Molecular structure inference from molecular spectra – notes on machine learning methods

ROMAN ANDREEV (andreevr ◎ ethz ch)

Inferring molecule structure from its ^{1}H-NMR spectrum

Inferring SMILES from simulated ^{1}H-NMR multiplets

and the sum formula

Training dataset and transformers model

- Daniel Lowe. *Chemical reactions from US patents (1976–Sep2016)*. figshare, 2017. 10.6084/m9.figshare.5104873.v1
- Marvin Alberts et al. “Unraveling Molecular Structure: A Multimodal Spectroscopic Dataset for Chemistry”. In: *NeurIPS 2024 Datasets and Benchmarks Track*. 2024. 10.48550/arXiv.2407.17492
- Marvin Alberts et al. *Learning the Language of NMR: Structure Elucidation from NMR spectra using Transformer Models*. 2023. 10.26434/chemrxiv-2023-8wxcz

Training dataset and transformers model

- **Dataset:** USPTO reaction dataset[1]
  - 1,435,481 reactions
  - Realistic molecules & common chemicals
- **Unique molecules:**
  - Initially: 1,675,439
  - Filter: 5 < # heavy atoms < 35
  - Allowed elements: C, H, O, N, S, P, Si, B, halogens
  - After filtering: 1,416,499
  - Simulation w/ MestReNova (all spectra): **794,403** molecules
- **Spectra simulated:** ^{1}H-NMR, IR, ^{13}C-NMR, HSQC-NMR, MS/MS
- "... **deuterated chloroform** as solvent. Default settings were used"
- **Representation:** SMILES

[1] Lowe, 2017 [1]

Training dataset

SRC-TRAIN.TXT

- C 15 H 24 N 2 1HNMR | 6.86 6.83 d 1H J 0.68 | 5.98 5.96 s 1H | 4.22 4.19 s 2H | 3.83 3.74 p 1H J 6.74 | 3.55 3.50 m 2H | 2.24 2.21 s 3H | 2.05 1.99 m 2H | 1.35 1.32 s 5H | 1.26 1.22 d 6H J 6.67
- C 30 H 28 F N O 3 1HNMR | 7.39 7.30 m 4H | 7.30 7.05 m 11H | ... | 2.60 2.46 m 2H

...

TGT-TRAIN.TXT

- C c 1 c c 2 c (c c 1 N) N (C (C) C) ... (C) C
- O = C (C C c 1 c c c c c 1) N C (C c 1 c c c (O c 2 c c c c c 2) c c 1) C (O) c 1 c c c (F) c c 1

...

Training dataset

SRC-TRAIN.TXT

- `C 15 H 24 N 2 1HNMR | 6.86 6.83 d 1H J 0.68 | 5.98 5.96 s 1H | 4.22 4.19 s 2H | 3.83 3.74 p 1H J 6.74 | 3.55 3.50 m 2H | 2.24 2.21 s 3H | 2.05 1.99 m 2H | 1.35 1.32 s 5H | 1.26 1.22 d 6H J 6.67`
- `C 30 H 28 F N O 3 1HNMR | 7.39 7.30 m 4H | 7.30 7.05 m 11H | ... | 2.60 2.46 m 2H`

...

TGT-TRAIN.TXT with explicit hydrogens (new)

- `[CH3] [c] 1 [cH] [c] 2 [c] ( [cH] [c] 1 [NH2] ) [N] ( [CH] ( [CH3] ) [CH3] ) [CH2] [CH2] [C] 2 ( [CH3] ) [ CH3 ]`

...

Input vocabulary

Constructed from 10k samples

| ×92120, J×52692, 1H×46441, 2H×27470, m×25035, s×14414, d×13282, C×10000, H×10000, 1HNMR×10000, dd×9417, 3H×9343, O×9018, N×8443, t×7266, 2×6096, ddd×4820, 3×4683, dt×3241, 4H×3220, 4×2473, F×2346, S×2314, Cl×2132, ddt×1913, q×1820, td×1690, 5H×1627, 6H×1622, 5×1430, dddd×1430, 0.92×1393, 7.32×1286, dq×1266, 14×1166, tt×1163, 17×1152, 16×1144, 15×1112, 19×1098, 18×1096, 2.19×1064, 20×1060, 7.14×1058, p×1047, 1.47×1040, 7.33×1028, 7.69×1024, 12×1011, 2.20×1009, 7.26×1008, 13×994, 7.51×973, 11×956, 21×947, Br×941, 8H×929, 2.10×921, 22×918, ...

Output vocabulary

Constructed from 10k samples

c×94533, C×68425, (×39107,)×39107, 1×25622, O×23680, 2×17598, =×13822, N×11314, n×9750, 3×6992, F×5100, Cl×2717, −×2531, 4×1826, S×1744, [C@@H]×1443, [C@H]×1257, Br×1000, [nH]×986, s×871, #×869, /×522, o×498, 5×270, I×181, [C@]×148, [N+]×130, [O−]×112, \×106, [C@@]×100, P×62, [N−]×46, [n+]×30, 6×8, [S@]×7, [2H]×4, [C−]×3, [PH2]×1, [N@@+]×1, p×1, [S@@]×1, [N@]×1

The “model”: encoder-decoder transformer

- The **encoder** converts input NMR tokens into 512-dimensional vectors via a trainable “embedding” with added “positional encoding” and random “dropout”.
- These embeddings are processed through 4 layers of self-attention and feedforward networks, each using “residual” (passthrough) connections and layer normalization (no masking is applied).
- The **decoder** generates SMILES tokens in an autoregressive manner from a fixed vocabulary of about 56 tokens.
- It first embeds the partial SMILES sequence with positional encoding, then, in each of its 4 layers, applies:
  - Masked self-attention (attending only to previous tokens),
  - Cross-attention over the encoder output, and
  - A feedforward network.
- A linear generator converts the decoder output into “logits” over the SMILES vocabulary, interpreted as the probability of the next token.

The “model”: encoder-decoder transformer

- The **encoder** converts input NMR tokens into 512-dimensional vectors via a trainable “embedding” with added “positional encoding” and random “dropout”.
- These embeddings are processed through 4 layers of self-attention and feedforward networks, each using “residual” (passthrough) connections and layer normalization (no masking is applied).
- The **decoder** generates SMILES tokens in an autoregressive manner from a fixed vocabulary of about 56 tokens.
- It first embeds the partial SMILES sequence with positional encoding, then, in each of its 4 layers, applies:
  - Masked self-attention (attending only to previous tokens),
  - Cross-attention over the encoder output, and
  - A feedforward network.
- A linear generator converts the decoder output into “logits” over the SMILES vocabulary, interpreted as the probability of the next token.

The “model”: encoder-decoder transformer

- The **encoder** converts input NMR tokens into 512-dimensional vectors via a trainable “embedding” with added “positional encoding” and random “dropout”.
- These embeddings are processed through 4 layers of self-attention and feedforward networks, each using “residual” (passthrough) connections and layer normalization (no masking is applied).
- The **decoder** generates SMILES tokens in an autoregressive manner from a fixed vocabulary of about 56 tokens.
- It first embeds the partial SMILES sequence with positional encoding, then, in each of its 4 layers, applies:
  - Masked self-attention (attending only to previous tokens),
  - Cross-attention over the encoder output, and
  - A feedforward network.
- A linear generator converts the decoder output into “logits” over the SMILES vocabulary, interpreted as the probability of the next token.

The “model”: encoder-decoder transformer

- The **encoder** converts input NMR tokens into 512-dimensional vectors via a trainable “embedding” with added “positional encoding” and random “dropout”.
- These embeddings are processed through 4 layers of self-attention and feedforward networks, each using “residual” (passthrough) connections and layer normalization (no masking is applied).
- The **decoder** generates SMILES tokens in an autoregressive manner from a fixed vocabulary of about 56 tokens.
- It first embeds the partial SMILES sequence with positional encoding, then, in each of its 4 layers, applies:
  - Masked self-attention (attending only to previous tokens),
  - Cross-attention over the encoder output, and
  - A feedforward network.
- A linear generator converts the decoder output into “logits” over the SMILES vocabulary, interpreted as the probability of the next token.

The “model”: encoder-decoder transformer

- The **encoder** converts input NMR tokens into 512-dimensional vectors via a trainable “embedding” with added “positional encoding” and random “dropout”.
- These embeddings are processed through 4 layers of self-attention and feedforward networks, each using “residual” (passthrough) connections and layer normalization (no masking is applied).
- The **decoder** generates SMILES tokens in an autoregressive manner from a fixed vocabulary of about 56 tokens.
- It first embeds the partial SMILES sequence with positional encoding, then, in each of its 4 layers, applies:
  - Masked self-attention (attending only to previous tokens),
  - Cross-attention over the encoder output, and
  - A feedforward network.
- A linear generator converts the decoder output into “logits” over the SMILES vocabulary, interpreted as the probability of the next token.

