecular graphs ranges from 1 to 139, with an average of 33.8 nodes (quartiles 20, 28, 40). Recall that we do not consider hydrogen atoms in our computations. The number of bonds (edges without multiplicities) m ranges from 0 to 153, with an average of 35.8 bonds (quartiles 21, 30, 43). The complexity of an instance can be roughly measured by the product $m_1 \cdot m_2$, for number of bonds m_1, m_2. This product ranges from 0 to 22,950, with an average of 1279.5 (quartiles 520, 900, 1593). We report quartiles (25 %, median, 75 %) as our subset contains both, very small and very large molecular structures. In the extreme case, a few "molecular structures" consist of a single atom only.

An *instance* consists of a pair of biomolecular structures. To avoid proliferating running times in our running time evaluation, we uniformly subsampled a set of 20,000 instances. To allow that resulting running times are comparable with other computations in this paper, we used the 19,994 molecular structure from above to generate the instances. We generated 199,870,021 pairs of molecular structures; from this set, we uniformly sampled 20,000 instances. For the subsample, the product $m_1 \cdot m_2$ ranges from 0 to 14,124, with an average of 1273.0 (quartiles 525, 910, 1584). The list of molecular structures (both the complete set of biomolecules, the subsample with 19,994 molecular structures, and the 20,000 subsampled structures pairs) are available for download.

The Maximum Common Edge Subgraph problem

Given two graphs $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$, their *maximum common edge subgraph* (MCES) is a graph $G_c = (V_c, E_c)$ with $V_c \subseteq V_i$, $E_c \subseteq E_i$, $i \in \{1,2\}$, such that $|E_c|$ is maximal. We define the *MCES distance* of the two graphs as $|E_1| + |E_2| - 2|E_c|$. Note that a maximum common edge subgraph minimizes this distance.

Different from other variants of the common subgraph problem, MCES does explicitly not require that the subgraph is connected; otherwise, displacing a single bond may necessitate to exclude almost half of the graphs from the common subgraph. It is easy to see that MCES is NP-hard, as it generalizes subgraph isomorphism which, in turn, can be easily be reduced from the clique problem[85]. See Raymond *et al.*[23] from 2002 for an early and Englert *et al.*[50] for a more recent discussion of the different problem variants as well as exact and heuristic algorithms for its solution. Exact methods guarantee that the MCES of two molecular structures is found, despite the computational hardness of the problem. One approach is based on finding a largest clique in the product graph of the line graphs of the two molecular structure input graphs, but has to deal with so-called ΔY exchanges that happen because of identical line graphs of the two small graphs K_3 and $K_{1,3}$. Restriction to finding *connected* common substructures lead to a speed-up, as observed by Koch[86]. It is noteworthy that clique-based algorithms are fastest when the subgraphs and hence, the cliques, are small; hard instances can easily be constructed by considering highly similar molecular structures, which are arguably the most interesting for our application. The program RASCAL[52] is arguably the commercial default for computing the MCES of small molecules based on cliques in a product graph; recently, an implementation of RASCAL has been made available in RDKit (version 2023.09.1). A lesser known approach, which has so far not been applied to the comparison of molecular structures, is based on integer linear programming[87].

Some algorithms reach improved running times by restricting the input to certain graph classes such as outerplanar graphs[88,89] but are therefore only of limited practical use. Next, certain algorithms do not consider the optimal MCES solution but rather, introduce additional constraints that have to be satisfied, to improve running times. Examples are limiting the number of connected components[25] and considering simplified graph representations[90]. We argue that all of these modifications, which are introduced because of the inability to efficiently compute the MCES rather than problem-immanent considerations, are again of only limited practical use.

We stress that beyond the simple "number of edge modifications" introduced above, there exist numerous other possibilities to transform the MCES into a distance between molecular structures[52]. We will not discuss this further here, as for the application of mapping all molecular structures into the plane, using absolute distances appears to be the appropriate choice.

Computing the Maximum Weight Common Edge Subgraph

We suggest a new approach to compare two molecular structures, which combines the computation of lower bounds as in [52], with an integer linear programming approach based on the formulation given in [87].

In molecular graphs, nodes are labeled by atom type and edges are weighted reflecting their bond order. We therefore compute the *maximum weight common edge subgraph* (MCES), a weighted variant of MCES. Formally, the input consists of two molecular graphs $M_1 = (G_1, a_1, b_1)$ and $M_2 = (G_2, a_2, b_2)$, where $a_i : V_i \to \Sigma$ denotes the atom type and $b_i : E_i \to \{1, 1.5, 2\}$ specifies the bond type (single, aromatic, or double), for $i \in \{1,2\}$. The task is to compute the maximum weight common edge subgraph $G_c = (V_c, E_c)$, that is, $V_c \subseteq V_i$ with $a_1(v) = a_2(v)$ for all $v \in V_c$, $E_c \subseteq E_i$ for $i \in \{1,2\}$, such that their *weighted distance*

$$d(M_1, M_2) = \sum_{e \in E_1} b_1(e) + \sum_{e \in E_2} b_2(e) - 2 \sum_{e \in E_c} \min\{b_1(e), b_2(e)\}$$

is minimal. Here, Σ is the set of elements, such as $\Sigma = \{\mathrm{C,H,N,O,P,S,B,F,Si,Cl,Se,Br,I}\}$. We reformulate this term and alternatively minimize

$$d(M_1, M_2) = \sum_{e \in E_c} \left| b_1(e) - b_2(e) \right| + \sum_{e \in E_1 \setminus E_c} b_1(e) + \sum_{e \in E_2 \setminus E_c} b_2(e) \ .$$

Precisely speaking, we solve the following problem: Given two molecular graphs M_1 and M_2 and a distance threshold T, compute their minimum weighted distance $w(M_1, M_2)$ if this is below T and otherwise an (ideally large) lower bound for $d(M_1, M_2)$.

We solve this problem using an algorithm engineering approach. It consists of first computing lower bounds on the distance. Should the maximum of these bounds already be larger than T, we report this maximum. Only if both bounds fail, we use an exact algorithm based on an integer linear program (ILP) to compute $\min\{d(M_1, M_2), T\}$.

The bounds are similar to those used in [52], but we additionally take the weights of the edges into account. The first bound can be computed in linear time using radix sort. It maps nodes onto each other based on their weighted degree. We first sort both node sets V_1 and V_2 by weighted degree. Then, we iteratively go through both lists by taking and removing the first pair and adding the absolute difference of the weighted degrees to the bound. Should one graph contain more nodes than the other, we also add the weighted degrees of the additional nodes to the bound. In the end, we divide the bound by two, because each edge has been considered twice in this calculation.

The second bound is also based on mapping nodes onto each other, but now we take the atom types of both nodes in an edge into account. We therefore, we construct bipartite graphs with nodes from each atom type on each side. Should the molecules have different numbers of an atom types, we fill up the graphs such that both sides have an equal number of nodes. We now link pairs of nodes in each bipartite graph and set the weight of this edge to the minimum weight to match the two nodes onto each other, which we compute using a complete enumeration approach for mapping the local neighborhoods. We calculate the final bound by computing minimum weight perfect matchings in each of the bipartite graphs, summing up the results and, again, dividing by two. We can compute this bound in cubic time using the Hungarian algorithm [91] or faster methods for finding minimum-weight perfect matchings.

We note that the second bound is stronger than the first, meaning that the value of the first bound can never be larger than that of the second. This follows because a matching for the second bound is also a valid matching for the first bound, but with smaller weight. In addition, we use the same denominator of two for both bounds. The advantage of the first bound is that it can be computed even faster. Since finding minimum matchings can be executed very fast in practice, we concentrated on the second bound throughout this paper.

To compute $\min\{d(M_1, M_2), T\}$, we use an integer linear programming (ILP) formulation. It is similar to the formulation in [87], but explicitly addresses weighted minimization and unmapped edges, and is faster to solve because of the atom labels and the threshold constraint. We introduce variables y_{ik} for each node pair $i \in V_1$ and $j \in V_2$ with $a(i) = a(j)$ to indicate whether i will be mapped to j in which case $y_{ik} = 1$. We introduce similar variables c_{ijkl} for mappable edge pairs $ij \in E_1$ and $kl \in E_2$ with $a(i) = a(j)$ and $a(k) = a(l)$ with corresponding weights $w_{ijkl} = |b(ij) - b(kl)|$. Finally, we have variables n_{ij} for all $ij \in E_1 \cup E_2$ to indicate whether an edges is not mapped. We denote by $N(i)$ the neighborhood of node i, that is the set of adjacent nodes.

Our ILP is as follows:

$$\min \quad \sum_{ijkl} c_{ijkl} w_{ijkl} + \sum_{ij \in E_1 \cup E_2} n_{ij} b(ij) \tag{1}$$

$$\text{s. t.} \quad \sum_{k \in V_2} y_{ik} \leq 1 \qquad \text{for all } i \in V_1 \tag{2}$$

$$\sum_{i \in V_1} y_{ik} \leq 1 \qquad \text{for all } k \in V_2 \tag{3}$$

$$\sum_{j \in N(i)} c_{ijkl} \leq y_{ik} + y_{il} \qquad \text{for all } i \in V_1, \text{for all } kl \in E_2 \tag{4}$$

$$\sum_{l \in N(k)} c_{ijkl} \leq y_{ik} + y_{jk} \qquad \text{for all } k \in V_2, \text{for all } ij \in E_1 \tag{5}$$

$$\sum_{kl \in E_2} c_{ijkl} + n_{ij} = 1 \qquad \text{for all } ij \in E_1 \tag{6}$$

$$\sum_{ij \in E_1} c_{ijkl} + n_{kl} = 1 \qquad \text{for all } kl \in E_2 \tag{7}$$

$$\sum_{ijkl} c_{ijkl} w_{ijkl} + \sum_{ij \in E_1 \cup E_2} n_{ij} b(ij) \leq T \tag{8}$$

$$y_{ik} \in \{0,1\} \qquad \text{for all mappable } i \in V_1 \text{ and } k \in V_2 \tag{9}$$

$$c_{ijkl} \in \{0,1\} \qquad \text{for all mappable } ij \in E_1 \text{ and } kl \in E_2 \tag{10}$$

$$n_{ij} \in \{0,1\} \qquad \text{for all } ij \in E_1 \cup E_2 \tag{11}$$

We implemented the approach using Python with packages `networkx` for graphs and graph algorithms and `pulp` for the ILP solver interface, which enables the usage of different optimizers like CPLEX, Gurobi or free libraries. For further details of our approach and implementation, see [54].

We stress that we can stop ILP computations at basically any time, and receive both upper and lower bounds for the objective function from the ILP solver. Similarly, we can start the ILP solver with a running time limit such as 5 min. This prohibits that few instances require hours of computation time, but nevertheless, provides a lower of bound for the myopic MCES distance.

Running time evaluations

All running times were measured on a 40-core Intel Xeon E5-2698 2.20GHz processor running 80 threads in parallel. Python version 3.9.7 on Ubuntu 20.04 was used to run the scripts; ILP computations were executed with CPLEX version 12.8 (`https://www.ibm.com/support/pages/cplex-optimization-studio-v128`). Running times of individual instances were estimated via wall clock time using Python's "time" module. We stress that the resulting running times are not optimized, in the sense that we did not exclude interrupts, thread switching etc. Furthermore, running two threads on a single core via hyperthreading reduces the overall running time of the complete batch, but increases the running time of each thread. Yet, the running times reported here are a good measure of what to expect when computations are executed in applications, where such running time optimizations are uncommon.

Notably, different settings were applied in the running time evaluations. For the complete set of instances, CPLEX was executed via `pulp` with the default settings. In this mode, CPLEX may use all cores available, potentially starting a large number of threads. Especially when already computing instances in parallel, this leads to suboptimal running times — both because of the overhead introduced by starting a large number of threads and because threads started by CPLEX may block other computations. For the subsampled set of 20k instances, CPLEX was executed with the parameter "threads" set to 1, restricting CPLEX to single-threaded mode. This option is only available when using the "CMD"-version of CPLEX in `pulp`.

