# NLP4Chemistry

## Language Models

Camilo Thorne, Saber Akhondi

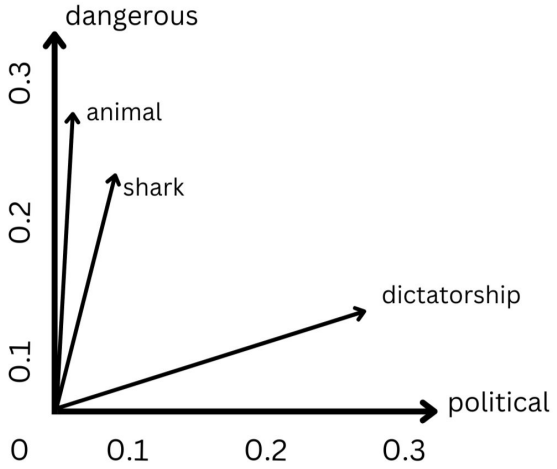
May 2024 - COLING/LREC '24

# Language Models

# Distributional hypothesis [Firth, 1957]

"You shall know a word by the company it keeps"

- The meaning of a word can be approximated by counting co-occurences with other words in sentences (bag of words - BOW - model)
- Co-occurrence counts give rise to **distributional vectors** and **distributional space**
- Meanings that are close = vectors that are close in distributional space

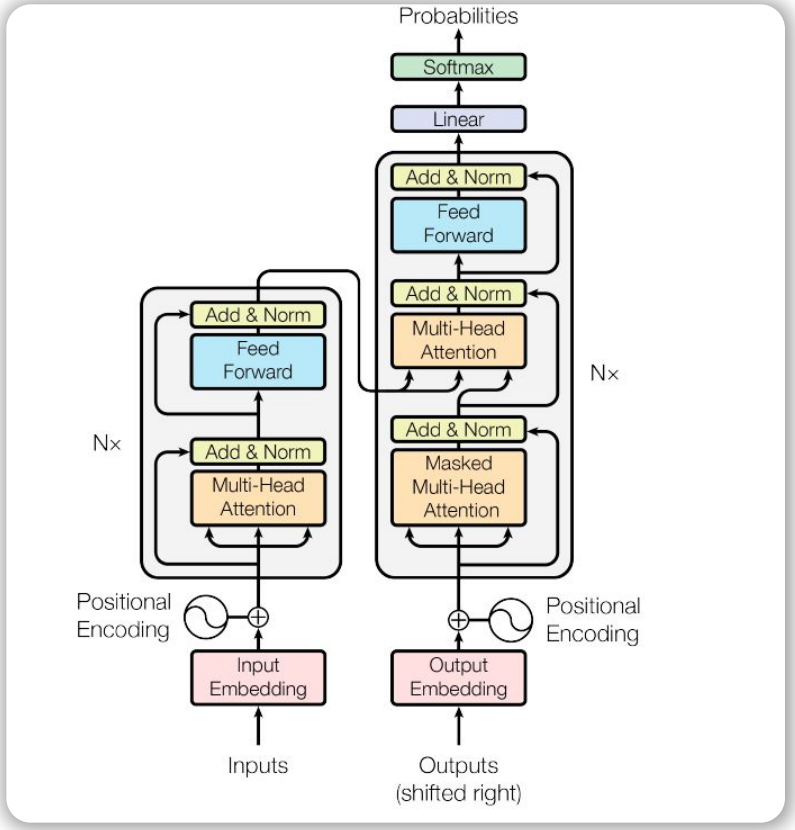



Language models learn distributional spaces via self-supervised learning!

# Neural language models [Vaishwani, 2017]

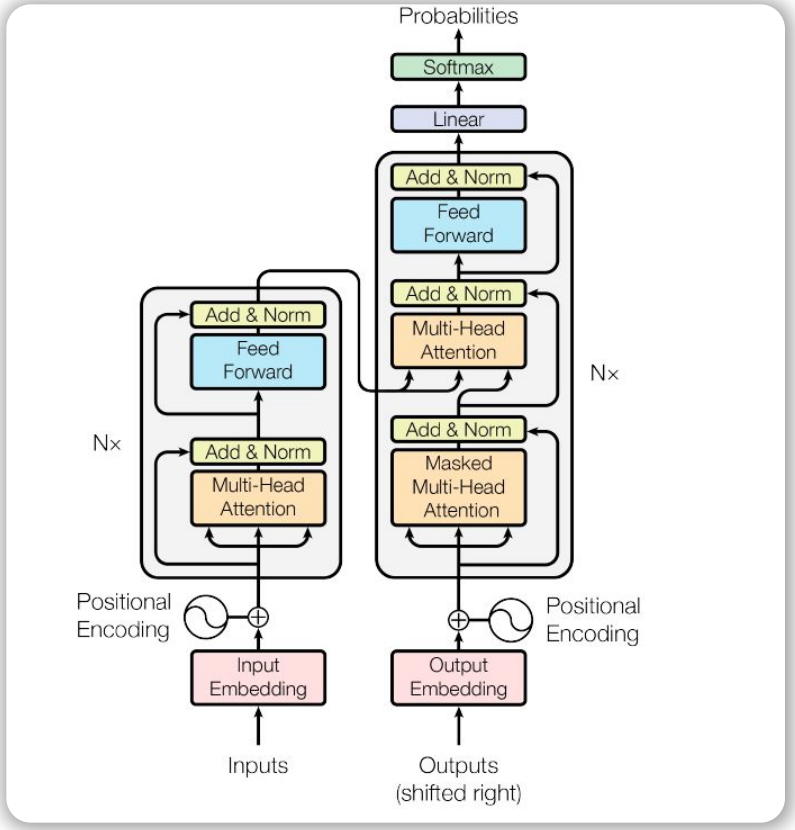
- Current language models based on the **transformer architecture**
- Deep neural networks resemble the way our brain functions
- Neurons (similar to human brain neurons) are known as parameters in language models
- The more neurons (bigger brain) means more learning capacity
- More learning capacity yields in better results

*Build **dense** distributional representations!*



# Neural language models – Seq2Seq (e.g. T5)

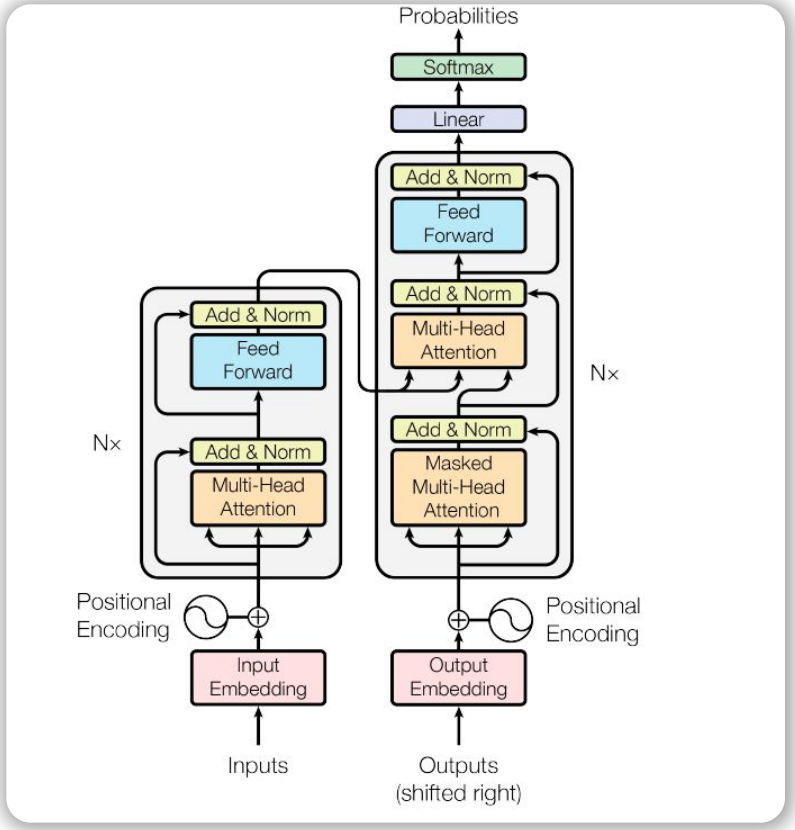
- Current language models based on the transformer architecture
- Transformers rely on the **attention** mechanism
- Neurons are known as **parameters** in language models
- More neurons means more learning capacity
- Pre-trained on very large corpora (billions of tokens)




Encoder-decoder => text2text transduction and translation

# Neural language models – encoder (e.g. BERT)

- Current language models based on the transformer architecture
- Transformers rely on the **attention** mechanism
- Neurons are known as **parameters** in language models
- More neurons means more learning capacity
- Pre-trained on very large corpora (billions of tokens)

