Journal of Cheminformatics

RESEARCH Open Access

Accelerating the inference of string generation-based chemical reaction models for industrial applications

Mikhail Andronov[1*], Natalia Andronova[5], Michael Wand[1,3], Jürgen Schmidhuber[1,4] and Djork-Arné Clevert[2]

Abstract

Transformer-based, template-free SMILES-to-SMILES translation models for reaction prediction and single-step retrosynthesis are of interest to computer-aided synthesis planning systems, as they offer state-of-the-art accuracy. However, their slow inference speed limits their practical utility in such applications. To address this challenge, we propose speculative decoding with a simple chemically specific drafting strategy and apply it to the Molecular Transformer, an encoder-decoder transformer for conditional SMILES generation. Our approach achieves over 3X faster inference in reaction product prediction and single-step retrosynthesis with no loss in accuracy, increasing the potential of the transformer as the backbone of synthesis planning systems. To accelerate the simultaneous generation of multiple precursor SMILES for a given query SMILES in single-step retrosynthesis, we introduce Speculative Beam Search, a novel algorithm tackling the challenge of beam search acceleration with speculative decoding. Our methods aim to improve transformer-based models' scalability and industrial applicability in synthesis planning.

Scientific contribution

We present the first application of speculative decoding to accelerate the inference of a transformer neural network for SMILES-to-SMILES conversion for reaction modeling. We propose a chemically specific simple drafting strategy for speculative decoding of SMILES. We also introduce Speculative Beam Search - the first method to accelerate beam search decoding from the transformer with speculative decoding.

Keywords CASP, Speculative decoding, Single-step retrosynthesis, Fast inference, Reaction prediction

Introduction

Automated planning of organic chemical synthesis, first formalized around fifty years ago [1], is one of the core technologies enabling computer-aided drug discovery. While first computer-aided synthesis planning (CASP) systems relied on manually encoded rules [2, 3], researchers now primarily focus on CASP methods powered by artificial intelligence techniques. The design principles of the latter were outlined in the seminal work by Segler et al. [4]: a machine learning-based single-step retrosynthesis model combined with a planning algorithm. The former proposes several candidate retrosynthetic chemical transformations for a given molecule, and the latter,

*Correspondence:
Mikhail Andronov
mikhail.andronov@idsia.ch
[1] IDSIA, USI, SUPSI, Lugano, Switzerland
[2] Machine Learning Research, Pfizer Research and Development, Berlin, Germany
[3] Institute for Digital Technologies for Personalized Healthcare, SUPSI, Lugano, Switzerland
[4] AI Initiative, KAUST, Thuwal, Saudi Arabia
[5] Lugano, Switzerland

e.g., Monte-Carlo Tree Search, uses those candidates to construct a synthesis tree. Single-step retrosynthesis models are now commonly developed in two paradigms: template-based models and template-free models. Besides retrosynthesis, one can also build a model to predict the products of chemical reactions (Fig. 1).

The principle of template-based models is to use a set of SMIRKS templates extracted from reaction data and a machine learning model for classification or ranking to select a matching template for a query SMILES that will, upon application, transform the query into the product SMILES (for product prediction), or the SMILES of possible reactants (for single-step retrosynthesis). In contrast, in template-free models, the query transforms into the result without resorting to SMIRKS templates, e.g., with a sequence of predicted graph edits [5, 6] or through "translation" of the query SMILES into the desired SMILES with a conditioned text generation model [7–9]. While CASP systems leveraging template-based single-step models proved to be effective [10], there is an interest in building CASP with template-free models instead, as they demonstrate state-of-the-art accuracy in both single-step retrosynthesis and reaction product prediction [11, 12]. Most accurate template-free models are currently conditional autoregressive SMILES generators based on the transformer architecture [13], which also serves as the backbone for Large Language Models (LLM) [14, 15]. Molecular Transformer [8] was the first SMILES-to-SMILES transformer for reaction modeling, which served as a basis for subsequent modifications [7, 9], which demonstrate state-of-the-art accuracy in reaction prediction and single-step retrosynthesis. Unfortunately, the high accuracy of such autoregressive models like Chemformer [7] comes at the cost of a slow inference speed [16], which hinders their successful adoption as part of industrial CASP systems.

Possible ways of addressing the problem of slow inference in such autoregressive transformers include reducing the number of parameters in the model while trying to preserve its accuracy using knowledge distillation [17, 18] from a larger model or reducing the computational complexity of the transformer's forward pass by a clever architectural design [19]. However, such approaches present trade-offs with the accuracy of the model.

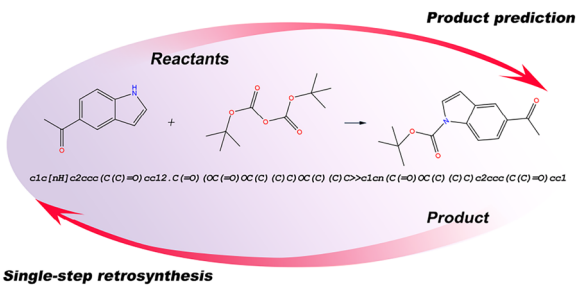

Fig. 1 Both reaction product prediction and single-step retrosynthesis can be formulated as SMILES-to-SMILES translation and approached with a model like an encoder-decoder transformer

Our work proposes another method to accelerate the inference from SMILES-to-SMILES translation models. We turn to speculative decoding [20, 21], a general technique for LLM inference acceleration, to make the Molecular Transformer [8] up to three times faster in reaction prediction and single-step retrosynthesis without changing the model architecture, training procedure and without compromising on accuracy. We propose a chemically specific drafting scheme for speculative decoding based on extracting subsequences of a query SMILES. We also design a speculative variant of beam search to accelerate the prediction of multiple candidate SMILES for a query in single-step retrosynthesis and product prediction.