For comparisons with RASCAL, the RDKit implementation in version 2023.09.3 was used; the time limit was set to one hour. As described above, for part of the comparison the similarity thresholds was modified. To obtain approximations when no exact computations were performed, the option "returnEmptyMCES" was enabled. To circumvent the internal limit on the size of the product graph, the

option "maxBondMatchPairs" in the C++ base of RDKit was exposed to Python interface, necessitating recompilation. Otherwise all settings were left at their default values.

The myopic MCES distance is a metric

We now prove that the myopic MCES distance is a metric. The proof proceeds in four steps: For completeness, we show that the first bound gives rise to a pseudometric. Next, we show that the second bound is a pseudometric, too. Third, we show that the exact MCES distance is a metric. Finally, we use all of these results to show that the myopic MCES distance is a metric, for all $T > 0$.

The first bound equals zero if the node set and weighted degrees are identical. It is symmetric because the absolute difference is symmetric. To see that the triangle inequality holds, we note that this bound is in fact the weight of a minimum matching between the two node sets. Now, two matchings between V_1 and V_2, and V_2 and V_3, respectively, can be combined into a matching between V_1 and V_3. Every edge weight is at most as large as the sum of edge weights in the two original matchings. Hence, the minimum matching between V_1 and V_3 will have weight at most as large as the sum of matching weights. Next, the same arguments also hold for the second bound, which, again, corresponds to a minimum weight matching, albeit with more sophisticated edge weights. Let d_B be the second bound.

Third, for the exact MCES distance d_E, it is understood that the weighted distance between two molecular graphs is symmetric, and that it is zero if and only the molecular graphs are identical. Proving the triangle inequality for the weighted MCES distance can be done analogously to the proof of Bunke *et al.*[92] who showed that a related MCS distance is a metric.

Fourth, we have shown that the bound d_B is a pseudometric, and that the exact distance d_E is a metric. We want to show that the myopic MCES distance d is a metric, too. It is straightforward to infer from the above that d is symmetric. Similarly, $d(x,x) = 0$ is clear. Next, $d(x,y) = 0$ implies $d_\mathrm{B}(x,y) = 0 < T$ and, hence, $0 = d(x,y) = d_\mathrm{E}(x,y)$. Since d_E is a metric, we infer $x = y$. Hence, what remains to be shown is that d fulfills the triangle inequality $d(x,z) \leq d(x,y) + d(y,z)$ for all x,y,z.

Recall that $d(x,y) = \min\{d_\mathrm{E}(x,y), T\}$ in case $d_\mathrm{B}(x,y) < T$. Clearly, $d(x,y) = d_\mathrm{B}(x,y) \geq T$ if $d_\mathrm{B}(x,y) \geq T$. We stress that $d_\mathrm{B}(x,y) \leq d(x,y)$ must hold even if $d(x,y) = \min\{d_\mathrm{E}(x,y), T\}$: In this case, $d_\mathrm{B}(x,y) < T$ and $d_\mathrm{B}(x,y) \leq d_\mathrm{E}(x,y)$ holds.

We make a case distinction on whether $d(x,z)$ was computed via the ILP or via bounds. (i) Assume $d(x,z) = \min\{d_\mathrm{E}(x,z), T\}$ holds. If at least one of the two distances $d(x,y)$ and $d(y,z)$ is at least T (say, $d(x,y) \geq T$) then

$$d(x,z) \;=\; \min\big\{d_\mathrm{E}(x,z), T\big\} \;\leq\; T \;\leq\; d(x,y) \;\leq\; d(x,y) + d(y,z). \tag{12}$$

This covers the case that one of the distances $d(x,y)$ and $d(y,z)$ was computed heuristically. So, $d(x,y) = d_\mathrm{E}(x,y)$ and $d(y,z) = d_\mathrm{E}(y,z)$. Now,

$$d(x,z) \;=\; \min\big\{d_\mathrm{E}(x,z), T\big\} \;\leq\; d_\mathrm{E}(x,y) \;\leq\; d_\mathrm{E}(x,y) + d_\mathrm{E}(y,z) = d(x,y) + d(y,z)$$

since d_E is a metric. (ii) Now, assume $d(x,z) = d_\mathrm{B}(x,z)$ was computed heuristically. But then, $d(x,z) = d_\mathrm{B}(x,z) \leq d_\mathrm{B}(x,y) + d_\mathrm{B}(y,z) \leq d(x,y) + d(y,z)$. □

The important step in this proof is (12): Here, we require that d is double thresholded via $d(x,z) = \min\{d_\mathrm{E}(x,z), T\}$. If we do not use double thresholding, then the resulting distance measure is not a metric, see Supplementary Fig. 14.

Data availability

Data used throughout this paper as well as computed MCES distances and corresponding UMAP embeddings are available at `https://github.com/boecker-lab/myopic-mces-data`; interactive UMAP visualizations are available at `https://mces-data.boeckerlab.uni-jena.de/`.

Code availability

Code is open source and freely available at `https://github.com/AlBi-HHU/myopic-mces`.

Conclusion

Machine learning datasets containing experimental data for molecular structures usually differ substantially from a uniform subset of biomolecular structures. More worrying is the fact that for most datasets, large regions of the biomolecular structure universe remain completely empty. We stress that this is a very practical issue, beyond the "correlation vs. causality" discussion. Using a well-known example from the literature: Even if we accept that child births in a country can be predicted from stork population sizes[93], it turns out that the model fails miserably for African countries. Consequently, the absence of any data for all countries outside of Europe should worry us massively on the domain of applicability of the model. We noted above the recent trend to train large end-to-end models *without* explicitly considering the domain of applicability[1–7]. We note in passing that datasets relying on quantum chemistry calculations instead of experimental values[94], usually do not share this issue.

Several guidelines for "good machine learning practice" in chemistry and the life sciences were published, due to the increasing importance of machine learning in these areas. For large models trained on small molecules, we suggest to include a distribution analysis of the training data into these recommendations; this may indicate whether the trained models are indeed predicting what they are claimed to predict. Otherwise, performance improvements using more complex machine learning models may be an adaption to the peculiarities of the training data, and may potentially result in no improvements in practice.

Our methods may allow us to spot datasets where the distribution of molecular structures is peculiar and potentially hazardous. On the other hand, if a dataset shows no peculiarities, this does not imply a "Carte Blanche" for machine learning. As an example, consider the MS/MS dataset that, in all of our analyses, did not appear peculiar. Yet, from our own experience with these data, we want to warn the reader that even here, the distribution of molecular structures may result in unexpected behavior of trained models and counterintuitive evaluation results[95,96].

We have shown that the presented method for estimating MCES distances exhibits fairly small running times. It is however important to note that the current implementation is Python-based, with hardly any optimizations applied. We conjecture that a C++ implementation of the MCES bounds can reach running times comparable to that of RDKit's RASCAL implementation, as it consists primarily of computing a maximum matching, one of the best-studied problems in theoretical computer science. It is understood that being able to swiftly estimate MCES distances allows for application beyond what we have discussed in this paper. In particular, MCES distances can be used as part of machine learning; notable examples are clustering, k-nearest neighbor and the radial basis function kernel. MCES can also be used to measure the diversity of a set of molecular structures[97,98]. Instead of simply using the Kullback-Leibler divergences[99] for this purpose, we suggest to measure, for every biomolecular structure, its distance to the (k-th) closest structure in the training data. As used here, the MCES distance measures an "absolute distance" between molecular structures; it is understood that it can be readily modified to take into account the sizes of the molecular structures[50]. Similarly, it may be modified to consider substructure relationships. Finally, we may incorporate differences in elements between the two structures where appropriate, for example for halogens.

Acknowledgments

F.K. and S.B. supported by Deutsche Forschungsgemeinschaft (BO 1910/23) and by the Ministry for Economics, Sciences and Digital Society of Thuringia (framework ProDigital, DigLeben-5575/10-9).

Author contributions

S.B. designed the research. F.K. executed all computations and evaluations, and prepared all figures. M.L. assembled the set of biomolecular structures and supported evaluations. J.S., G.K. and S.B. developed the method for computing myopic MCES distances, and J.S. implemented the method. S.B. and G.K. wrote the manuscript, with the help of the other authors.

Competing interests

The authors declare no competing interests.

References

1. Ekins, S., Puhl, A. C., Zorn, K. M., Lane, T. R., Russo, D. P., Klein, J. J., Hickey, A. J. & Clark, A. M. Exploiting machine learning for end-to-end drug discovery and development. *Nat Mater* **18,** 435–441 (2019).
2. Stokes, J. M., Yang, K., Swanson, K., *et al.* A Deep Learning Approach to Antibiotic Discovery. *Cell* **180,** 688–702.e13 (2020).
3. Urbina, F., Lentzos, F., Invernizzi, C. & Ekins, S. Dual Use of Artificial Intelligence-powered Drug Discovery. *Nat Mach Intel* **4,** 189–191 (2022).
4. Liu, G., Catacutan, D. B., Rathod, K., *et al.* Deep learning-guided discovery of an antibiotic targeting Acinetobacter baumannii. *Nat Chem Biol* **19,** 1342–1350 (2023).
5. Wong, F., Zheng, E. J., Valeri, J. A., *et al.* Discovery of a structural class of antibiotics with explainable deep learning. *Nature* (2023).
6. Lee, B. K., Mayhew, E. J., Sanchez-Lengeling, B., *et al.* A principal odor map unifies diverse tasks in olfactory perception. *Science* **381,** 999–1006 (2023).
7. Kroll, A., Ranjan, S., Engqvist, M. K. M. & Lercher, M. J. A general model to predict small molecule substrates of enzymes based on machine and deep learning. *Nat Commun* **14,** 2787 (2023).
8. Wu, Z., Ramsundar, B., Feinberg, E. N., Gomes, J., Geniesse, C., Pappu, A. S., Leswing, K. & Pande, V. MoleculeNet: a benchmark for molecular machine learning. *Chem Sci* **9,** 513–530 (2018).
9. Kapoor, S. & Narayanan, A. *Leakage and the Reproducibility Crisis in ML-based Science* 2022. arXiv: `2207.07048 [cs.LG]`.
10. Ball, P. Is AI leading to a reproducibility crisis in science? *Nature* **624,** 22–25 (2023).
11. Deng, J., Yang, Z., Wang, H., Ojima, I., Samaras, D. & Wang, F. A systematic study of key elements underlying molecular property prediction. *Nat Commun* **14,** 6395 (2023).
12. Walters, P. *We Need Better Benchmarks for Machine Learning in Drug Discovery* `https://practicalcheminformatics.blogspot.com/2023/08/we-need-better-benchmarks-for-machine.html` (2024).
13. Morgan, H. L. The generation of a unique machine description for chemical structures – a technique developed at chemical abstracts service. *J Chem Doc* **5,** 107–113 (1965).
14. Lajiness, M. S. Dissimilarity-based compound selection techniques. *Perspect Drug Discov Des* **7/8,** 65–84 (1997).
15. Flower, D. R. On the Properties of Bit String-Based Measures of Chemical Similarity. *J Chem Inf Comput Sci* **38,** 379–386 (1998).
16. Dixon, S. L. & Koehler, R. T. The hidden component of size in two-dimensional fragment descriptors: side effects on sampling in bioactive libraries. *J Med Chem* **42,** 2887–2900 (1999).
17. Godden, J. W., Xue, L. & Bajorath, J. Combinatorial preferences affect molecular similarity/diversity calculations using binary fingerprints and Tanimoto coefficients. *J Chem Inf Comput Sci* **40,** 163–166 (2000).
18. Martin, Y. C., Kofron, J. L. & Traphagen, L. M. Do structurally similar molecules have similar biological activity? *J Med Chem* **45,** 4350–4358 (2002).
19. Raymond, J. W. & Willett, P. Effectiveness of graph-based and fingerprint-based similarity measures for virtual screening of 2D chemical structure databases. *J Comput Aided Mol Des* **16,** 59–71 (2002).
20. Holliday, J. D., Salim, N., Whittle, M. & Willett, P. Analysis and display of the size dependence of chemical similarity coefficients. *J Chem Inf Comput Sci* **43,** 819–828 (2003).
21. Maggiora, G., Vogt, M., Stumpfe, D. & Bajorath, J. Molecular similarity in medicinal chemistry. *J Med Chem* **57,** 3186–3204 (2014).