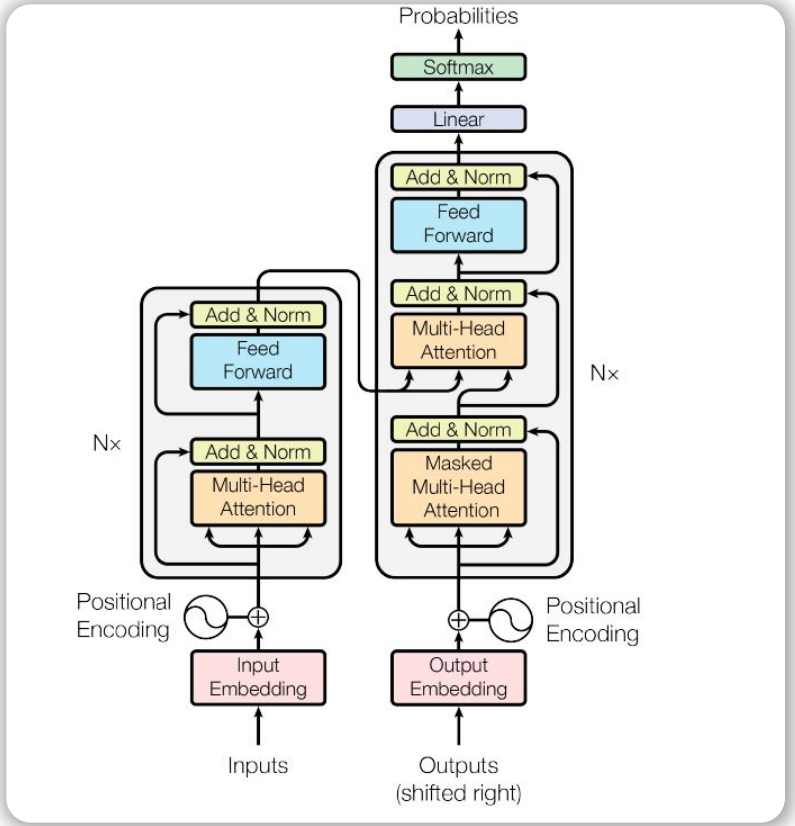



Encoder => embeddings and feature extraction

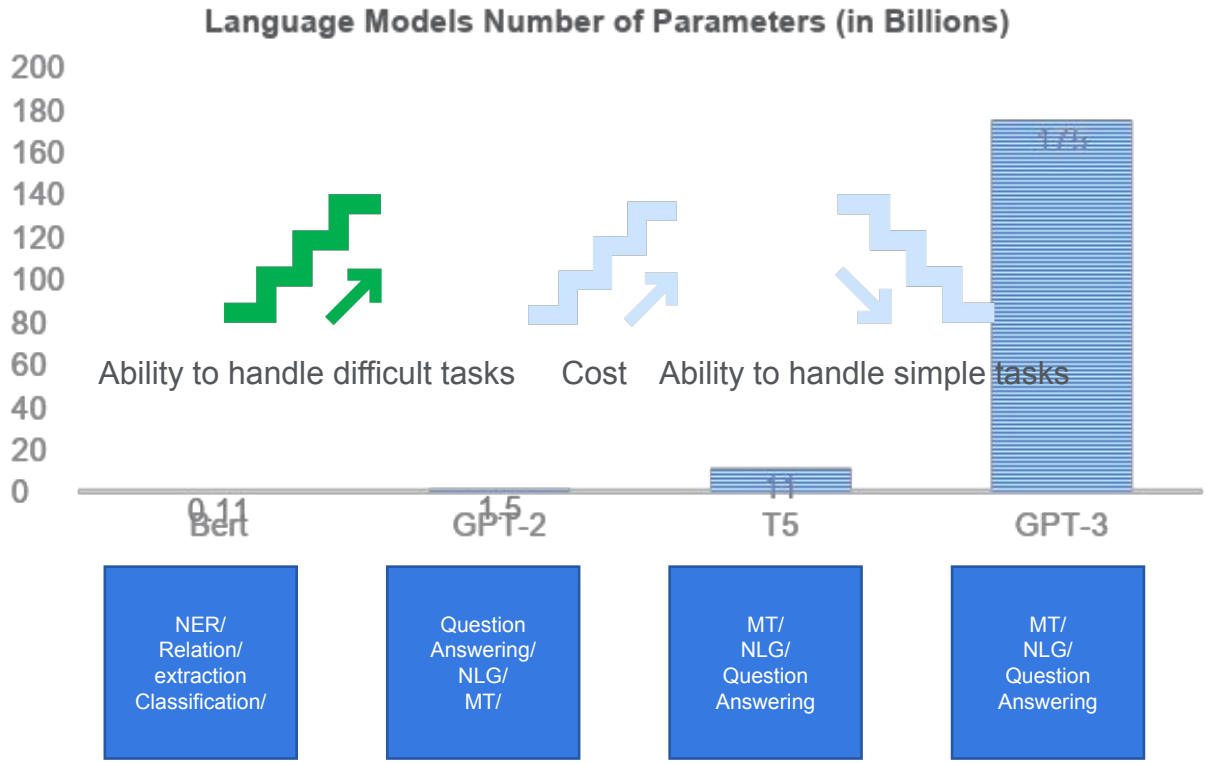
# Neural language models – decoder (e.g. GPTs)

- Current language models based on the transformer architecture
- Transformers rely on the **attention** mechanism
- Neurons are known as **parameters** in language models
- More neurons means more learning capacity
- Pre-trained on very large corpora (billions of tokens)

Decoder => text generation

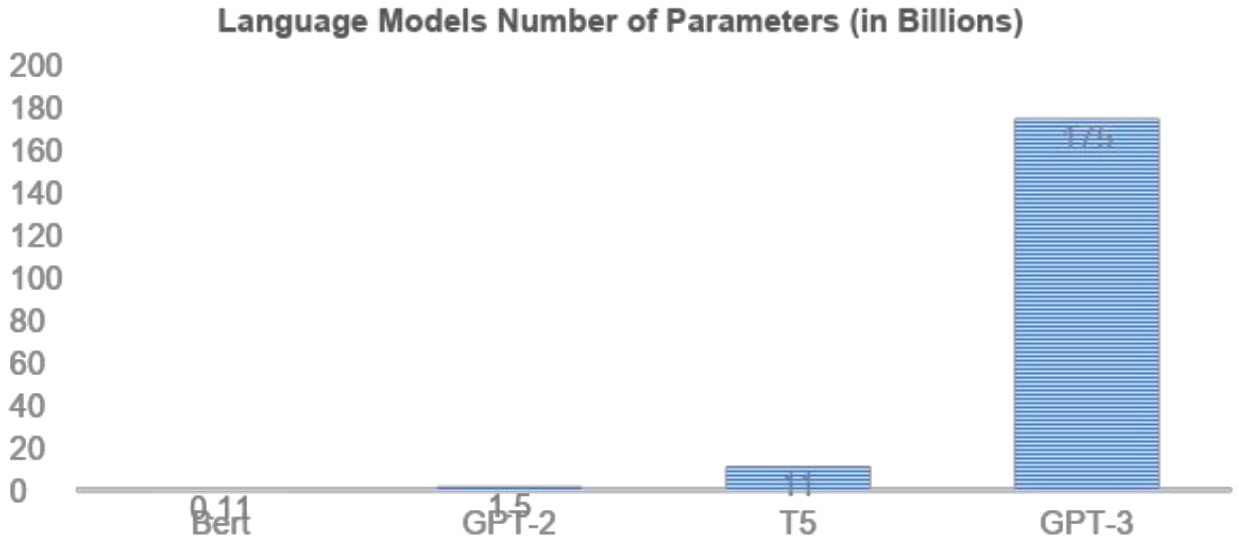


# Language models – parameter distribution



# Language models – parameter distribution

| Use cases that can be addressed | | | | |
|---|---|---|---|---|
| | Entities<br>Relations<br>Classifications | Question Answering<br>Language generation<br>translations | MT/ NLG/ Question Answering | MT/ NLG/ Question Answering |



# SMILES and natural language

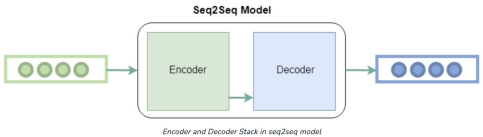
- SMILES strings can be seen as a "sentence" or **stream of tokens**
- Example: **`CC[N+](C)(C)Cc1ccccc1Br`**
  - atom tokenizer: `'C', 'C', '[N+]', '(', 'C', ')', '(', 'C', ')', 'C', 'c', '1', 'c', 'c', 'c', 'c', 'c', '1', 'Br'`
  - character tokenizer: `'C', 'C', '[', 'N', '+', ']', '(', 'C', ')', '(', 'C', ')', 'C', 'c', '1', 'c', 'c', 'c', 'c', 'c', '1', 'B', 'r'`
  - BPE tokenizer: `'CC', '[N+](C)', '(C)C', 'c1ccccc1', 'Br'`
  - 4-mer tokenizer: `'CC[N+]C', 'C[N+]C)', '[N+]C)C', 'C)C)', ')C)C', 'C)Cc', ')Ccc', 'Cccc', 'cccc', 'cccc', 'cccc', 'ccc6', 'cc6Br'`

Can be embedded in a text-based distributional space via a language model!!!

