

STOUT: SMILES to IUPAC names using Neural Machine translation

Kohulan Rajan¹, Achim Zielesny² & Christoph Steinbeck^{1*}

¹Institute for Inorganic and Analytical Chemistry, Friedrich-Schiller-University Jena, Lessingstr. 8, 07743 Jena, Germany

²Institute for Bioinformatics and Chemoinformatics, Westphalian University of Applied Sciences, August-Schmidt-Ring 10, D-45665 Recklinghausen, Germany

*Corresponding author email: christoph.steinbeck@uni-jena.de

Abstract

Chemical compounds can be identified through a graphical depiction, a suitable string representation, or a chemical name. A universally accepted naming scheme for chemistry was established by the International Union of Pure and Applied Chemistry (IUPAC) based on a set of rules. Due to the complexity of this rule set a correct chemical name assignment remains challenging for human beings and there are only a few rule-based cheminformatics toolkits available that support this task in an automated manner.

Here we present **STOUT (SMILES-TO-IUPAC-name translator)**, a deep-learning neural machine translation approach to generate the IUPAC name for a given molecule from its SMILES string as well as the reverse translation, i.e., predicting the SMILES string from the IUPAC name. The open system demonstrates a test accuracy of about 90% correct predictions, also incorrect predictions show a remarkable similarity between true and predicted compounds.

Graphical Abstract



Keywords:

Neural Machine Translation, chemical language, IUPAC names, SMILES, DeepSMILES, SELFIES, deep neural network, attention mechanism, recurrent neural network

Introduction

Assigning names to chemical compounds so that an author can refer to them in the text of a scientific article, book or a patent has a long history. In the early days and even still today, such names were often chosen based on physicochemical or perceptible properties, but also named after species, people, named after fictional characters, related to sex, bodily functions, death and decay, religion or legend, or other [1]. Usually, this makes it impossible to conclude from the name to the chemical structure of the compound. To overcome this dilemma, the International Union of Pure and Applied Chemistry (IUPAC) established a set of rules and guidelines for chemical nomenclature [2–5] where the systematic name can be generated from the structure and substructures of the chemical compound and the structure of the compound can be regenerated based on the name. Often, more than one systematic IUPAC name can be generated for the same compound. In their current draft of the Blue Book, the IUPAC therefore introduced the IUPAC preferred name, preferring one of the possible names over all others.

Other types of string representations of molecules, such as SMILES [6], InChI [7], and SMARTS [8] are more concise forms of line representations. These are designed to be machine-readable, they are in principle also human-readable, and have been incorporated into many major open-source and proprietary cheminformatics toolkits. They are not commonly used in text to denominate chemical compounds for the purpose of recognition by human readers.

IUPAC name generation, due to its algorithmic complexity and the large set of rules, is missing in many cheminformatics toolkits in general. For a human, IUPAC name generation for more than a handful of molecules is cumbersome. People therefore resort to the few available automatic tools for IUPAC name generation.

Among the available and reliable solutions is the “molconvert” software, a command-line program in Marvin Suite 20.15 from ChemAxon (<https://www.chemaxon.com>) [9]. It is available for researchers under an academic license. Open-source programs such as the Chemistry Development Kit (CDK) [10], RDKit [11], or Open Babel [12] do not (yet) provide any algorithms that can automate the process of IUPAC naming for molecules.

With this work we report a proof-of-concept application of Neural Machine Translation (NMT) for the conversion of machine-readable chemical line notations into IUPAC names and back. The large training set was generated with ChemAxon’s molconvert software and we would like to emphasise that this work would not have been possible without the generous offer by ChemAxon for the academic scientific community to use the software for free. We also like to point out that the purpose of this work is not to make ChemAxon’s tool obsolete. As a deterministic tool, it will continue to be the first choice for practical naming tasks in databases.

For the work presented here, we were inspired by Google's multiple NMT models and came up with the idea to build a **SMILES-TO-IUPAC-name** translator called STOUT. STOUT was developed based on language translation and language understanding. We treated the two chemical representations as two different languages - each SMILES string and corresponding IUPAC name was treated as two different sentences that have the same meaning in reality.

All these language models can only achieve greater than 90% accuracy with sufficient data to train them on. The majority of state-of-the-art language translation models are trained on millions of words and sentences to achieve such high levels of accuracy. Moreover, to train such large models in an adequate amount of time dedicated and powerful machine learning hardware is required. In this work, we report substantially shortened training times for our models using Google's Tensor Processing Units (TPU).

Methods

Using deep machine learning methods such as NMT for SMILES-to-IUPAC-name translation is a completely data-driven task so that high quality data from a reliable source is mandatory. In this work, datasets were created for SMILES-to-IUPAC-name translation as well as for IUPAC-name-to-SMILES translation respectively.

Data

All molecules were obtained from Pubchem [13], one of the openly available large, small molecule databases, where the entire PubChem database was downloaded from its FTP site in SDF format. Using the CDK, explicit hydrogens were removed from the molecules and their topological structures were converted to SMILES strings. The obtained 111 million molecules were filtered according to the rule set of our previous DECIMER work [14], i.e., molecules must

- have a molecular weight of fewer than 1500 Daltons,
- not possess any counter ions,
- contain only C, H, O, N, P, S, F, Cl, Br, I, Se and B,
- not contain any hydrogen isotopes(D, T),
- have between 3 and 40 bonds,
- not contain any charged group,
- contain implicit hydrogens only, except in functional groups,

to arrive at a dataset of 81 million molecules. These selected SMILES strings were converted into IUPAC names using Chemaxon's molconvert software, a command-line program in Marvin Suite 20.15 from ChemAxon (<https://www.chemaxon.com>).

Using SMILES strings directly for training Neural Networks (NN) may cause various problems due to their intricate structure which is difficult to split into separate meaningful tokens necessary for the machine input. To tackle this problem, two other representations are available, DeepSMILES

[15] and SELFIES [16]. For a discussion of the problem of string tokenization for deep learning, we refer our readers to those two publications. Our results confirm the superiority of SELFIES for the task discussed here and in our work on Optical Chemical Entity Recognition [14]. Thus, for this work all SMILES strings were converted into SELFIES using a custom python script.