22. Bajusz, D., Rácz, A. & Héberger, K. Why is Tanimoto index an appropriate choice for fingerprint-based similarity calculations? *J Cheminf* **7,** 20 (2015).
23. Raymond, J. W. & Willett, P. Maximum common subgraph isomorphism algorithms for the matching of chemical structures. *J Comput Aided Mol Des* **16,** 521–533 (2002).
24. Ehrlich, H.-C. & Rarey, M. Maximum common subgraph isomorphism algorithms and their applications in molecular science: a review. *Wiley Interdiscip Rev Comput Mol Sci* **1,** 68–79 (2011).
25. Schmidt, R., Krull, F., Heinzke, A. L. & Rarey, M. Disconnected Maximum Common Substructures under Constraints. *J Chem Inf Model* **61,** 167–178 (2021).
26. McInnes, L., Healy, J. & Melville, J. *UMAP: Uniform Manifold Approximation and Projection for Dimension Reduction* 2020. arXiv: `1802.03426`.
27. Ertl, P., Roggo, S. & Schuffenhauer, A. Natural product-likeness score and its application for prioritization of compound libraries. *J Chem Inf Model* **48,** 68–74 (2008).
28. Jayaseelan, K. V., Moreno, P., Truszkowski, A., Ertl, P. & Steinbeck, C. Natural product-likeness score revisited: an open-source, open-data implementation. *BMC Bioinf* **13,** 106 (2012).
29. Dobson, C. M. Chemical space and biology. *Nature* **432** (2004).
30. Van der Maaten, L. & Hinton, G. Visualizing Data using t-SNE. *J Mach Learning Res* **9,** 2579–2605 (2008).
31. Kruskal, J. B. & Wish, M. *Multidimensional scaling* **11** (Sage, 1978).
32. Chari, T., Banerjee, J. & Pachter, L. *The Specious Art of Single-Cell Genomics* 2021. eprint: `https://www.biorxiv.org/content/early/2021/08/26/2021.08.25.457696.full.pdf`.
33. Djoumbou-Feunang, Y., Eisner, R., Knox, C., *et al.* ClassyFire: automated chemical classification with a comprehensive, computable taxonomy. *J Cheminformatics* **8,** 61 (2016).
34. Sun, E. D., Ma, R. & Zou, J. Dynamic visualization of high-dimensional data. *Nat Comput Sci.* https://doi.org/10.1038/s43588-022-00380-4 (2022).
35. Subramanian, G., Ramsundar, B., Pande, V. & Denny, R. A. Computational Modeling of β-Secretase 1 (BACE-1) Inhibitors Using Ligand Based Approaches. *J Chem Inf Model* **56,** 1936–1949 (2016).
36. Martins, I. F., Teixeira, A. L., Pinheiro, L. & Falcao, A. O. A Bayesian Approach to in Silico Blood-Brain Barrier Penetration Modeling. *J Chem Inf Model* **52,** 1686–1697 (2012).
37. Novick, P. A., Ortiz, O. F., Poelman, J., Abdulhay, A. Y. & Pande, V. S. SWEETLEAD: an In Silico Database of Approved Drugs, Regulated Chemicals, and Herbal Isolates for Computer-Aided Drug Discovery. *PLOS ONE* **8** (2013).
38. Delaney, J. S. ESOL: Estimating Aqueous Solubility Directly from Molecular Structure. *J Chem Inf Comput Sci* **44,** 1000–1005 (2004).
39. Wenlock, M. & Tomkinson, N. *Experimental in vitro DMPK and physicochemical data on a set of publicly disclosed compounds.* `https://doi.org/10.6019/CHEMBL3301361`. 2015.
40. Kuhn, M., Letunic, I., Jensen, L. J. & Bork, P. The SIDER database of drugs and side effects. *Nucleic Acids Res* **44,** D1075–D1079 (2016).
41. Altae-Tran, H., Ramsundar, B., Pappu, A. S. & Pande, V. Low Data Drug Discovery with One-Shot Learning. *ACS Cent Sci* **3,** 283–293 (2017).
42. Domingo-Almenara, X., Guijas, C., Billings, E., Montenegro-Burke, J. R., Uritboonthai, W., Aisporna, A. E., Chen, E., Benton, H. P. & Siuzdak, G. The METLIN small molecule dataset for machine learning-based retention time prediction. *Nat Commun* **10,** 5811 (2019).
43. Richard, A. M., Judson, R. S., Houck, K. A., *et al.* ToxCast Chemical Landscape: Paving the Road to 21st Century Toxicology. *Chem Res Toxicol* **29,** 1225–1251 (2016).
44. Richard, A. M., Huang, R., Waidyanatha, S., *et al.* The Tox21 10K Compound Library: Collaborative Chemistry Advancing Toxicology. *Chem Res Toxicol* **34,** 189–216 (2021).
45. Huang, R., Xia, M., Nguyen, D.-T., Zhao, T., Sakamuru, S., Zhao, J., Shahane, S. A., Rossoshek, A. & Simeonov, A. Tox21Challenge to Build Predictive Models of Nuclear Receptor and Stress Response Pathways as Mediated by Exposure to Environmental Chemicals and Drugs. *Front Environ Sci* **3** (2016).

46. Wang, M., Carver, J. J., Phelan, V. V., *et al.* Sharing and community curation of mass spectrometry data with Global Natural Products Social Molecular Networking. *Nat Biotechnol* **34,** 828–837 (2016).
47. Horai, H., Arita, M., Kanaya, S., *et al.* MassBank: A public repository for sharing mass spectral data for life sciences. *J Mass Spectrom* **45,** 703–714 (2010).
48. Yang, K., Swanson, K., Jin, W., *et al.* Analyzing Learned Molecular Representations for Property Prediction. *J Chem Inf Model* **59,** 3370–3388 (2019).
49. Wang, Y., Wang, J., Cao, Z. & Barati Farimani, A. Molecular contrastive learning of representations via graph neural networks. *Nat Mach Intell* **4,** 279–287 (2022).
50. Englert, P. & Kovács, P. Efficient Heuristics for Maximum Common Substructure Search. *J Chem Inf Model* **55,** 941–955 (2015).
51. Agrawal, M., Kayal, N. & Saxena, N. PRIMES is in P. *Ann Math* **160,** 781–793 (2004).
52. Raymond, J. W., Gardiner, E. J. & Willett, P. RASCAL: Calculation of Graph Similarity using Maximum Common Edge Subgraphs. *Comput J* **45,** 631–644 (2002).
53. Johnson, M. *Relating metrics, lines and variables defined on graphs to problems in medicinal chemistry* in *Graph theory with applications to algorithms and computer science* (John Wiley & Sons, Inc., 1985), 457–470.
54. Seipp, J. *Fast Maximum Common Edge Subgraph Computation for Comparing Similar Molecular Structures* MA thesis (Dept. of Computer Science, Heinrich Heine University Düsseldorf, 2021).
55. Schmidt, R., Ehmki, E. S. R., Ohm, F., Ehrlich, H.-C., Mashychev, A. & Rarey, M. Comparing Molecular Patterns Using the Example of SMARTS: Theory and Algorithms. *J Chem Inf Model* **59,** 2560–2571 (2019).
56. Roy, B. Transitivité et connexité. *C R Acad Sci Paris,* 216–218 (1959).
57. Kim, H., Wang, M., Leber, C., Nothias, L.-F., Reher, R., Kang, K. B., van der Hooft, J. J. J., Dorrestein, P., Gerwick, W. & Cottrell, G. NPClassifier: A Deep Neural Network-Based Structural Classification Tool for Natural Products. *ChemRxiv,* 12885494.v1 (2020).
58. Kim, S., Thiessen, P. A., Bolton, E. E., *et al.* PubChem Substance and Compound databases. *Nucleic Acids Res* **44,** D1202–D1213 (2016).
59. Tanimoto, T. T. *An elementary mathematical theory of classification and prediction* tech. rep. (International Business Machines Corporation, New York, 1958).
60. Durant, J. L., Leland, B. A., Henry, D. R. & Nourse, J. G. Reoptimization of MDL keys for use in drug discovery. *J Chem Inf Comput Sci* **42,** 1273–1280 (2002).
61. Bolton, E. E., Wang, Y., Thiessen, P. A. & Bryant, S. H. in (eds Wheeler, R. A. & Spellmeyer, D. C.) 217–241 (Elsevier, 2008).
62. Rogers, D. & Hahn, M. Extended-connectivity fingerprints. *J Chem Inf Model* **50,** 742–754 (2010).
63. Capecchi, A., Probst, D. & Reymond, J.-L. One molecular fingerprint to rule them all: drugs, biomolecules, and the metabolome. *J Cheminformatics* **12,** 1–15 (2020).
64. Pattanaik, L. & Coley, C. W. Molecular Representation: Going Long on Fingerprints. *Chem Previews* **6,** 1204–1207 (2020).
65. Sandfort, F., Strieth-Kalthoff, F., Kühnemund, M., Beecks, C. & Glorius, F. A Structure-Based Platform for Predicting Chemical Reactivity. *Chem* **6,** 1379–1390 (2020).
66. Kanehisa, M., Sato, Y., Kawashima, M., Furumichi, M. & Tanabe, M. KEGG as a reference resource for gene and protein annotation. *Nucleic Acids Res* **44,** D457–D462 (2016).
67. Hastings, J., de Matos, P., Dekker, A., *et al.* The ChEBI reference database and ontology for biologically relevant chemistry: enhancements for 2013. *Nucleic Acids Res* **41,** D456–D463 (2013).
68. Wishart, D. S., Feunang, Y. D., Marcu, A., *et al.* HMDB 4.0: the human metabolome database for 2018. *Nucleic Acids Res* **46,** D608–D617 (2018).
69. Ramirez-Gaona, M., Marcu, A., Pon, A., Guo, A. C., Sajed, T., Wishart, N. A., Karu, N., Djoumbou Feunang, Y., Arndt, D. & Wishart, D. S. YMDB 2.0: a significantly expanded version of the yeast metabolome database. *Nucleic Acids Res* **45,** D440–D445 (D1 2017).
70. Hawkins, C., Ginzburg, D., Zhao, K., *et al.* Plant Metabolic Network 15: A resource of genome-wide metabolism databases for 126 plants and algae. *J Integr Plant Biol* **63,** 1888–1905 (2021).