# Language models for chemistry

By modelling the problem using input-output sequence pairs (SMILES, or just sentences) models learn a probability distribution over words (or word sequences) used to predict the most likely next word in a sentence based on previous entry.

Here: we can represent chemical reactions as a language.

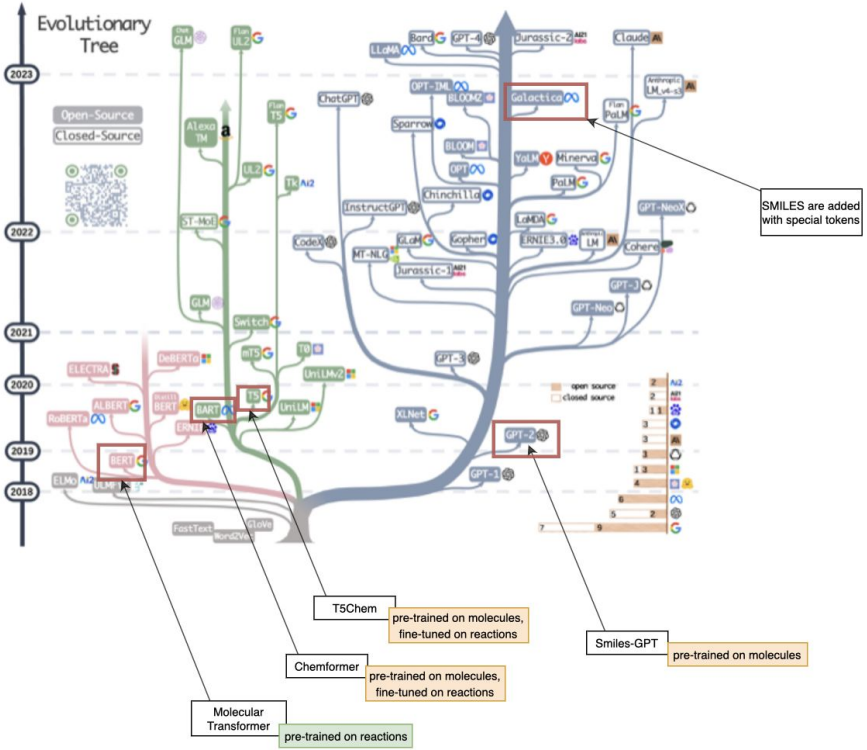



**Chemistry LMs:**

Encoder models: BERT

Encoder-decoder models: BART & T5

Decoder models: GPT-2, Galactica, GPT-3




See: https://github.com/Mooler0410/LLMsPracticalGuide [11]

# Model Inference

# Language modeling and generation

- Language models predict by generating the most likely continuation to an input text / sequence of input tokens
- Generation is normally incremental (left to right)
- Mathematically, language models estimate a joint probability distribution of inputs and outputs:
  - the generated is the output sequence that maximizes the joint probability
  - the joint probability can be decomposed into a product of single-step, token-wise probabilities via the chain law
  - for simplicity, models reason with log-likelihoods (sums)

Note: In principle many supervised learning tasks can be reformulated as generation tasks!

# Chain law

Idea of **LM/Softmax**: dense features $\Rightarrow$ discrete predictions

Probability of next token:

$$p(w_t|w_1,\ldots,w_{t-1};\theta) = \text{softmax}(\mathbf{W}_{out}\mathbf{h}_{t-1} + \mathbf{b}_{out})$$

Probability of full sequence (chain law):

$$p(w_{1:T}) = \prod_t p(w_t|w_1,\ldots,w_{t-1})$$

# Next token prediction

**Seq2Seq notation**: Denote $x$ the source input; $\mathcal{V}$ the vocabulary $w_t$ a random variable for the $t$-th target token with support $\mathcal{V}$; $y_{1:T}$ the ground-truth output; $\hat{y}_{1:T}$ the predicted output; and finally, $p(w_{1:T}|x;\theta) = \prod_t p(w_t|w_{1:t-1}, x;\theta)$ is the model distribution.

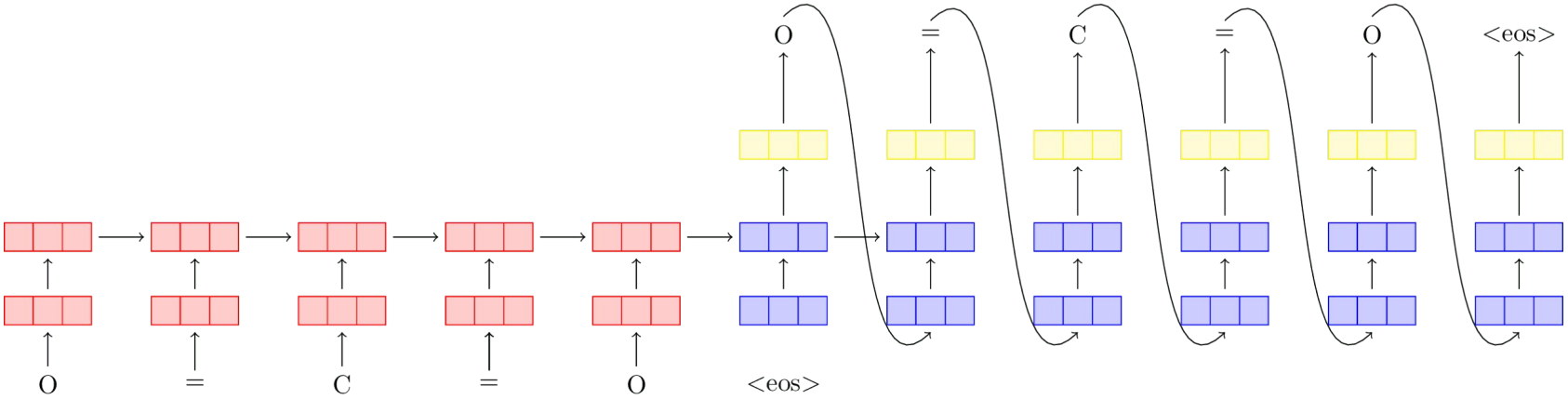
**Train objective**: Given source-target pairs $(x, y_{1:T})$, minimize next token prediction loss (NTPL) of each word independently, conditioned on *gold* history $y_{1:t-1}$

$$\mathcal{L}_{\text{NTPL}}(\theta) = -\sum_t \log p(w_t = y_t|y_{1:t-1}, x;\theta)$$

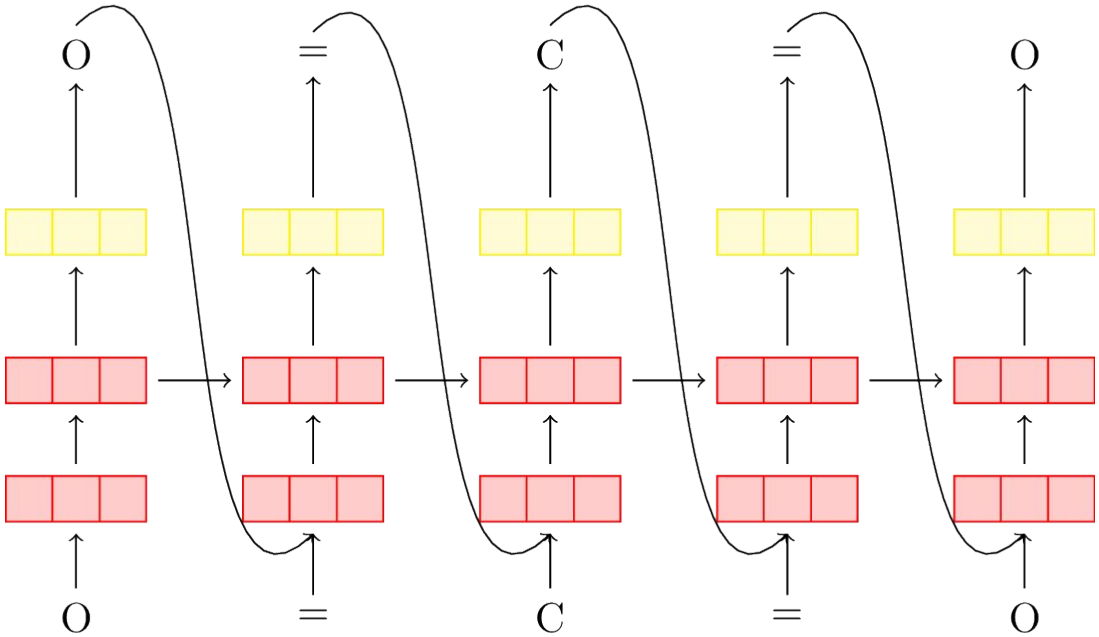
**Test objective**: Structured prediction

$$\hat{y}_{1:T} = \text{argmax}_{w_{1:T}} \sum_t \log p(w_t|w_{1:t-1}, x;\theta)$$