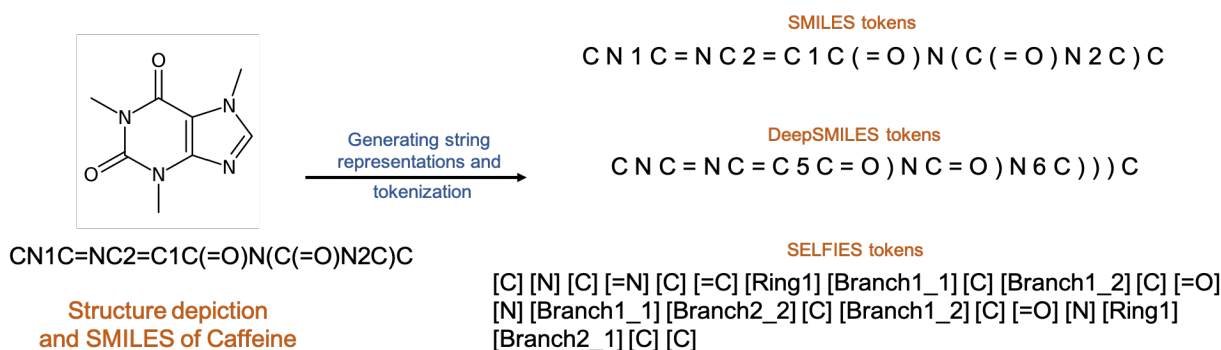


Figure 1: SMILES, DeepSMILES and SELFIES split into tokens which are separated by a space character.

Two datasets were constructed, a 30 million and a 60 million molecules set with SELFIES and corresponding IUPAC names, where the 60 million set contained all 30 million molecule entries of the former. Every SELFIES string and IUPAC name was split into separate tokens using the space character as a delimiter. SELFIES were split according to a closed square bracket "]" and an open square bracket "[". For IUPAC names a small set of rules was applied to split them uniformly: After every,

- open bracket "(", "{" and "[",
- close bracket ")", "}" and "]",
- dash symbol "-",
- full stop ".",
- comma ",",

a space character was added as a delimiter. After adding the delimiter, the dataset was padded to fit to the maximum length, a "start" token was added to each string to indicate the beginning of the string, and an "end" token was added at the end of the string. The strings were tokenized and saved into small TFRecord files for training with GPUs or TPUs. Finally, two SELFIES-to-IUPAC-name datasets and two IUPAC-name-to-SELFIES datasets - with 30 million (exactly 30728192) and 60 million (exactly 61440000) molecules each - were generated.

Network

The NMT network follows the implementation reported by Google for their language translation models, which itself is built on the network designed by Luong et al. [17] for neural machine

translation, using a soft attention mechanism developed by Bahdanau et al. [18]. It is based on an autoencoder-decoder architecture and is written on Python 3 with Tensorflow 2.3.0 [19] at the backend. The encoder network and the decoder network use Recurrent Neural Networks (RNNs) with Gated Recurrent Units (GRU). The input strings are passed to the encoder and the output strings to the decoder. The encoder network generates the encoder output and the encoder hidden state. The attention weight is calculated by the attention mechanism implemented in the network. Encoder output with attention weights then create the context vector. Meanwhile, the decoder output is passed through an embedding layer. The output generated by the embedding layer and the context vector are concatenated and passed on to the GRUs of the decoder. An Adam optimizer with a learning rate of 0.0005 is applied and Sparse Categorical cross-entropy is used to calculate the loss with a modified loss function. A batch size of 256 Strings is used for a GPU and a global batch size of 1024 Strings for a TPU where the global batch size is divided between the nodes.

For SELFIES-to-IUPAC-name and IUPAC-name-to-SELFIES translation the same network architecture is used with the input/output datasets simply being swapped. Figure 2 shows the STOUT architecture for SMILES-to-IUPAC-name translation.

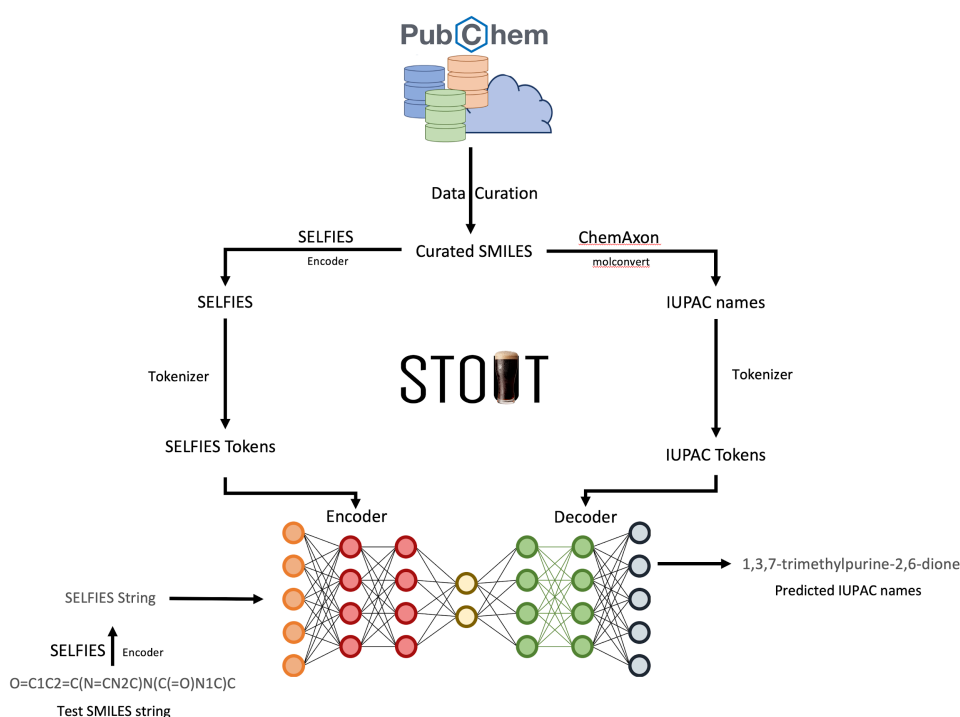


Figure 2: STOUT architecture for SMILES-to-IUPAC-name translation

Model training

For large datasets, training a neural network efficiently is a challenging task. As an initial test, the network was trained with 15 Million molecules on a server with an nVidia Tesla V100 GPU, 384GB of RAM, and two Intel(R) Xeon(R) Gold 6230 processors. The average training epoch was evaluated to be about 27 hours so that training of larger datasets appeared to be prohibitive. With more than 100 epochs of training time used in our training described below, those 27 hours per epoch translate into almost 4 months of training time, with multiples of that for training with 30 Mio or 60 Mio structures. Thus, the training scripts were modified to use Tensor Processing Units (TPUs) available on the Google cloud using the TensorFlow distributed learning API. A corresponding training with TPU V3-8 units (with 8 nodes each) reduced the average training epoch to about 2 hours.

