A Appendix

A.1 Extended results

We benchmark models in terms of retrieval accuracy as described (§ 4.3) for both the `NIST20` and `NPLIB1` datasets (Table A1, A2). We recreate retrieval experiments using the full PubChem retrieval library in Table A3. We reproduce main text results with standard error values included in Tables A4, A5, and A6. We showcase additional spectra predictions from our model trained on `NIST20` in Figure A1.

Table A1: `NIST20` spectra prediction retrieval top k accuracy for different values of k. All values represent the mean across three separate random seeds ± the standard error of the mean for a single test set.

top k	1	2	3	4	5	8	10
Random	0.026 ± 0.000	0.052 ± 0.001	0.076 ± 0.001	0.098 ± 0.001	0.120 ± 0.000	0.189 ± 0.001	0.233 ± 0.002
3DMolMS	0.055 ± 0.002	0.105 ± 0.000	0.146 ± 0.002	0.185 ± 0.004	0.225 ± 0.004	0.332 ± 0.003	0.394 ± 0.004
FixedVocab	0.172 ± 0.002	0.304 ± 0.002	0.399 ± 0.001	0.466 ± 0.004	0.522 ± 0.006	0.638 ± 0.005	0.688 ± 0.003
NEIMS (FFN)	0.105 ± 0.002	0.243 ± 0.006	0.324 ± 0.006	0.387 ± 0.006	0.440 ± 0.007	0.549 ± 0.005	0.607 ± 0.002
NEIMS (GNN)	0.175 ± 0.003	0.305 ± 0.001	0.398 ± 0.001	0.462 ± 0.002	0.515 ± 0.003	0.632 ± 0.003	0.687 ± 0.003
`SCARF`	$\mathbf{0.187 \pm 0.004}$	$\mathbf{0.321 \pm 0.006}$	$\mathbf{0.417 \pm 0.004}$	$\mathbf{0.486 \pm 0.004}$	$\mathbf{0.541 \pm 0.005}$	$\mathbf{0.652 \pm 0.004}$	$\mathbf{0.708 \pm 0.005}$

Table A2: `NPLIB1` spectra prediction retrieval top k accuracy for different values of k. All values represent the mean across three separate random seeds ± the standard error of the mean for a single test set.

top k	1	2	3	4	5	8	10
Random	0.033 ± 0.001	0.061 ± 0.005	0.092 ± 0.003	0.118 ± 0.003	0.141 ± 0.006	0.216 ± 0.006	0.258 ± 0.006
3DMolMS	0.087 ± 0.001	0.159 ± 0.010	0.218 ± 0.004	0.268 ± 0.006	0.317 ± 0.006	0.427 ± 0.008	0.488 ± 0.005
FixedVocab	0.193 ± 0.003	$\mathbf{0.314 \pm 0.004}$	$\mathbf{0.390 \pm 0.003}$	$\mathbf{0.448 \pm 0.005}$	$\mathbf{0.492 \pm 0.001}$	$\mathbf{0.587 \pm 0.005}$	0.635 ± 0.006
NEIMS (FFN)	$\mathbf{0.195 \pm 0.003}$	0.313 ± 0.002	0.388 ± 0.003	0.447 ± 0.006	0.488 ± 0.002	0.585 ± 0.007	0.624 ± 0.010
NEIMS (GNN)	0.174 ± 0.007	0.285 ± 0.004	0.362 ± 0.002	0.422 ± 0.001	0.471 ± 0.002	0.586 ± 0.007	$\mathbf{0.640 \pm 0.005}$
`SCARF`	0.135 ± 0.007	0.242 ± 0.001	0.320 ± 0.001	0.389 ± 0.004	0.444 ± 0.002	0.569 ± 0.001	0.630 ± 0.008

Table A3: Retrieval accuracy on `NIST20` for a single 500 molecule subset of the test set using a library of 49 decoys or all decoys contained in PubChem ("None"). Results were repeated for 3 random training seeds of the model and are shown ± the standard error of the mean. The top value is shown in bold.

PubChem limit	50			None		
Top k	1	2	3	1	2	3
FixedVocab	0.168 ± 0.003	0.308 ± 0.003	0.410 ± 0.005	0.145 ± 0.004	$\mathbf{0.258 \pm 0.004}$	0.323 ± 0.001
NEIMS (FFN)	0.102 ± 0.002	0.237 ± 0.003	0.315 ± 0.004	0.087 ± 0.003	0.183 ± 0.008	0.236 ± 0.007
NEIMS (GNN)	0.169 ± 0.003	0.300 ± 0.004	0.402 ± 0.005	0.138 ± 0.004	0.239 ± 0.005	0.312 ± 0.008
`SCARF`	$\mathbf{0.204 \pm 0.009}$	$\mathbf{0.326 \pm 0.005}$	$\mathbf{0.432 \pm 0.009}$	$\mathbf{0.167 \pm 0.003}$	$\mathbf{0.258 \pm 0.001}$	$\mathbf{0.336 \pm 0.003}$

A.2 Dataset preparation

`NIST20` [35] is prepared by extracting all positive-mode experimental spectra collected in higher-energy collision-induced dissociation (HCD) mode (i.e., collected on Orbitrap mass spectrometers). Spectra are filtered, so that we keep only those for which the associated molecule (M) has (i) a mass under 1,500 Da, (ii) contains only elements from a predefined set (i.e,. "C", "N", "P", "O", "S", "Si", "I", "H", "Cl", "F", "Br", "B", "Se", "Fe", "Co", "As", "Na", "K"), and (iii) is charged with common adduct types (i.e., "[M+H]+", "[M+Na]+", "[M+K]+", "[M-H2O+]+", "[M+NH3+H]+", and "[M-2H2O+H]+"). Because non-standard empirical spectra databases [48] often do not include the measured collision energies, we pool all collision energies for each compound-adduct pairing to create a single spectrum. We refer the reader to Young et al. [55] for detailed instructions for purchasing and extracting the `NIST20` dataset.