# Text2Text / Seq2Seq generation



# Auto-regressive Text Generation

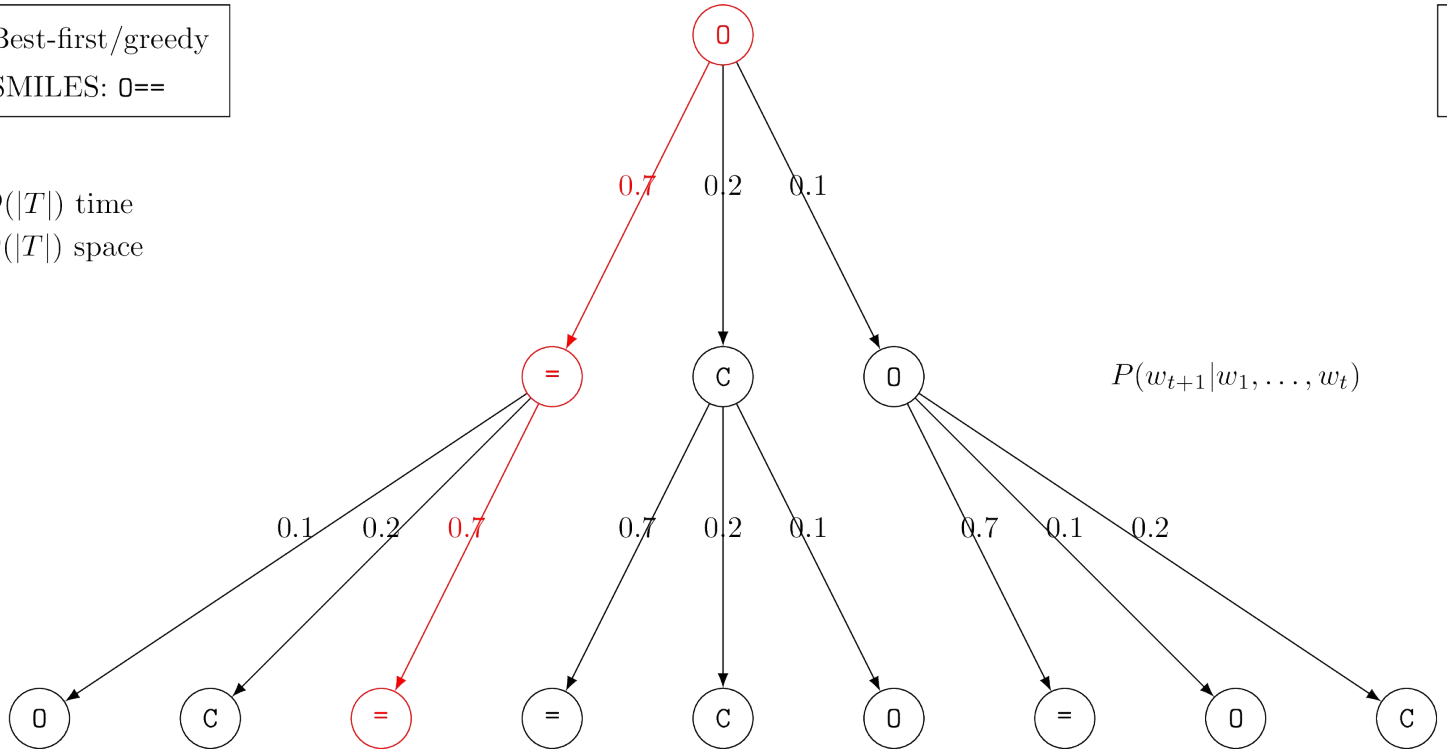


# Greedy search

Best-first/greedy
SMILES: O==

$O(|T|)$ time
$O(|T|)$ space

$T = 3$
$V = \{\texttt{O}, \texttt{=}, \texttt{C}\}$

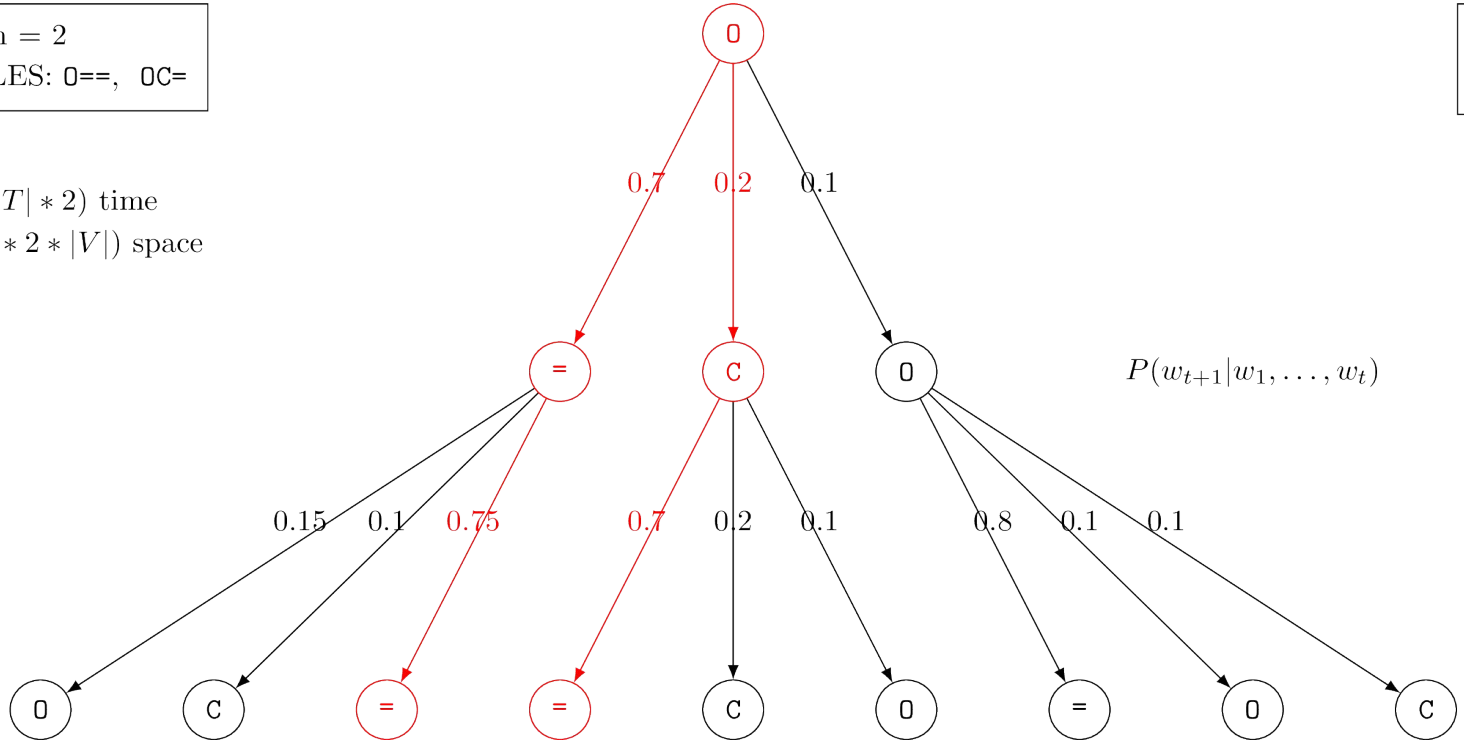


# Beam search

Beam = 2
SMILES: O==, OC=

$O(|T| * 2)$ time
$O(|T| * 2 * |V|)$ space

$T = 3$
$V = \{O, =, C\}$



# Sampling search - idea

In **top-$k$ sampling**, sampling is restricted to the most likely words $w$ that complete $w_1 \cdots w_{t-1}$:

$$w_t \sim P_k(w|w_1 \cdots w_{t-1})$$

where $P_k(w|w_1 \cdots w_{t-1})$ is the restriction of $P(w|w_1 \cdots w_{t-1})$ to its $k$ most likely outcomes.