71. Caspi, R., Altman, T., Billington, R., *et al.* The MetaCyc database of metabolic pathways and enzymes and the BioCyc collection of Pathway/Genome Databases. *Nucleic Acids Res* **42,** D459–D471 (2014).
72. Shinbo, Y., Nakamura, Y., Altaf-Ul-Amin, M., Asahi, H., Kurokawa, K., Arita, M., Saito, K., Ohta, D., Shibata, D. & Kanaya, S. in *Plant Metabolomics* (eds Saito, K., Dixon, R. A. & Willmitzer, L.) 165–181 (Springer-Verlag, 2006).
73. Gu, J., Gui, Y., Chen, L., Yuan, G., Lu, H.-Z. & Xu, X. Use of natural products as chemical library for drug discovery and network pharmacology. *PLoS One* **8,** 1–10 (2013).
74. Weber, R. J. M., Li, E., Bruty, J., He, S. & Viant, M. R. MaConDa: A publicly accessible mass spectrometry contaminants database. *Bioinformatics* **28,** 2856–2857 (2012).
75. Fonger, G. C., Hakkinen, P., Jordan, S. & Publicker, S. The National Library of Medicine's (NLM) Hazardous Substances Data Bank (HSDB): background, recent enhancements and future plans. *Toxicology* **325,** 209–216 (2014).
76. Banerjee, P., Erehman, J., Gohlke, B., Wilhelm, T., Preissner, R. & Dunkel, M. Super Natural II - a database of natural products. *Nucleic acids research* **43,** D935–D939 (2015).
77. Sorokina, M., Merseburger, P., Rajan, K., Yirik, M. A. & Steinbeck, C. COCONUT online: Collection of Open Natural Products database. *J Cheminf* **13,** 2 (2021).
78. Mohammed Taha, H., Aalizadeh, R., Alygizakis, N., *et al.* The NORMAN Suspect List Exchange (NORMAN-SLE): facilitating European and worldwide collaboration on suspect screening in high resolution mass spectrometry. *Environ Sci Eur* **34,** 104 (2022).
79. Nelson, S. J., Johnston, W. D. & Humphreys, B. L. in *Relationships in the organization of knowledge* (eds Bean, C. A. & Green, R.) 171–184 (Kluwer Academic Publishers, 2001).
80. Gaulton, A., Hersey, A., Nowotka, M., *et al.* The ChEMBL database in 2017. *Nucleic Acids Res* **45,** D945–D954 (2017).
81. Schymanski, E. L., Kondić, T., Neumann, S., Thiessen, P. A., Zhang, J. & Bolton, E. E. Empowering large chemical knowledge bases for exposomics: PubChemLite meets MetFrag. *J Cheminf* **13,** 19 (2021).
82. Mendez, D., Gaulton, A., Bento, A. P., Chambers, J., De Veij, M., Félix, E., Magariños, M. P., Mosquera, J. F., Mutowo, P., Nowotka, M., *et al.* ChEMBL: towards direct deposition of bioassay data. *Nucleic Acids Res* **47,** D930–D940 (2019).
83. Dührkop, K., Shen, H., Meusel, M., Rousu, J. & Böcker, S. Searching molecular structure databases with tandem mass spectra using CSI:FingerID. *Proc Natl Acad Sci USA* **112,** 12580–12585 (2015).
84. Dührkop, K., Fleischauer, M., Ludwig, M., Aksenov, A. A., Melnik, A. V., Meusel, M., Dorrestein, P. C., Rousu, J. & Böcker, S. SIRIUS 4: a rapid tool for turning tandem mass spectra into metabolite structure information. *Nat Methods* **16,** 299–302 (2019).
85. Karp, R. M. in *Complexity of Computer Computations* (eds Miller, R. E. & Thatcher, J. W.) 85–103 (Plenum Press, New York, 1972).
86. Koch, I. Enumerating all connected maximal common subgraphs in two graphs. *Theoret Comput Sci* **250,** 1–30 (2001).
87. Bahiense, L., Manić, G., Piva, B. & de Souza, C. C. The maximum common edge subgraph problem: A polyhedral investigation. *Discrete Appl Math* **160,** 2523–2541 (2012).
88. Schietgat, L., Ramon, J. & Bruynooghe, M. A polynomial-time maximum common subgraph algorithm for outerplanar graphs and its application to chemoinformatics. *Ann Math Artif Intell* **69,** 343–376 (2013).
89. Akutsu, T. & Tamura, T. *A Polynomial-Time Algorithm for Computing the Maximum Common Subgraph of Outerplanar Graphs of Bounded Degree* in *Mathematical Foundations of Computer Science (MFCS 2012)* (eds Rovan, B., Sassone, V. & Widmayer, P.) (Springer, Berlin, Heidelberg, 2012), 76–87.
90. Nouleho Ilemo, S., Barth, D., David, O., Quessette, F., Weisser, M.-A. & Watel, D. Improving graphs of cycles approach to structural similarity of molecules. *PloS one* **14,** e0226680 (2019).
91. Edmonds, J. & Karp, R. M. Theoretical Improvements in Algorithmic Efficiency for Network Flow Problems. *J ACM* **19,** 248–264 (1972).

92. Bunke, H. & Shearer, K. A graph distance metric based on the maximal common subgraph. *Pattern Recogn Lett* **19,** 255–259 (1998).
93. Matthews, R. Storks Deliver Babies ($p = 0.008$). *Teaching Stat* **22,** 36–38 (2000).
94. Ramakrishnan, R., Dral, P. O., Rupp, M. & von Lilienfeld, O. A. Quantum chemistry structures and properties of 134 kilo molecules. *Scientific Data* **1,** 140022 (2014).
95. Dührkop, K., Nothias, L. F., Fleischauer, M., *et al.* Systematic classification of unknown metabolites using high-resolution fragmentation mass spectra. *Nat Biotechnol* **39,** 462–471 (2021).
96. Hoffmann, M. A., Nothias, L.-F., Ludwig, M., Fleischauer, M., Gentry, E. C., Witting, M., Dorrestein, P. C., Dührkop, K. & Böcker, S. High-confidence structural annotation of metabolites absent from spectral libraries. *Nat Biotechnol* **40,** 411–421 (2022).
97. Dunn, T. B., Seabra, G. M., Kim, T. D., Juárez-Mercado, K. E., Li, C., Medina-Franco, J. L. & Miranda-Quintana, R. A. Diversity and Chemical Library Networks of Large Data Sets. *J Chem Inf Model.* https://doi.org/10.1021/acs.jcim.1c01013 (2021).
98. Skinnider, M. A., Stacey, R. G., Wishart, D. S. & Foster, L. J. Chemical language models enable navigation in sparsely populated chemical space. *Nat Mach Intell* **3,** 759–770 (2021).
99. Kullback, S. & Leibler, R. A. On Information and Sufficiency. *Ann Math Stat* **22,** 79–86 (1951).
100. Pedregosa, F., Varoquaux, G., Gramfort, A., *et al.* Scikit-learn: Machine Learning in Python. *J Mach Learn Res* **12,** 2825–2830 (2011).
101. Virtanen, P., Gommers, R., Oliphant, T. E., *et al.* SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nat Methods* **17,** 261–272 (2020).
102. Kruskal, J. On the Shortest Spanning Subtree of a Graph and the Traveling Salesman Problem. *Proc Amer Math Soc* **7,** 48–50 (1956).

Supplement

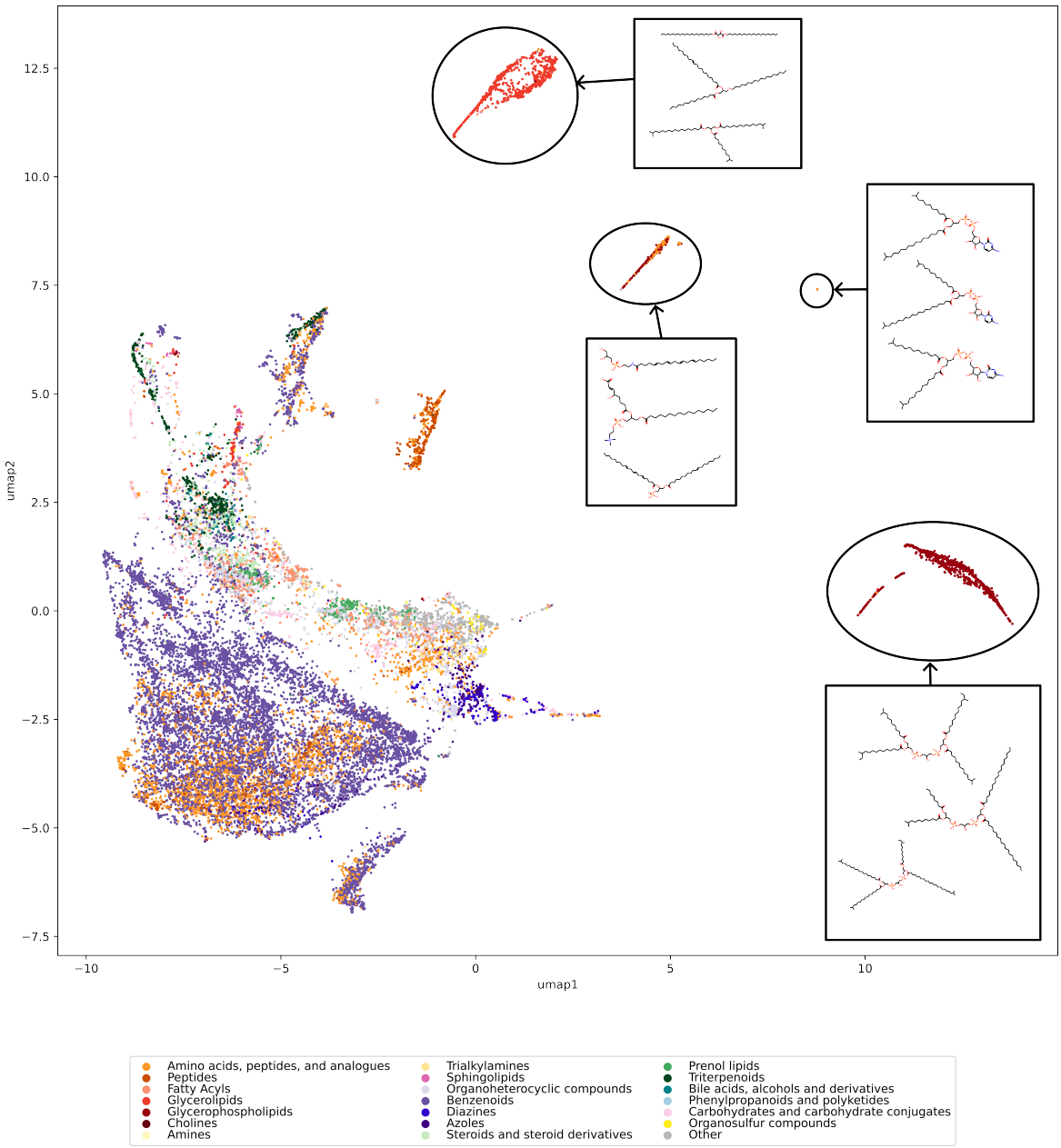

Fig. 6. Supplement: Initial map of biomolecular structures. Outlier clusters are highlighted and annotated with three exemplary structures drawn at random for each cluster. Different from Fig. 1 and Supplementary Fig. 8, the UMAP embedding was computed from all 19,994 subsampled biomolecular structures. This includes the 1898 molecular structures removed from Fig. 1. In contrast, Supplementary Fig. 8 shows the 1898 molecular structures added back into the UMAP embedding computed from 18,096 molecular structures. See Fig. 1 for color-coding.

