NON-CONFIDENTIAL

SmiZip – Ancient Relic or Buried Treasure?

Noel O'Boyle
12th RDKit UGM Mainz

Disclaimer

The material that follows is a presentation of general background information about Sosei Group Corporation and its subsidiaries (collectively, the “Company”) as of the date of this presentation. This material has been prepared solely for informational purposes and is not to be construed as a solicitation or an offer to buy or sell any securities and should not be treated as giving investment advice to recipients. It is not targeted to the specific investment objectives, financial situation or particular needs of any recipient. It is not intended to provide the basis for any third party evaluation of any securities or any offering of them and should not be considered as a recommendation that any recipient should subscribe for or purchase any securities.

The information contained herein is in summary form and does not purport to be complete. Certain information has been obtained from public sources. No representation or warranty, either express or implied, by the Company is made as to the accuracy, fairness, or completeness of the information presented herein and no reliance should be placed on the accuracy, fairness, or completeness of such information. The Company takes no responsibility or liability to update the contents of this presentation in the light of new information and/or future events. In addition, the Company may alter, modify or otherwise change in any manner the contents of this presentation, in its own discretion without the obligation to notify any person of such revision or changes.

This presentation contains “forward-looking statements,” as that term is defined in Section 27A of the U.S. Securities Act of 1933, as amended, and Section 21E of the U.S. Securities Exchange Act of 1934, as amended. The words “believe”, “expect”, “anticipate”, “intend”, “plan”, “seeks”, “estimates”, “will” and “may” and similar expressions identify forward looking statements. All statements other than statements of historical facts included in this presentation, including, without limitation, those regarding our financial position, business strategy, plans and objectives of management for future operations (including development plans and objectives relating to our products), are forward looking statements. Such forward looking statements involve known and unknown risks, uncertainties and other factors which may cause our actual results, performance or achievements to be materially different from any future results, performance or achievements expressed or implied by such forward looking statements. Such forward looking statements are based on numerous assumptions regarding our present and future business strategies and the environment in which we will operate in the future. The important factors that could cause our actual results, performance or achievements to differ materially from those in the forward looking statements include, among others, risks associated with product discovery and development, uncertainties related to the outcome of clinical trials, slower than expected rates of patient recruitment, unforeseen safety issues resulting from the administration of our products in patients, uncertainties related to product manufacturing, the lack of market acceptance of our products, our inability to manage growth, the competitive environment in relation to our business area and markets, our inability to attract and retain suitably qualified personnel, the unenforceability or lack of protection of our patents and proprietary rights, our relationships with affiliated entities, changes and developments in technology which may render our products obsolete, and other factors. These factors include, without limitation, those discussed in our public reports filed with the Tokyo Stock Exchange and the Financial Services Agency of Japan. Although the Company believes that the expectations and assumptions reflected in the forward-looking statements are reasonably based on information currently available to the Company's management, certain forward looking statements are based upon assumptions of future events which may not prove to be accurate. The forward looking statements in this document speak only as at the date of this presentation and the company does not assume any obligations to update or revise any of these forward statements, even if new information becomes available in the future.

This presentation does not constitute an offer, or invitation, or solicitation of an offer, to subscribe for or purchase any securities. Neither this presentation nor anything contained herein shall form the basis of any contract or commitment whatsoever. Recipients of this presentation are not to construe the contents of this summary as legal, tax or investment advice and recipients should consult their own advisors in this regard.

This presentation and its contents are proprietary confidential information and may not be reproduced, published or otherwise disseminated in whole or in part without the Company’s prior written consent. These materials are not intended for distribution to, or use by, any person or entity in any jurisdiction or country where such distribution or use would be contrary to local law or regulation.

This presentation contains non-GAAP financial measures. The non-GAAP financial measures contained in this presentation are not measures of financial performance calculated in accordance with IFRS and should not be considered as replacements or alternatives profit, or operating profit, as an indicator of operating performance or as replacements or alternatives to cash flow provided by operating activities or as a measure of liquidity (in each case, as determined in accordance with IFRS). Non-GAAP financial measures should be viewed in addition to, and not as a substitute for, analysis of the Company's results reported in accordance with IFRS.

References to "FY" in this presentation for periods prior to 1 January 2018 are to the 12-month periods commencing in each case on April 1 of the year indicated and ending on March 31 of the following year, and the 9 month period from April 1 2017 to December 31 2017. From January 1 2018 the Company changed its fiscal year to the 12-month period commencing in each case on January 1. References to "FY" in this presentation should be construed accordingly.

sosei HEPTARES

Sosei Heptares GPCR Structure-Based Drug Discovery Company

Delivered 25+ pre-clinical candidates, produced 10+ clinical candidates

>6 new pre-clinical candidates expected in the next 2 years for internal and collaboration programs