Model test

In order to evaluate the models' performance, independent test sets for the 30 Million and the 60 Mio molecules training sets were constructed with 1.9 million molecules each. A uniform and highly similar frequency distribution of unique SELFIES tokens in training and test data was ensured by corresponding test molecule selection. The SELFIES-to-IUPAC-name translation and the reverse IUPAC-name-to-SELFIES translation were tested with the same set.

To assess the predictive accuracy BLEU scoring [20] was used (see appendix for details). In addition, Tanimoto similarities were calculated between original and predicted strings using PubChem fingerprints. For the predictions of IUPAC names as an output, the IUPAC names were re-converted to canonical SMILES using OPSIN 2.5 [21] with the resulting SMILES being utilized for Tanimoto similarity calculations.

Results and Discussion

Computational considerations

Table 1 summarizes the increase in unique SELFIES/IUPAC-name tokens with the increase of data set size. Note, that the token set sizes differ significantly from dataset to dataset with smaller differences for the SELFIES tokens and considerable ones for the IUPAC name tokens. This also reflects the need for sufficient memory to fit in the big sets of tokens without chopping the dataset.

Table 1: Number of unique SELFIES and IUPAC-name tokens for each dataset.

Dataset Size	Number of SELFIES tokens	Number of IUPAC tokens
15 Million	33	56834

30 Million	35	73864
60 Million	38	93270

As already stated above, training a 15 Million molecule dataset on a TPU V3-8 requires 2 hours per epoch which is 13 times faster than using a GPU V100. Using a TPU V3-32 allows for an additional 4 times faster performance in comparison to a TPU V3-8 and is 54 times faster compared to a GPU V100, see figure 3.

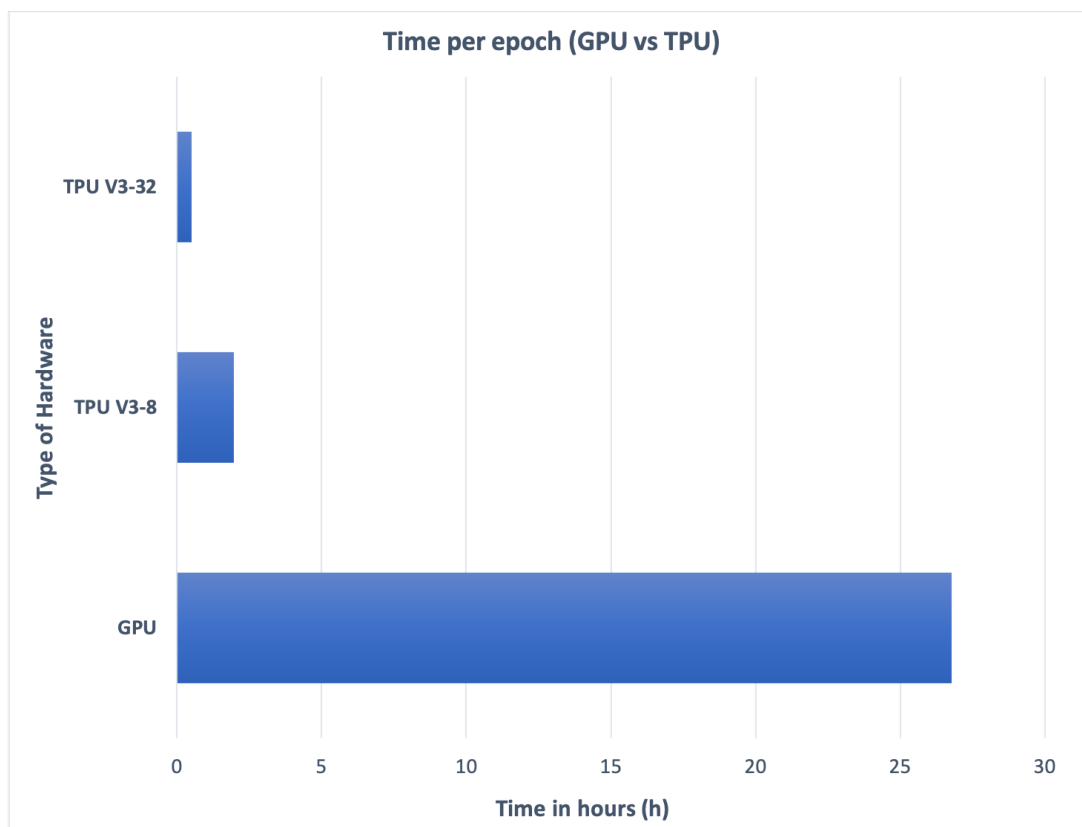


Figure 3: Average training time per epoch on different hardware (lower is better).

Figure 4 shows the different training times per epoch of the different datasets on TPU V3-8 units where all models were trained for more than 100 epochs until convergence. The difference between the SELFIES-to-IUPAC-name and IUPAC-name-to-SELFIES training is caused by the different number of I/O tokens of each dataset: For the SELFIES-to-IUPAC-name translation, the output tokens are derived from the IUPAC names whereas for the IUPAC-name-to-SELFIES translation the output tokens are taken from SELFIES strings. Since SELFIES strings are smaller and less complex than IUPAC name strings the IUPAC-name-to-SELFIES translation is considerably faster.