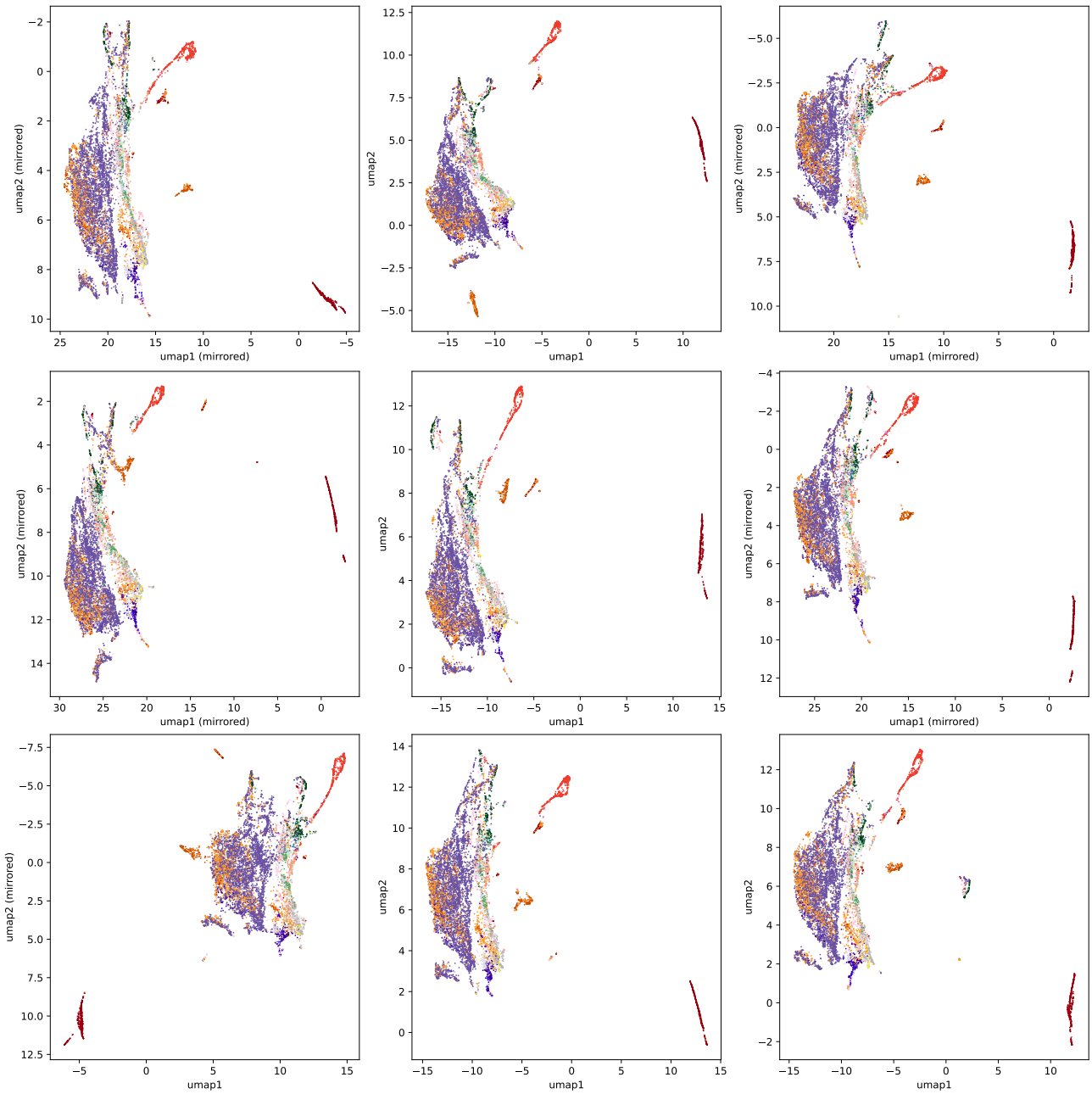

Fig. 7. Supplement: UMAP embeddings vary when other biomolecular structures are subsampled. It is understood that the (random and uniform) choice of 19,994 biomolecular structures affects the UMAP embedding. To investigate the effect of subsampling, we further uniformly subsubsampled nine times 10,000 structures, and created the corresponding UMAP embeddings. Compound classes are color-coded as in Fig. 1. For easier visual inspection, some of the plots have been mirrored, as indicated.

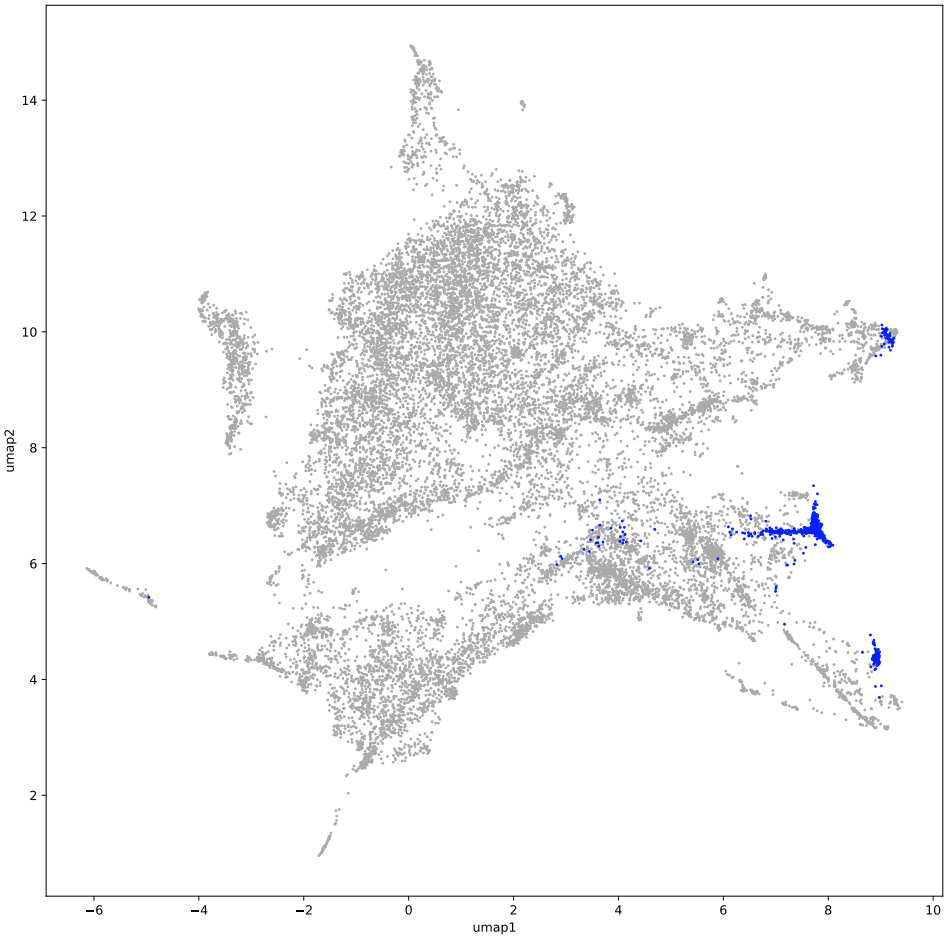

Fig. 8. Supplement: Full map of biomolecular structures. We did not recompute the UMAP embedding for this figure. Rather, the same UMAP embedding as in Fig. 1 was used. UMAP embedding computed from the 18,096 molecular structures. Shown in this plot are all 19,994 molecular structures subsampled from the biomolecular structures. Highlighted are the 1898 molecular structures not shown in Fig. 1. Compare to Supplementary Fig. 6.

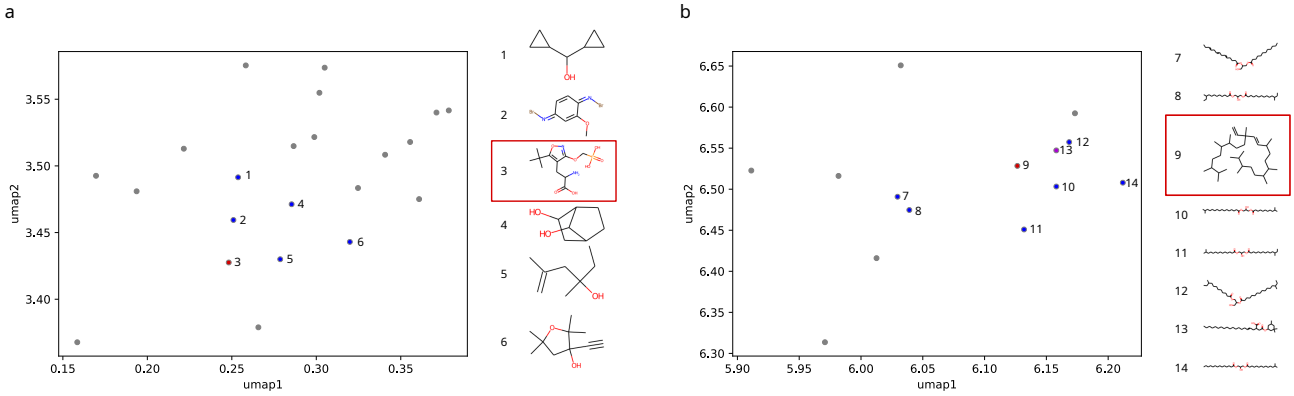

Fig. 9. Supplement: UMAP embeddings are clearly far from perfect. Both examples show a section of the MCES-based UMAP embedding of biomolecular structures (compare to Fig. 1). (a) This section of the embedding contains structures that do not share much similarity with one another, but with seemingly no better possible placement by UMAP. For instance, the structure labeled "3" is a member of the SMRT dataset mapped onto the UMAP space, with large myopic MCES distances (from 17.5 to 20.5) to all other labeled structures. (b) In this example, the structure labeled "9" clearly does not belong in this cluster of fatty acyls. A possible explanation for the strange placement might be, that the "outlier"-structure has approximately the same myopic MCES distance to all other labeled compounds (18 or 19 except for structure "13", where the distance is 15); this is similar for the structure labeled "13" (albeit with a smaller distance: 10), which intuitively seems to be structurally much more closely related to the cluster. Numerous similar examples to those two cases can be found throughout the embedding.

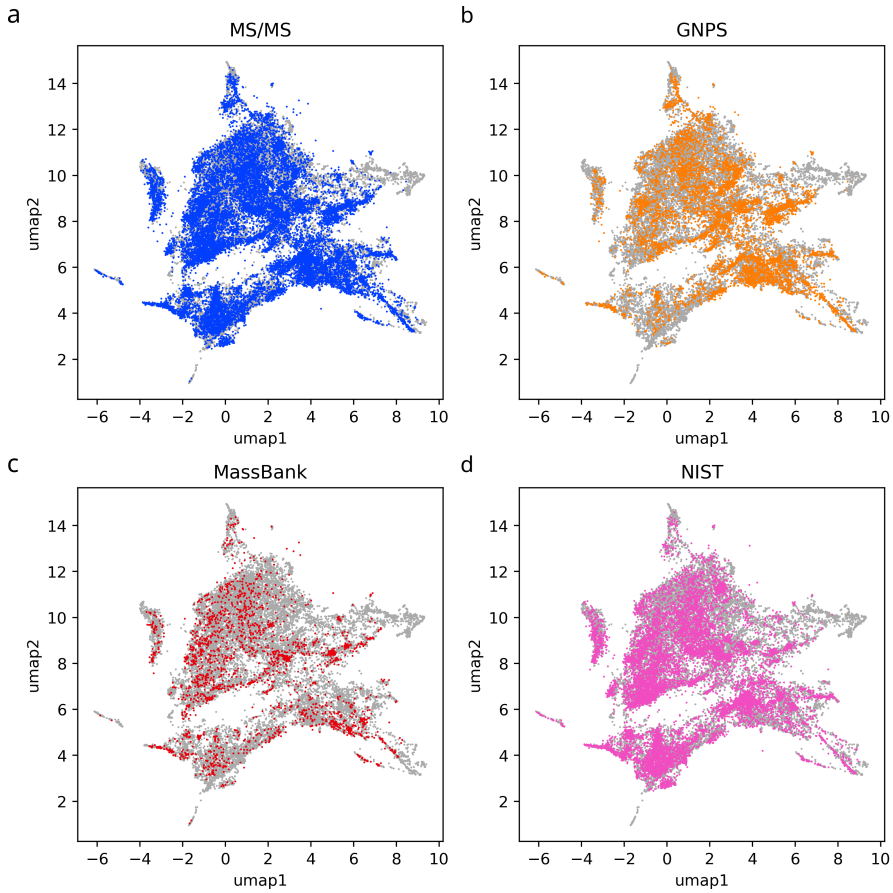

Fig. 10. Supplement: Maps of molecular structures in the MS/MS training dataset. Shown are both the total 18,848 molecular structures in the dataset (a) as well as those of the individual spectral libraries GNPS (b), MassBank (c), and NIST (d). A total of 36,944 molecular structures are shown in all plots combined. We stress that the assignment of molecular structures to the three libraries is not perfect: A molecular structure might be contained in more than one library but is displayed only for one. Hence, the plots *must not be overinterpreted to compare the coverage of the three libraries*. What we can learn from the three library plots is that each library covers the universe of biomolecular structures rather well. We use the same UMAP embedding as in Fig. 1. No molecular structures fall outside of this plot. Biomolecular structures are shown in light gray.