R&D CENTRE
CAMBRIDGE, UK (Heptares)
~170 EMPLOYEES

HEADQUARTERS
TOKYO, JAPAN (Sosei K.K.)
~25 EMPLOYEES

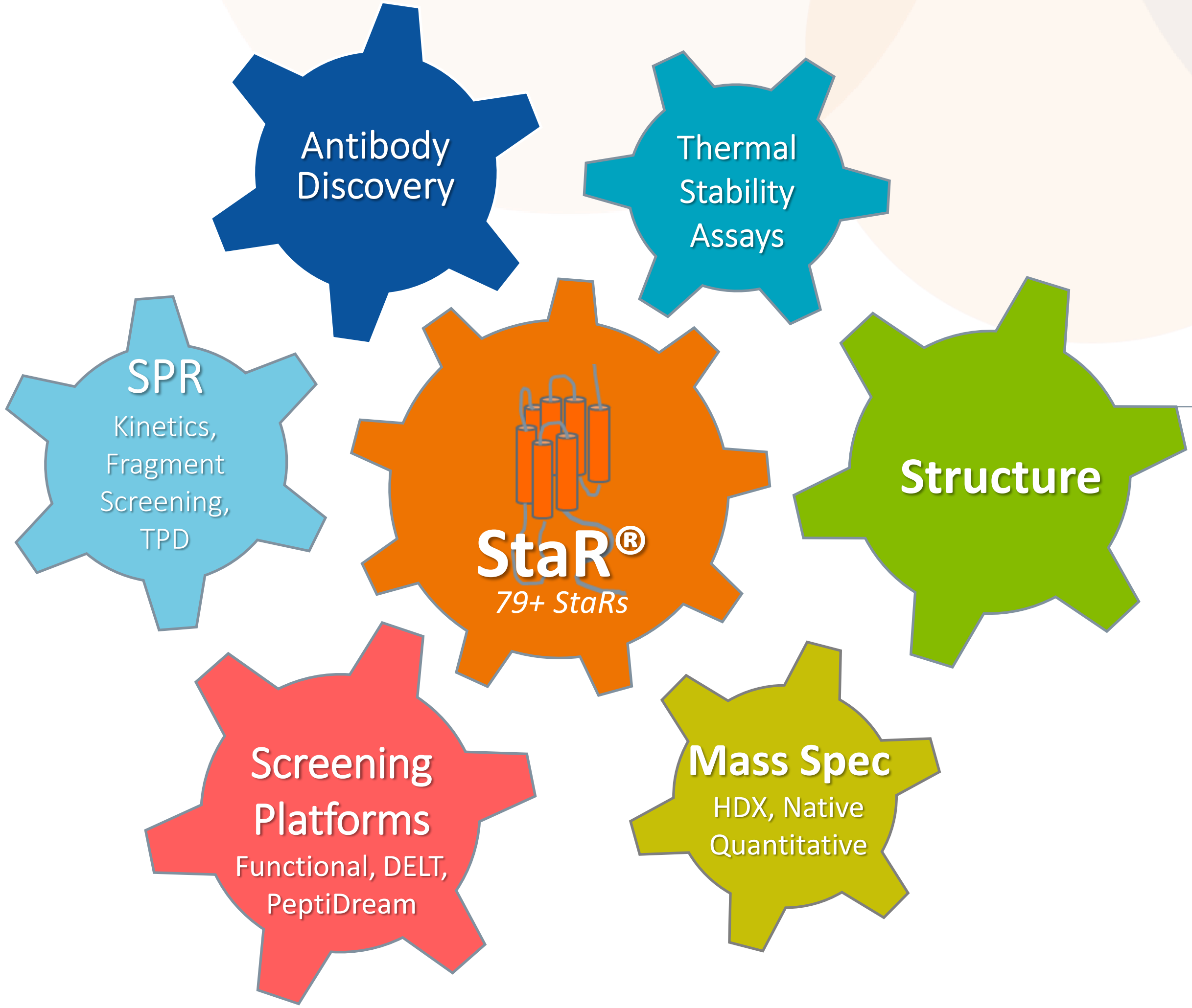
DRUG DISCOVERY/EARLY DEVELOPMENT DRIVEN BY STAR® / SBDD ENGINE					LATE STAGE DEVELOPMENT	
PRE-DISCOVERY	DISCOVERY	PRECLINICAL	PH 1a	PH 1b	PH 2/3	APPROVAL
StaR® Technology Structure Hit ID Target Validation ✓	**Structure Based Drug Design** Chemistry Pharmacology ✓	DMPK Toxicology Translation ✓	SAD MAD Experimental Medicine ✓	Dosing assessment Proof of Mechanism ✓	Tactical development team sources global tx for Japanese patients PMDA Clinical Efficacy ✓	PMDA Pharmacovigilance ✓

Programs advanced to PoM or PoC before partnering / seeded into co-owned investment vehicles