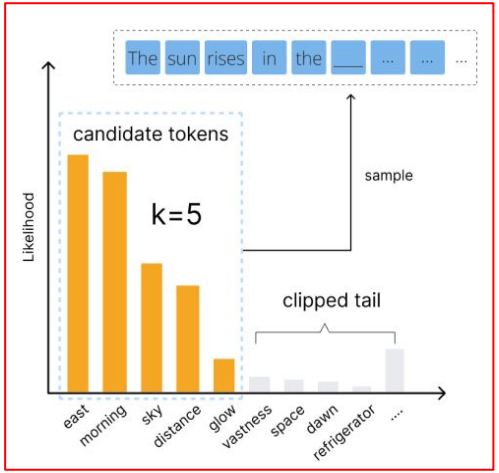
In **temperature sampling** with temperature $t$ on the other hand we sample as follows:

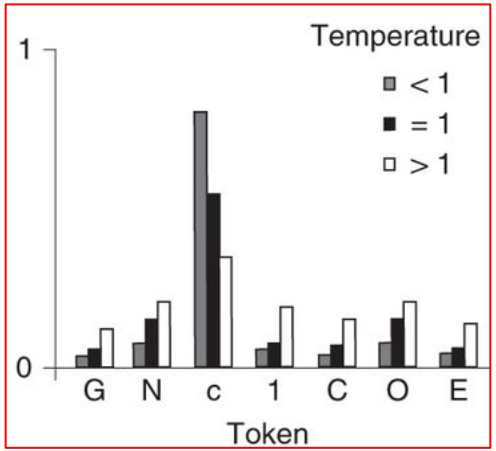
$$w_t \sim P_t(w|w_1 \cdots w_{t-1})$$

where

$$P_t(w|w_1 \cdots w_{t-1}) \approx \frac{\exp(\theta(w_1 \cdots w_{t-1}w)) \cdot t^{-1}}{\sum_{w \in W} \exp(\theta(w_1 \cdots w_{t-1}w)) \cdot t^{-1}}$$

The temperature parameter $t$ has the effect of "flattening" $P(w|w_1 \cdots w_{t-1})$, viz., the higher this number, the more uniform the distribution and thus the more "random" the sampling.



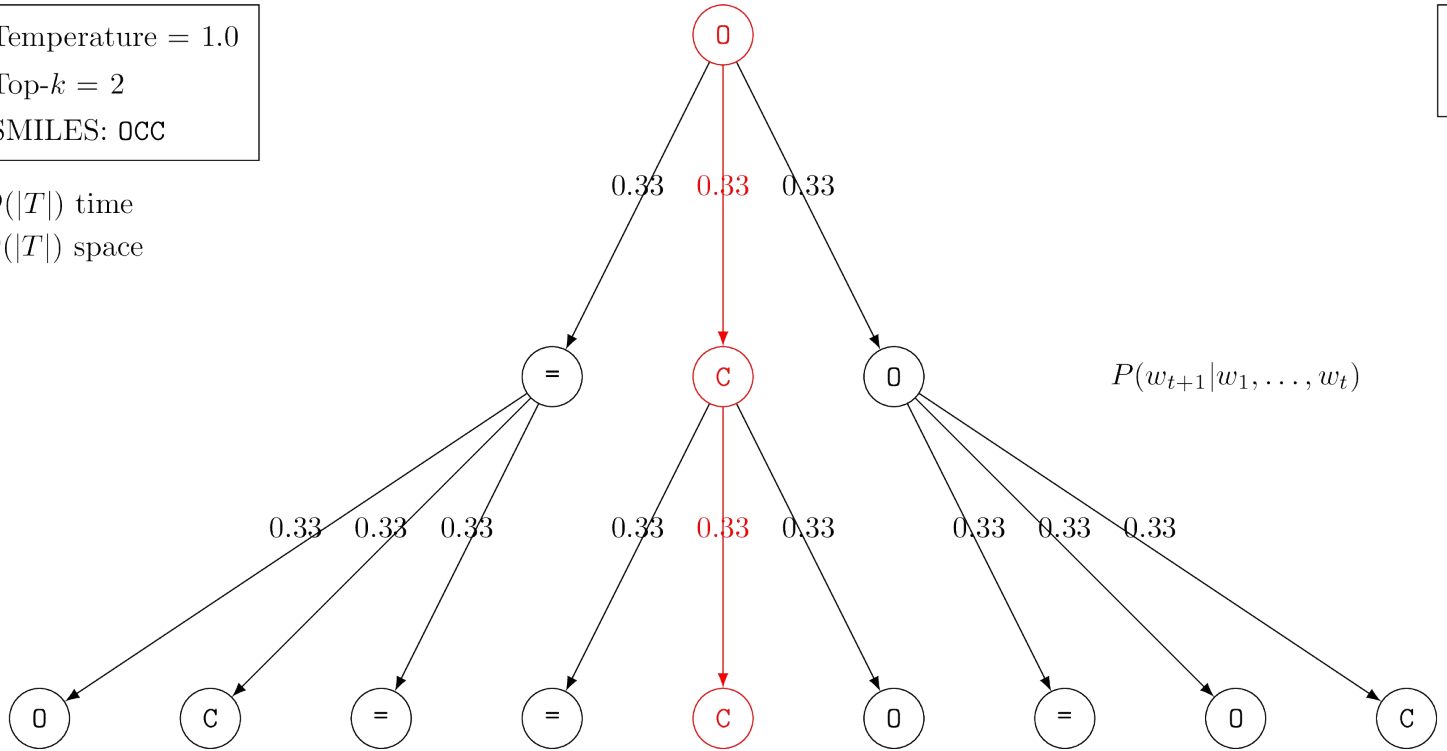



# Sampling search

Temperature = 1.0
Top-$k$ = 2
SMILES: OCC

$O(|T|)$ time
$O(|T|)$ space

$T = 3$
$V = \{\texttt{O}, \texttt{=}, \texttt{C}\}$



# LLM classification

LLMs do not classify in the way we intuitively think! Instead of a simple next-token prediction objective forward pass for the whole model vocabulary $V$, as we do when generating text:

$$o^* = \text{argmax}_{o \in V} P(o|\vec{i});$$

we instead solve:

$$o^* = \text{argmax}_{o \in V | \{o_1, \ldots, o_k\}} P(o|\vec{i}).$$

In practice, we a) estimate the conditional probability $P(o_j|\vec{i})$ of label $o_j$ being a continuation of input $i$, for $1 \leq j \leq k$, and b) return the label with maximum probability.
In other words, we run a forward pass, restrict attention to $\{o_1, \ldots, o_k\} \subseteq V$, and select as prediction the most likely token **in this subset only** (vs. in the full conditional probability distribution that applies to all of $V$).

# Evaluation

# Evaluation metrics

Depending on the task, different metrics are used!

| Task | Definition | Metric |
|---|---|---|
| Classification | Classify | Accuracy |
| Text generation | Generate text | Perplexity |
| Text2Text generation | "Transduce" input sequences into output sequences | BLEU, ROUGE |
| Closed or open book question answering | Generate answer to question or resp. extract answer from input | Token-level F1-score |

# BLEU

Given: $\hat{s} = (\hat{w}_1, \ldots, \hat{w}_m)$ a generated sequence (of tokens/words), $s = (w_1, \ldots, w_m)$ a reference sequence (of tokens/words), $C(t, s)$ is the number of times n-gram $t$ occurs in $s$, $s_{s<i}$ is the set of words/tokens of $s$ up to position $i$ and $G_n[s]$ the set of n-grams of $s$, then:

$$p_n(\hat{s}; s) = \frac{\sum_{i=1}^{m} \sum_{t \in G_n[\hat{s}_{<i}]} \min(C(s, \hat{s}_{<i}), C(s, s_{<i}))}{\sum_{i=1}^{m} \sum_{t \in G_n[\hat{s}_{<i}]} C(s, \hat{s}_{<i})} \qquad \text{(overlap)}$$

$$\mathrm{BP}(\hat{s}; s) = \exp(-\max(0, r/c - 1)) \qquad \text{(brevity penalty)}$$

$$\mathrm{BLEU}_k = \mathrm{BP}(\hat{s}; s) \cdot \exp(\sum_{n=1}^{\infty} k_n \cdot \ln p_n(\hat{s}; s)) \qquad \text{(BLEU)}$$