Methods

Speculative decoding

Autoregressive sequence models, such as variants of the Transformer [13, 14, 22], generate sequences token by token, and every prediction of the next token requires a forward pass of the model. Such a process may be computationally expensive, especially for models with billions of parameters. Therefore, an intriguing question arises whether one could generate several tokens in one forward pass of the model, thus completing the output faster. Speculative decoding [20, 21] answers positively. Recently proposed as a method of inference acceleration for Large Language Models, it is based on the draft-and-verify idea. At every generation step, one can append some draft sequence to the sequence generated by the model so far and see if the model "accepts" the draft tokens.

If the draft sequence has length N, in the best case, the model adds $N + 1$ tokens to the generated sequence in one forward pass, and in the worst case, it adds one token as in standard autoregressive generation. The acceptance rate for one generated sequence is the number of accepted draft tokens divided by the total number of tokens in the generated sequence. One can also test multiple draft sequences in one forward pass taking advantage of parallelization, and choose the best one. Speculative decoding does not affect the content of the predicted sequence in any way compared to token-by-token decoding.

One can freely choose a way of generating draft sequences. For LLMs, one would usually use another smaller language model that performs its forward pass faster than the main LLM [20] or additional generation heads on top of the LLM's backbone [23]. However, one

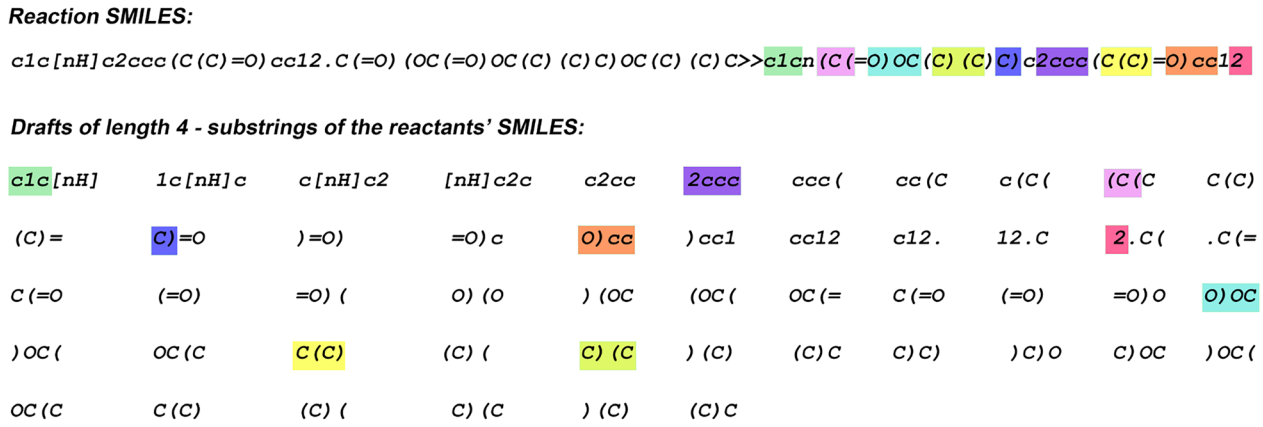

Fig. 2 Speculative decoding accelerates product prediction with the Molecular Transformer or a similar autoregressive SMILES generator. Before generating an output sequence, we prepare a list of subsequences of a desired length, e.g., four, of the tokenized query SMILES of reactants. Then, at every generation step, the model can copy up to four tokens from one of the draft sequences to the output, thus generating from one to five tokens in one forward pass. Colors highlight the best draft at every decoding step

can also construct draft sequences without calling any learned functions. For example, generate random draft sequences, even though their acceptance rate will be minimal.

Speculative decoding may imply the training of a separate model for drafting, but it does not require any changes to the architecture or training procedure of the main autoregressive model that generates the output.

Drafting strategy for SMILES

In reaction product prediction and single-step retrosynthesis as SMILES-to-SMILES conversion problems (Fig. 1), the source and target sequences are often relatively similar. In a chemical reaction, large fragments of reactants typically remain unchanged, which means that the SMILES of products and reactants have many common substrings. It is especially true if reactant and product SMILES are aligned to minimize the edit distance between them [12]. This characteristic of chemical reaction data enables a convenient heuristic for speculative decoding. Specifically, one can extract multiple substrings of a chosen length D from the query SMILES and use them as draft sequences with a high acceptance rate. Such an approach to draft generation is conceptually reminiscent of the LINGO method [24], which fragments a SMILES string into a collection of fixed-length overlapping substrings for QSPR modeling and molecular similarity estimation. Figure 2 demonstrates our drafting method in product prediction. Before generating the target string, we can assemble a list of token subsequences from the source sequence (reactant tokens) with a sliding window of a desired length (4 in this example) and stride 1. Then, at every generation step, we can try all draft sequences in one forward pass of the model to see if the model can copy up to 4 tokens from one of them. The draft token acceptance rate in this example reaches 78%. We can also make the drafting strategy "smarter" by only considering the substrings of the query SMILES that start with the token currently concluding the generated output and stripping the first token in those substrings (Fig. 3). This strategy would partially address the haunting problem of throughput-latency trade-off (section 3.1), and we use this strategy in our experiments as well (e.g., in section 3.3.2).

Speculative beam search

Speculative decoding was initially formulated for the accelerated generation of a single sequence for a given query, e.g. with greedy decoding or nucleus sampling. However, generating one sequence per query has limited utility in the context of reaction modeling. In CASP, the single-step retrosynthesis model must suggest multiple different reactant sets for every query product so that the planning algorithm can choose from them. Also, in product prediction, it is possible that a reaction results in a mixture of products. Therefore, the real problem is to accelerate the generation of multiple SMILES for a single given SMILES. When using a SMILES-to-SMILES transformer, beam search is the standard choice for generating a set of outputs instead of one.