All spectrum intensities are square-root transformed to provide higher weighting to lower intensity peaks, normalized to a maximum intensity of 1 (i.e., through dividing by the maximum intensity),

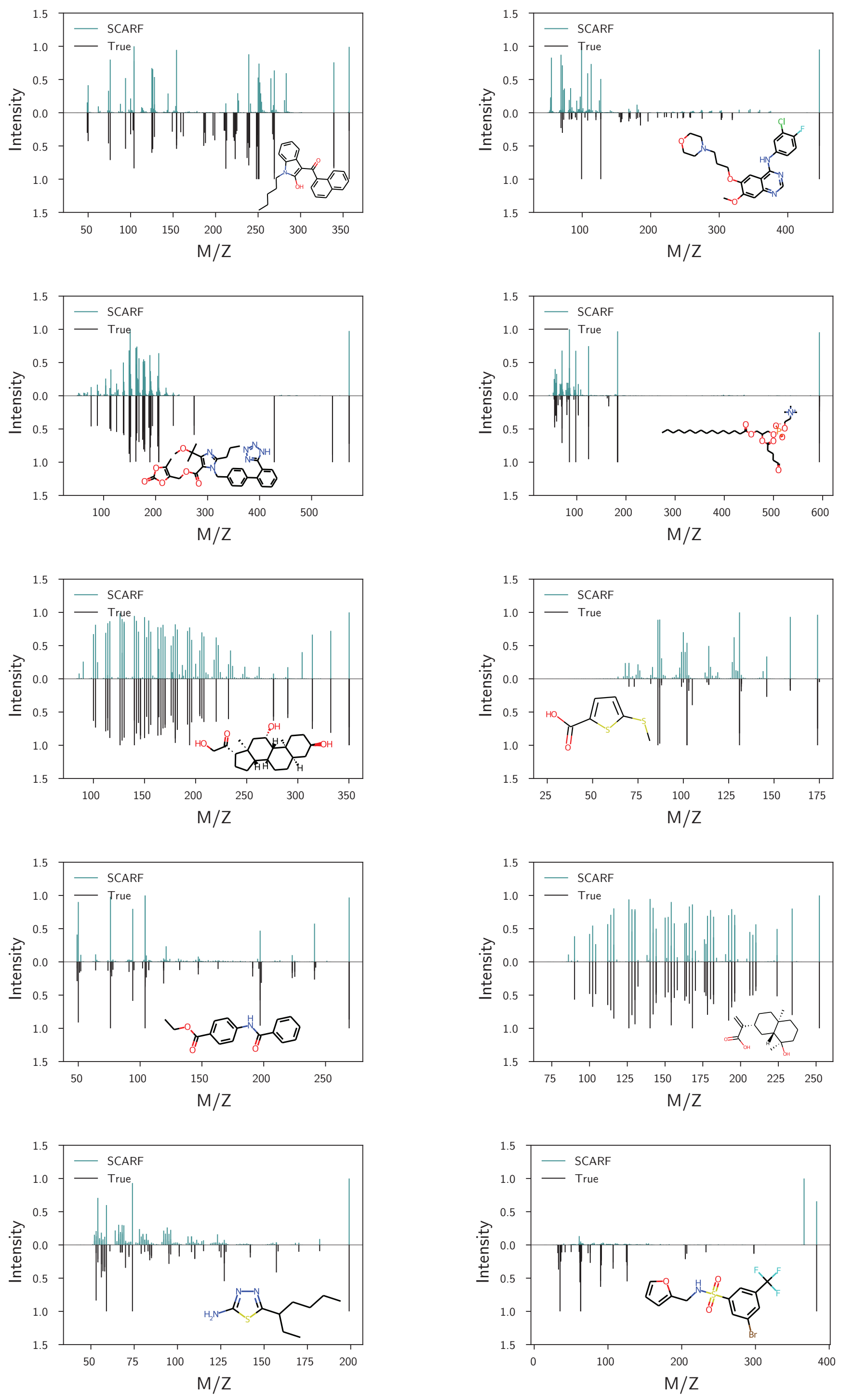

Figure A1: Example spectra predictions from the NIST20 dataset for 10 randomly selected test molecules. The ground truth spectra are shown underneath in black, with predictions above in teal. Molecules are shown inset.

Table A4: Model coverage of true peak formulae as determined by MAGMa at various max formula cutoffs for the NPLIB1 dataset. Results are calculated for a single held out test split, shown ± the standard error of the mean across three random seeds for all models that were retrained. The best value (i.e., highest) is typeset in bold for each column.