# Accuracy

Let $Y = \{y_1, \ldots, y_k\}$ be a set (array) of labels, and $\hat{Y} = \{\hat{y}_1, \ldots, \hat{y}_k\}$ the corresponding model predictions. Accuracy is defined as the complement (difference) of prediction error:

$$\text{Acc} = 1 - \text{Err} = 1 - \frac{|\{\hat{y}_i \mid \hat{y}_i \neq y_i\}|}{|Y|}$$

# Perplexity

The best language model $\hat{P}$ is the one that best predicts (generates) an unseen test set $S$, i.e., gives the highest $\hat{P}(w_1 \dots w_n)$ for all $W = w_1 \dots w_n \in S$. Perplexity can be defined as the inverse of the probability of $S$, where the predicted probability of $W$ is normalized by the sequence's length $n$:

$$\begin{aligned} PP(W) &= \hat{P}(w_1 \dots w_n)^{-1/n} \\ &= \sqrt[n]{\frac{1}{\hat{P}(w_1 \dots w_n)}} \\ &= \sqrt[n]{\frac{1}{\prod_{i=1}^{n} \hat{P}(w_i | w_1 \dots w_{i-1})}} \\ &\propto \exp(-\frac{1}{n} \sum_{i=1}^{n} \log \hat{P}(w_i | w_1 \dots w_{i-1}))) \end{aligned}$$

Minimizing $PP(W)$, for all $W \in S$, is equivalent to maximising $\hat{P}$. The best language model is thus the model that **returns the lowest perplexity** on $S$.

# Token-level F1-score

For *token-level* precision, recall and F1-score the idea is to consider the predicted $pred = \{t_1, \ldots, t_n\}$ and reference $gold = \{t'_1, \ldots, t'_m\}$ sequences as *sets* of tokens and then measure IR-style set-based (macro-averaged) metrics as in:

$$
\begin{aligned}
P &= \frac{|pred \cap gold|}{|pred|} \\
R &= \frac{|pred \cap gold|}{|gold|} \\
F1 &= \frac{2PR}{P+R}
\end{aligned}
$$

# ROUGE-N

The ROUGE-n metric measures the n-gram overlap between a reference word/token sequence $S = (w_1, \ldots, w_k)$ and a predicted or generated word/sequence $\hat{S} = (\hat{w}_1, \ldots, \hat{w}_k)$. Let $G_{<n}[S] = G_1[S] \cup \cdots \cup G_n[S]$ (the union of all p-grams with $p \leq n$) and $G_{<n}[\hat{S}] = G_1[\hat{S}] \cup \cdots \cup G_n[\hat{S}]$ (idem), then:

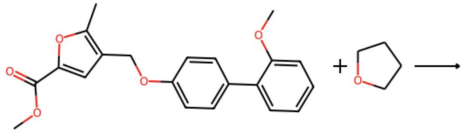
$$\text{ROUGE-n} = \frac{|G_{<n}[S] \cap G_{<n}[\hat{S}]|}{|G_{<n}[S]|}$$

# Molecular Synthesis

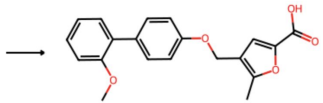
# Problem

- **Single-step RT:**
  - predicts possible reactants given a target molecule
  - we consider only single-outcome reactions, multi-outcome reactions can be decomposed into multiple single-outcome reactions
- **Multi-step RT:**
  - predicts a sequence of reactions, with specified max lengths of the RT pathways, and every reaction is a single step reaction.
  - predicts the reactants until they belong to a set of commercially available reactants.
  - planning is a failure, if we achieved the max length of the RT pathway, and the returned reactants aren't commercially available.

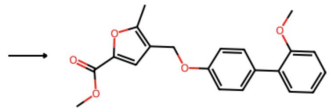
```
rxn1 = Reactions.ReactionFromSmarts("COC(=O)c1cc(COc2ccc(-c3ccccc3OC)cc2)c(C)o1.C1CCOC1>>",
                useSmiles=True)
Draw.ReactionToImage(rxn1)
```



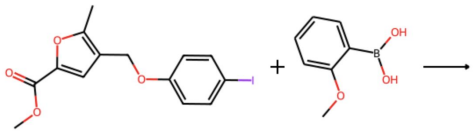
```
rxn2 = Reactions.ReactionFromSmarts(">>" + tokenizer.decode(output[0], skip_special_tokens=True),
                useSmiles=True)
Draw.ReactionToImage(rxn2)
```



```
rxn = Reactions.ReactionFromSmarts(">>COC(=O)c1cc(COc2ccc(-c3ccccc3OC)cc2)c(C)o1",
                useSmiles=True)
Draw.ReactionToImage(rxn)
```



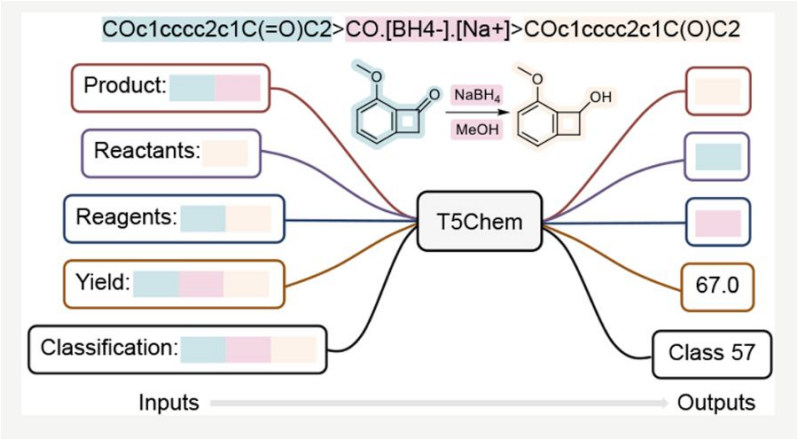
```
rxn = Reactions.ReactionFromSmarts(tokenizer.decode(output[0], skip_special_tokens=True) + ">>",
                useSmiles=True)
Draw.ReactionToImage(rxn)
```



# T5Chem model [Lu, 2022]

T5 – adopted to the chemical reaction prediction tasks:

- Reaction type classification on **USPTO_TPL**
- Forward reaction prediction on **USPTO_MIT**
- Single-step retrosynthesis on **USPTO_50K**
- Reaction yield prediction on C-N coupling reactions
- Reagent suggestion




+ introduction of a new unified multitask reaction prediction data set **USPTO_500_MT**

+ can be used to train and test these 4 different tasks + new task: reagent suggestion.

+ use of SHAP to explain T5Chem predictions at the functional group level.

https://github.com/HelloJocelynLu/t5chem

# T5Chem model [Lu, 2022]

Similar to the original Transformer, (1) instead of using the Layer Norm bias, it places a layer normalization outside the residual path and (2) uses a different, relative position embedding scheme

*The idea of multitasking predictions is inspired by the natural human learning process that knowledge learned from one task should be helpful to other related tasks*

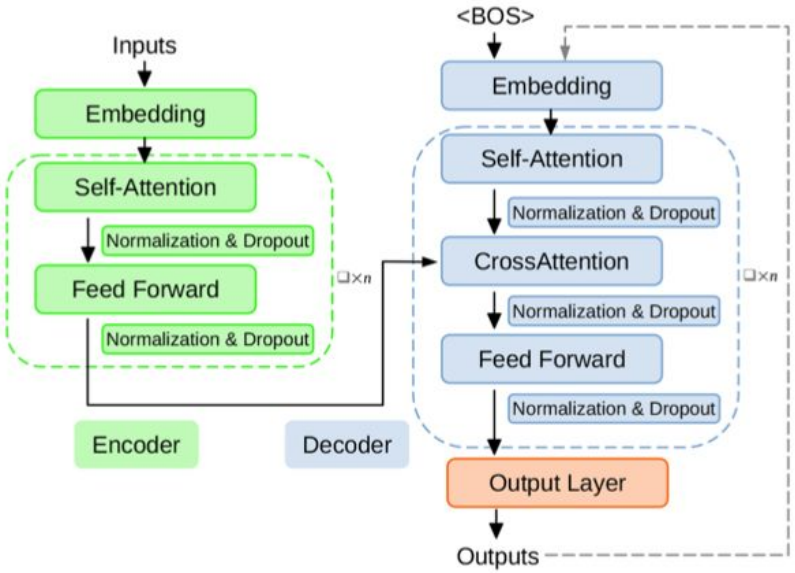



**Figure 1.** Illustration of the general transformer architecture[65] used in T5[69] and T5 Chem.

To specify tass, a task-specific prompt should be added at the beginning of the input sequence before, e.g.:

- **"Product"**: for reaction product prediction
- **"Reactants"**: for single-step RT
- etc.