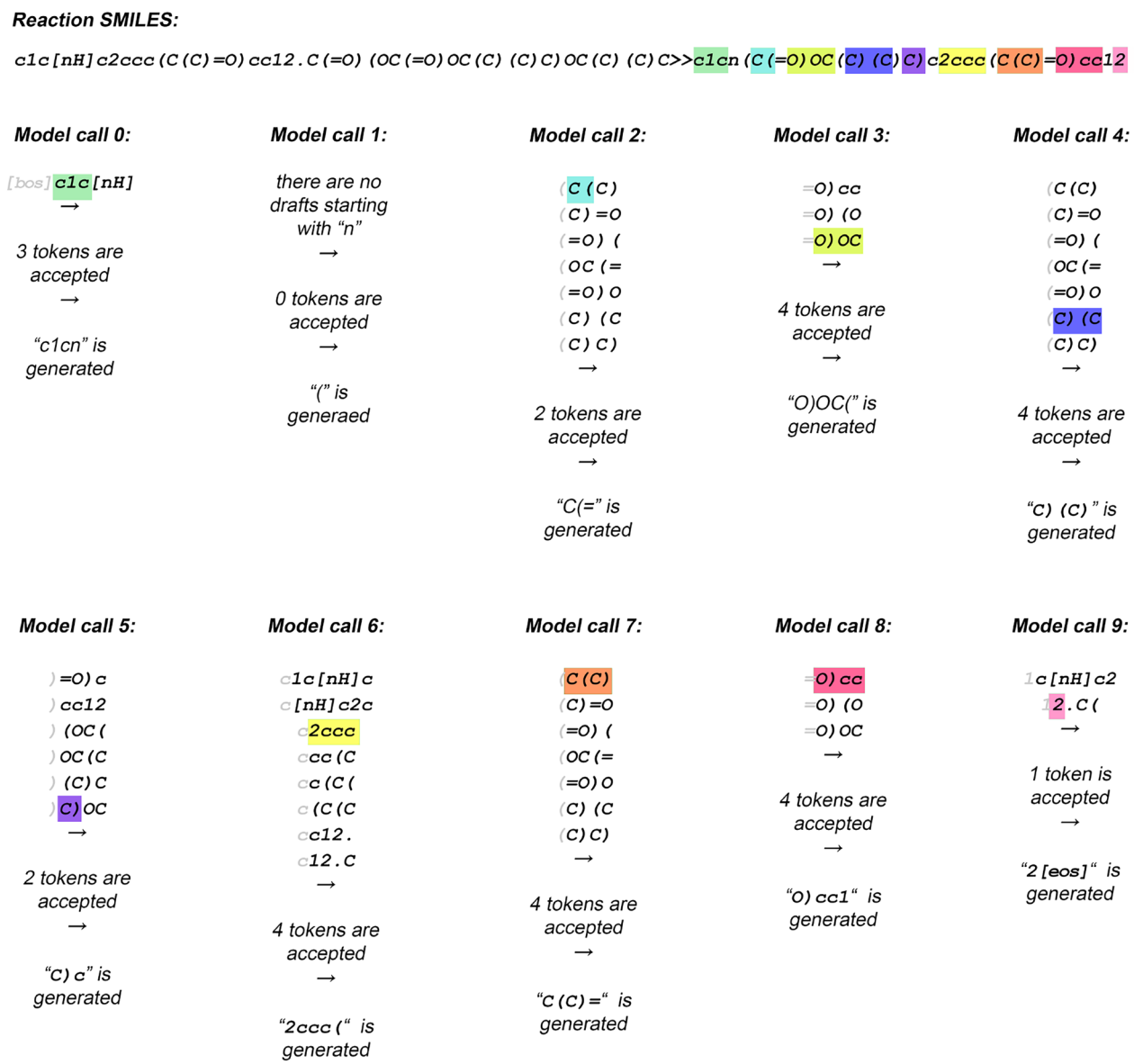

Fig. 3 One improvement over the naive drafting strategy in Fig. 2 is to only consider drafts that start with the token that currently concludes the generated output. For example, for the first call of the model we consider only those substrings of the source sequence that start with the [BOS] token, and there is only one such subsequence. We strip the first token (in gray) from the chosen subsequence and use the remainder as a draft. After two calls, we consider only the subsequences that start with "(" and strip this token to finalize drafts, and so on

We found a way to accelerate beam search with speculative decoding. The main idea behind it is that at every decoding step, we use a draft sequence to generate multiple candidate sequences of different lengths in one forward pass, which we then order by probabilities and keep the best K of them. We call our algorithm "speculative beam search" (SBS). Algorithm 1 provides an outline of the speculative beam search procedure we implement for our experiments. The algorithm features the following functions.

Algorithm 1 Speculative Beam Search

Require: $\mathcal{S}_{B\times L_s}$ - query token sequence (tensor of integers), K - number of best sequences to maintain, N - number of drafts, D - number of tokens in a draft. B is the batch size, L_s is the source sequence length, L_g is the length of the output generated so far, L_{gm} is the maximum requested length of the generated sequence, V is model's vocabulary size, E is the dimensionality of token embeddings.

$\mathcal{M}_{B\times L_s\times E} \leftarrow \text{ENCODER}(\mathcal{S})$
$\mathcal{G}_{B\times K\times 1} \leftarrow [\text{BOS}]$
$\mathcal{D}_{B\times N\times D} \leftarrow \text{MAKEDRAFTS}(\mathcal{S}, N, D)$
repeat
  $\mathcal{GD}_{BKN\times L_g+D} \leftarrow \text{CONCAT}(\mathcal{G}, \mathcal{D})$
  $\mathcal{L}_{BKN\times L_g+D\times V} \leftarrow \text{DECODER}(\mathcal{GD}, \mathcal{M})$
  ${\mathcal{D}^*}_{B\times K\times D} \leftarrow \text{BESTDRAFT}(\mathcal{L})$
  $\mathcal{T}_{B\times M\times L_g+D+1} \leftarrow \text{SEQTREE}(\mathcal{D}^*, \mathcal{L}, \mathcal{G})$
  $\mathcal{G}_{B\times K\times L_g+D+1} \leftarrow \text{EXTRACTTOP}(\mathcal{T}, K)$
until $L_g = L_{gm}$; **return** $\mathcal{G}$

Functions in the SBS algorithm

MAKEDRAFTS Produces N subsequences of length D from every query sequence to use as drafts according to our heuristic drafting scheme (section 2.2). The subsequences are extracted by a sliding window with a uniform step size. If N is larger than L_s, the extra sequences are padded with the most frequent non-service token in the vocabulary.