What is “attention”? – a glorified transistor

It is a mechanism that computes a weighted sum of input features, enabling the model to focus on the most relevant parts of a sequence. Introduced in “Attention is all you need” by Vaswani et al. [4].

$$\underset{\longleftrightarrow}{\text{softmax}}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$

- For each token, a trainable **query** vector is compared to a set of trainable **key** vectors (from the same sequence in self-attention or from the encoder in cross-attention) via a dot-product.
- The resulting similarity scores are scaled and passed through a “softmax” (over the keys) to obtain **attention weights**.
- These weights determine how much each token’s associated trainable **value** vector contributes to the output.

Training and model predictions

- Model by Alberts et al. [2], $\sim$ 30M trainable parameters
- 679'195 training samples, batch size 4k
- Re-trained on NVIDIA 1$\times$A10 GPU[2] ($\sim$35h), up to 250k batches
- The model generates several hypotheses ranked by "score", i.e., log-likelihood of the prediction according to the model
- Top-N accuracy evaluated on 1'000 or 10'000 unseen samples
- "Beam search" is a heuristic to find the *most likely* hypotheses; use beams size 100 (i.e., 100 concurrent hypotheses, keep top 10)

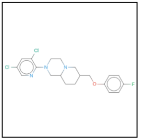
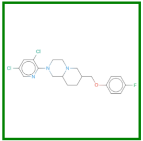
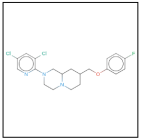
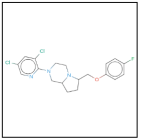
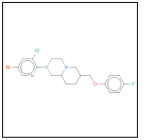
[2]Thanks to Lambda Cloud for compute credits

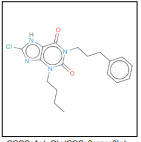
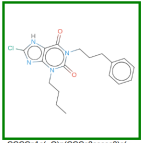
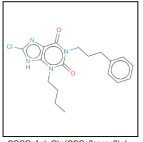
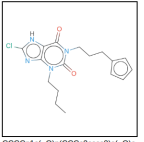
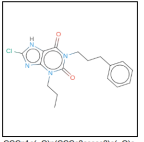
Evaluation

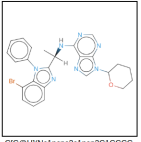
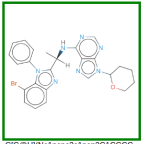
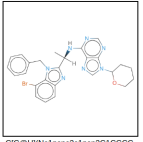
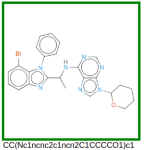
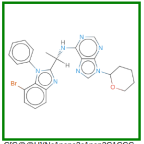
What is a *correct* prediction by the trained model?

- ✗ Predicted SMILES matches exactly
- ✓ Canonicalize before comparison
- ✓ Chirality-aware
- ✓ Exclude hypotheses with wrong sum formula

https://numpde.github.io/nmr-to-structure-lite/translate/f_figures.html

Reference	-0.0367	-0.1433	-0.1732	-0.1817
				
Fc1ccc(OCC2CCC3CN(c4ncc(Cl)cc4Cl)CCN3C2)cc1	Fc1ccc(OCC2CCC3CN(c4ncc(Cl)cc4Cl)CCN3C2)cc1	Fc1ccc(OCC2CCN3CCN(c4ncc(Cl)c4Cl)CC3C2)cc1	Fc1ccc(OCC2CCC3CN(c4ncc(Cl)cc4Cl)CCN3C2)cc1	Fc1ccc(OCC2CCC3CN(c4ncc(Br)cc4Cl)CCN3C2)cc1

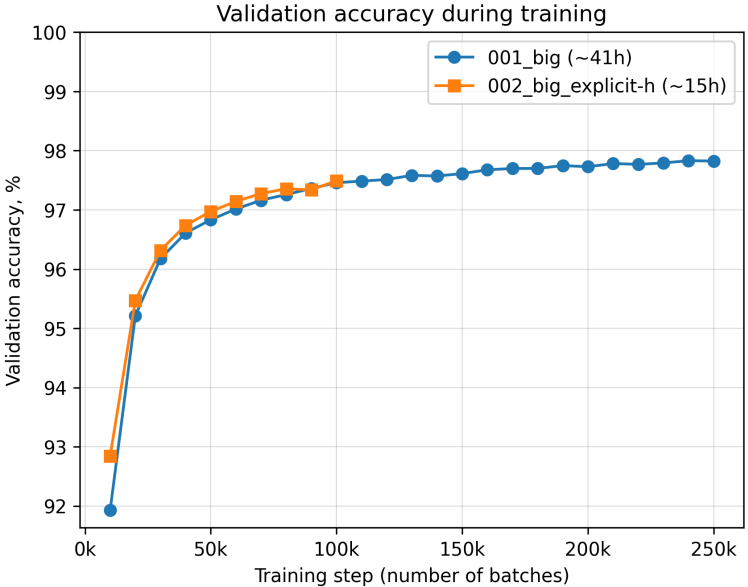
Reference	-0.0368	-0.1825	-0.1893	-0.1957
				
CCCCn1c(=O)n(CCCc2ccccc2)c(=O)c2[nH]c(Cl)nc21	CCCCn1c(=O)n(CCCc2ccccc2)c(=O)c2nc(Cl)[nH]c21	CCCCn1c(=O)n(CCCc2ccccc2)c(=O)c2nc(Cl)[nH]c21	CCCCn1c(=O)n(CCCc2ccccc2)c(=O)c2[nH]c(Cl)nc21	CCCn1c(=O)n(CCCc2ccccc2)c(=O)c2[nH]c(Cl)nc21

Reference	-0.0378	-0.1039	-0.1161	-0.1345
				
C[C@H](Nc1ncnc2c1ncn2C1CCCCO1)c1nc2ccccc(Br)c2n1-c1ccccc1	C[C@H](Nc1ncnc2c1ncn2C1CCCCO1)c1nc2ccccc(Br)c2n1-c1ccccc1	C[C@H](Nc1ncnc2c1ncn2C1CCCCO1)c1nc2ccccc(Br)c2n1Cc1ccccc1	CC(Nc1ncnc2c1ncn2C1CCCCO1)c1nc2ccccc(Br)c2n1-c1ccccc1	C[C@@H](Nc1ncnc2c1ncn2C1CCCCO1)c1nc2ccccc(Br)c2n1-c1ccccc1

Evaluation

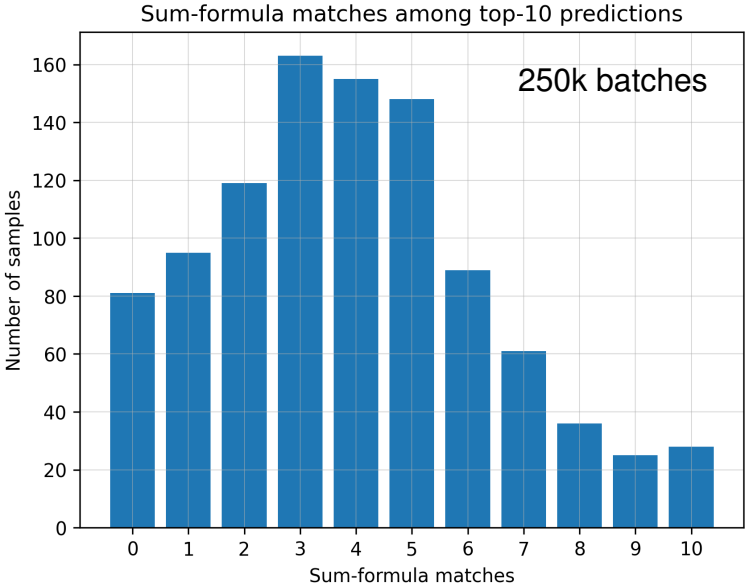
Validation accuracy during training:

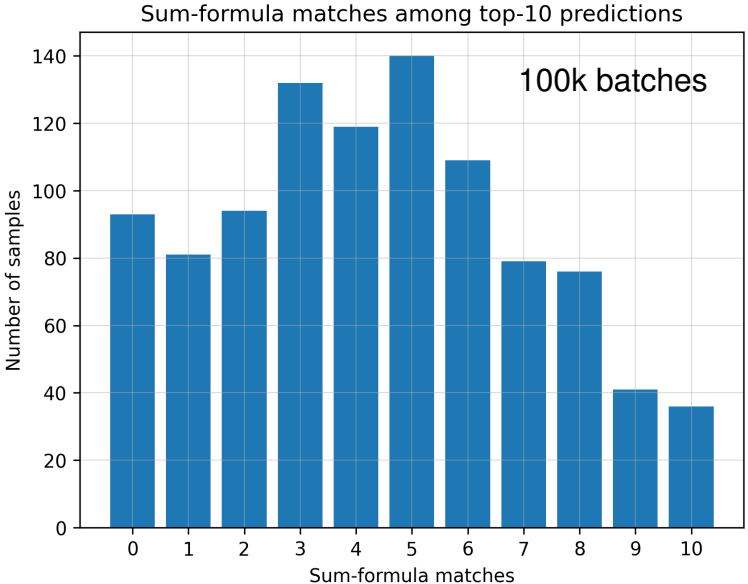
next-token prediction accuracy over the validation dataset with “teacher forcing”.