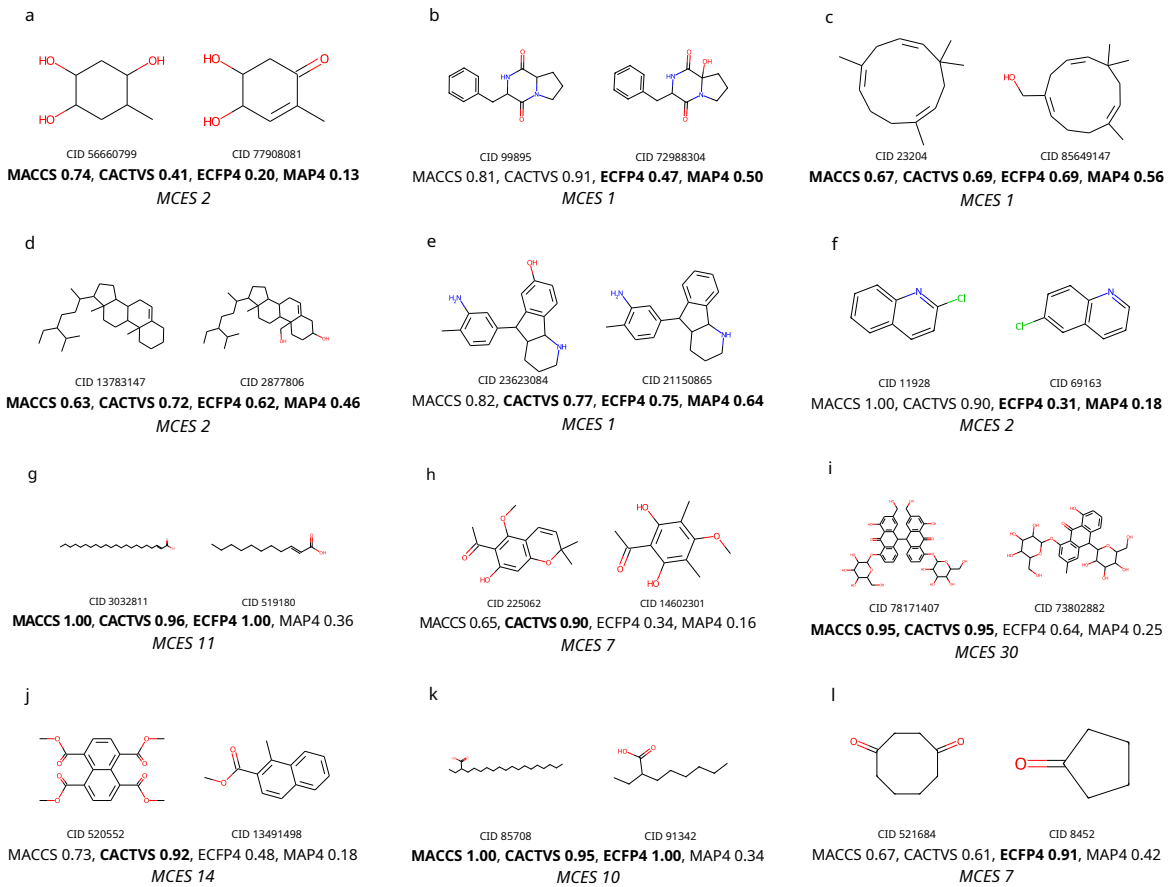

Fig. 11. Supplement: Tanimoto coefficients can differ substantially from perceived structural similarity. Shown are Tanimoto coefficients for MACCS, CACTVS, ECFP4 (Morgan), and MAP4 fingerprints; high values imply high similarity. The corresponding Jaccard distance is one minus this value. (a–f) Molecular structure pairs with high perceived structural similarity but relatively low Tanimoto coefficient, for one or more fingerprint types. (g–l) Molecular structure pairs with low perceived structural similarity but very high Tanimoto coefficient for at least one fingerprint type. All molecular structures are biomolecular structures; "CID" is the PubChem compound identifier number. It is understood that similar "pathological cases" exist for other fingerprint type. It is also understood that the problem cannot be solved by using another fingerprint-based similarity measure, such as the Sørensen-Dice coefficient. For comparison, we also report the myopic MCES distance between each molecular structure pair.

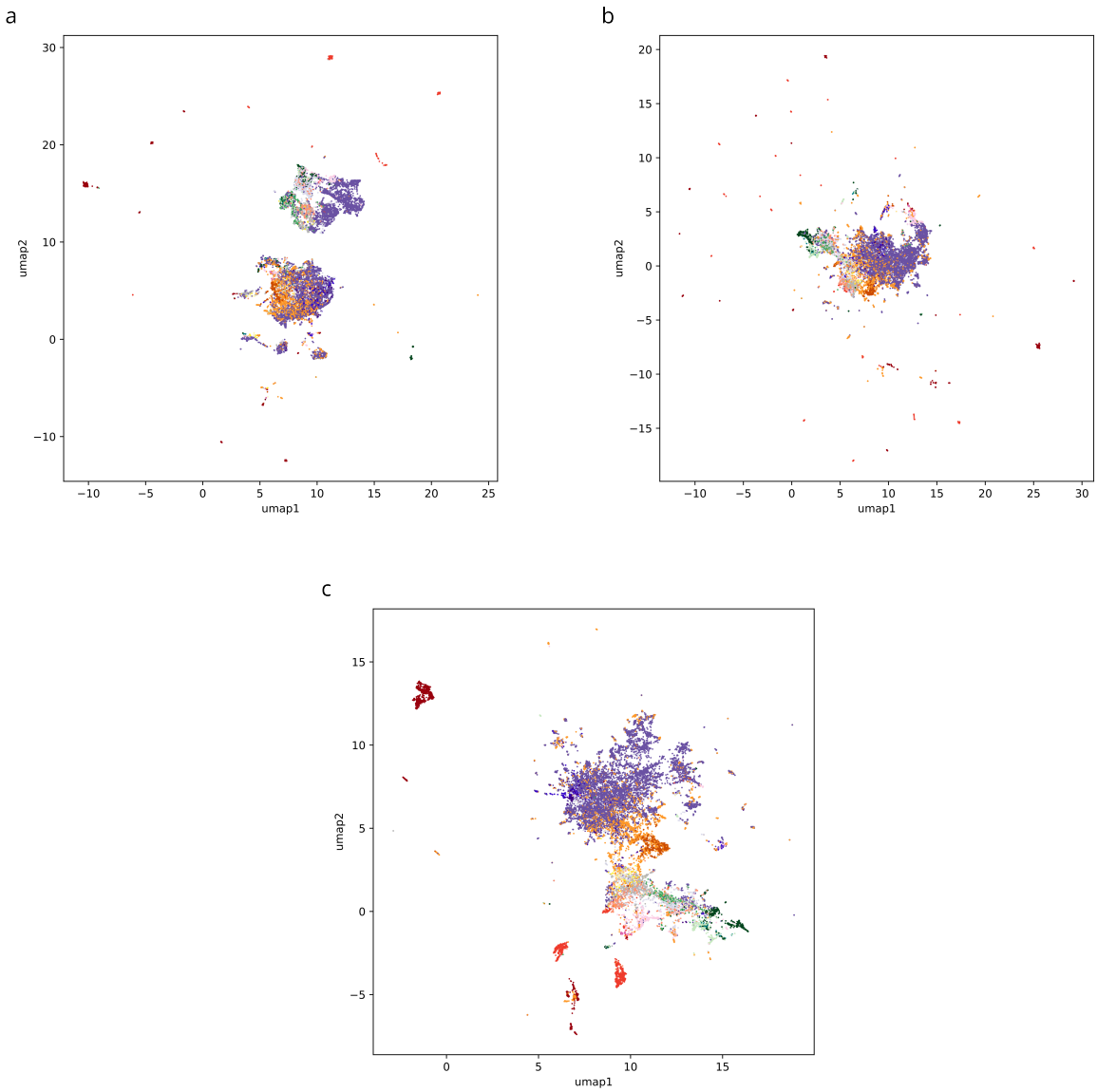

Fig. 12. Supplement: UMAP embedding using Tanimoto coefficients. We use and display all 19,994 biomolecular structures. Jaccard distances (one minus Tanimoto coefficients) were computed from MACCS fingerprints (a), ECFP4 (Morgan) fingerprints (b), and MAP4 fingerprints (c). The resulting distances are used to create corresponding UMAP embeddings. Compound classes are color-coded as in Fig. 1. Different from the MCES-based plots, for both types of fingerprints we observe numerous outlier clusters, as well as singleton outliers. Certain compound classes are separated in distant clusters. Compare to Supplementary Fig. 6 where outliers were not removed.

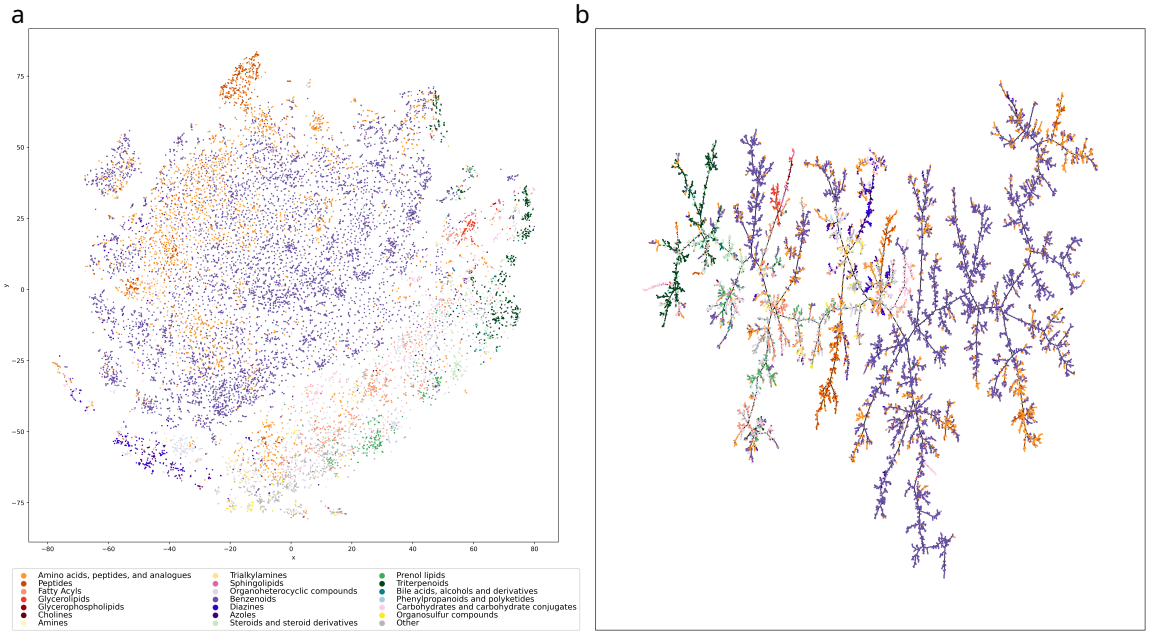

Fig. 13. Supplement: t-SNE and Minimum Spanning Tree visualizations. As alternative visualization methods to UMAP, t-SNE (a) and Minimum Spanning Tree computation (b) are applied to the pairwise myopic MCES distances of the 18,096 biomolecular structures (outlier lipid clusters excluded). The color-coding of compound classes from Fig. 1 is used. t-SNE computations are performed with scikit-learn[100], using the default settings (version 1.1.3). The Minimum Spanning Tree is computed using SciPy[101] version 1.10.1, based on Kruskal's algorithm[102].