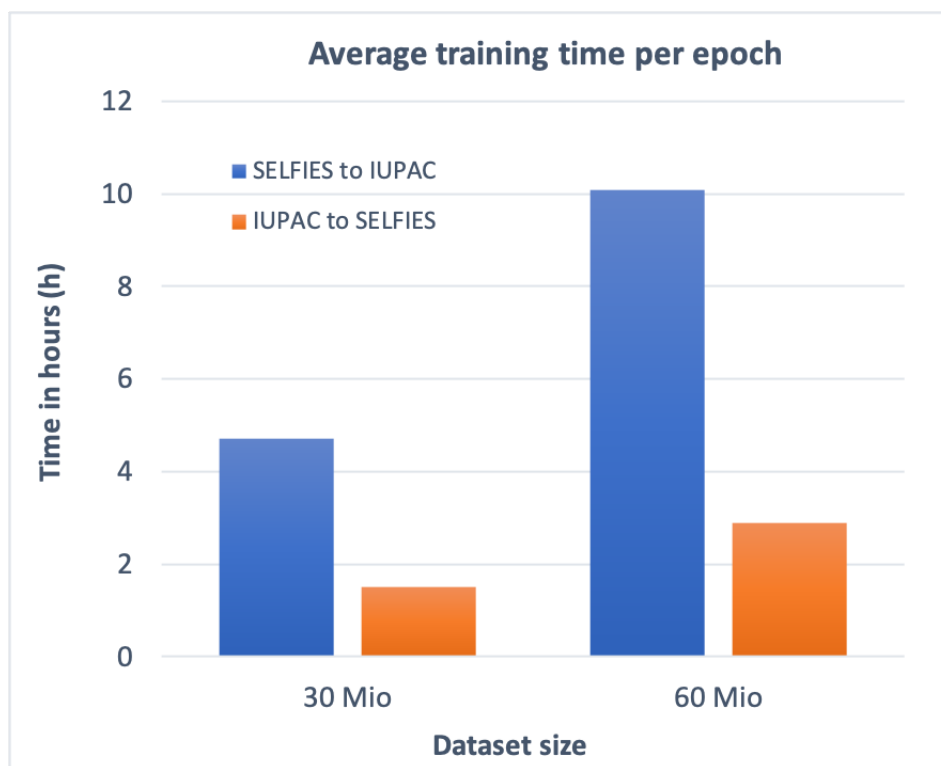


Figure 4: Average training time per epoch for different datasets using TPU V3-8.

Test results

SELFIES-to-IUPAC-name translation

Table 2 summarizes the overall accuracy and the BLEU scores for the 30 million and the 60 million molecules dataset. A predicted string with a BLEU score of 1.0 means 100% accuracy.

Table 2: Overall accuracy and BLEU scores.

Training dataset size	30 Mio	60 Mio
Overall accuracy	95.35%	89.95%
Total number of strings with BLEU 1.0	84.07%	66.95%

BLEU-1	0.98	0.95
BLEU-2	0.97	0.93
BLEU-3	0.96	0.92
BLEU-4	0.95	0.90

The 30 million molecules dataset achieves an accuracy of over 95%, the 60 million molecules dataset slightly less than 90%. Although the 60 million molecules data set has more molecules to train on - which should lead to an increased accuracy - it has to be taken into account that it also contains a considerably increased number of input and output tokens (compare above). The input and output space volumes to be learned are considerably increased in comparison to the 30 million molecules data set. Thus, the larger the number of tokens, the more difficult it becomes to learn the IUPAC names. This demonstrates the need for more training data within the same set of tokens, i.e. same input/output space volumes, to not compare “apples and oranges”.

In order to assess the network’s ability to learn “chemistry” we calculated the Tanimoto similarities between the predicted and the original molecules by translating the original and the predicted IUPAC names back to SMILES strings using OPSIN. The IUPAC names that OPSIN was able to translate back to SMILES strings were counted as valid IUPAC names while the others were counted as invalid. Only the valid IUPAC-name-to-SMILES translation was used for the Tanimoto similarity calculation. The Average Tanimoto was measured on the Valid IUPAC-name-to-SMILES. Additionally, both Tanimoto calculations were readjusted to the number of data points present on the test dataset (see table 3).

Table 3: Tanimoto similarities.

Training dataset size	30 Mio	60 Mio
Invalid IUPAC names	5.18%	15.78%
Valid IUPAC names	94.82%	84.22%
100 % accurate IUPAC names	84.07%	66.95%
Tanimoto 1.0 count on the total test dataset	85.25%	69.56%
Tanimoto 1.0 count on valid IUPAC names	89.91%	82.59%
Average Tanimoto (measured for total test dataset)	0.93	0.82

Average Tanimoto (measured for valid IUPAC names)	0.98	0.97
---	------	------

The invalid IUPAC names are the ones that were rejected by OPSIN and could not be converted into SMILES. This inability is the result of errors with the IUPAC names being predicted. In most cases, the IUPAC name predictions failed because

- they did not contain a comma,
- some of them were missing a close bracket symbol corresponding to the open bracket symbol,
- the valence of an atom was wrong,
- a certain block of text was uninterpretable,
- they failed to assign all bonds correctly,
- of a disagreement between lengths of bridges and alkyl chain length,
- the IUPAC names were too long.

Table 4 presents a few examples of IUPAC names that could not be converted to SMILES strings with an explanation of the failure.

Table 4: Failed IUPAC-name-to-SMILES translations.