Coverage @	10	30	300	1000
Random	0.004	0.014	0.126	0.336
Frequency	0.090	0.151	0.466	0.688
CFM-ID	**0.170**	0.267	–	–
Autoregressive	0.072 ± 0.001	0.082 ± 0.002	0.095 ± 0.001	0.099 ± 0.000
SCARF-D	0.158 ± 0.001	0.284 ± 0.003	0.681 ± 0.002	0.856 ± 0.002
SCARF-F	0.155 ± 0.002	0.306 ± 0.003	0.708 ± 0.003	0.859 ± 0.001
SCARF	0.164 ± 0.009	$\mathbf{0.309 \pm 0.014}$	$\mathbf{0.724 \pm 0.013}$	$\mathbf{0.879 \pm 0.004}$

Table A5: Model coverage of true peak formulae as determined by MAGMa at various max formula cutoffs for the NIST20 dataset. Results are calculated for a single held out test split, shown ± the standard error of the mean across three random seeds for all models that were retrained. The best value (i.e., highest) is typeset in bold for each column.

Coverage @	10	30	300	1000
Random	0.009	0.026	0.232	0.532
Frequency	0.173	0.275	0.659	0.830
CFM-ID	0.197	0.282	–	–
Autoregressive	0.204 ± 0.001	0.262 ± 0.002	0.309 ± 0.005	0.317 ± 0.006
SCARF-D	0.248 ± 0.001	0.425 ± 0.002	0.839 ± 0.002	0.941 ± 0.001
SCARF-F	0.249 ± 0.001	0.476 ± 0.002	0.855 ± 0.000	0.943 ± 0.001
SCARF	$\mathbf{0.308 \pm 0.002}$	$\mathbf{0.552 \pm 0.001}$	$\mathbf{0.907 \pm 0.002}$	$\mathbf{0.968 \pm 0.001}$

filtered to exclude any noise peaks with normalized intensity under 0.003, and subsetted to only the top 50 highest intensity peaks. All peaks are mass-shifted by the weight of the parent adduct (i.e,. if the spectrum is "[M+H]+", the weight of a proton is subtracted from each child peak).

A.2.1 Product formulae assignments

Because the precursor ion and adduct species are known for the training dataset, we subtract the precursor adduct mass from every peak in the training set, and attempt to annotate each peak with a plausible product formula (i.e., a subset of the true precursor formula).

We opt to constrain the training product formulae to be subsets of contiguous heavy atoms of the parent molecule as derived with the MAGMa algorithm [40].

We note two important limitations of these heuristics. First, by using molecular substructures to annotate product formulae, our model is less prone to correctly identifying complex rearrangements. Second, it is also possible for adduct switching to occur. Namely, if the precursor ion has a sodium adduct ("[M+Na]+"), some of the product formulae may actually switch and acquire a hydrogen adduct instead. We assume no adduct switching in our formulation, instead focusing on the novelty of the prefix tree decoding approach, as these represent data labeling challenges, rather than modeling challenges.

In addition, any predictive models of product formulae distributions will more closely predict spectra that would be produced on instrumentation similar to the training sets utilized [9, 11]. Given this, we encourage users of such models to treat these predictions as putative, rather than experimentally valid.

Table A6: Spectra prediction in terms of cosine similarity, coverage (proportion of ground-truth peaks that are covered by the top 100 non-zero predictions), validity (the fraction of predicted peaks for which a chemically plausible explanation is possible), and time. Best value in each column is typeset in bold (higher is better for all metrics but time). Values are shown ± the standard error of the mean computed across three random seeds on a single test set for all models that could be retrained (i.e., not CFM-ID).

Dataset	NIST20			NPLIB1			
	Cosine sim.	Coverage	Valid	Cosine sim.	Coverage	Valid	Time (s)
CFM-ID	0.412 ± 0.000	0.278 ± 0.000	$\mathbf{1.00 \pm 0.000}$	0.377 ± 0.000	0.235 ± 0.000	$\mathbf{1.00 \pm 0.000}$	1114.7
3DMolMS	0.510 ± 0.000	0.734 ± 0.001	0.94 ± 0.001	0.394 ± 0.002	0.507 ± 0.001	0.92 ± 0.000	**3.5**
FixedVocab	0.704 ± 0.000	0.788 ± 0.000	$\mathbf{1.00 \pm 0.000}$	$\mathbf{0.568 \pm 0.002}$	$\mathbf{0.563 \pm 0.001}$	$\mathbf{1.00 \pm 0.000}$	5.5
NEIMS (FFN)	0.617 ± 0.000	0.746 ± 0.001	0.95 ± 0.001	0.491 ± 0.002	0.524 ± 0.001	0.95 ± 0.000	3.9
NEIMS (GNN)	0.694 ± 0.000	0.780 ± 0.000	0.95 ± 0.001	0.521 ± 0.002	0.547 ± 0.003	0.94 ± 0.001	4.9
SCARF	$\mathbf{0.726 \pm 0.001}$	$\mathbf{0.807 \pm 0.000}$	$\mathbf{1.00 \pm 0.000}$	0.536 ± 0.007	0.552 ± 0.008	$\mathbf{1.00 \pm 0.000}$	21.1

Table A7: Spectra prediction accuracy comparing inclusion (Cosine sim.) and exclusion (Cosine sim. (no MS1)) of the precursor mass. For all compounds, the peak at the mass of the input compound is masked in the prediction and ground truth to compute Cosine sim. (no MS1). All results represent an average on a single test set across three random seeds.