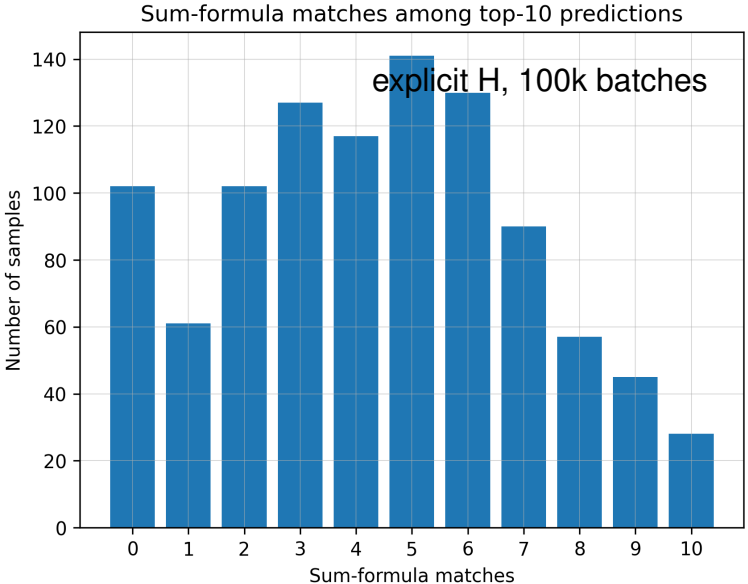

Validation accuracy during training
001_big (~41h)
002_big_explicit-h (~15h)
Validation accuracy, %
Training step (number of batches)
100
99
98
97
96
95
94
93
92
0k
50k
100k
150k
200k
250k

Evaluation

Do predictions have the correct sum-formula?

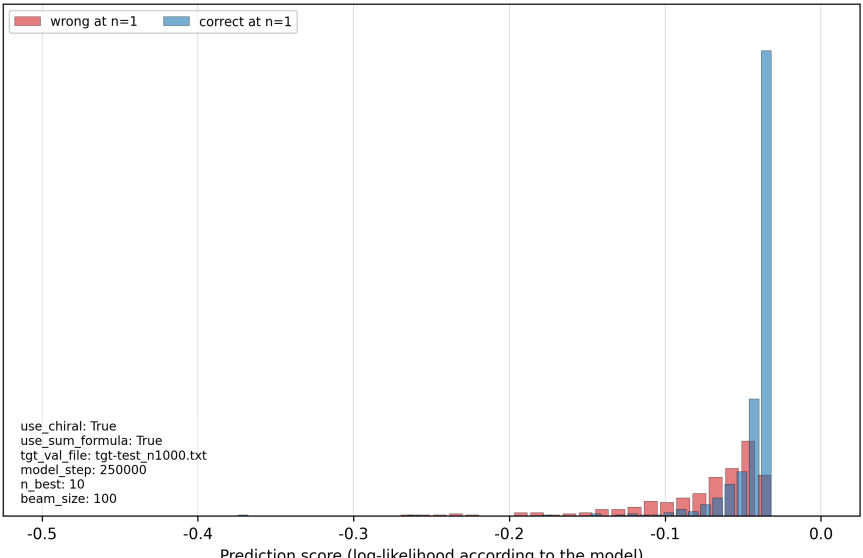

Sum-formula matches among top-10 predictions
250k batches
Number of samples
0
20
40
60
80
100
120
140
160
0
1
2
3
4
5
6
7
8
9
10
Sum-formula matches

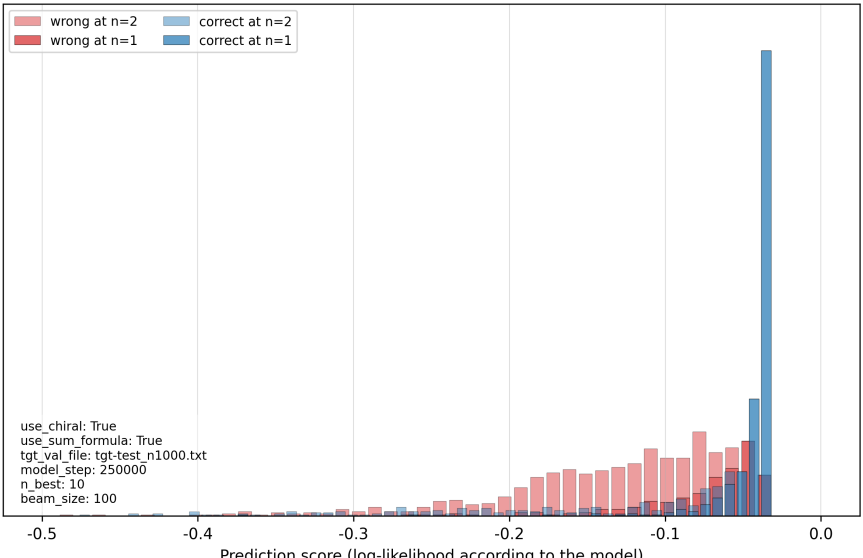

Sum-formula matches among top-10 predictions
100k batches
Number of samples
Sum-formula matches
0
20
40
60
80
100
120
140
0
1
2
3
4
5
6
7
8
9
10

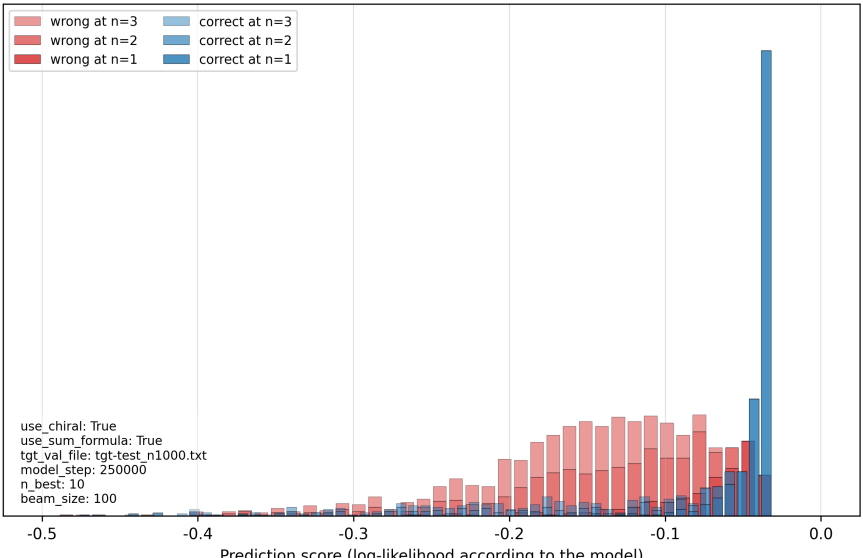

Sum-formula matches among top-10 predictions
explicit H, 100k batches
Number of samples
0
20
40
60
80
100
120
140
0
1
2
3
4
5
6
7
8
9
10
Sum-formula matches

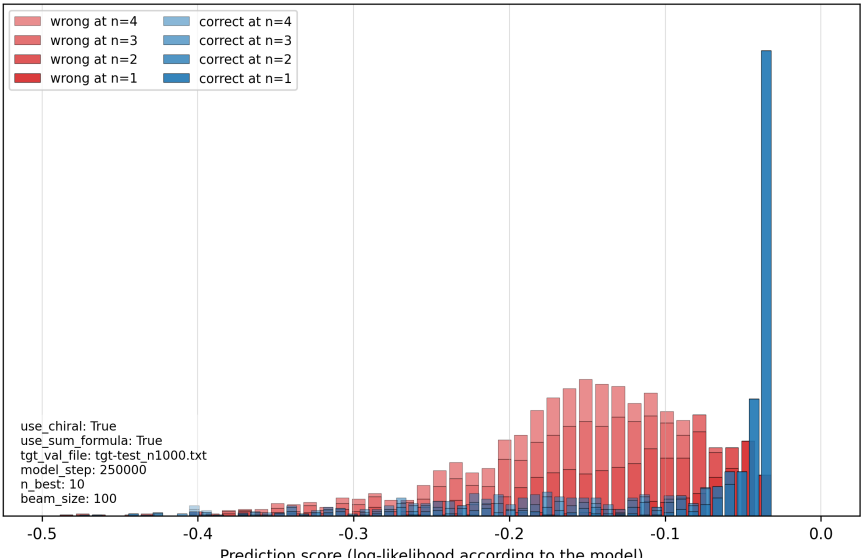
Evaluation

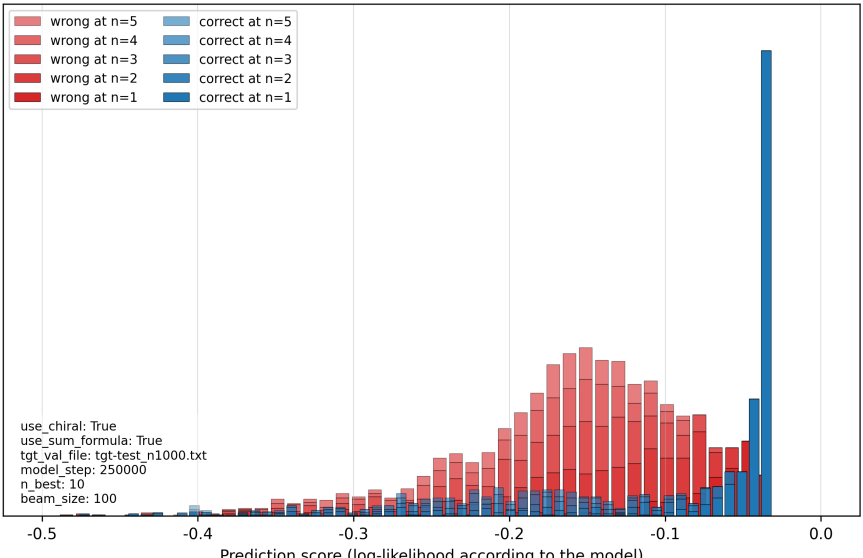
Which top-n predictions are correct?