IUPAC names	Reason for failure (OPSIN error messages)
1. N-[6-(2,3-diaminopropylidene)-1-methyl-1,2,4a,5,6,8a-hexahydroquinolin-6-yl]-N-methylpropanamide	Atoms are in an unphysical valency state. Element: C valency: 5
2. 2-[[[[(3-ethoxypropyl)amino]({[2-(2-fluorophenyl)ethyl]amino})methylidene]amino)-N,N-dimethylacetamide	Unmatched opening bracket found
3. 3'-(propan-2-yl)-2',3',4',5',6',7',8',8'a-octahydro-2'H-spiro[imidazole-4,1'-indolizin]-2-amine	The following being uninterpretable: 2',3',4',5',6',7',8',8'
4. ({2',6'-difluoro-2',6'-dimethyl-[1,1'-biphenyl]-4-yl)methyl}(propyl)amine	Failed to assign all double bonds
5. 1,4,5-trimethyl-1-[1,2-dimethylpropyl]-2-methyl-1-propylbicyclo[12.2.1]tetradeca-1,5-diene	Disagreement between lengths of bridges and alkyl chain length

The Tanimoto similarity index 1.0 count with 85% (30 million molecules set) of the test data is already remarkable but the astonishing average Tanimoto similarity over 0.9 (30 million molecules set) suggests that an “understanding” of the “language of chemistry” emerged. In addition, it becomes obvious that the number of predictions with a Tanimoto similarity of 1.0 is greater than the number of predictions with a BLEU score of 1.0, see table 5: Although there are different IUPAC names, using OPSIN to re-translate these names led to SMILES representations with similar or even identical chemical graphs, see figure 5. This also illustrates the extent to which the model is capable to successfully generalise the information of the training data.

Table 5: Predicted IUPAC name strings with a Tanimoto similarity index of 1.0 but a low BLEU score.

No	IUPAC names		BLEU Score	IUPAC names translated into SMILES using OPSIN		Tanimoto similarity Index
	Original	Predicted		Original	Predicted	
1.	N,N-dimethyl-2-[(11-oxa-2-thia-5,7-diazadodecan-6-ylidene)amino]acetamide	2-{{{(3-methoxypropyl)amino}{{[2-(methylsulfany)ethyl]amino}}methylidene}amino)-N,N-dimethylacetamide	0.13	<chem>CN(C(CN=C(NCCSC)NCCCOC)=O)C</chem>	<chem>COCCCN(CN(CCCSC)=NCC(=O)N(C)C</chem>	1.0
2.	(1Z,3Z,5Z,7Z,9Z)-cyclohexacos-1,3,5,7,9-pentaene	cyclotetracos-1,3,5,7-tetraene	0.20	<chem>C1=CC=CC=CC=CC=CC=CC=CC=CC=CC=CC=CC1</chem>	<chem>C1=CC=CC=CC=CC=CC=CC=CC=CC=CC=CC=CC1</chem>	1.0
3.	4-[1-(4-methoxypyrimidin-2-yl)piperidine-4-carbonyl]thiomorpholine	4-methoxy-2-[4-(thiomorpholin-4-carbonyl)piperidin-1-	0.23	<chem>COC1=NC(=NC=C1)N2CCC(CC2)C(=O)N3CCSCC3</chem>	<chem>COC1=NC(=NC=C1)N2CC(CC2)C(=O)N3CCSCC3</chem>	1.0

9.	N-butyl-N-[(3-chloro-6-hydrazinylpyridin-2-yl)methyl]aniline	N-[(3-chloro-6-hydrazinylpyridin-2-yl)methyl]-N-butylaniline	0.83	<chem>C(CCC)N(C1=CC=CC=C1CC2=N(C(=CC=C2Cl))NN)C(C1)=CC=CC=C1</chem>	1.0
10.	[2-hydroxy-3-(propan-2-yloxy)propyl][(3-methoxypropyl][(5-methylfuran-2-yl)methyl]amine	[2-hydroxy-3-(propan-2-yloxy)propyl][(5-methylfuran-2-yl)methyl]amine	0.90	<chem>OC(CN(CCC=1OC(=CC(OC)CC=1OC(1)C)CCCO)C)COC(C)C(C)C</chem>	1.0

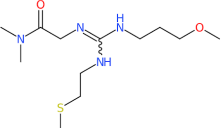
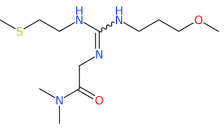
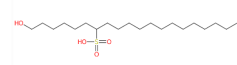
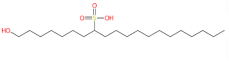
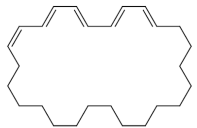
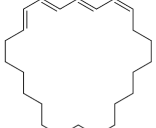
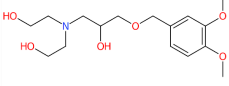
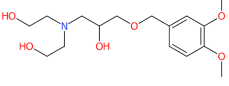
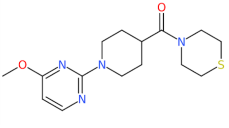
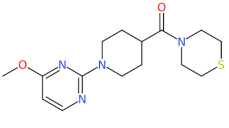
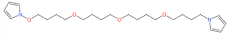
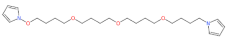
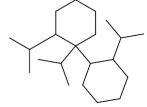
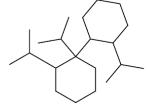
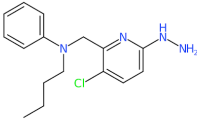
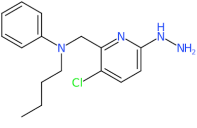
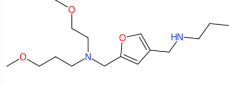
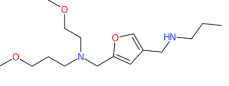
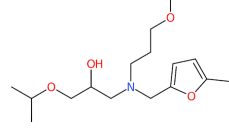
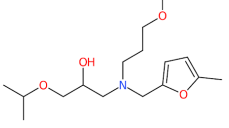
No	Original Molecule	Predicted Molecule	No	Original Molecule	Predicted Molecule
1.			6.		
2.			7.		
3.			8.		
4.			9.		
5.			10.		

Figure 5: Chemical structures depicted with the CDK depiction generator for predictions with Tanimoto similarity 1.0 but low BLEU score.

IUPAC-name-to-SELFIES translation

The IUPAC-name-to-SELFIES translation was tested with the same 1.9 million test molecules as the SELFIES-to-IUPAC-name model before, but in reverse order. As a measure for current top performance, OPSIN is able to convert 98.17% of IUPAC names generated by the molconvert algorithm back to SMILES. Table 6 summarizes the overall accuracy, the calculated BLEU scores, and the Tanimoto similarities that were carried out on the test molecules for IUPAC-name-to-SELFIES translation.