ENCODER Forward pass of the encoder of the Molecular Transformer. Gives the embeddings of the source sequences $\mathcal{M}$ to be used in the decoder.

DECODER Forward pass of the decoder of the Molecular Transformer. Gives the logits $\mathcal{L}$.

CONCAT: Concatenates every sequence decoded so far with all its corresponding draft sequences.

BESTDRAFT Uses the logits predicted by the decoder to select the best draft for every resulting sequence and discard the others. The best draft is the one with the largest number of accepted tokens. The model accepts a token at a given position if that token is among the top three most probable tokens that the model predicts at that position. There can also be fewer than top three contestants for acceptance, if the sum of their probabilitis exceeds a threshold like in nucleus sampling, e.g., 99.75%. For example, if one of the tokens gets 100% predicted probability, as it often happens in our SMILES-to-SMILES experiments, that token will be the only candidate for acceptance.

SEQTREE The core of the SBS algorithm. Proposes many candidate sequences for the decoding step based on the single accepted draft. The candidate sequences may have different lengths. The candidate sequences are organized in a tree, in which every path from the root to a leaf gives a candidate sequence continuation. Figure 4 demonstrates an example. M is the largest number of candidate continuations in the batch.

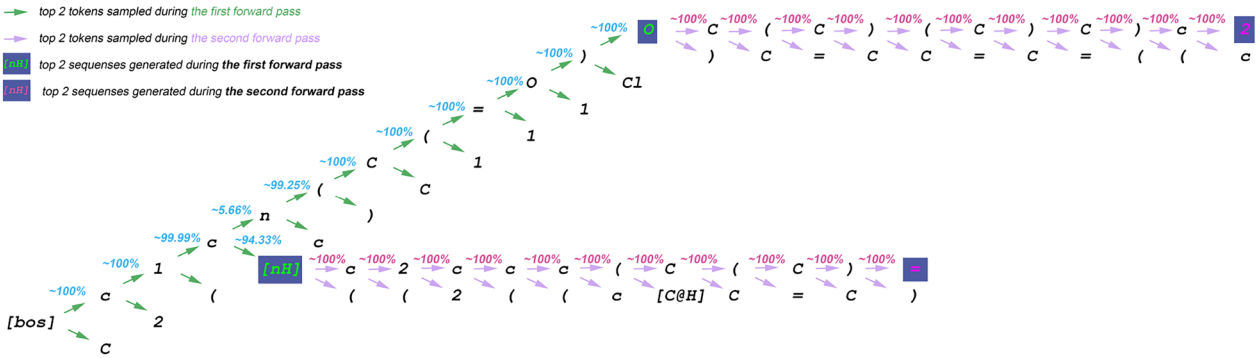

Fig. 4 An example of two first iterations of the sampling of candidate sequence trees in our speculative beam search with two generated sequences maintained for one query sequence in the batch. Here, we select the two best candidates at each iteration. The draft length D in this example is 10. The best draft for the first iteration is **c1cn(C(=O)**. It gets concatenated with the BOS token. The first forward pass generates 12 candidate sequences. The best sequences in the first iteration are **c1c[nH]** and **c1cn(C(=O)O**, and they become the "beams". In the second iteration, the best draft for the first "beam" is **c2ccc(C(C)**, and for the second one it is **C(C)(C)C)c**. The second forward pass generates 24 sequences overall, which all get sorted by their probabilities. The most probable sequences after the second iteration are **c1c[nH]c2ccc(C(C)=** and **c1cn(C(=O)OC(C)(C)C)c2**, and they become the generated sequences for the next iteration. In this example, after two iterations SBS generates two sequences of lengths 15 and 22, respectively, whereas the standard beam search would have generated two sequences of length 2

EXTRACTTOP: Orders the candidate sequences by probabilities, keeps the K ones with the highest probabilities and discards the others.

Model

We demonstrate the application of our method to the Molecular Transformer (MT) [8], an encoder-decoder transformer model for SMILES-conditioned SMILES generation.

The original Molecular Transformer [8] adopts OpenNMT [25], a general framework for neural machine translation, for SMILES-to-SMILES translation. Since the code in this framework is complex and intractable to customize, we decided to re-implement the model in PyTorch Lightning to keep only the necessary code and have more design freedom in the model's inference procedure implementation. We do not provide comparisons with the decoding speed from the original MT because its underlying OpenNMT inference implementation is optimized with various techniques that we do not use for the conceptual simplicity. Those techniques are independent of speculative decoding yet compatible with it, and the OpenNMT implementation would benefit from speculative decoding as much as the implementation in our demonstration.

Training details

For product prediction, we train our model with the same hyperparameters as in Schwaller et al. [8] with four encoder and decoder layers, eight heads, embedding dimensionality of 256, and feedforward dimensionality of 2048, which results in 11,4 million parameters. For single-step retrosynthesis, we set the hyperparameters as in n Zhong et al. [12] (six encoder and decoder layers, eight heads, embedding dimensionality of 256, and feedforward dimensionality of 2048), which results in 17,4 million parameters. The dictionary is the same for the encoder and the decoder in both models. We use the Adam optimizer for both models.

Data

We used the open reaction data from US patents [26] for training all models. We trained the model for reaction product prediction as in the original paper [8] on the USPTO MIT dataset, a standard benchmark for product prediction, without reactant-reagent separation. We trained the model for single-step retrosynthesis on USPTO 50K, a standard dataset for the task. In this dataset, we augmented every reaction in the training set 20 times using SMILES augmentation [9] with root-aligned SMILES [12]. We followed the standard atomwise tokenization procedure [8] to tokenize SMILES.

Results and discussion

Our implementation of the Molecular Transformer successfully reproduces the accuracy scores of the original MT [8] that relies on OpenNMT. Comparing our MT and the original MT, we observe at most 0.2 percentage points discrepancy of top-1 to top-5 accuracy in product prediction with beam search (Table 1). Having verified the correctness of our MT's implementation in this way, we replace the standard decoding procedures for the MT with our methods based on speculative decoding and achieve a significant speed-up in product prediction and single-step retrosynthesis. We report our experiments done on one NVIDIA Tesla V100 GPU with 32GB of memory.