Inferred SMILES (top-1 accuracy: 62.80%)
wrong at n=1
correct at n=1
use_chiral: True
use_sum_formula: True
tgt_val_file: tgt-test_n1000.txt
model_step: 250000
n_best: 10
beam_size: 100
-0.5
-0.4
-0.3
-0.2
-0.1
0.0
Prediction score (log-likelihood according to the model)


Inferred SMILES (top-2 accuracy: 71.30%)
wrong at n=2
wrong at n=1
correct at n=2
correct at n=1
use_chiral: True
use_sum_formula: True
tgt_val_file: tgt-test_n1000.txt
model_step: 250000
n_best: 10
beam_size: 100
-0.5
-0.4
-0.3
-0.2
-0.1
0.0
Prediction score (log-likelihood according to the model)

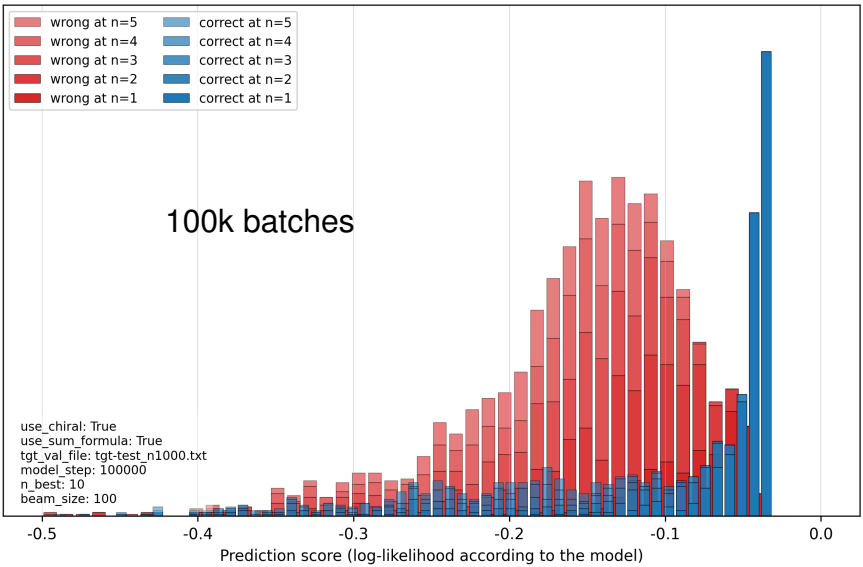

Inferred SMILES (top-3 accuracy: 74.50%)
wrong at n=3
wrong at n=2
wrong at n=1
correct at n=3
correct at n=2
correct at n=1
use_chiral: True
use_sum_formula: True
tgt_val_file: tgt-test_n1000.txt
model_step: 250000
n_best: 10
beam_size: 100
-0.5
-0.4
-0.3
-0.2
-0.1
0.0
Prediction score (log-likelihood according to the model)

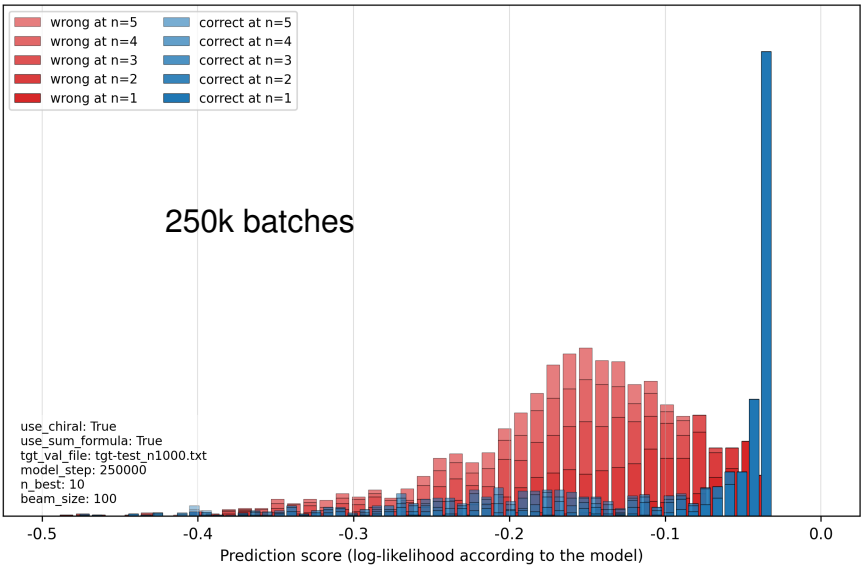

Inferred SMILES (top-4 accuracy: 76.50%)
wrong at n=4
wrong at n=3
wrong at n=2
wrong at n=1
correct at n=4
correct at n=3
correct at n=2
correct at n=1
use_chiral: True
use_sum_formula: True
tgt_val_file: tgt-test_n1000.txt
model_step: 250000
n_best: 10
beam_size: 100
-0.5
-0.4
-0.3
-0.2
-0.1
0.0
Prediction score (log-likelihood according to the model)


Inferred SMILES (top-5 accuracy: 77.30%)
wrong at n=5
wrong at n=4
wrong at n=3
wrong at n=2
wrong at n=1
correct at n=5
correct at n=4
correct at n=3
correct at n=2
correct at n=1
use_chiral: True
use_sum_formula: True
tgt_val_file: tgt-test_n1000.txt
model_step: 250000
n_best: 10
beam_size: 100
-0.5
-0.4
-0.3
-0.2
-0.1
0.0
Prediction score (log-likelihood according to the model)

Evaluation

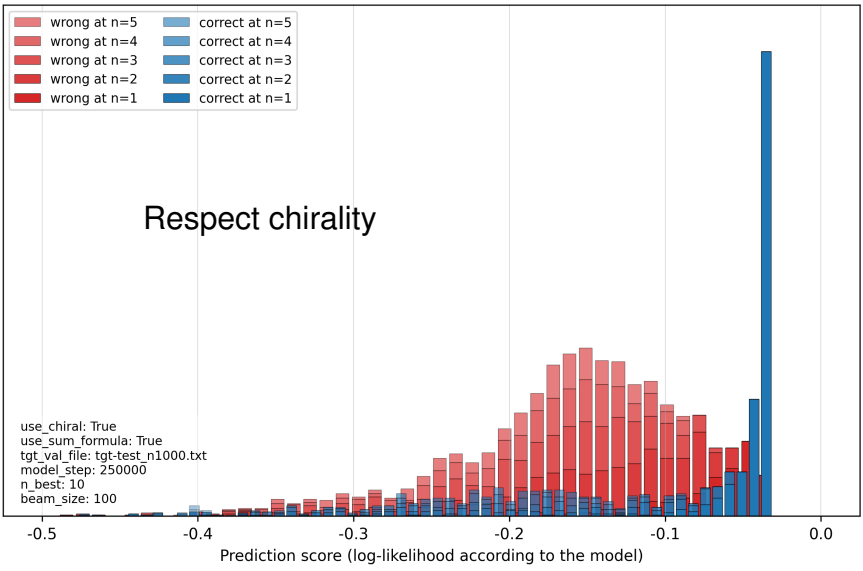
100k batches vs 250k batches

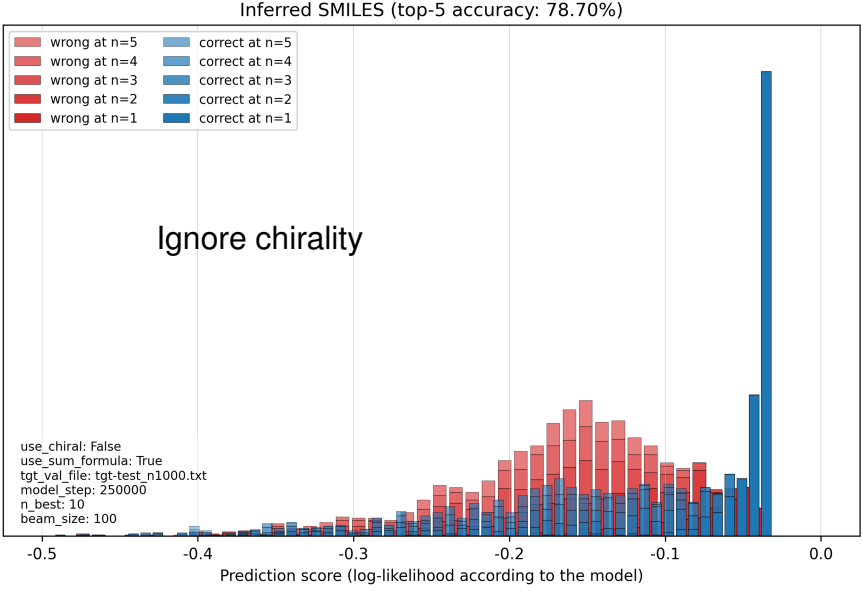

Inferred SMILES (top-5 accuracy: 74.70%)
wrong at n=5
wrong at n=4
wrong at n=3
wrong at n=2
wrong at n=1
correct at n=5
correct at n=4
correct at n=3
correct at n=2
correct at n=1
100k batches
use_chiral: True
use_sum_formula: True
tgt_val_file: tgt-test_n1000.txt
model_step: 100000
n_best: 10
beam_size: 100
-0.5
-0.4
-0.3
-0.2
-0.1
0.0
Prediction score (log-likelihood according to the model)