Table 6: Overall accuracy, BLEU Scores, and Tanimoto similarity calculations.

	30 Mio	60 Mio
Overall accuracy	94.72%	92.14%
Total number of predicted strings with BLEU 1.0	80.76%	68.98%
BLEU-1	0.98	0.97
BLEU-2	0.97	0.95
BLEU-3	0.96	0.94
BLEU-4	0.95	0.92
Tanimoto Calculations		
Average Tanimoto similarity index	0.96	0.94
Number of predicted strings with Tanimoto 1.0	82.54%	73.10%

The 30 million molecules result is near the OPSIN top performance. The difference between the 30 and 60 million molecules data sets is considerably reduced in comparison with SELFIES-to-IUPAC-name translation since the SELFIES token output space volume difference is smaller than the IUPAC-name token output space difference before. Again, the predictions with Tanimoto

similarity index 1.0 exceed those with BLEU scores 1.0. The reason for this is that BLEU is calculated by mapping word to word for an original and predicted SELFIES string while Tanimoto similarity is calculated according to the corresponding chemical structure, see table 7 and figure 6. To improve these results more molecules with the same set of unique tokens would be needed.

Table 7: Predicted SELFIES with low BLEU scores and Tanimoto similarity 1.0.

No.	SELFIES		BLEU Score	SELFIES decoded back into SMILES		Tanimoto similarity Index
	Original	Predicted		Original	Predicted	
1.	[O]=[C][N][C][N]=[C][N]=[C][C][Expl=Ring1][Branch1_2][N]=[C][Ring1][Branch2_3][C][Branch1_2][C][=O][N][C]	[O]=[C][Branch1_1][Ring1][N][C][C]=[N][C][C][=N][C][=N][C][Expl=Ring1][Branch1_2][N][C][Ring1][Branch2_3][=O]	0.19	O=C1NC=2N=CN=C C2N=C1C(=O)NC	O=C(NC)C1=NC=2C=NC=NC2NC1=O	1.0
2.	[O]=[C][C]=[C][Branch1_1][Branch1_3][C][Branch1_2][C][=O][O][C][C][Branch1_1][C][O]=[C][N][Ring1][O]	[O]=[C][Branch1_1][Ring1][O][C][C]=[C][C][Branch1_2][C][=O][N][C][=C][Ring1][Branch1_3][O]	0.26	O=C1C=C(C(=O)OC)C(O)=CN1	O=C(OC)C1=CC(=O)NC=C1O	1.0
3.	[O]=[N][C][C]=[C][Branch1_1][Ring1][C]=[O][C]=[C][Branch1_1][Ring2][C][Expl=Ring1][Branch2_1][C][O]	[O]=[N][C][=C][C][Branch1_1][Ring1][C]=[O][C][C][Branch1_2][Ring2][=C][Ring1][Branch2_1][C][O]	0.28	O=NC=1C=C(C(=O)C=C(C1)CO	O=NC1=CC(C=O)=CC(=C1)CO	1.0
4.	[O]=[C][C][Branch1_1][C][F]=[C][Branch1_1][O][N][C]=[C][Ring1][Br	[O]=[C][C][Branch1_2][O]=[C]	0.32	O=C1C(F)=C(NC=C1	O=C1C(=CN(C(=C1F)C)C(1.0

	anch1_3][C][Branch1_1][C][F][F][C]][N][C][Branch1_2][Branch1_1][=C][Ring1][Branch1_2][F][C][C][Branch1_1][C][F][F]		C(F)F)C	F)F	
5.	[Cl][C][C][Branch1_1][C][N][=C][N][=C][S][N][=C][Branch1_1][Branch2_3][C][Ring1][Branch1_1][Expl=Ring1][Branch2_3][C]	[Cl][C][=C][Branch1_1][C][N][C][=N][C][S][N][=C][Branch1_1][Branch1_2][C][Ring1][Branch2_3][Expl=Ring1][Branch1_1][C]	0.43	CIC=1C(N)=CN=C2SN=C(C21)C	CIC1=C(N)C=NC=2SN=C(C12)C	1.0
6.	[F][C][Branch1_1][C][F][C][=N][C][Branch1_2][O][=C][Branch1_1][Branch1_2][C][=C][Ring1][Branch1_2][I][C][Br][C][O]	[F][C][Branch1_1][C][F][C][N][=C][Branch1_1][O][C][Branch1_2][Branch1_2][=C][C][Expl=Ring1][Branch1_2][I][C][Br][C][O]	0.51	FC(F)C1=NC(=C(C=C1)CBr)CO	FC(F)C=1N=C(C(=CC1)CBr)CO	1.0
7.	[N][#C][C][C][=C][C][Branch1_2][C][=O][C][C][C][Ring1][Branch1_2][=C][C][Expl=Ring1][Branch2_3][I]	[N][#C][C][C][=C][C][Branch1_2][C][=O][C][C][Ring1][Branch1_2][I][C][Branch1_2][C][=O][C][C][Ring1][Branch2_2]	0.64	N#CC=1C=C2C(=O)CCC2=CC1I	N#CC=1C=C2C(=CC1)C(=O)CC2	1.0
8.	[O][=C][C][C][C][Branch2_1][Ring1][Ring1][C][N][=C][Branch1_1][Branch1_2][C][=C][C][Expl=Ring1][[O][=C][C][C][C][Branch2_1][Ring1][Ring1][C]	0.70	O=C1CCC(C=2N=C(C(=CC2)C(F)	O=C1CCC(C=2NC(=CC=C2)C(F)(F)F)C	1.0