Dataset	NIST20		NPLIB1	
	Cosine sim.	Cosine sim. (no MS1)	Cosine sim.	Cosine sim. (no MS1)
CFM-ID	0.412	0.289	0.377	0.326
3DMolMS	0.510	0.517	0.394	0.390
FixedVocab	0.704	0.637	**0.568**	**0.505**
NEIMS (FFN)	0.617	0.557	0.491	0.454
NEIMS (GNN)	0.694	0.620	0.521	0.477
SCARF	**0.726**	**0.663**	0.536	0.466

A.2.2 Dataset statistics

To probe the composition of our two primary datasets, we investigate both the molecular weight and chemical classes contained in the NPLIB1 and NIST20 datasets. We find that the average molecular weight is higher for NPLIB1 (Figure A3A), consistent with the increased complexity of natural product molecules. We additionally compute chemical classes of the compounds using NPClassifier [26] to identify the types of compounds present in both datasets (Figure A3B-C). NPLIB1 is enriched for steroids, coumarins, and various complex alkaloid natural products. On the other hand, NIST20 is enriched for small peptides, nicotinic acid alkaloids, and pseudoalkaloids, among others. While these descriptions are helpful to identify the dataset composition, chemical compound classification is itself a learned classification and should be interpreted cautiously.

Table A8: NIST20 retrieval accuracy averaged across three random seeds of model training stratified by the weight of the target molecule.

Dataset	NIST20						
Molecular weight	0 - 200	200 - 300	300 - 400	400 - 500	500 - 600	600 - 700	700 - 2000
Num. compounds	624	1358	900	387	129	56	89
Random	0.024	0.021	0.027	0.025	0.031	0.048	0.101
3DMolMS	0.039	0.041	0.057	0.077	0.070	0.125	**0.199**
FixedVocab	0.143	0.184	**0.168**	0.172	**0.196**	**0.220**	0.161
NEIMS (FFN)	0.110	0.122	0.092	0.078	0.111	0.083	0.064
NEIMS (GNN)	0.155	0.192	0.164	**0.182**	0.147	0.214	0.161
SCARF	**0.191**	**0.211**	0.165	0.163	0.168	0.161	0.169

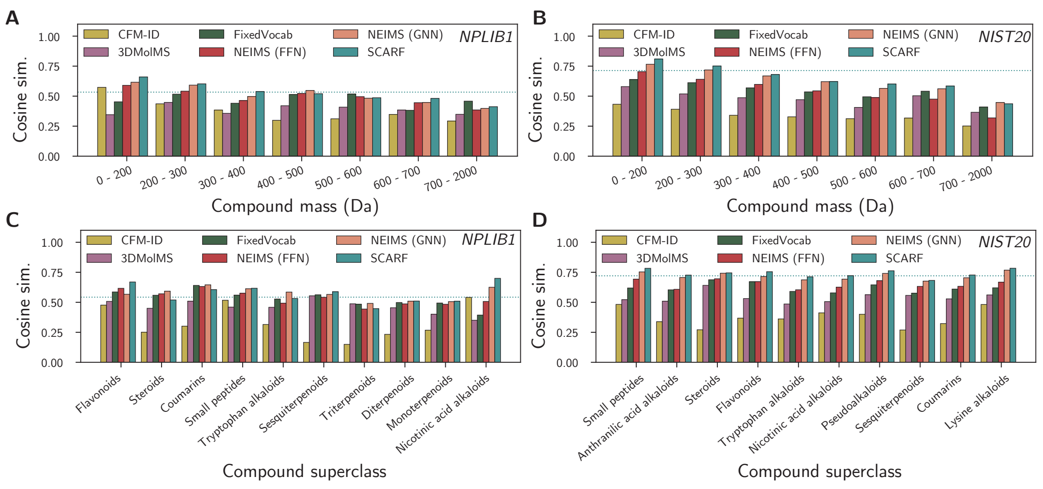

Figure A2: Cosine similarity of predicted spectra is stratified across molecular weight for both `NPLIB1` (**A**) and `NIST20` (**B**). We further stratify results across putative chemical classes of input molecules using NPClassifier [26] for both `NPLIB1` (**C**) and `NIST20` (**D**). The dotted line indicates the average predictive cosine similarity of `SCARF` across all examples and averaged over three random splits.

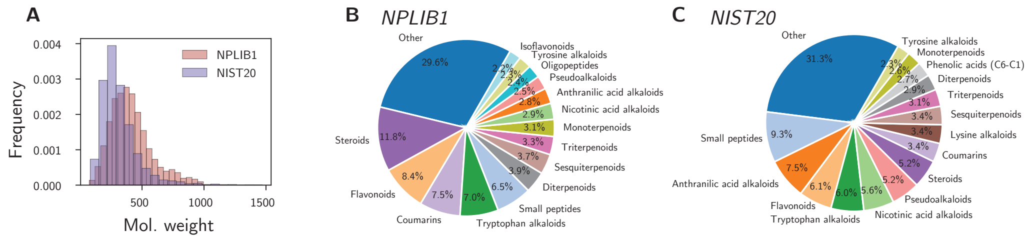

Figure A3: Spectra dataset molecule characterizations. **A.** Distribution of the molecular weight of compounds across `NPLIB1` and `NIST20`. **B-C.** Chemical classes contained in `NPLIB1` (B) and `NIST20` (C) with the top 15 classes shown and all others grouped in 'Other'. Chemical classes are computed using NPClassifier [26].

A.3 Baselines

We further describe select baseline models.