# T5Chem model [Lu, 2022]

**Modifications of original T5 architecture into:**
(1) the molecular generation head for all sequence-to-sequence tasks (i.e., reactions product prediction, single-step RT, reagent suggestion),
(2) the classification head for reaction type classification task,
(3) the regression head for product yield prediction.

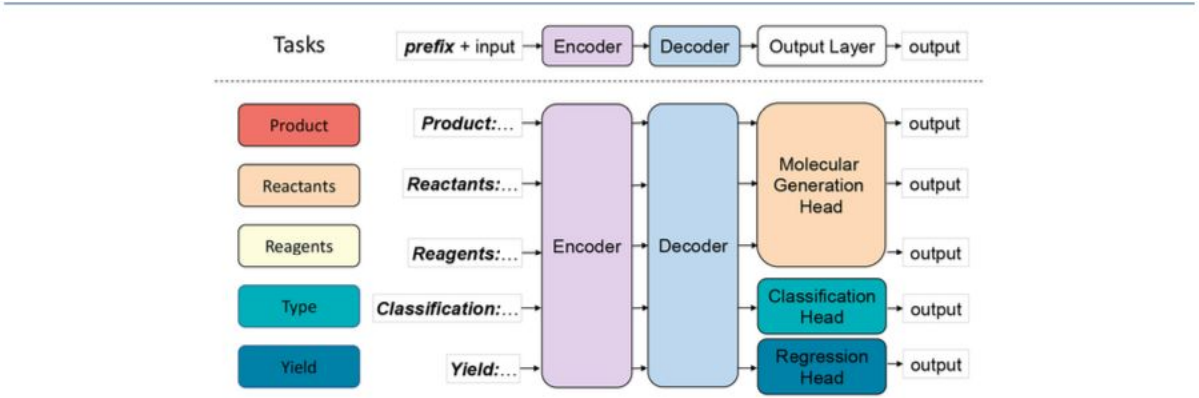



**Figure 2.** Multitasking of T5Chem. Five different tasks are shown. All models have a general structure of encoder, decoder, and output layers. However, they may have different prompts and head types, depending on task types.

**Tokenization:** character-level tokenization, which splits reaction SMILES into a single alphabet letters, digit, or special symbols. [e.g., atom tokenization runs a WordPiece tokenization algorithm over SMILES strings using regular expression].

**Advantages:** simplification, flexibility, and much smaller vocabulary size. Neither atom tokenization or SELFIES outperform character-level tokenization for these 5 tasks.

# T5Chem model [Lu, 2022]

For sequence-to-sequence tasks the input and output sequences share the same vocabulary. **The output layer, molecular generation head, shares weights with the input embedding layer, and produces a probability distribution over the whole vocabulary space.**

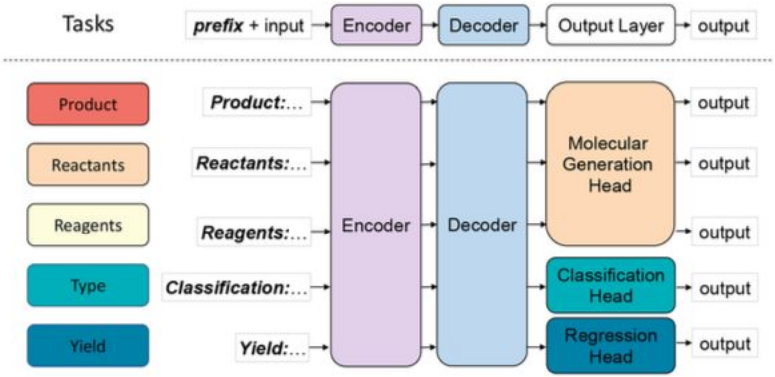



**Figure 2.** Multitasking of T5Chem. Five different tasks are shown. All models have a general structure of encoder, decoder, and output layers. However, they may have different prompts and head types, depending on task types.

**MLM objective** is used for model pre-training in self-supervised manner. 97million molecules from Pubchem were used for pre-training. Mask rate: 15% (randomly masked tokens, and the goal is to predict correct tokens that have been masked).

80% of the time, these were replaced with <mask> token, or 10% of the time replaced by another random token in the vocabulary, and 10% of the time remained the same. After that, fine-tuning for supervised downstream tasks.

# T5Chem model (seq2seq)

Molecular generation head (MGH): produces a probability distribution over the whole vocabulary.

MGH and classification head use cross entropy loss as the loss function.

**MGH repeatedly generated molecules token by token, until the "end of the sentence token" is generated, or the maximum lengths of the allowed prediction is reached.**

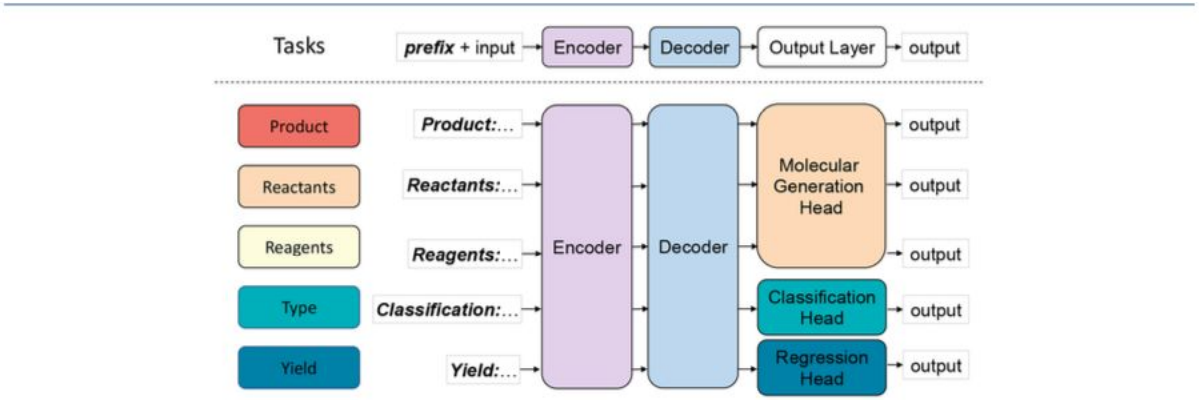



**Figure 2.** Multitasking of T5Chem. Five different tasks are shown. All models have a general structure of encoder, decoder, and output layers. However, they may have different prompts and head types, depending on task types.

**For all tasks, T5Chem uses a whole encode and decoder architecture**, with four layers and eight attention heads. Max vocab = 100, which is 70 tokens from character-level tokenization of SMILES of Pubchem and USPTO molecules, five special structural tokens (like <mask>, <unk>), six prompting tokens, and 19 placeholders. <pad> token is used as the first input token for the decoder to initialize the decoding process. Placeholders are there to use them to add any other prompting tokens if needed.

# T5Chem model – datasets

**Table 1. Data Set Splits Used for the Experiments**

| Data set | Train | Valid | Test | Total | Task |
|---|---|---|---|---|---|
| USPTO_TPL[56a] | 360,545 | 40,059 | 44,511 | 445,115 | Reaction type classification |
| USPTO_MIT[12] | 409,035 | 30,000 | 40,000 | 479,035 | Forward prediction |
| USPTO_50k[29a] | 40,029 | 5004 | 5004 | 50,037 | Retrosynthesis |
| C–N Coupling[44a,b] (Random splits) | 2767 | – | 1188 | 3955 | Reaction yield prediction |
| C–N Coupling[44,a,b] (Out-of-sample test 1) | 3057 | – | 898 | 3955 | Reaction yield prediction |
| C–N Coupling[44,a,b] (Out-of-sample tests 2, 4) | 3055 | – | 900 | 3955 | Reaction yield prediction |
| C–N Coupling[44,a,b] (Out-of-sample test 3) | 3058 | – | 897 | 3955 | Reaction yield prediction |
| USPTO_500_MT[a] | 116,360 | 12,937 | 14,238 | 143,535 | Multitask prediction |

[a]Contains stereochemical information. [b]With reactants/reagents separation.

**USPTO_50K for single-step RT:** For a given product as an input, it finds reactant combinations that could generate the input compound. This dataset is a filtered version of Lowe's patent dataset. Contains 50k reactions that have been classified into 10 broad reaction types. Here stereochemical information wasn't removed, and model wasn't provided with reaction type.