	Branch1_2][C][Branch1_1][C][F][Branch1_1][C][F][F][C][Ring1][#C]	[N][C][Branch 1_2][Branch1_ 2][=C][C][=C][R ing1][Branch1 _2][C][Branch1 _1][C][F][Branc h1_1][C][F][F][C][Ring1][#C]		(F)F)C1	1	
9.	[N][#C][C][=C][Branch1_1][N][C][=C][C][=C][N][=C][N][Ring1][Branc h2_2][Ring1][Branch1_1][C]	[N][#C][C][=C][Branch1_1][N] [C][=C][C][=C][N][=C][N][Ring 1][Branch1_1][Ring1][Branch 2_2][C]	0.83	N#CC1=C(C=CC2=C N=CN12)C	N#CC1=C(C= CC2=CN=CN 21)C	1.0
10.	[O][B][Branch1_1][S][O][C][Branc h1_1][C][C][Branch1_1][C][C][C][Ring1][Branch1_3][Branch1_1][C] [C][C][C][=C][C][=C][Branch1_1][B ranch2_1][C][=C][Ring1][Branch1 _2][C][C][C][C][C][C]	[O][B][Branch1 _1][S][O][C][Br anch1_1][C][C] [Branch1_1][C] [C][C][Ring1][B ranch1_3][Bra nch1_1][C][C][C][C][=C][C][=C][Branch1_1][B ranch2_3][C][= C][Ring1][Bran ch1_2][C][C][C][C][C][C]	0.92	O1B(OC(C)C)C1(C)C)C2=CC=C (C=C2CCC)CCC	O1B(OC(C)(C)C1(C)C)C2 =CC=C(C=C2 CCCCC)C	1.0

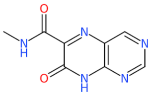
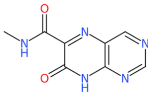
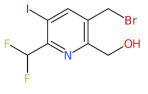
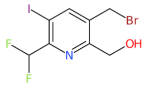
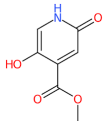
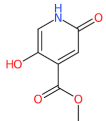
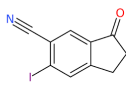
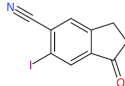
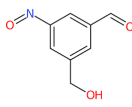
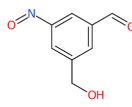
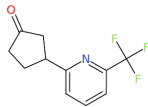
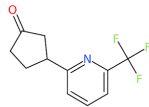
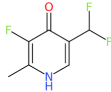
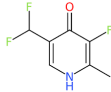
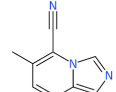
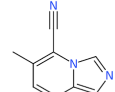
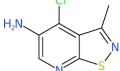
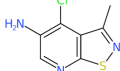
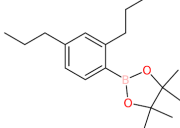
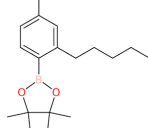
No	Original Molecule	Predicted Molecule	No	Original Molecule	Predicted Molecule
1.			6.		
2.			7.		
3.			8.		
4.			9.		
5.			10.		

Figure 6: Chemical structures depicted with the CDK depiction generator for predictions with Tanimoto similarity 1.0 and low BLEU score.

Conclusion

With this work, purely data-driven deep learning models for translation between different chemical entity representations are reported. We show that deep learning models are able to capture the essence of SMILES to IUPAC name string conversion (and vice versa) with reaching the 90% accuracy threshold. Despite this promising finding, any large scale and uncured application should be currently handled with care.

With more data and additional training epochs STOUT is expected to further improve its prediction accuracy in the future. At best, it may finally play in the ballpark of the rule-based systems which further on define the possible top performance. Using the TPU platform will enable the models to be trained in an acceptable amount of time in the order of a few weeks. In addition, STOUT may be extended to alternative sophisticated models used in language translation and understanding, such as BERT [22].

Appendix

BLEU scoring for machine translations is a scoring metric introduced in 2002 used to compare a predicted sentence with the original sentence. Each predicted word is compared with the original, and each word is called a unigram or a 1-gram. In longer sentences we can also compare word pairs or bigrams. Here, we calculated BLEU-1 for unigram comparison, BLEU-2 for the bigram comparison, BLEU-3 for 3-gram comparison and BLEU-4 for 4-gram comparison.

In order to compare the predicted IUPAC name with the original IUPAC name a sentence-to-sentence comparison should be done, so we used the sentence BLEU scoring function inbuilt in Python Natural Language Toolkit [23]. We use the original IUPAC name as the reference string and the predicted IUPAC name as the candidate string to calculate BLEU. If all words in the candidate match the reference, a perfect score of 1.0 will be awarded.

Reference : 1 , 3 , 7 - trimethylpurine - 2 , 6 - dione

Candidate : 1 , 3 , 7 - trimethylpurine - 2 , 6 - dione

BLEU score: **1.0**

BLEU-1: 1.00

BLEU-2: 1.00

BLEU-3: 1.00

BLEU-4: 1.00

BLEU score will reduce according to the following,

- each wrong word matches
- each wrong word pair matches
- length of the candidate string is longer/ shorter than reference string
- order of the predicted words are wrong

For these a penalty will be awarded so the overall score will decrease. Few examples are given below.

Wrong word

Reference : 1 , 3 , 7 - trimethylpurine - 2 , 6 - dione

Candidate : 1 , 3 , 7 - trimethylpurine - 2 , 6 - **trione**

BLEU score: **0.91**

BLEU-1: 0.92

BLEU-2: 0.92

BLEU-3: 0.92

BLEU-4: 0.91

Wrong word pair

Reference : 1 , 3 , 7 - trimethylpurine - 2 , 6 - dione

Candidate : 1 , 3 , 7 - trimethylpurine - 2 , 6 , **trione**

BLEU score : **0.82**

BLEU-1: 0.85

BLEU-2: 0.84

BLEU-3: 0.85

BLEU-4: 0.82

Shorter prediction

Reference : 1 , 3 , 7 - trimethylpurine - 2 , 6 - dione

Candidate : 1 , 3 , 7 - trimethylpurine - 2

BLEU score : **0.64**

BLEU-1: 0.64

BLEU-2: 0.64

BLEU-3: 0.64

BLEU-4: 0.64

Longer Prediction

Reference : 1 , 3 , 7 - trimethylpurine - 2 , 6 - dione

Candidate : 1 , 3 , 7 - trimethylpurine - 2 , 6 - dione , **6 - dione, 6 - dione**