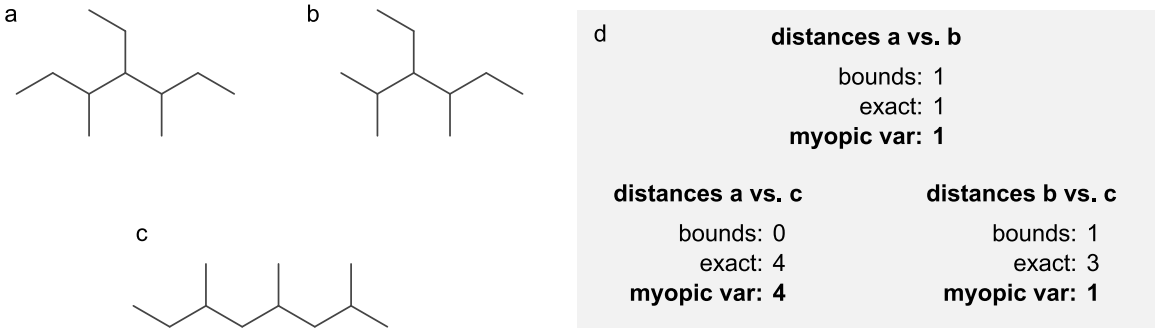

Fig. 14. Supplement: Double thresholding is needed to enforce the triangle inequality. Different from the myopic MCES distance used in the rest of the paper, we now consider a variant ("myopic var") that does not employ double thresholding: We again first compute the MCES bounds, then decide whether the distance bound is below T. In this case, we compute the exact MCES distance. Different from the rest of the paper, we then use this exact MCES distance, *even if it is larger than* T. As shown in (d), the resulting distance violates the triangle inequality for molecular structures (a,b,c): For $T = 1$ the "myopic var" distance between a and c is 4, which is strictly larger than the sum of "myopic var" distances from a to b and from b to c. The problem is resolved for the myopic MCES distance as there, the distance between a and c is only $\min\{4, T\} = 1$.

Table 2. Supplement: Structure databases that contribute to the set of biomolecular structures used herein. For each database the total number of molecular structures (total) and the number of structures that are absent of all preceding databases (additional) is shown. A subset of structures from PubChem (PC) which are annotated with categories of biological relevance is included.

Database	total	additional	Ref.	URL
KEGG	14,362	14,362	[66]	https://www.kegg.jp
ChEBI	56,219	42,621	[67]	https://www.ebi.ac.uk/chebi
HMDB	95,973	85,942	[68]	https://www.hmdb.ca
YMDB	1,864	135	[69]	http://www.ymdb.ca
PlantCyc	5,503	1,072	[70]	https://plantcyc.org/
MetaCyc	13,059	3,175	[71]	https://metacyc.org
KNApSAcK	42,168	31,967	[72]	http://www.knapsackfamily.com/KNApSAcK/
UNPD	161,976	112,057	[73]	http://pkuxxj.pku.edu.cn/UNPD [offline]
MaConDa	91	31	[74]	https://www.maconda.bham.ac.uk/
HSDB	4,319	1,080	[75]	ftp://ftp.nlm.nih.gov/nlmdata/.hsdblease/
Super Natural II	227,931	95,291	[76]	http://bioinformatics.charite.de/supernatural
COCONUT	380,169	104,404	[77]	https://coconut.naturalproducts.net
NORMAN suspect list	64,121	47,251	[78]	https://www.norman-network.com/nds/SLE/
MeSH-annotated PC	82,337	37,523	[58, 79]	https://pubchem.ncbi.nlm.nih.gov
PC “bio and metabolites”	138,753	30,500	[58]	https://pubchem.ncbi.nlm.nih.gov/classification
PC “drug”	7,649	651	[58]	https://pubchem.ncbi.nlm.nih.gov/classification
PC “safety and toxic”	148,962	109,792	[58]	https://pubchem.ncbi.nlm.nih.gov/classification
PC “food”	4,305	243	[58]	https://pubchem.ncbi.nlm.nih.gov/classification

Table 3. Supplement: Molecular structure dataset sizes. Number of molecular structures in the considered molecular structure training datasets. Some molecular structures failed the standardization via PubChem and were discarded, see column “After standardization”. Few molecular structures resulted in failed MCES computations and were also discarded, see column “After MCES computations”. For the *SMRT* dataset we uniformly subsampled 10,000 standardized molecular structures. The number “Total in plot” refers to the union of the molecular structures in the dataset and the 18,096 biomolecular structures.

Dataset	Number of compounds	After standardization	After MCES computations	Total in plot	ref.
BACE	1513	1194	1194	19,290	[35]
BBBP	2039	1662	1662	19,758	[36]
ClinTox	1478	1476	1475	19,571	[37]
Delaney	1128	1128	1127	19,223	[38]
Lipo	4200	4200	4200	22,296	[39]
SIDER	1427	1420	1389	19,485	[40, 41]
SMRT	80,037	10,000	10,000	28,096	[42]
ToxCast	8576	8517	8502	26,598	[43]
Tox21	7831	7783	7763	25,859	[44, 45]
MS/MS	18,848	18,848	18,848	36,944	[46, 47]