**USPTO_500_MT for Multitask Reaction Prediction:** they created a new dataset that is applicable for multiple reaction predictions tasks, but IMO it's a bit unclear which data exactly they used, and how it overlaps with USPTO_50k. They combined all the datasets for 5 tasks, in a way to avoid data leakage, so probably took "intersection".

# T5Chem model – results

## Performance on multitask prediction:

Two groups:

1. forward reaction prediction, single-step RT, and reagents prediction, since they share the same architecture
2. reaction classification and reaction yield prediction

Multitask performance with combined loss function on all 5 tasks got worse performance!

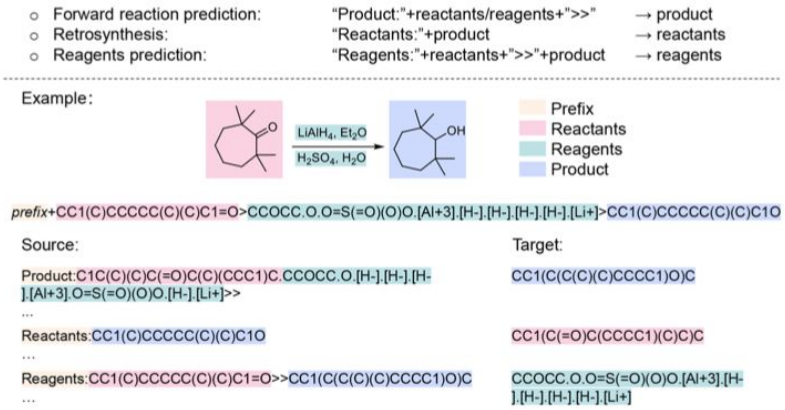



**Figure 3.** Mix training data set from different tasks. In the multitask training scheme, we combined the forward reaction prediction task, retrosynthesis task, and reagents prediction task together. Every input instance starts with a task-specific prompt, followed by actual input. In forward reaction prediction, the model takes reactants and reagents (without separation) as a source sequence. In retrosynthesis, the model takes only product SMILES as a source sequence. In reagents prediction, the model takes both reactants and product SMILES as inputs. A reduction reaction is shown as an example.

Classification and regression are less similar to the other three seq2seq tasks, but for training (embedding, encoder and decoder) shared the same weights among all the tasks!

# T5Chem model – results

**Performance on multitask prediction:**

- Only one model was trained with the mixed dataset, and then tested separately on the three test sets from different tasks.
- Also checking validity of SMILES: SMILES string that can be parsed by the RDKit package.
- Sometimes accuracy and invalidity might be high at the same time

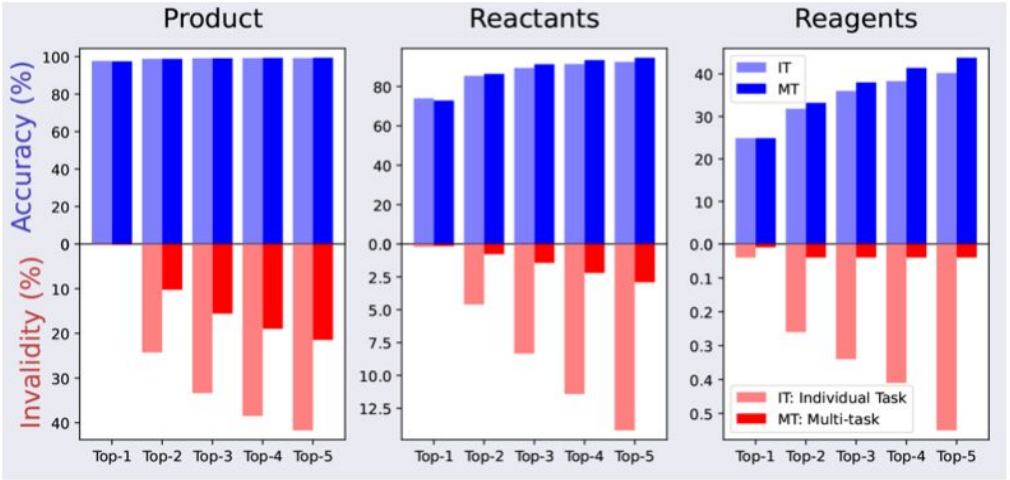
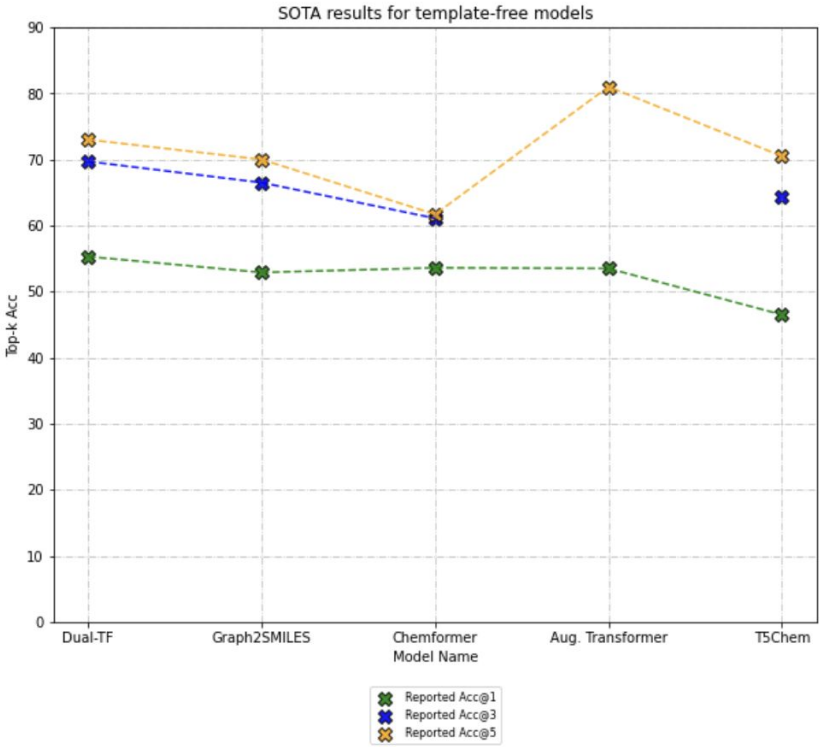



**Figure 4.** Test results on USPTO_500_MT. We evaluated both accuracy and SMILES invalidity for top-k predictions on different tasks. Individual tasks and multitask training schemes have comparable performance in accuracy for all three tasks. But combined training gives much lower invalid SMILES rate. Note that one may have a high top-k accuracy and high SMILES invalidity at the same time.

- ✔ The model achieves comparable performance as being trained on separate tasks in terms of top-k accuracy.
- ✔ The 3 tasks are closely related and are possible to learn at once.
- ✔ T5Chem generates much less grammatically invalid molecules when being trained on mixed training data.
- ✔ Leveraging the knowledge from different yet related tasks is helpful to build a more robust model.

# SOTA models for single-step RT




| Model | Acc@1 | Acc@3 | Acc@5 |
|---|---|---|---|
| Dual-TF [8] | 55.3 | 66.7 | 73.0 |
| Graph2SMILES [9] | 52.9 | 66.5 | 70.0 |
| Chemformer [10] | 53.6 | 61.1 | 61.7 |
| Aug. Transformer [11] | 53.5 | NaN | 81.0 |
| T5Chem [12] | 46.5 | 64.4 | 70.5 |

Table 1. Scores reported by the authors

| Model | Acc@1 | Acc@3 | Acc@5 |
|---|---|---|---|
| Dual-TF | No code available | | |
| Graph2SMILES* Retraining was expensive so we used the provided checkpoints | 86.6 | 90.2 | 90.86 |
| Chemformer (fine-tuned by us) | 50.8 | 57.8 | 58.6 |
| Aug. Transformer | - | - | - |
| T5Chem (fine-tuned by us) | 44.5 | 63.6 | 69.5 |

Table 2. Scores reproduced by us, by fine-tuning the pretrained models

Note: We did not spend a lot of time on hyperparameter optimization, therefore we consider the reported scores as reproducible.

# References

- 2014
  - Neural Machine Translation by Jointly Learning to Align and Translate - https://arxiv.org/abs/1409.0473

- 2017 - Attention Is All You Need - https://arxiv.org/abs/1706.03762

- 2022 - Unified Deep Learning
  Model for Multitask Reaction Predictions with Explanation
  - https://pubs.acs.org/doi/abs/10.1021/acs.jcim.1c01467

- 2023
  - ChemCrow: Augmenting large-language models with chemistry tools - https://arxiv.org/abs/2304.05376

- 2024
  - Leveraging large language models for predictive chemistry - https://www.nature.com/articles/s42256-023-00788-1

**Thank you!**