BLEU score : **0.59**

BLEU-1: 0.63

BLEU-2: 0.59

BLEU-3: 0.58

BLEU-4: 0.48

Wrong order of predictions

Reference : 1 , 3 , 7 - trimethylpurine - 2 , 6 - dione

Candidate : 1 , 3 , 7 - trimethylpurine - **6 , 2** - dione

BLEU score: **0.65**

BLEU-1: 1.00

BLEU-2: 0.82

BLEU-3: 0.74

BLEU-4: 0.65

List of abbreviations

BLEU: Bilingual Evaluation Understudy

BERT: Bidirectional Encoder Representations from Transformers

CDK: Chemistry Development Kit

DECIMER: Deep IEarning for Chemical Image Recognition

FTP: File Transfer Protocol

GPU: Graphics Processing Unit

IUPAC: International Union of Pure and Applied Chemistry

InChI: International Chemical Identifier

NMT: Neural Machine Translation

OPSIN: Open Parser for Systematic IUPAC Nomenclature

RAM: Random Access Memory

RNN: Recurrent Neural Network

SDF: Structure Data File

SELFIES: Self-Referencing Embedded Strings

SMARTS: SMILES arbitrary target specification

SMILES: Simplified Molecular-Input Line-Entry System

STOUT: Smiles TO iUpac Translator

TPU: Tensor Processing Units

TFRecord: TensorFlow Record file

VRAM: Video Random Access Memory

Availability of data and materials

The code for STOUT and the trained models are available at <https://github.com/Kohulan/Smiles-TO-iUpac-Translator>

Declarations

Competing interests

AZ is co-founder of GNWI - Gesellschaft für naturwissenschaftliche Informatik mbH, Dortmund, Germany.

Funding

The authors acknowledge funding by the Carl-Zeiss-Foundation.

Authors' contributions

KR developed the software and performed the data analysis. CS and AZ conceived the project and supervised the work. All authors contributed to and approved the manuscript.

Acknowledgments

We cordially acknowledge the company ChemAxon for making their deterministic IUPAC name generator available for free for academic purposes, without which this project would not have been possible.

We are also grateful for the company Google making free computing time on their TensorFlow Research Cloud infrastructure available to us.

References

1. Contributors to Wikimedia projects (2004) List of chemical compounds with unusual names. https://en.wikipedia.org/wiki/List_of_chemical_compounds_with_unusual_names. Accessed 1 Dec 2020
2. Favre HA, Powell WH (2013) Nomenclature of Organic Chemistry: IUPAC Recommendations and Preferred Names 2013. Royal Society of Chemistry
3. Nomenclature of Inorganic Chemistry – IUPAC Recommendations 2005. Chem Int 27:25–26
4. Inczedy J, Lengyel T, Ure AM, Gelencsér A, Hulanicki A, Others (1998) Compendium of analytical nomenclature. Hoboken: Blackwell Science
5. Union internationale de chimie pure et appliquée. Physical, International Union of Pure and Applied Chemistry. Physical and Biophysical Chemistry Division (2007) Quantities, Units and Symbols in Physical Chemistry. Royal Society of Chemistry
6. Weininger D (1988) SMILES, a chemical language and information system. 1. Introduction to methodology and encoding rules. J Chem Inf Comput Sci 28:31–36

7. Heller SR, McNaught A, Pletnev I, Stein S, Tchekhovskoi D (2015) InChI, the IUPAC International Chemical Identifier. *J Cheminform* 7:23
8. Website. Daylight Inc. 4. SMARTS—a language for describing molecular patterns. <http://www.daylight.com/dayhtml/doc/theory/theory.smarts.html>. Accessed 16 Dec 2020
9. ChemAxon - Software Solutions and Services for Chemistry & Biology. <https://www.chemaxon.com>. Accessed 23 Nov 2020
10. Steinbeck C, Han Y, Kuhn S, Horlacher O, Luttmann E, Willighagen E (2003) The Chemistry Development Kit (CDK): an open-source Java library for Chemo- and Bioinformatics. *J Chem Inf Comput Sci* 43:493–500
11. Website. RDKit: open-source cheminformatics. <https://www.rdkit.org>. Accessed 26 Nov 2020
12. O’Boyle NM, Banck M, James CA, Morley C, Vandermeersch T, Hutchison GR (2011) Open Babel: An open chemical toolbox. *J Cheminform* 3:33
13. Kim S, Chen J, Cheng T, et al (2019) PubChem 2019 update: improved access to chemical data. *Nucleic Acids Res* 47:D1102–D1109
14. Rajan K, Zielesny A, Steinbeck C (2020) DECIMER: towards deep learning for chemical image recognition. *J Cheminform* 12:65
15. O’Boyle N, Dalke A DeepSMILES: An Adaptation of SMILES for Use in Machine-Learning of Chemical Structures. <https://doi.org/10.26434/chemrxiv.7097960>
16. Krenn M, Häse F, Nigam A, Friederich P, Aspuru-Guzik A (2020) Self-referencing embedded strings (SELFIES): A 100% robust molecular string representation. *Mach Learn: Sci Technol* 1:045024
17. Luong M-T, Pham H, Manning CD (2015) Effective Approaches to Attention-based Neural Machine Translation. *arXiv:1508.04025 [cs.CL]*
18. Bahdanau D, Cho K, Bengio Y (2014) Neural Machine Translation by Jointly Learning to Align and Translate. *arXiv:1409.0473 [cs.CL]*
19. Abadi M, Agarwal A, Barham P, et al (2016) TensorFlow: Large-Scale Machine Learning on Heterogeneous Distributed Systems. *arXiv:1603.04467 [cs.DC]*
20. Papineni K, Roukos S, Ward T, Zhu W-J (2002) BLEU: a method for automatic evaluation of machine translation. In: *Proceedings of the 40th annual meeting of the Association for Computational Linguistics*. pp 311–318
21. Lowe DM, Corbett PT, Murray-Rust P, Glen RC (2011) Chemical name to structure: OPSIN, an open source solution. *J Chem Inf Model* 51:739–753
22. Devlin J, Chang M-W, Lee K, Toutanova K (2018) BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. *arXiv:1810.04805v2 [cs.CL]*
23. Bird S, Klein E, Loper E (2009) *Natural Language Processing with Python: Analyzing Text with the Natural Language Toolkit*. “O’Reilly Media, Inc.”