Throughput-latency trade-off

Before discussing the application of speculative decoding to SMILES generation, we should emphasize the existence of the *throughput-latency trade-off* in the transformer. When processing the test dataset, one can increase the batch size to improve throughput, as the processing will require fewer model calls. A larger batch size typically reduces the time needed to process the entire dataset. However, this improvement comes at the cost of increased latency per batch. The forward pass becomes slower as larger batches demand more computational resources and memory. At a certain point, the increase in latency may outweigh the benefits of higher throughput, ultimately leading to slower overall decoding. Simultaneously improving latency and throughput is challenging, and one typically must carefully balance the two.

Product prediction

We tested our MT with speculative decoding for product prediction on USPTO MIT mixed, i.e., without an explicit separation between reactants and reagents. The test dataset in this benchmark comprises 40 thousand reactions.

Table 1 The top-5 accuracy of the Molecular Transformer (MT) in predicting reaction products on USPTO MIT (mixed) in the original report and our reimplementation. Both models use beam search with beam size 5 to generate five predictions for every query. Our model successfully reproduces the accuracy of the original model with negligible discrepancy

Accuracy	Original MT	Our MT	Δ
Top-1, %	88.6	88.4	−0.2
Top-2, %	92.4	92.5	0.1
Top-3, %	93.5	93.7	0.2
Top-5, %	94.5	94.7	0.2

Greedy decoding acceleration

When serving a transformer as an AI assistant that predicts reaction products, one could start by using greedy decoding for inference. Table 2 summarizes our experiments with greedy decoding from MT on the test set of USPTO MIT. On the V100 GPU, our greedy speculative decoding is faster than the greedy decoding by 3.53, 2.84, 1.57, and 1.18 times on average at batch sizes 1, 4, 16, and 32, respectively. Here, the throughput-latency trade-off comes into play. With our heuristic drafting, we ensure a high acceptance rate by trying multiple drafts in parallel for every sequence in the batch. However, it can significantly inflate the effective batch size of the model's input. As a result, the latency of the forward pass will worsen, reversing all benefits of speculative decoding at larger batch sizes. We find the balance between throughput and latency by doing a grid search over the number of drafts and the draft length on a fraction (500 reactions) of the test dataset (Fig. 5). For batch size 1, 23 drafts of 17 tokens give the best speed-up to greedy decoding. For batch size 4, 15 drafts of 14 tokens for every sequence work best. For batch size 16, it is 7 drafts of 7 tokens. For batch size 32, it is 3 drafts of 5 tokens (Fig. 5A). The results in Table 2 are given for the corresponding drafting parameters. At even larger batch sizes, the effect of speculative greedy decoding reverses, and it becomes slower than standard greedy decoding.

Even accelerating greedy decoding inference with the batch size of 1 would be sufficient for an improved user experience with reaction prediction assistants: chemists would usually enter one query at a time in a user interface of a reaction model like IBM RXN.

The model's accuracy is 88.3% with greedy decoding, and it is not affected when switching to speculative greedy decoding.

Beam search acceleration

We compare the inference time of standard beam search with beam size 5 and our speculative beam search in product prediction (Table 3). Standard beam search completes in around 298 min, 167 min, 152 min, and 88 min on average at batch sizes 1, 2, 3, and 4, respectively. SBS works consistently faster, and gives a speed-up of around 2.12, 1.53, 1.63, and 1.07 times. The speed-up becomes smaller as the batch size increases due to the increased latency of a single forward pass of the model. Before comparing the methods, we determined the optimal drafting parameters for SBS using a grid search as in the greedy speculative decoding experiments. In Table 3, SBS uses 23 drafts of 10 tokens for batch size 1, 10 drafts of 14 tokens for batch size 2, 10 drafts of 9 tokens for batch size 3, and 7 drafts of 10 tokens for batch size 4 (Fig. 5B). The optimal number of simultaneous drafts decreases to combat the increasing forward pass latency as the batch size grows.

While speculative greedy decoding guarantees that the generated outputs will be the same as the outputs generated by standard greedy decoding, our SBS does not guarantee the same outputs as beam search. However, in practice the difference is negligible. The top-1 accuracy of SBS in product prediction is 88.4 %, the top-3 accuracy is 93.7 %, and the top-5 accuracy is 94.7 % (Table 6A). There is practically no difference in accuracy compared to standard beam search in product prediction (Table 6A): the accuracy percentage differs in the second decimal

Table 2 Wall time of the model's inference on the USPTO MIT test set in reaction product prediction with standard and speculative greedy decoding. "B" stands for batch size. The average time and the standard deviation are estimated based on five runs. The number of drafts and the draft length in the speculative greedy algorithm is different for every batch size to ensure the most speed-up by balancing the throughput-latency trade-off: 23 drafts of length 17 for B=1, 15 drafts of length 14 for B=4, 7 drafts of length 7 for B=16, 3 drafts of length 5 for B=32

Decoding	Time (B=1), min	Time (B=4), min	Time (B=16), min	Time (B=32), min
Greedy	190.5 ± 36.0	81.8 ± 12.3	28.7 ± 2.3	16.8 ± 1.8
Speculative Greedy	53.9 ± 3.6	28.8 ± 3.2	18.2 ± 1.0	14.2 ± 0.3

Table 3 Wall time of the model's inference on the USPTO MIT test set in reaction product prediction with beam search (BS) and speculative beam search (SBS). The number of generated sequences (denoted by "K") is 5. "B" stands for batch size. The number of drafts and the draft length in the speculative beam search is different for every batch size. The average time and the standard deviation are estimated based on five runs

Decoding (K=5)	Time (B=1), min	Time (B=2), min	Time (B=3), min	Time (B=4), min
Beam search	297.9 ± 20.2	166.5 ± 32.0	151.8 ± 19.4	87.7 ± 13.8
SBS	140.4 ± 7.2	109.1 ± 3.6	93.3 ± 1.3	82.1 ± 0.6

place in the Top-5 case but not in Top-1 or Top-3. The number of invalid predicted SMILES also differs insignificantly for the fifth prediction, not the first or the third.