A.3.1 CFM-ID baseline

`CFM-ID` [2] is a long standing and important approach to fragmentation prediction. Because `CFM-ID` is fit using a time intensive EM training approach on an analogous dataset, we utilize the pretrained Docker implementation provided by the authors in line with [34]. `CFM-ID` has two options for predicting molecules in either positive or negative adduct mode with "H" adducts (i.e., "[M+H]+" or "[M-H]-"). To directly compare to our method, we predict spectra in positive mode and remove hydrogens from all predicted peaks, as all training peaks are shifted by removing their adducts.

`CFM-ID` also produces predictions at three collision energies (i.e., low , medium, or high fragmentation). Because we opt not to include these, we merge these predictions and re-normalize the result to a maximum of 1.

A.3.2 Autoregressive baseline

When considering the task of generating spectrum formulae candidates, we compare `SCARF-Thread` to an autoregressive recurrent neural network baseline, which is based around a long short term memory (LSTM) module [21].

The LSTM generates formulae consecutively from a single concatenated encoding of the input molecule and input full formula. At each step in the recurrent process, a one-hot encoding of the previous predicted element count is concatenated to a one-hot encoding of the element type being predicted in the current step. By embedding this information into the network, we can avoid predicting element types that do not appear in the parent molecule's molecular formula. If the parent molecular formula has 5 element types, each autoregressively predicted formula requires generating only 5 element type counts; this eliminates the need for a stop condition between each formula. Formulae are generated autoregressively, from highest to lowest intensity. When predicting the counts of the next element type, we employ the same difference and forward count prediction strategy as used in `SCARF` for fair comparison. The model is trained with a cross entropy loss and full hyperparameters are listed in Table A10.

A.3.3 NEIMS baseline

NEIMS [50] is a highly efficient binned spectrum prediction approach, originally developed for gas chromatography-mass spectra (GC-MS). To enable a fair comparison, we optimize its hyperparameters on our dataset and add higher resolution bins. Furthermore, we also train a graph neural network-based version "NEIMS (GNN)" (in addition to the network that more closely matches Wei et al. [50]'s original model and operates on the molecular fingerprint, "NEIMS (FFN)"). The adduct type is either concatenated to all atom features (for NEIMS (GNN)) or to the fingerprint vector (for NEIMS (FFN)).

A.3.4 3DMolMS baseline

3DMolMS [23] is a binned spectrum prediction approach developed simultaneously to this work. Unlike the other binned NEIMS approach, 3DMolMS utilizes a point-based deep neural network model operating on the point cloud of an input molecule. To project a 2D molecule or SMILES string into 3D space, a single 3D conformer is first generated using RDKit [39]. After several 3D convolutions, the atom-wise representations are pooled, covariates corresponding to the settings of the machine and experiment are concatenated, and the result is projected into a fixed length binned spectrum.

To enable a fair comparison, we copy the 3DMolMS architecture into our modeling framework with minor tweaks to the network. Rather than use variable sized hidden layers, we fix hidden layer sizes to a single value across convolutions. In addition, we only use the covariate of the adduct type for consistency with our model, excluding collision energy and instrument type. We hyperparameter optimize the model independently. We find that the performance of this model is substantially lower than the NEIMS baseline, likely due to the additional use of the "difference" prediction module in the NEIMS approach that allows the network to predict intensities at both fragments and neutral losses.

A.3.5 FixedVocab baseline

Concurrently to our work, Murphy et al. introduce an alternative formula prediction strategy for mass spectrum prediction, a model they term GRAFF-MS. Unlike `SCARF`, GRAFF-MS utilizes a fixed vocabulary of molecular formulae and molecular formula differences, predicting intensities at each such value without learning to encode the formulae. These formulae and formula differences are selected in a greedy fashion based upon their frequency in the training set.

Because no code was released for this approach at the time this work was conducted, we reimplement a variant of their approach that emphasizes the use of a fixed vocabulary of formulae and differences. For methodological consistency, we utilize equivalent formula annotations as used by `SCARF` (i.e., one annotation per peak), do not model collision energies or instrument types, and utilize the same graph encoder as `SCARF` for encoding each molecule. We treat the number of fragment and difference formulae as a tunable hyperparameter (which is optimized along with the rest of the hyperparameters – see §A.5.6). We mask all invalid formulae and differences and utilize a cosine similarity loss with the original spectrum to train the model. To convert predicted formula and difference intensities into a binned spectrum, each formula-intensity pairing is projected into its respective binned position using a scatter max calculation. We note that because alternative adducts and isotopes are not labeled in our preprocessing step, we do not predict isotopic or adduct variants for each fragment.

Given the differences between codebases, it is possible that the performance of our reimplementation does not exactly match the original implementation, and we instead refer to it as "FixedVocab" rather than GRAFF-MS in table presentations. An earlier version of our work understated the FixedVocab model's performance due to an implementation decision along these lines (specifically, not including the "0" neutral loss as a predicted vocabulary entry). This has since been rectified, increasing the accuracy of the FixedVocab model.