Inferred SMILES (top-5 accuracy: 77.30%)
wrong at n=5
wrong at n=4
wrong at n=3
wrong at n=2
wrong at n=1
correct at n=5
correct at n=4
correct at n=3
correct at n=2
correct at n=1
250k batches
use_chiral: True
use_sum_formula: True
tgt_val_file: tgt-test_n1000.txt
model_step: 250000
n_best: 10
beam_size: 100
-0.5
-0.4
-0.3
-0.2
-0.1
0.0
Prediction score (log-likelihood according to the model)

Evaluation

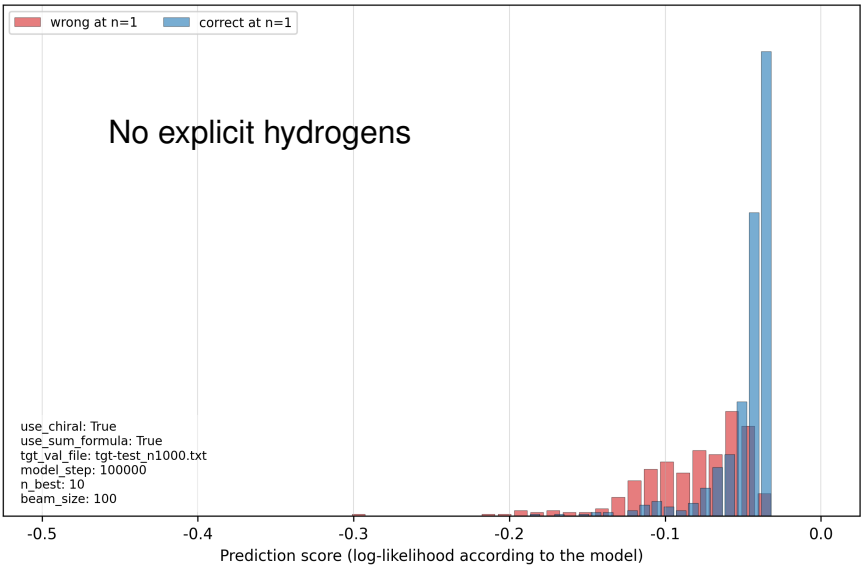
chirality: on vs off

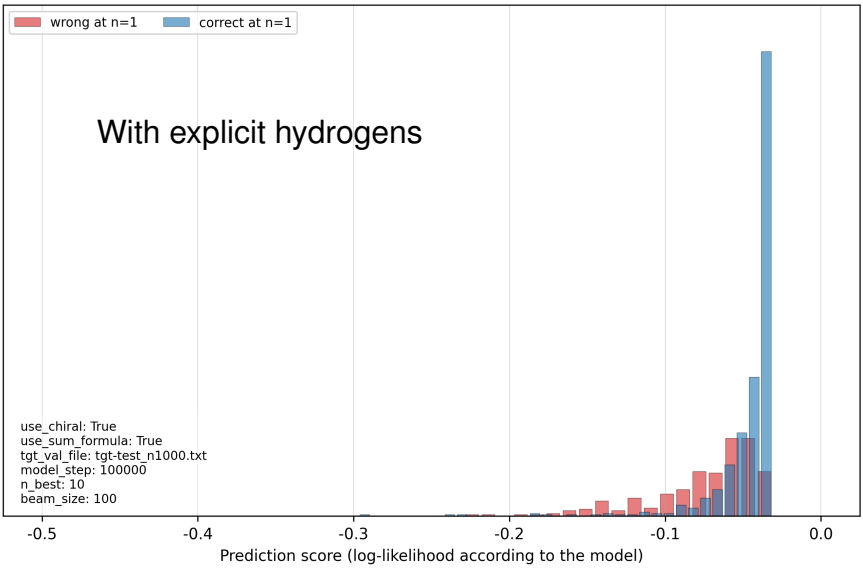

Inferred SMILES (top-5 accuracy: 77.30%)
wrong at n=5
wrong at n=4
wrong at n=3
wrong at n=2
wrong at n=1
correct at n=5
correct at n=4
correct at n=3
correct at n=2
correct at n=1
Respect chirality
use_chiral: True
use_sum_formula: True
tgt_val_file: tgt-test_n1000.txt
model_step: 250000
n_best: 10
beam_size: 100
-0.5
-0.4
-0.3
-0.2
-0.1
0.0
Prediction score (log-likelihood according to the model)

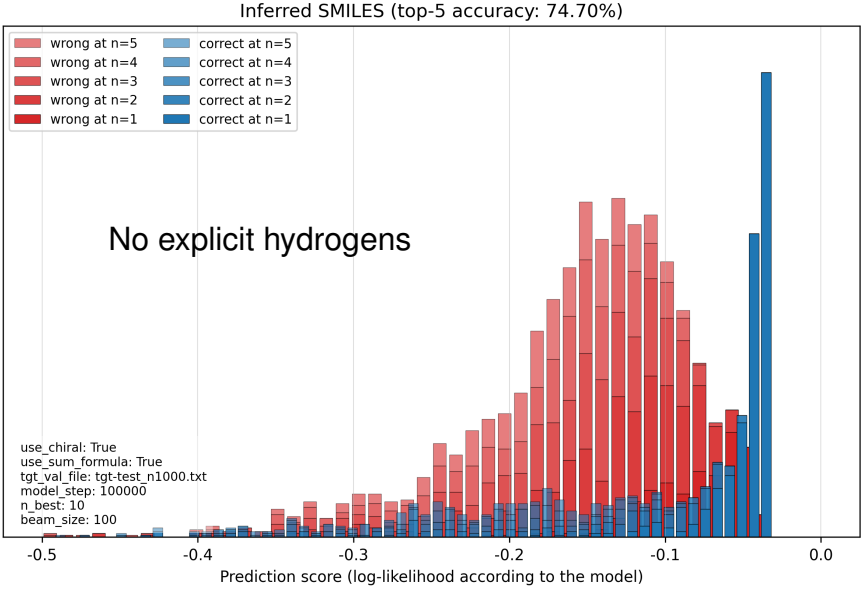

Inferred SMILES (top-5 accuracy: 78.70%)
wrong at n=5
wrong at n=4
wrong at n=3
wrong at n=2
wrong at n=1
correct at n=5
correct at n=4
correct at n=3
correct at n=2
correct at n=1
Ignore chirality
use_chiral: False
use_sum_formula: True
tgt_val_file: tgt-test_n1000.txt
model_step: 250000
n_best: 10
beam_size: 100
-0.5
-0.4
-0.3
-0.2
-0.1
0.0
Prediction score (log-likelihood according to the model)

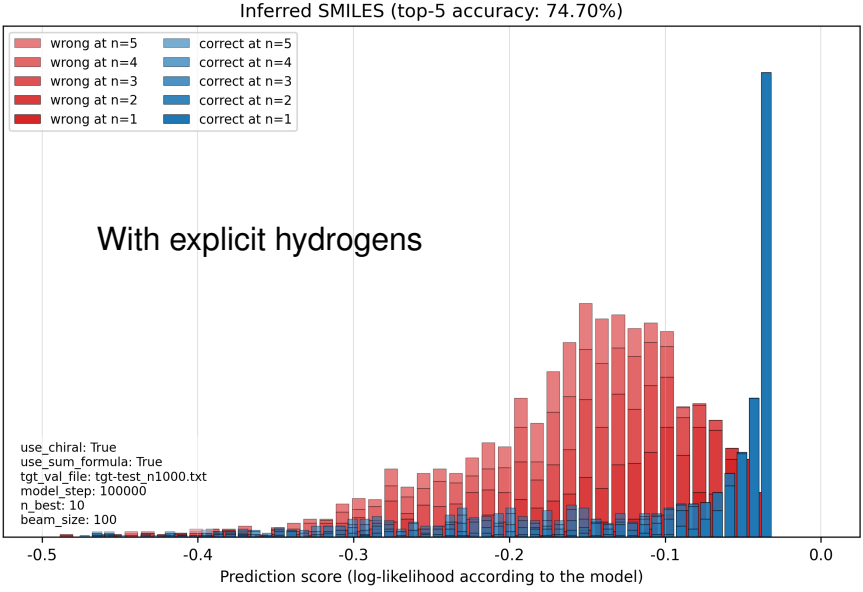
Evaluation

explicit hydrogens: no vs yes


Inferred SMILES (top-1 accuracy: 58.40%)
wrong at n=1
correct at n=1
No explicit hydrogens
use_chiral: True
use_sum_formula: True
tgt_val_file: tgt-test_n1000.txt
model_step: 100000
n_best: 10
beam_size: 100
-0.5
-0.4
-0.3
-0.2
-0.1
0.0
Prediction score (log-likelihood according to the model)