Single-step retrosynthesis

We carried out single-step retrosynthesis experiments on USPTO 50K, in which the training dataset was augmented 20 times. The augmentation procedure is to construct alternative root-aligned [12] SMILES for every dataset entry. This augmentation minimizes the edit distance between reactants and products, which simplifies training, pushes the model's accuracy higher, and increases the acceptance rate in our speculative decoding method. We did not augment the test set; it comprises 5007 reactions.

Beam search at different beam widths

Greedy decoding is less relevant to single-step retrosynthesis than to product prediction. Therefore, we describe only the acceleration of beam search with speculative decoding.

The wall time our retrosynthesis model takes to process the USPTO 50K test set with beam search and batch size 1 is around 67 min, 70 min, 71 min, and 72 min when generating 5, 10, 15, and 20 predictions for every query SMILES, respectively (Table 4), although we notice the spread of the wall time to be above 10 min in all cases. The average wall time increases as expected with the number of maintained hypotheses as the effective batch size becomes larger. When we replace the standard beam search with our SBS at the same batch size and the number of generated sequences, the generation finishes in around 29, 39, 46, and 47 min, accelerating the inference by around 2.3, 1.8, 1.56, and 1.53 times, respectively. Like in the case of greedy speculative decoding, we run a greed search over the number of drafts and the number of tokens in drafts on 500 reactions from the test set to find the optimal combinations of parameters that give the best acceleration. SBS with B=1 and K=5 uses 15 drafts of length 11, SBS with B=1 and K=10 uses 10 drafts of length 10, with B=1 and K=15 it also uses 10 drafts of length 10, and with B=1 and K=20 it uses 5 drafts of length 14 (Fig. 5C). To summarize, our SBS works consistently faster than beam search at batch size fixed to one, although the effect becomes less pronounced as the number of outputs per sequence grows.

Beam search at different batch sizes

We also check the behaviour of beam search and SBS at varying batch size and a fixed number of hypotheses per sequence. We keep the latter to be equal to 10, as it is a realistic setting for synthesis planning which also does not inflate the effective batch size too much. As Table 5 shows, the standard beam search finishes in 70, 42, 27, and 19 min on average with batch size 1, 2, 4, and 8, respectively. SBS finishes faster: in 39, 30, 28, and 20 min, respectively. The speed-up is 1.78 with batch size 1 and 1.4 with batch size 2, and with batch sizes 4 and 8 SBS becomes slightly slower than beam search. The number of drafts and draft length is again optimized beforehand with the help of a grid search. To further combat the latency increase overwhelming the throuthput benefits, we tried changing the drafting strategy. In the variant of SBS, which we here call "SBS with smart drafts", we first select only the drafts that start with the same token as the last token generated by the model so far, and then strip

Table 4 Wall time of the model's inference on the USPTO 50k test set (5K reactions) in single-step retrosynthesis with beam search and speculative beam search (SBS). Batch size is 1. "B" stands for batch size, and "K" stands for the number of generated sequences (beam size for beam search). The number of drafts and the draft length in the speculative beam search is different for every K. The average time and the standard deviation are estimated based on five runs

Decoding (B=1)	Time (K=5), min	Time (K=10), min	Time (K=15), min	Time (K=20), min
Beam search	66.7 ± 11.1	70.0 ± 13.2	71.3 ± 14.9	72.2 ± 15.9
SBS	28.9 ± 2.9	39.2 ± 3.4	45.5 ± 1.2	47.2 ± 1.2

Table 5 Wall time of the model's inference on the USPTO 50k test set (5K reactions) in single-step retrosynthesis with beam search and speculative beam search (SBS). "B" stands for batch size, and "K" stands for the number of generated sequences (beam size for beam search). Here K is equal to 10. The number of drafts and the draft length in the speculative beam search is different for every B. The average time and the standard deviation are estimated based on five runs

Decoding (K=10)	Time (B=1), min	Time (B=2), min	Time (B=4), min	Time (B=8), min
Beam search	70.0 ± 13.2	42.1 ± 6.7	26.9 ± 3.5	18.7 ± 1.2
SBS	39.2 ± 3.4	30.1 ± 0.8	28.1 ± 0.3	20.1 ± 0.2
SBS (smart drafts)	22.7 ± 1.3	16.6 ± 0.4	14.8 ± 0.1	13.4 ± 0.1