A.4 Retrieval subsets vs. PubChem

We restrict our retrieval experiments in §4.3 to only the top 49 decoys per test case for two reasons. First, from a practical perspective, running the forward model on every isomer match in PubChem (approx. thousands each, >1,000,000 for only 500 test cases) makes benchmarking across all considered models substantially more challenging both for this work and also future work. Second, we also believe that this top 50 challenge represents a more realistic setting. In practice, retrieval compound libraries will often be carefully crafted and designed to contain molecules similar to the unknown molecule rather than all possible isomers (using either prior knowledge or "backward" models such as CSI:FingerID [12] and MIST [19]). We conduct a side-by-side analysis on a small 500 molecule subset of the test set comparing the setting described above (with 49 decoys) to a setting with no limit on the number of decoys. The results are shown in Table A3, showing that `SCARF` still performs well in this setting.

A.5 Model details

Here, we describe details of our model's training setup, architecture, and hyperparameters that were omitted from the main text. Definitive details can also be found in the code at `https://github.com/samgoldman97/ms-pred`.

A.5.1 Training

We train each of our models on a single RTX A5000 NVIDIA GPU (CUDA Version 11.6), making use of the Torch Lightning [15] library to manage the training. `SCARF-Thread` and `SCARF-Weave` take on the order of 1.5 and 2.5 hours of wall time to train respectively.

A.5.2 Molecule encoding

Within both `SCARF-Thread` and `SCARF-Weave`, a key component is an encoding of the molecular graph using a message passing graph neural network, $\mathsf{gnn}(\mathcal{M})$. Such graph neural network models are now well described [3, 7, 20], so we will skip a detailed explanation of them here. In our experiments, we use gated graph sequence neural networks [30]. We made use of the implementation of this network in the DGL library [49] and use as atom features those shown in Table A9 (which are computed using RDKit [39] or DGL [49]).

Table A9: Graph neural network (GNN) atom features.

Name	Description
Element type	one-hot encoding of the element type
Degree	one-hot encoding of number of bonds atom is associated with
Hybridization type	one-hot encoding of the hybridization (SP, SP2, SP3, SP3D, SP3D2)
Charge	one-hot encoding of atom's formal charge (from -2 to 3)
Ring-system	binary flag indicating whether atom is part of a ring
Atom mass	atom's mass as a `float`
Chiral tag	atom's chiral tag as one-hot encoding
Adduct type	one-hot encoding of the adduct ion
Random walk embed steps	positional encodings of the nodes computed using DGL

A.5.3 Molecular formulae representations

When forming representations of formulae (including formulae prefixes) we use a count-based encoder, $\mathsf{counts}(\boldsymbol{f})$. This encoder takes in as input the counts of all individual elements in the formula (which also can be "undefined" for counts of elements not yet specified – indicated as '$*$' in Figure 3B) and returns a vector representation in $\mathbb{R}^d$. The encoder is based upon the Fourier feature mapping proposed by Tancik et al. [43], but using only sin basis functions (to reduce the number of parameters required by our networks). Tancik et al. [43] has shown that such features perform better than encoding integers directly; furthermore, compared to learned representations, using Fourier features enables us (at least in principle) to deal with counts at test time that have not been seen during training.

To be precise, each possible count, $v \in \mathbb{N}_0$, is encoded by our counts-based encoder into the vector:

$$\mathsf{abs}\left(\left[\sin\left(\frac{2\pi v}{T_1}\right), \sin\left(\frac{2\pi v}{T_2}\right), \sin\left(\frac{2\pi v}{T_3}\right), \ldots\right]\right),$$

where the periods (T_1, T_2, etc.) are set at increasing powers of two that enable us to discriminate all possible element counts given in the input, and $\mathsf{abs}(\cdot)$ is the absolute value function such that we get positive embeddings. For the "undefined" count we learn a separate encoding of the same dimensionality.

A.5.4 Further details of `SCARF-Thread`

Pseudo-code for the `SCARF-Thread` model is shown in Algorithm A.1. Note that the second for loop (on the line marked ‡) does not depend on previous iterations of the loop, so that in practice we perform this computation in parallel. At training time we use teacher forcing (§3.3), meaning the first for loop (marked †) is only run sequentially at inference time.

The function $\mathsf{scarf\text{-}thread\text{-}net}(\cdot)$ represents the network shown in Figure 3B and generates the set of subsequent valid element counts given a prefix (i.e., the child nodes of a given prefix node). As discussed in the main text, we treat this as a multi-label binary classification task and predict the binary label for each possible count using forward and difference MLPs (Eq. 3). We fix a maximum possible element count (i.e., the number of possible classes in this classification problem), $N = 160$. We do not allow product formulae to have more of a given element than is present in the precursor formula, $\boldsymbol{\mathcal{F}}$, and we achieve this by setting the probability of these classes to zero.