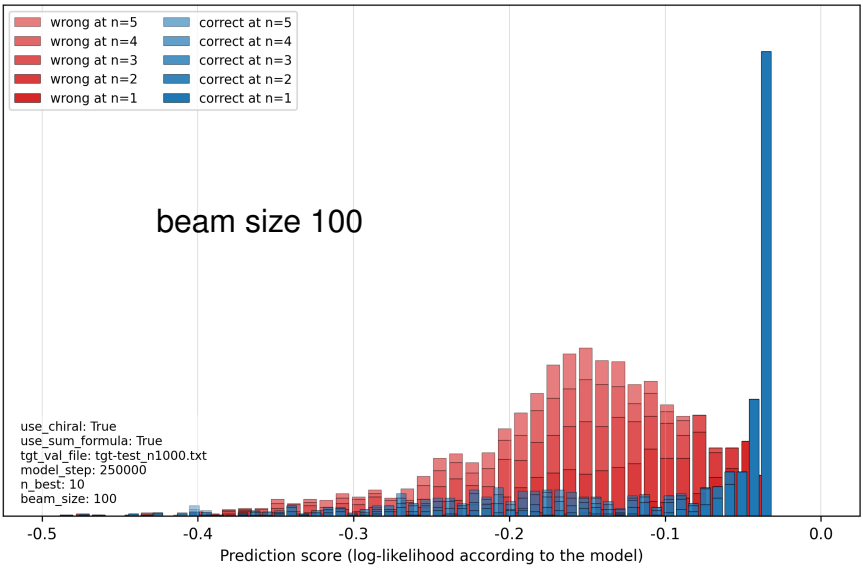

Inferred SMILES (top-1 accuracy: 59.60%)
wrong at n=1
correct at n=1
With explicit hydrogens
use_chiral: True
use_sum_formula: True
tgt_val_file: tgt-test_n1000.txt
model_step: 100000
n_best: 10
beam_size: 100
-0.5
-0.4
-0.3
-0.2
-0.1
0.0
Prediction score (log-likelihood according to the model)

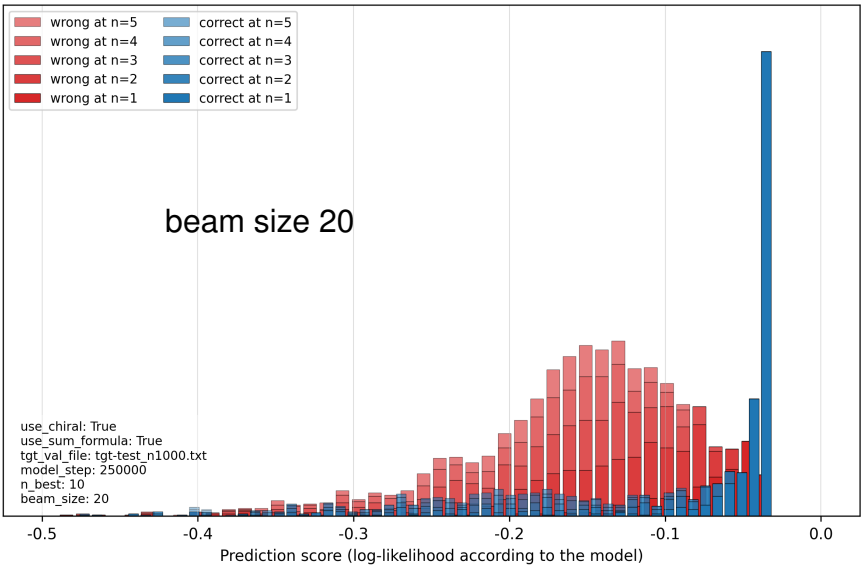

Inferred SMILES (top-5 accuracy: 74.70%)
wrong at n=5
wrong at n=4
wrong at n=3
wrong at n=2
wrong at n=1
correct at n=5
correct at n=4
correct at n=3
correct at n=2
correct at n=1
No explicit hydrogens
use_chiral: True
use_sum_formula: True
tgt_val_file: tgt-test_n1000.txt
model_step: 100000
n_best: 10
beam_size: 100
-0.5
-0.4
-0.3
-0.2
-0.1
0.0
Prediction score (log-likelihood according to the model)

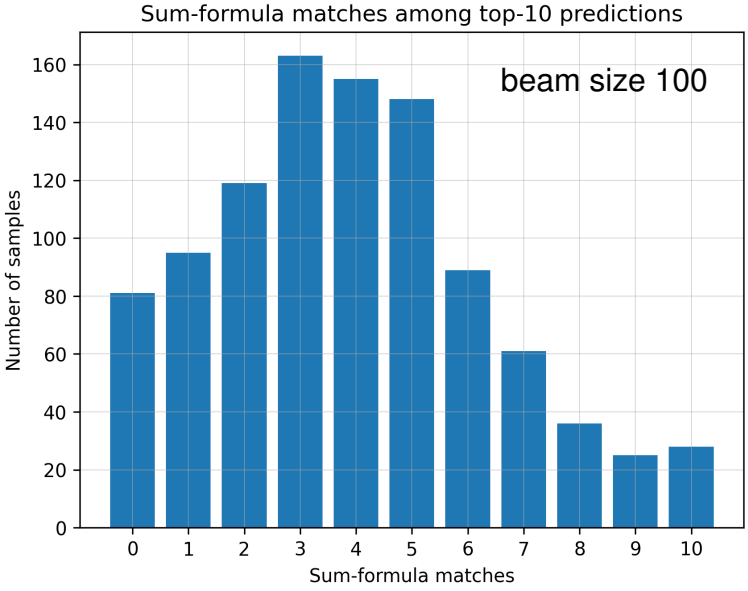

Inferred SMILES (top-5 accuracy: 74.70%)
wrong at n=5
wrong at n=4
wrong at n=3
wrong at n=2
wrong at n=1
correct at n=5
correct at n=4
correct at n=3
correct at n=2
correct at n=1
With explicit hydrogens
use_chiral: True
use_sum_formula: True
tgt_val_file: tgt-test_n1000.txt
model_step: 100000
n_best: 10
beam_size: 100
-0.5
-0.4
-0.3
-0.2
-0.1
0.0
Prediction score (log-likelihood according to the model)

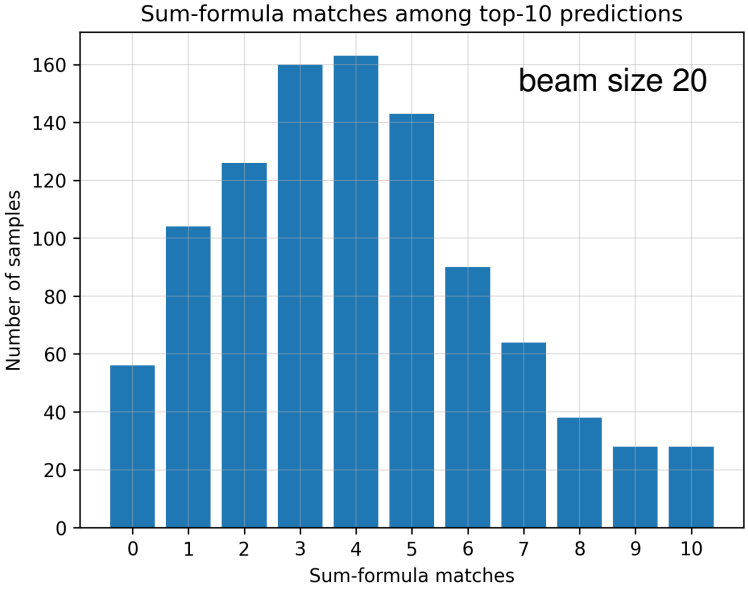
Evaluation

beam size (in hypothesis search): 100 vs 20


Inferred SMILES (top-5 accuracy: 77.30%)
wrong at n=5
wrong at n=4
wrong at n=3
wrong at n=2
wrong at n=1
correct at n=5
correct at n=4
correct at n=3
correct at n=2
correct at n=1
beam size 100
use_chiral: True
use_sum_formula: True
tgt_val_file: tgt-test_n1000.txt
model_step: 250000
n_best: 10
beam_size: 100
-0.5
-0.4
-0.3
-0.2
-0.1
0.0
Prediction score (log-likelihood according to the model)