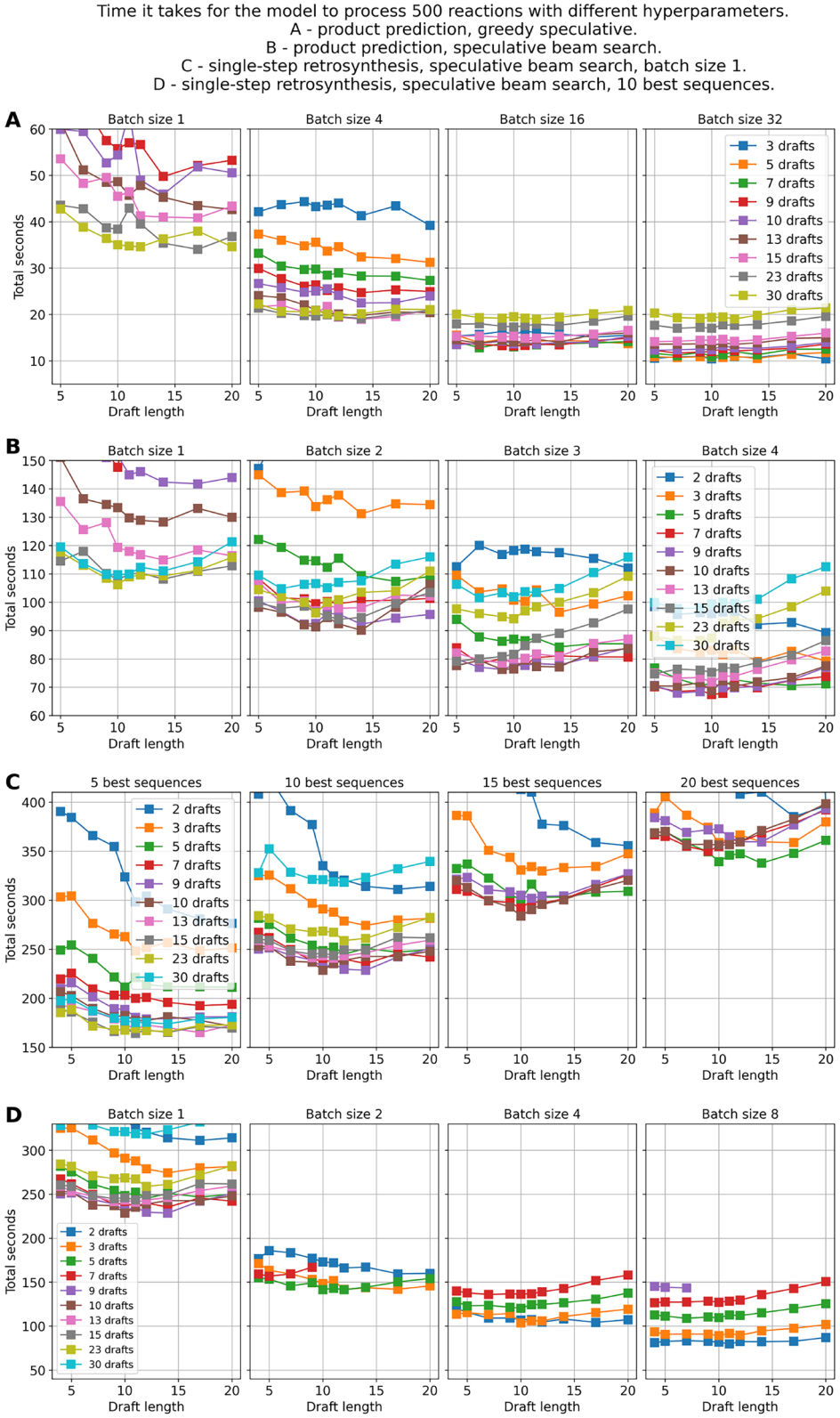

Fig. 5 Results of the grid search over the draft length and the number of drafts: the model needs different time to give predictions for 500 queries from the full test dataset with different combinations of hyperparameters in reaction prediction and single-step retrosynthesis. We ran the grid search once without estimating the spread of processing time

Table 6 The top-N accuracy of our model and the proportion of invalid SMILES in the N-th prediction with different decoding strategies in product prediction on USPTO MIT (A) and in single-step retrosynthesis on USPTO 50K (B). The difference in accuracy between standard and speculative beam search is negligible

(A) Product prediction		**Top-1, %**	**Top-3, %**	**Top-5, %**
Accuracy	Beam search	88.425	93.690	94.733
	SBS	88.425	93.690	94.720
		Pred. 1, %	Pred. 3, %	Pred. 5, %
Invalid SMILES	Beam search	0.232	8.270	13.182
	SBS	0.232	8.275	13.285
(B) Single-step retrosynthesis		**Top-1, %**	**Top-5, %**	**Top-10, %**
Accuracy	Beam search	52.077	82.069	88.918
	SBS	52.077	82.069	89.038
	SBS (smart drafts)	52.077	82.069	88.978
		Pred. 1, %	Pred. 5, %	Pred. 10, %
Invalid SMILES	Beam search	0.799	3.534	8.107
	SBS	0.799	3.534	8.007
	SBS (smart drafts)	0.799	3.534	8.187

the first draft token (Fig. 3). This strategy allows to reduce the number of drafts tried at the same time and reduce the effective batch size while maintaining high acceptance rate. The smart drafting allows SBS to consistently outperform the speed of beam search on batch sizes from 1 to 8 (Table 5), improving it by 3.1, 2.5, 1.8, and 1.4 times, respectively.

The accuracy when using SBS is the same as with beam search in top-1 and top-5; in top-10, it differs only in the decimals of a percent point (Table 6B). The proportion of invalid predictions also changes insignificantly. Thus, our SBS accelerates the MT's generation of multiple reactant sets several times without having to compromise on accuracy. Such a speed-up could make the autoregressive transformer a more attractive basis for single-step models in multi-step synthesis planning.

Limitations

The benefits of speculative decoding in accelerating greedy decoding and beam search decoding of SMILES with our drafting methods become less manifested with the increase of the batch size or the number of generated outputs per sequence. Since our methods rely on applying multiple drafts in parallel to every sequence being generated in the batch at every step, the effective batch size of the model's input inflates from BK to BKN, where B is the batch size, K is the number of generated outputs per sequence, and N is the number of drafts. This inflation leads to the increase in latency of the model's forward pass, which may cancel the benefits of reducing the number of model calls when BKN is large enough. Nonetheless, we are confident that the proposed speculative decoding methods are promising for building transformer-based CASP systems and reaction prediction assistants. The acceleration will likely extend into larger B and K with better drafting strategies that use fewer drafts (ideally one) with a high acceptance rate, and with other inference acceleration techniques used in conjunction with speculative decoding, such as kv-caching or quantization.

Conclusion

We combine speculative decoding and chemical insights to accelerate inference in the Molecular Transformer, a SMILES-to-SMILES translation model. Our methods make processing the test set up to three times faster in both single-step retrosynthesis on USPTO 50K and reaction product prediction on USPTO MIT compared to the standard decoding procedures (greedy decoding and beam search). We propose a novel Speculative Beam Search method to accelerate the generation of multiple output SMILES per query. Our method aims at making state-of-the-art template-free SMILES-generation-based models such as the Molecular Transformer more suitable for industrial applications such as computer-aided synthesis planning systems.

Author contributions

MA and NA conceptualized the paper idea. MA and NA wrote the code and conducted the experiments. MA wrote the manuscript with inputs from all co-authors. JS, DC and MW acquired the research funding, administered the project and provided supervision.