Algorithm A.1: Pseudo-code for `SCARF-Thread`, which generates prefix trees from a root node autoregressively, one level at a time.

```
Data: Input molecule, $\mathcal{M}$, with corresponding input formula, $\boldsymbol{\mathcal{F}}$.
Result: Set of product formulae, $\rho_e = \{\boldsymbol{f}^i\}_{i=1}^n$.
1   $\boldsymbol{h}_\mathcal{M} \leftarrow \mathsf{gnn}(\mathcal{M})$ ;                  ▷ Form embedding of precursor molecule.
2   $\rho_0 \leftarrow \{*\}$ ;          ▷ Store the set of initial prefixes which is just the undefined formula, $*$.
3†  for $j \in [1, \ldots, e]$ do                                   ▷ Loop over all possible elements.
4       $\rho_j \leftarrow \{\}$ ;                             ▷ Create the set of prefixes the next time around.
5       $\boldsymbol{h}_j \leftarrow \mathsf{one\text{-}hot}(j)$ ;              ▷ Encoding of which element we are predicting the count of.
6‡      for $\boldsymbol{f}'_{<j} \in \rho_{j-1}$ do (in parallel)                  ▷ Loop over all current prefixes.
7           $\boldsymbol{c}' = [\boldsymbol{h}_\mathcal{M}, \mathsf{counts}(\boldsymbol{f}'_{<j}), \mathsf{counts}(\boldsymbol{\mathcal{F}} - \boldsymbol{f}'_{<j}), \boldsymbol{h}_j]$ ;     ▷ Create context vector, Eq. 2
8           $\{f_j^{i'}\}_{i'=1}^{n'} \leftarrow \mathsf{scarf\text{-}thread\text{-}net}(\boldsymbol{c}', \boldsymbol{\mathcal{F}})$ ;   ▷ Predict the set of valid next element counts under this prefix.
9           $\rho_j \leftarrow \rho_j \cup \mathsf{create\text{-}new\text{-}prefixes}(\boldsymbol{f}'_{<j}, \{f_j^{i'}\}_{i'=1}^{n'})$ ;   ▷ Create new prefixes for the next element.
return $\rho_e$
```

A.5.5 Further details of `SCARF-Weave`

As discussed in the main text, `SCARF-Weave` is based off Lee et al. [29]'s Set Transformer. After forming the input encoding using the molecule and count-based encoder (§A.5.2 & §A.5.3), we further refine this embedding using an MLP (multi-layer perceptron) network. The output of this is passed into a series of l_3 Transformer [44] layers (§A.5.6 defines the exact number used in the experiments) with 8 attention heads each.

We use a cosine distance loss to train the parameters of SCARF-Weave. This loss is also used for the FFN and GNN baselines (Table 2). To ensure consistency with the baselines, we first project the output of SCARF-Weave into a binned histogram representation (§A.5.6 defines the number of bins used); for each bin we take the max intensity across all applicable formulae. Given a predicted binned spectra, $\hat{\boldsymbol{s}}$, and the ground-truth binned spectra, $\boldsymbol{s}$, the cosine distance is defined as the negative of the cosine similarity (computed using PyTorch's `torch.cosine_similarity` function [36]):

$$\text{cos-sim}\left(\hat{\boldsymbol{s}}, \boldsymbol{s}\right) = \frac{\hat{\boldsymbol{s}} \cdot \boldsymbol{s}}{\max\left(\|\hat{\boldsymbol{s}}\|_2 \|\boldsymbol{s}\|_2,\ \epsilon\right)}, \tag{A.1}$$

where $\epsilon = 1 \times 10^{-8}$ is used to ensure numerical stability.

A.5.6 Hyperparameters

To enable fair comparison across models, hyperparameters were tuned for SCARF, the FFN binned prediction baseline, and the GNN binned prediction baseline. Parameters were tuned using RayTune [31] with Optuna [1] and an ASHAScheduler. Each model was allotted 50 different hyperoptimization trials for fitting. Models were hyperparameter optimized on a smaller $10,000$ spectra subset of NIST20. Parameters are detailed in Table A10.

A.6 Limitations and future work

We outline several potential directions for future work to address limitations of this work.

1. *Improving the gold standard training annotation pre-processing.* Because SCARF is flexible in that it can match distributions of formula assignments, a key step to improving and building upon this approach is to develop more robust assignments of formula to training spectra. This includes adding complexity and removing potential assumptions, such as allowing annotations to account for rearrangement, elimination, or charge transfer. A second goal is to identify potentially low quality training spectra, such as ones that emerge from mixtures, and remove these from the inputs. Another potential way to handle such cases would be to model each spectrum peak as an *ensemble* of potential equivalent-mass formulae, which would be particularly helpful in relating SCARF to inverse models such as MIST [19] in which the structure of the molecule and formula identity of each peak cannot be known *a priori*.
2. *Incorporating other model covariates.* Incorporating collision energy features explicitly into the model, as well as negative-ion mode inputs, will increase its usability. This could be enabled by aggregating public data containing these annotations.
3. *Featurizing molecule inputs using different or more powerful molecular encoders.* Recent and simultaneous work to this used a pretrained graph encoder as part of a binned spectrum prediction approach, MassFormer [55]. It is possible to include more powerful molecule or formula encoders into SCARF.
4. *Consideration of interpretability by subgraph attribution and combination with ICEBERG.* Following this initial work, we developed a second model, ICEBERG [18], that uses a similar two step modeling approach, but instead encodes fragments, not formula. This increases accuracy and robustness, especially for retrieval, but substantially slows the model. In comparison to ICEBERG, SCARF still has several benefits including speed, the lack of required substructure labeling, and ability to capture potential skeletal rearrangements of molecules (i.e., discontinuities in structure that may not be possible to model by only breaking bonds). An open question and exciting opportunity in the future is to combine these two levels of abstraction and make formula-level predictions with graph-level attribution or featurization.
5. *Retrieval-specific loss functions to enhance retrieval performance.* A significant finding of this work was the noise associated with the retrieval task and lack of correlation with spectrum prediction performance as measured by cosine similarity. Future work may consider how to more directly define loss functions that reflect the task of retrieval.

Table A10: Model and baseline hyperparameters.