Inferred SMILES (top-5 accuracy: 80.20%)
wrong at n=5
wrong at n=4
wrong at n=3
wrong at n=2
wrong at n=1
correct at n=5
correct at n=4
correct at n=3
correct at n=2
correct at n=1
beam size 20
use_chiral: True
use_sum_formula: True
tgt_val_file: tgt-test_n1000.txt
model_step: 250000
n_best: 10
beam_size: 20
-0.5
-0.4
-0.3
-0.2
-0.1
0.0
Prediction score (log-likelihood according to the model)


Sum-formula matches among top-10 predictions
beam size 100
Number of samples
0
20
40
60
80
100
120
140
160
0
1
2
3
4
5
6
7
8
9
10
Sum-formula matches

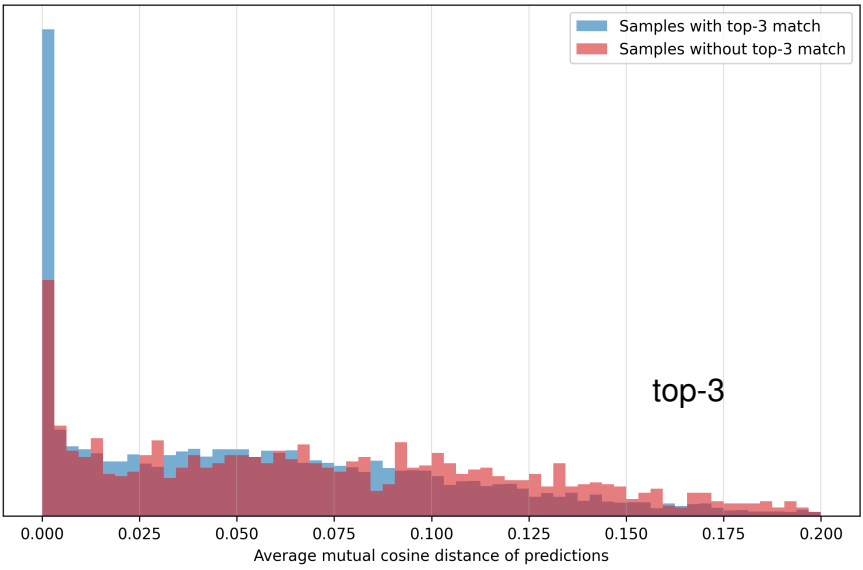

Sum-formula matches among top-10 predictions
beam size 20
Number of samples
0
20
40
60
80
100
120
140
160
0
1
2
3
4
5
6
7
8
9
10
Sum-formula matches

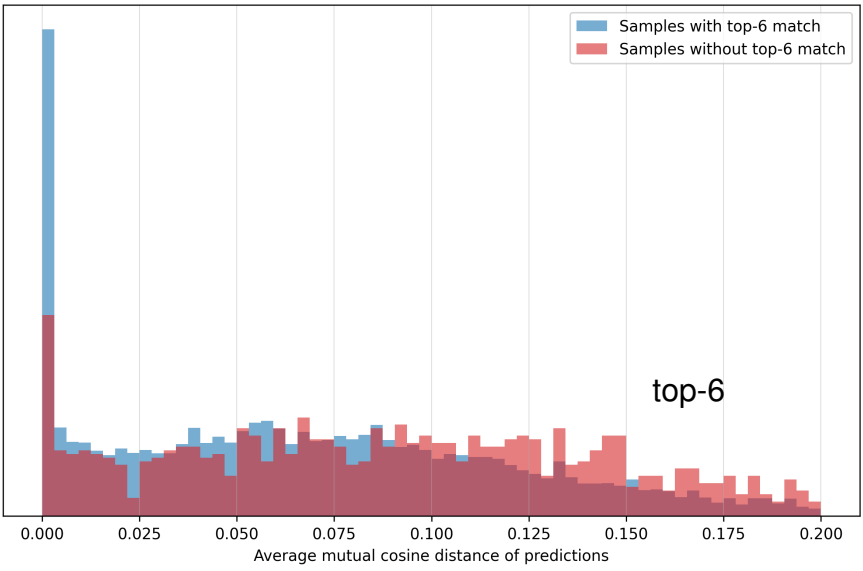
Evaluation

Average mutual cosine distance among top-n predictions,
or “dispersion”

... computed using the molecule-to-vector mapping ChemBERTa[3]
(self-supervised transformer pretrained on SMILES)

[3] Ahmad et al., 2022 [5]


Dispersion of top-3 predictions (normalized)
Samples with top-3 match
Samples without top-3 match
top-3
0.000
0.025
0.050
0.075
0.100
0.125
0.150
0.175
0.200
Average mutual cosine distance of predictions


Dispersion of top-6 predictions (normalized)
Samples with top-6 match
Samples without top-6 match
top-6
0.000
0.025
0.050
0.075
0.100
0.125
0.150
0.175
0.200
Average mutual cosine distance of predictions

Bayes:

$$P(\checkmark \mid d \text{ is small}) = \frac{P(d \text{ is small} \mid \checkmark)\, P(\checkmark)}{P(d \text{ is small} \mid \checkmark)\, P(\checkmark) + P(d \text{ is small} \mid \text{✗})\, P(\text{✗})}$$

Given

$$\frac{P(d \text{ is small} \mid \text{✗})}{P(d \text{ is small} \mid \checkmark)} \approx \frac{1}{2}, \quad P(\checkmark) \approx 80\%, \quad P(\text{✗}) \approx 20\%,$$

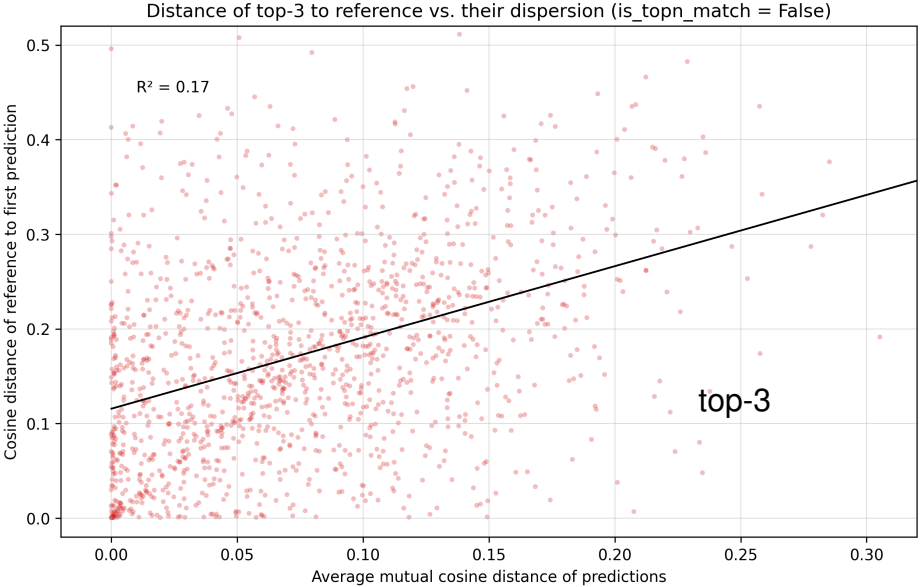
we have

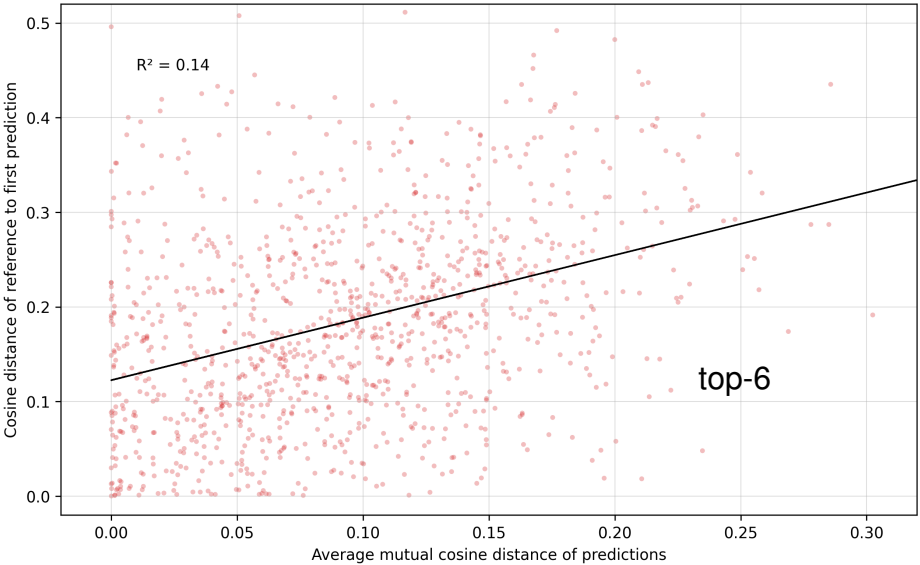
$$P(\checkmark \mid d \text{ is small}) \approx \frac{80\%}{80\% + \frac{1}{2} \times 20\%} \approx 89\%.$$

$\rightsquigarrow$ Very small prediction dispersion indicates a top-n match.
Similarly, high dispersion indicates a mismatch.