Funding
This study was partially funded by the European Union's Horizon 2020 research and innovation program under the Marie Skłodowska-Curie Innovative Training Network European Industrial Doctorate grant agreement No. 956832 "Advanced machine learning for Innovative Drug Discovery, and also by the Horizon Europe funding programme under the Marie Skłodowska-Curie Actions Doctoral Networks grant agreement "Explainable AI for Molecules - AiChemist" No. 101120466.

Data availability
The instructions for downloading the data to replicate the results are also in the GitHub repository https://github.com/Academich/translation-transformer

Declarations

Ethics approval and consent to participate
Not applicable

Consent for publication
Not applicable

Competing interests
The authors have no Conflict of interest to declare.

Received: 29 December 2024 Accepted: 16 February 2025
Published online: 10 March 2025

References

1. Pensak DA, Corey EJ (1977) LHASA-logic and heuristics applied to synthetic analysis. ACS Publications
2. Gasteiger J, Pförtner M, Sitzmann M, Höllering R, Sacher O, Kostka T, Karg N (2000) Computer-assisted synthesis and reaction planning in combinatorial chemistry. Perspect Drug Dis Design 20:245–264
3. Johnson PY, Burnstein I, Crary J, Evans M, Wang T (1989) Designing an expert system for organic synthesis: the need for strategic planning. ACS Publications
4. Segler MH, Preuss M, Waller MP (2018) Planning chemical syntheses with deep neural networks and symbolic AI. Nature 555(7698):604–610
5. Bradshaw J, Kusner MJ, Paige B, Segler MH, Hernández-Lobato JM (2018) A generative model for electron paths. arXiv preprint arXiv:1805.10970
6. Sacha M, Błaz M, Byrski P, Dabrowski-Tumanski P, Chrominski M, Loska R, Włodarczyk-Pruszynski P, Jastrzebski S (2021) Molecule edit graph attention network: modeling chemical reactions as sequences of graph edits. J Chem Inf Model 61(7):3273–3284
7. Irwin R, Dimitriadis S, He J, Bjerrum EJ (2022) Chemformer: a pre-trained transformer for computational chemistry. Mach Learning Sci Technol 3(1):015022
8. Schwaller P, Laino T, Gaudin T, Bolgar P, Hunter CA, Bekas C, Lee AA (2019) Molecular transformer: a model for uncertainty-calibrated chemical reaction prediction. ACS Central Sci 5(9):1572–1583
9. Tetko IV, Karpov P, Van Deursen R, Godin G (2020) State-of-the-art augmented NLP transformer models for direct and single-step retrosynthesis. Nat Commun 11(1):5575
10. Genheden S, Thakkar A, Chadimová V, Reymond J-L, Engkvist O, Bjerrum E (2020) Aizynthfinder: a fast, robust and flexible open-source software for retrosynthetic planning. J Cheminf 12(1):70
11. Meng Z, Zhao P, Yu Y, King I (2023) A unified view of deep learning for reaction and retrosynthesis prediction: current status and future challenges. arXiv preprint arXiv:2306.15890
12. Zhong Z, Song J, Feng Z, Liu T, Jia L, Yao S, Wu M, Hou T, Song M (2022) Root-aligned SMILES: a tight representation for chemical reaction prediction. Chem Sci 13(31):9023–9034
13. Vaswani A, Shazeer N, Parmar N, Uszkoreit J, Jones L, Gomez AN, Kaiser Ł, Polosukhin I(2017) Attention is all you need. Advances in neural information processing systems 30
14. Brown T, Mann B, Ryder N, Subbiah M, Kaplan JD, Dhariwal P, Neelakantan A, Shyam P, Sastry G, Askell A et al (2020) Language models are few-shot learners. Adv Neural Inf Proc Sys 33:1877–1901
15. Zhao WX, Zhou K, Li J, Tang T, Wang X, Hou Y, Min Y, Zhang B, Zhang J, Dong Z, etal (2023) A survey of large language models. arXiv preprint arXiv:2303.18223
16. Torren-Peraire P, Hassen AK, Genheden S, Verhoeven J, Clevert D-A, Preuss M, Tetko IV (2024) Models matter: the impact of single-step retrosynthesis on synthesis planning. Dig Dis 3(3):558–572
17. Hartog PB, Westerlund AM, Tetko IV, Genheden S (2024) Investigations into the efficiency of computer-aided synthesis planning. J Chem Inf Model. https://doi.org/10.1021/acs.jcim.4c01821
18. Hinton G, Vinyals O, Dean J (2015) Distilling the knowledge in a neural network. arXiv preprint arXiv:1503.02531
19. Tay Y, Dehghani M, Bahri D, Metzler D (2022) Efficient transformers: A survey. ACM Comput. Surv. 55(6)
20. Leviathan Y, Kalman M, Matias Y (2023) Fast inference from transformers via speculative decoding. In International Conference on Machine Learning, pp. 19274–19286. PMLR
21. Xia H, Ge T, Wang P, Chen S-Q, Wei F, Sui Z (2023) Speculative decoding: exploiting speculative execution for accelerating seq2seq generation. Findings Assoc Comput Linguist EMNLP 2023:3909–3925
22. Schmidhuber J (1992) Learning to control fast-weight memories: an alternative to recurrent nets. Neural Comput 4(1):131–139
23. Cai T, Li Y, Geng Z, Peng H, Lee JD, Chen D, Dao T (2024) Medusa: Simple LLM inference acceleration framework with multiple decoding heads. arXiv preprint arXiv:2401.10774
24. Vidal D, Thormann M, Pons M (2005) Lingo, an efficient holographic text based method to calculate biophysical properties and intermolecular similarities. J Chem Inf Modeling 45(2):386–393
25. Klein G, Kim Y, Deng Y, Nguyen V, Senellart J, Rush AM (2018) OpenNMT: neural machine translation toolkit
26. Lowe DM (2012) Extraction of chemical structures and reactions from the literature. Ph.D. dissertation, University of Cambridge, Cambridge, UK. https://doi.org/10.17863/CAM.16293

Publisher's Note