Model	Parameter	Grid	Value
Autoregressive	learning rate	$[1e-4, 1e-3]$	0.0009
	learning rate scheduler	-	StepDecay (5,000)
	learning rate decay	$[0.7, 1.0]$	0.85
	dropout	$\{0.0, 0.1, 0.2, 0.3\}$	0.2
	hidden size, d	$\{128, 256, 512\}$	512
	gnn layers	$[1, 6]$	1
	rnn layers	$[1, 3]$	3
	batch size	$\{8, 16, 32, 64\}$	64
	weight decay	$\{0, 1e-6, 1e-7\}$	$1e-6$
	use differences (Eq.3)	{True, False}	True
	conv type	-	GatedGraphConv
	random walk embed steps (Table A9)	[0,20]	20
	graph pooling	{mean, attention}	mean
NEIMS (FFN)	learning rate	$[1e-4, 1e-3]$	0.00087
	learning rate scheduler	-	StepDecay (5,000)
	learning rate decay	$[0.7, 1.0]$	0.722
	dropout	$\{0.0, 0.1, 0.2, 0.3\}$	0.0
	hidden size, d	$\{64, 128, 256, 512\}$	512
	layers, l	$\{1, 2, 3\}$	2
	batch size	$\{16, 32, 64, 128\}$	128
	weight decay	$\{0, 1e-6, 1e-7\}$	0
	use differences (Eq.3)	{True, False}	True
	num bins (§A.5.5)	-	$15,000$
NEIMS (GNN)	learning rate	$[1e-4, 1e-3]$	0.00052
	learning rate scheduler	-	StepDecay (5,000)
	learning rate decay	$[0.7, 1.0]$	0.767
	dropout	$\{0.0, 0.1, 0.2, 0.3\}$	0.0
	hidden size, d	$\{64, 128, 256, 512\}$	512
	layers, l	$[1, 6]$	4
	batch size	$\{16, 32, 64\}$	64
	weight decay	$\{0, 1e-6, 1e-7\}$	$1e-7$
	use differences (Eq.3)	{True, False}	True
	num bins (§A.5.5)	-	$15,000$
	conv type	-	GatedGraphConv
	random walk embed steps (Table A9)	[0,20]	19
	graph pooling	{mean, attention}	mean
3DMolMS	learning rate	$[1e-4, 1e-3]$	0.00074
	learning rate scheduler	-	StepDecay (5,000)
	learning rate decay	$[0.7, 1.0]$	0.86
	dropout	$\{0.0, 0.1, 0.2, 0.3\}$	0.3
	hidden size, d	$\{64, 128, 256, 512\}$	256
	layers, l	$[1, 6]$	2
	top layers	$[1, 3]$	2
	neighbors, k	$[3, 6]$	5
	batch size	$\{16, 32, 64\}$	16
	weight decay	$\{0, 1e-6, 1e-7\}$	$1e-6$
	num bins (§A.5.5)	-	$15,000$
FixedVocab	learning rate	$[1e-4, 1e-3]$	0.00018
	learning rate scheduler	-	StepDecay (5,000)
	learning rate decay	$[0.7, 1.0]$	0.92
	dropout	$\{0.0, 0.1, 0.2, 0.3\}$	0.3
	hidden size, d	$\{64, 128, 256, 512\}$	512
	layers, l	$[1, 6]$	6
	batch size	$\{16, 32, 64\}$	64
	weight decay	$\{0, 1e-6, 1e-7\}$	$1e-6$
	num bins (§A.5.5)	-	$15,000$
	conv type	-	GatedGraphConv
	random walk embed steps (Table A9)	[0,20]	11
	graph pooling	{mean, attention}	mean

Table A10 – continued from previous page

Model	Parameter	Grid	Value
	formula library size	{1000, 5000, 10000, 25000, 50000}	5000
`SCARF-Thread`	learning rate	$[1e-4, 1e-3]$	0.000577
	learning rate scheduler	-	StepDecay (5,000)
	learning rate decay	$[0.7, 1.0]$	0.894
	dropout	$\{0.0, 0.1, 0.2, 0.3\}$	0.3
	hidden size, d	$\{128, 256, 512\}$	512
	mlp layers, l_1	$[1, 3]$	2
	gnn layers, l_2 (§A.5.2)	$[1, 6]$	4
	batch size	$\{8, 16, 32, 64\}$	16
	weight decay	$\{0, 1e-6, 1e-7\}$	$1e-6$
	use differences (Eq.3)	{True, False}	True
	conv type	-	GatedGraphConv
	random walk embed steps (Table A9)	[0,20]	20
	graph pooling	{mean, attention}	mean
`SCARF-Weave`	learning rate	$[1e-4, 1e-3]$	0.00031
	learning rate scheduler	-	StepDecay (5,000)
	learning rate decay	$[0.7, 1.0]$	0.962
	dropout	$\{0.0, 0.1, 0.2, 0.3\}$	0.2
	hidden size, d	$\{128, 256, 512\}$	512
	mlp layers, l_1 (§A.5.5)	$[1, 3]$	2
	gnn layers, l_2 (§A.5.2)	$[1, 6]$	3
	transformer layers, l_3 (§A.5.5)	$[0, 3]$	2
	batch size	$\{4, 8, 16, 32, 64\}$	32
	weight decay	$\{0, 1e-6, 1e-7\}$	0
	num bins (§A.5.5)	-	$15,000$
	conv type	-	GatedGraphConv
	random walk embed steps (Table A9)	[0,20]	7
	graph pooling	{mean, attention}	attention