Evaluation

Distance of the top prediction to the reference
vs
average distance among top-n predictions


Distance of top-3 to reference vs. their dispersion (is_topn_match = False)
R² = 0.17
top-3
Cosine distance of reference to first prediction
Average mutual cosine distance of predictions
0.0
0.1
0.2
0.3
0.4
0.5
0.00
0.05
0.10
0.15
0.20
0.25
0.30


Distance of top-6 to reference vs. their dispersion (is_topn_match = False)
R² = 0.14
top-6
Cosine distance of reference to first prediction
Average mutual cosine distance of predictions
0.0
0.1
0.2
0.3
0.4
0.5
0.00
0.05
0.10
0.15
0.20
0.25
0.30

Molecular descriptors

We have a baseline confidence of $\sim$80%.

By looking at chemical properties of the molecules, can we improve our confidence of a top-n match?

In other words, construct a method such that: if it says “match”, we are confident that the transformer model prediction is indeed correct.

This is called “precision”, computed as

$$\text{Precision} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Positives}}$$

$\rightsquigarrow$ What makes molecules easier/harder to predict?

Chemical descriptors I

- **Molecular symmetry number:** counts unique canonical atom ranks; a higher value indicates lower overall symmetry. [link]
- **Number of chiral centers:** counts stereogenic atoms that can lead to non-superimposable mirror images. [link]
- **Number of diastereotopic protons:** counts hydrogen atoms in non-equivalent chemical environments (via CIP assignments).
- **Average Gasteiger charge:** computes the mean partial atomic charge using the Gasteiger method, reflecting the electron distribution. [link]
- **Fused ring count:** counts rings that are fused (sharing at least two atoms) with another ring, affecting rigidity and aromaticity.

Chemical descriptors II

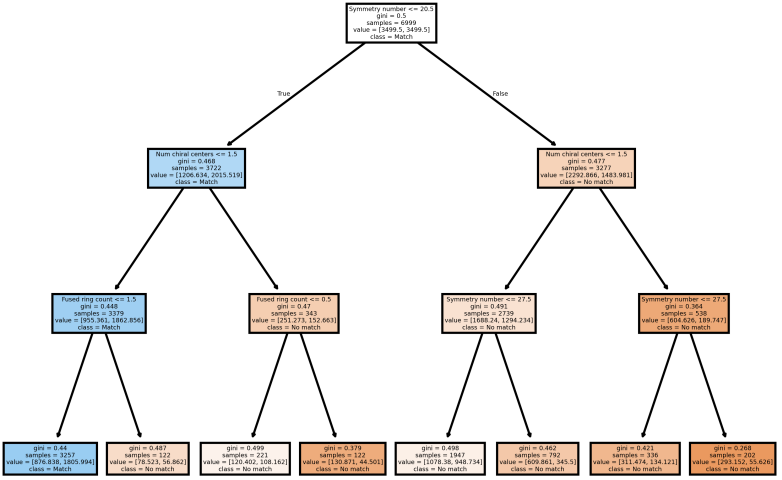
- **Bridgehead protons:** counts hydrogen atoms attached to heavy atoms shared by multiple rings, influencing steric effects and molecular stability.
- **Rotatable bonds:** enumerates bonds that are rotatable (typically single bonds outside rings), determining flexibility. [link]
- **Internal hydrogen bonds:** estimates potential intramolecular hydrogen bonding interactions by pairing donor (N/O with attached H) and acceptor (N/O with degree > 1) atoms.
- **Alpha heteroatom protons:** counts hydrogen atoms attached to carbons adjacent to heteroatoms (O or N).

cf. [code]

Chemical descriptors – decision tree

We train a “decision tree” on these molecular descriptors to predict whether there is a top-n match. Use the unseen validation dataset.

Caveat: we use the descriptors of the “unknown” reference molecule.

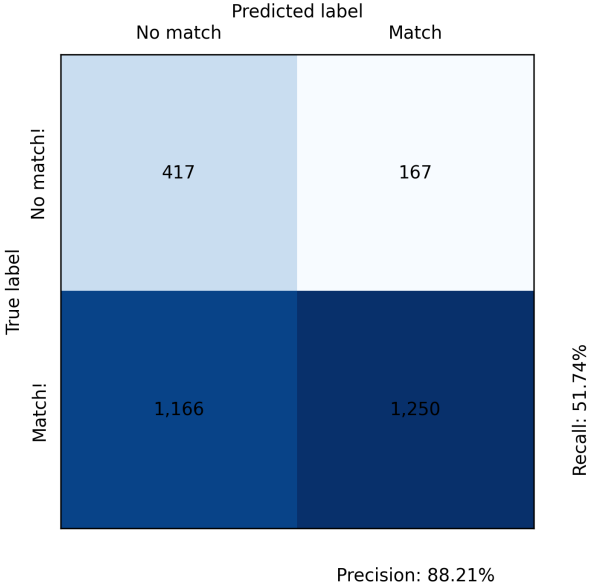

Symmetry number <= 20.5
gini = 0.5
samples = 6999
value = [3499.5, 3499.5]
class = Match
True
False
Num chiral centers <= 1.5
gini = 0.468
samples = 3722
value = [1206.634, 2015.519]
class = Match
Num chiral centers <= 1.5
gini = 0.477
samples = 3277
value = [2292.866, 1483.981]
class = No match
Fused ring count <= 1.5
gini = 0.448
samples = 3279
value = [955.361, 1862.856]
class = Match
Fused ring count <= 0.5
gini = 0.47
samples = 343
value = [251.273, 152.663]
class = No match
Symmetry number <= 27.5
gini = 0.491
samples = 2739
value = [1688.24, 1294.234]
class = No match
Symmetry number <= 27.5
gini = 0.364
samples = 538
value = [604.626, 189.747]
class = No match
gini = 0.44
samples = 3257
value = [876.838, 1805.994]
class = Match
gini = 0.487
samples = 122
value = [78.523, 56.862]
class = No match
gini = 0.499
samples = 221
value = [120.402, 108.162]
class = No match
gini = 0.379
samples = 122
value = [130.871, 44.501]
class = No match
gini = 0.498
samples = 1947
value = [1078.38, 948.734]
class = No match
gini = 0.462
samples = 792
value = [609.861, 345.5]
class = No match
gini = 0.421
samples = 336
value = [311.474, 134.121]
class = No match
gini = 0.268
samples = 202
value = [293.152, 55.626]
class = No match

IF:

- Symmetry number ≤ 20 (basically, size)
- Num chiral centers ≤ 1
- Fused ring count ≤ 1

then the decision tree indicates a top-n match.

In this case, we are $\sim$90% confident that the transformer model got it right.


Predicted label
No match
Match
True label
No match!
Match!
417
167
1,166
1,250
Recall: 51.74%
Precision: 88.21%

Maybe next time

- Functional groups as additional input
- Noisy spectra / multiplets
- Fine-tuning & evaluation on experimental data
- Multimodal input: better data?
- “Inductive bias" with graphs
- Fine-tuning an off-the-shelf self-supervised model
- Diffusion-like SMILES generation?
- Feedback loop (á la Jonas [6], Devata et al. [7])

thx.

this slide is intentionally left blank

References I

[1] Daniel Lowe. *Chemical reactions from US patents (1976–Sep2016)*. figshare, 2017. 10.6084/m9.figshare.5104873.v1.

[2] Marvin Alberts, Oliver Schilter, Federico Zipoli, Nina Hartrampf, and Teodoro Laino. "Unraveling Molecular Structure: A Multimodal Spectroscopic Dataset for Chemistry". In: *NeurIPS 2024 Datasets and Benchmarks Track*. 2024. 10.48550/arXiv.2407.17492.

[3] Marvin Alberts, Federico Zipoli, and Alain C. Vaucher. *Learning the Language of NMR: Structure Elucidation from NMR spectra using Transformer Models*. 2023. 10.26434/chemrxiv-2023-8wxcz.

References II

[4] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. "Attention Is All You Need". In: *Advances in Neural Information Processing Systems* 30 (2017), pp. 5998–6008. 10.48550/arXiv.1706.03762.

[5] Walid Ahmad, Elana Simon, Seyone Chithrananda, Gabriel Grand, and Bharath Ramsundar. "ChemBERTa-2: Towards Chemical Foundation Models". In: *ELLIS Machine Learning for Molecule Discovery Workshop 2021*. 2022. 10.48550/arXiv.2209.01712.

[6] Eric Jonas. "Deep Imitation Learning for Molecular Inverse Problems". In: *Advances in Neural Information Processing Systems*. Vol. 32. 2019.

References III

[7] Sriram Devata, Bhuvanesh Sridharan, Sarvesh Mehta, Yashaswi Pathak, Siddhartha Laghuvarapu, Girish Varma, and U. Deva Priyakumar. "DeepSPInN – Deep reinforcement learning for molecular structure prediction from infrared and 13C NMR spectra". In: *Digital Discovery* (2024). 10.1039/D4DD00008K.