	biomol.	BACE	BBBP	ClinTox	Delaney	Lipo	SIDER	Tox21	ToxCast	SMRT	MS/MS
Fatty acid esters	14,7%	1	23	54	36	24	44	287	291	2,706	1,377
Carboxylic acids	12.3%	25	189	265	0	448	313	911	1,126	1,362	4,148
Polyols	12,0%	1	65	73	23	49	89	221	253	88	2,867
Prenol lipids	11.3%	4	13	36	23	18	31	245	256	223	1,833
Acetals	10,7%	14	108	80	18	47	86	255	273	2,060	2,410
Carbohydrates and carbohydrate conjugates	10.3%	0	64	78	19	34	91	207	235	440	2,447
Lactones	9,7%	2	40	57	15	38	47	184	197	355	2,162
Pyrans	9.4%	1	12	17	10	71	19	85	90	462	2,297
Benzopyrans	9.2%	47	21	20	11	79	16	102	113	571	2,339
Trialkylamines	9.0%	51	550	291	1	995	281	838	900	16,206	1,428
Aralkylamines	8.9%	595	371	257	6	767	270	636	693	5,814	1,604
Cyclic alcohols and derivatives	8.8%	1	178	139	39	33	150	264	288	229	2,002
1-benzopyrans	8.8%	47	21	20	11	78	16	100	110	548	2,255
Glycosyl compounds	8.4%	0	51	55	11	26	62	141	155	77	2,164
Dicarboxylic acids and derivatives	8.0%	6	82	114	11	23	118	422	508	256	1,670
Tertiary alcohols	7,5%	5	170	118	42	92	117	299	318	379	1,568
Pyranones and derivatives	7.2%	0	10	12	9	70	9	70	78	379	1,946
1-hydroxy-4-unsubstituted benzenoids	7.2%	3	44	45	25	83	43	251	264	275	1,852
Aryl ketones	7.1%	11	91	58	11	139	49	294	317	1,227	1,546
Monosaccharides	7,0%	0	49	58	9	23	66	145	170	425	1,586
Organic phosphoric acids and derivatives	6.9%	0	3	19	6	2	22	92	99	13	461
O-glycosyl compounds	6,8%	0	44	42	8	1	43	102	107	27	1,883
Phosphate esters	6.8%	0	3	16	6	2	17	84	86	8	439
Alkyl phosphates	6,6%	0	3	13	5	2	14	67	71	3	422
Pyrroles	6.6%	129	66	63	5	378	85	170	186	9,364	1,312
Cyclic ketones	6.3%	2	168	121	50	51	125	285	312	589	1,051
Vinylogous acids	6.0%	0	51	41	10	57	37	179	203	330	1,465
Piperidines	5.9%	48	241	133	7	718	129	391	422	20,264	1,171
Indoles and derivatives	5.6%	79	64	60	5	256	81	173	182	4,420	1,376
Glycerophospholipids	5.5%	0	0	2	0	0	1	5	6	0	217
Dialkyl phosphates	5.5%	0	0	3	2	0	5	28	30	0	250
Dialkylamines	5.4%	530	139	138	1	404	147	342	385	4,459	831
Tricarboxylic acids and derivatives	5.3%	2	7	12	1	1	18	73	74	13	368
Alpha,beta-unsaturated carboxylic esters	5.2%	7	12	27	7	11	24	191	198	2,572	1,075
Enoate esters	5.2%	7	12	27	7	11	24	189	196	2,572	1,074
Pyrrolidines	5.0%	92	113	85	7	193	121	215	247	10,646	867
Tetracarboxylic acids and derivatives	4.9%	1	3	7	0	0	10	24	27	2	162
Steroids and steroid derivatives	4.9%	0	129	119	48	14	116	231	244	64	1,130
Tetrahydrofurans	4.5%	0	22	41	12	15	43	138	146	831	833
Glycerolipids	4.5%	0	2	3	2	1	3	24	20	1	145
Indoles	4.4%	70	53	49	2	228	68	138	148	2,713	1,181
Naphthalenes	4.4%	3	44	39	46	88	41	207	234	421	740
Monoalkylamines	4.3%	42	96	117	1	132	150	281	331	417	1,447
Glycerophosphoglycerophosphoglycerols	4.2%	0	0	0	0	0	0	0	0	0	0
Cardiolipins	4.1%	0	0	0	0	0	0	0	0	0	0
Organic salts	4.1%	7	77	81	11	46	133	441	1,623	344	593
Furans	4.0%	10	24	17	8	81	16	84	88	4,250	451
Substituted pyrroles	3.9%	106	26	43	2	271	61	118	130	7,531	885
N-acyl-alpha amino acids and derivatives	3.9%	55	121	118	0	106	139	164	201	945	1,358
Chromones	3,9%	0	3	4	4	50	6	33	34	207	1,134
Aryl alkyl ketones	3.8%	9	63	30	5	83	23	168	177	848	833
Triradylcglycerols	3.8%	0	0	0	1	0	1	4	3	0	21
Methyl esters	3.8%	4	19	23	18	30	28	167	179	1,737	783
Alpha amino acid amides	3.8%	65	55	89	4	149	121	128	138	6,551	978
Alkanolamines	3.7%	524	147	134	6	215	135	317	356	743	530
Triacylglycerols	3.6%	0	0	0	1	0	1	4	3	0	21
Amino acids	3.4%	7	83	127	0	160	156	242	269	609	1,220
Coumarins and derivatives	3.4%	0	7	6	7	21	3	26	31	151	874
Aldehydes	3.4%	1	17	6	16	10	10	136	153	94	527
Triterpenoids	3.3%	0	0	7	0	0	5	22	25	4	505
Organobromides	3.3%	33	34	13	49	95	14	203	218	2,943	233
Sulfenyl compounds	3.2%	11	66	53	49	217	58	250	272	14,691	529
Benzofurans	3.2%	1	11	12	6	23	11	53	61	406	322
Diazinanes	3.1%	8	167	85	34	406	69	188	206	10,959	436
Hydroxysteroids	3.0%	0	122	81	37	9	91	158	169	43	804
Tertiary alkylarylamines	3.0%	14	158	90	34	617	81	265	296	16,162	486
Styrenes	2,9%	12	12	14	6	19	11	110	120	241	753
Fluorobenzenes	2.9%	421	77	37	13	446	53	171	184	13,321	430
N-arylamides	2,9%	21	20	34	13	256	30	133	143	11,714	342
Alkaloids and derivatives	2.8%	0	89	40	6	79	46	139	158	482	726
Sesquiterpenoids	2,8%	0	0	5	3	1	4	42	47	17	386
Dialkylarylamines	2.8%	14	91	62	22	585	63	214	241	15,677	436
Substituted imidazoles	2.7%	30	72	72	8	317	93	222	267	7,902	557
Piperazines	2.7%	7	137	73	1	399	60	169	184	10,321	399
Diterpenoids	2,6%	0	2	10	1	3	10	38	37	16	489
Allyl-type 1,3-dipolar organic compounds	2.6%	5	25	30	57	54	31	337	369	654	331
3-alkylindoles	2.6%	41	37	39	1	65	57	95	100	1,318	616
Benzenediols	2.6%	0	12	22	5	15	19	67	74	40	666
Flavonoids	2.6%	0	1	1	2	7	3	20	21	60	841
Aryl bromides	2.6%	33	22	10	26	95	14	114	123	2,906	211
Fatty acids and conjugates	2.5%	5	23	44	0	12	57	163	251	523	1,209
Benzoic acid esters	2.5%	3	14	20	20	36	16	224	221	849	516
Organic nitro compounds	2.5%	5	24	29	57	54	29	332	364	649	316
Benzamides	2,4%	163	34	47	6	449	42	150	156	6,092	346
Organic oxoazanium compounds	2.4%	5	24	25	56	54	24	321	341	649	306
1,2-aminoalcohols	2,4%	523	122	106	1	187	108	252	279	406	364
Methoxyphenols	2.4%	0	3	4	4	6	4	37	38	113	552
C-nitro compounds	2,4%	5	24	25	56	54	24	318	337	649	304
Hydropyridines	2.4%	38	28	23	4	144	20	56	63	4,569	309
Quinolines and derivatives	2.4%	24	48	51	11	370	44	165	179	2,965	515
Hexoses	2.3%	0	9	13	1	0	16	31	29	21	741
N-substituted imidazoles	2.3%	28	70	69	8	278	76	204	250	7,811	451
Peptides	2.3%	15	34	59	0	49	100	73	82	714	886
Oxolanes	2.3%	8	2	8	0	34	11	25	25	750	355
Phenolic glycosides	2.3%	0	4	3	2	0	8	14	13	15	592
Nitroaromatic compounds	2.3%	5	18	23	51	52	22	293	311	640	287
Hydroxy acids and derivatives	2.3%	1	11	24	0	8	34	55	89	8	604
N-alkylpyrrolidines	2.2%	37	90	34	0	102	40	122	144	2,001	322
Cyclohexenones	2.2%	0	48	46	28	9	44	101	113	303	412
Hydroxyflavonoids	2.2%	0	1	0	2	6	2	14	15	47	671
Coumarans	2.1%	0	36	20	5	12	24	36	43	71	265
Acetamides	2.1%	164	32	31	12	127	31	94	95	1,457	340
Dimethoxybenzenes	2.1%	4	12	7	1	80	12	58	66	3,275	467
Epoxides	2.1%	0	9	11	3	2	17	68	75	25	347
Monoterpenoids	2.1%	4	11	11	19	14	10	135	139	186	418
Aromatic alcohols	2.0%	7	55	62	17	134	59	170	190	377	442
Carboximidamides	2.0%	408	33	54	7	50	74	125	139	15,530	320
Gamma butyrolactones	2.0%	0	4	8	4	4	9	31	30	51	382
⋮											
Benzenesulfonyl compounds	2,0%	12	28	46	12	421	45	258	281	5,850	308
Aminopyrimidines and derivatives	1.9%	35	43	63	13	720	70	126	137	6,917	510
Organic anions	1,9%	2	32	63	16	45	48	330	348	11	631
N-alkylpiperazines	1.9%	4	107	56	0	223	45	132	143	4,488	269
Beta hydroxy acids and derivatives	1,9%	1	11	20	0	8	29	38	67	8	489
Primary carboxylic acid amides	1.9%	8	33	48	5	194	80	104	117	641	442
Organic zwitterions	1,9%	6	33	25	15	34	56	150	359	454	343
N-acyl-alpha amino acids	1.9%	8	6	26	0	25	48	49	57	88	999
Quaternary ammonium salts	1,9%	3	36	68	0	17	46	269	291	7	418
1,2-diols	1.8%	1	9	13	7	36	19	48	65	30	391
Phenylketones	1.8%	8	54	28	4	69	26	154	167	468	415
Oxosteroids	1.8%	0	122	84	41	12	90	152	157	42	499
Carbamic acids and derivatives	1.8%	8	57	45	26	113	49	122	123	576	235
Hemiacetals	1.8%	0	8	20	4	1	20	40	35	8	242
Carbamate esters	1.7%	8	57	45	26	113	49	120	121	576	235
Carboxylic acid imides	1,7%	0	68	33	54	55	29	104	108	2,105	267
Pyrazoles	1.7%	36	23	13	6	351	17	61	65	7,997	263
Nitrobenzenes	1.7%	5	7	13	42	33	14	231	247	471	221
Aryl thioethers	1.7%	9	102	48	11	201	29	143	156	4,310	199
Vinylogous esters	1.7%	0	4	3	6	8	3	22	25	222	444
5-hydroxyflavonoids	1.7%	0	1	0	2	2	2	14	15	39	515
Cinnamic acids and derivatives	1,7%	2	3	6	1	9	4	41	40	63	549
Disaccharides	1.6%	0	9	11	4	0	15	33	36	4	519
Benzylamines	1.6%	230	23	26	1	229	20	84	92	2,168	191
Dihydrofurans	1.6%	0	5	7	2	1	4	23	26	414	263
Pyridinones	1.6%	10	7	10	1	131	8	20	22	2,162	241
Dicarboximides	1.6%	0	66	33	52	45	28	90	94	2,032	246
Phenethylamines	1.6%	455	34	57	0	189	85	113	125	189	570
Fatty acyl glycosides	1.6%	0	0	2	0	0	6	13	14	7	410
Diazanaphthalenes	1.6%	5	19	27	6	280	24	78	84	4,645	271
Thiophenes	1.6%	52	28	28	2	246	33	83	91	7,254	171
Alpha,beta-unsaturated carbonyl compounds	1.6%	1	14	17	10	16	17	78	84	1,244	354
Benzodioxoles	1.6%	14	9	9	3	40	12	50	52	1,856	262
Bromobenzenes	1.6%	23	11	6	22	42	9	84	88	2,352	116
Thiazoles	1.6%	25	42	29	7	276	33	122	134	5,998	195
Terpene lactones	1.5%	0	0	0	1	0	0	5	4	14	253
Phenanthrenes and derivatives	1.5%	0	41	26	26	25	36	72	84	31	242
Guanidines	1.5%	255	20	36	3	28	64	79	89	1,952	230
Phenylmethylamines	1.5%	230	22	23	1	190	18	74	80	1,794	162
Flavones	1.5%	0	1	1	1	5	2	14	15	39	600
Oligosaccharides	1.5%	0	3	7	3	0	6	20	19	1	311
Terpene glycosides	1.5%	0	0	0	0	0	0	7	9	0	456
Pyridinecarboxylic acids and derivatives	1.5%	84	41	26	4	246	25	134	141	3,806	273
Oxazinanes	1.4%	6	23	15	0	285	25	84	92	4,669	163
Sugar acids and derivatives	1.4%	0	1	5	0	0	8	9	23	31	292
Alkyl chlorides	1.4%	0	60	33	72	10	44	274	298	32	159
Tetralins	1.4%	13	80	45	9	12	53	91	97	224	264
Ketals	1.4%	0	33	22	4	3	20	53	53	146	200
Catechols	1.4%	0	9	16	2	6	13	37	45	24	381
4'-hydroxyflavonoids	1.4%	0	0	0	2	4	1	7	8	18	428
Alkylarylthioethers	1.4%	9	37	23	10	143	12	71	82	3,688	132
Alkyl-phenylketones	1.4%	6	43	12	2	48	10	98	107	392	277
Morpholines	1.4%	6	21	15	0	285	24	82	89	4,628	156
Amphetamines and derivatives	1.4%	454	21	47	0	89	76	80	88	99	468
Dialkylthioethers	1.4%	12	116	63	13	37	62	123	152	691	287
Oxepanes	1.4%	0	1	3	1	0	1	15	15	14	203
Furanones	1.3%	0	4	6	2	1	3	22	25	92	225
Isoflavonoids	1.3%	0	0	0	1	25	0	9	9	20	314
Alpha,beta-unsaturated ketones	1.3%	1	14	17	6	16	17	56	61	1,244	322
Methoxyanilines	1.3%	8	13	6	3	150	10	61	70	5,481	115
Triazoles	1.3%	5	18	21	9	119	25	81	88	5,773	177
Benzodiazines	1.3%	5	10	22	6	250	21	65	71	3,821	220
2-heteroaryl carboxamides	1.3%	56	11	20	1	245	22	49	48	10,889	205
Vinyl halides	1.3%	7	24	31	31	112	21	152	170	20,463	183
Imidazopyrimidines	1.2%	0	28	28	10	96	32	70	73	911	208
Alpha amino acid esters	1.2%	27	19	25	2	12	21	47	55	364	283
Naphthols and derivatives	1.2%	0	27	13	5	10	14	62	64	22	131
Pyrrolidones	1.2%	57	29	15	7	19	15	51	54	2,713	197
Benzenesulfonamides	1.2%	6	20	41	9	277	35	141	152	5,591	231
Aromatic monoterpenoids	1.2%	4	6	7	4	13	8	45	44	172	222
Glucuronic acid derivatives	1.2%	0	0	3	0	0	6	4	5	1	208
Butenolides	1.2%	0	4	6	2	1	3	18	21	90	205
Enamines	1.2%	23	9	12	2	12	11	34	38	7,333	75
Oligopeptides	1.2%	0	4	21	0	2	32	26	30	7	251
Hydroxybenzoic acid derivatives	1.2%	0	11	17	4	17	13	85	89	33	330
Pyridoindoles	1.2%	0	11	5	0	6	3	16	16	162	257
Acryloyl compounds	1.2%	1	5	1	1	4	4	26	32	1,236	287
Enones	1.2%	1	5	1	1	4	4	26	32	1,236	287
Carboximidic acids and derivatives	1.2%	62	39	29	17	12	17	93	119	128	225
Halobenzoic acids and derivatives	1.2%	2	8	27	5	118	21	119	125	2,298	138
7-hydroxyflavonoids	1.2%	0	0	0	2	4	1	10	11	19	375
Purines and purine derivatives	1.2%	0	28	28	9	92	32	69	73	466	294
Aminopyridines and derivatives	1.2%	55	17	21	3	279	15	61	65	3,587	134
Flavonoid glycosides	1.2%	0	1	0	0	0	1	6	6	18	339
Fatty alcohols	1.2%	0	2	7	17	3	8	65	64	5	248
Glucuronides	1.2%	0	0	2	0	0	5	3	3	1	202
Beta carbolines	1.2%	0	11	5	0	5	3	16	16	160	247
Hydroxy fatty acids	1.1%	4	7	18	0	6	27	36	42	11	445
Tetrahydroisoquinolines	1.1%	8	15	15	0	15	13	29	33	997	147
Benzoic acids	1.1%	2	12	32	0	79	25	156	169	203	306
Azolines	1.1%	261	26	23	7	29	18	90	102	4,133	151
Pyrrolidine-2-ones	1.1%	57	29	15	7	17	15	44	46	2,702	165
Monoalkyl phosphates	1.1%	0	2	9	0	2	5	22	16	3	159
Anthracenes	1.1%	0	27	9	15	3	11	66	74	27	135
Pyrrolidine carboxylic acids and derivatives	1.1%	23	8	33	0	16	60	44	48	2,800	246
3-hydroxysteroids	1.1%	0	7	7	9	2	17	32	39	16	336
Fatty acyl glycosides of mono- and disaccharides	1.1%	0	0	1	0	0	4	11	12	7	317
Alkyl glycosides	1.1%	0	0	0	0	0	1	7	11	7	299
N-acylpiperidines	1.1%	12	6	4	2	61	5	18	19	5,572	252
Carboximidic acids	1.1%	62	33	26	10	9	15	64	86	122	188
Enols	1.1%	0	19	10	1	0	9	22	32	16	108
O-glucuronides	1.0%	0	0	2	0	0	4	3	3	1	180
N-acylpyrrolidines	1.0%	2	2	28	0	38	55	38	42	3,202	185
Hydrazines and derivatives	1.0%	0	12	12	3	9	17	49	60	6,548	58
Flavonoid O-glycosides	1.0%	0	1	0	0	0	1	6	6	17	301
Carboxylic acid salts	1.0%	2	19	14	0	8	26	117	249	3	353
Resorcinols	1.0%	0	1	3	2	7	3	17	17	12	262
Naphthopyrans	1.0%	0	3	3	0	0	3	6	6	6	192

Table 4. Supplement: Compound class distribution. Similar to Table 4, the distribution of frequently occurring compound classes in all datasets described in this paper is shown. Here, classes occurring in at least 1 % of the biomolecular structures are displayed. Again, compound classes with at least 15 occurrences in all datasets are omitted; this applies to 95 compound classes.

Table 5. Supplement: Full compound class distribution. As an extension to Table 1 and Supplementary Table 4, this file contains all compound classes.