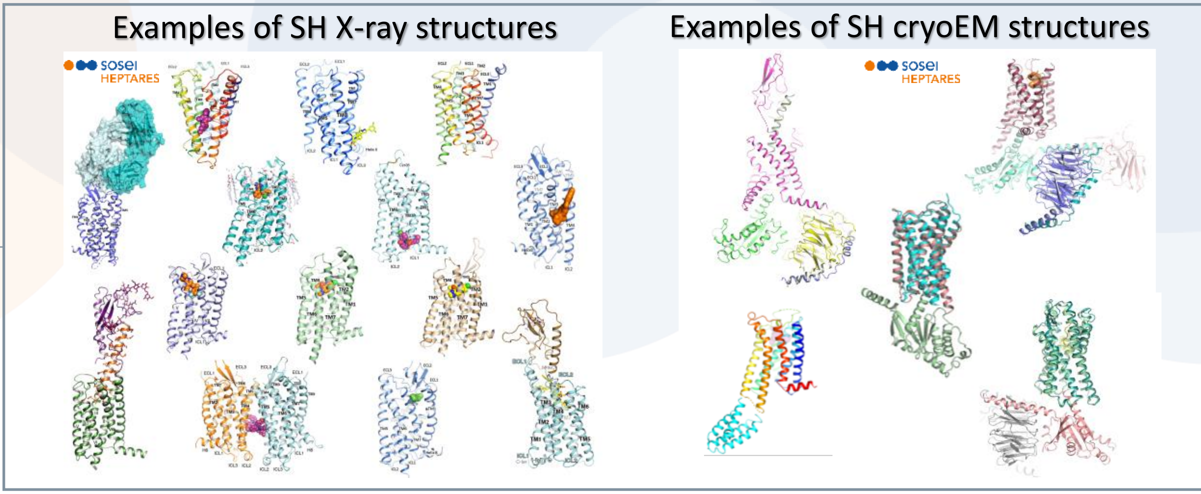
sosei HEPTARES

https://soseiheptares.com

StaR® Technology enabling GPCR Structure-Based Drug Discovery



X-ray crystallography platform complemented by Cryo-EM structural enablement to expand scope of GPCR SBDD



StaRs double small molecule ligand bound GPCR structures for SH SBDD

PDB * → Without structure:
~253 GPCRs
~636 modalities**

20 GPCRs: only peptide

141 GPCRs

12 GPCRs: only Apo

89 peptide:
3 **SH**

374 SM:
72 **SH**
113 GPCRs

Sosei **H**eptares *

Total:
328 SM
13 not SM

SH not in PDB:
256 SM
11 not SM

42 GPCRs

* September 2022, unique GPCR-ligand complexes
** Assuming 403 non-olfactory GPCRs x 2 modalities (agonist/PAM, antagonist/NAM)

SmiZip in theory

SMILES Multigram Compression

Roger Sayle[1] and Jack Delany[2]

[1] Metaphorics LLC, Santa Fe, New Mexico

[2] Daylight CIS, Santa Fe, New Mexico

Mug01, 6-9 March 2001, Santa Fe, New Mexico, USA.

What is SmiZip?

- A compression method for short strings, developed for SMILES strings by Roger Sayle in 1998[1]
- SmiZip itself was never published in a journal, but full details were presented at the Daylight MUG01 in Santa Fe (2001):
  - https://www.daylight.com/meetings/mug01/Sayle/SmiZip/sld001.htm (view online)
  - https://www.daylight.com/meetings/mug01/Sayle/SmiZip.ppt (PowerPoint)
- While used internally at NextMove, SmiZip is pretty much unknown in the broader community and no public implementation exists

[1] See email from Andrew Dalke (“Version 1.1, November 1998”)

How does SmiZip work?

- The basic unit of storage on a computer is a byte, which can have 256 values (0-255)
- A typical set of SMILES strings only uses 70 different values (e.g. value 67 for ‘C’, value 64 for ‘@’)
- Let’s associate the other ~190 values with an n-gram of length > 1 (i.e. a run of two or more characters)
  - The list of n-grams is chosen to maximally compress a training set
- Encoding: a query SMILES string is optimally encoded in terms of the 256 n-grams, and then converted to byte values

Compression is fast, but requires finding the optimal encoding of the SMILES as n-grams

```
COc1ccc(c(c1)OC)/C=C/c2ccccc2O

CO   c1ccc(c(c1)   OC)   /C=C/   c2ccccc2   O

     62      119     234     177     89     79

                  >wê±YO
```

Decompression is very fast

- Here, relative size after compression is 6/30 = 20%
  - Typical size is 25-30% depending on the dataset.
- **Note:** compression and decompression must use the same list of n-grams

Why use SmiZip?

- Everywhere a SMILES can be used, a zipped SMILES could be used instead
  - Decompression is very fast
- Save diskspace/RAM
  - Could be the difference between fast in-memory database and slow on-disk
  - Ultra-large databases: Enamine REAL/Space, etc
  - Chemistry web apps, or phone apps
- Unlike gzip, SmiZip…
  - Is suitable for compressing short strings like SMILES, e.g. for storing in a database table
  - Preserves the ability to randomly-access a compressed file
- May yield improved performance for algorithms
- The compressed format may be useful in its own right…

Multigram Compression

- Efficient compression is more tricky…

Given a simple alphabet of only “A” and “B”.
with the set of multigrams “A”, “B”, “AB” and “BAA”.
Encode the string “ABAA”.

The greedy solution uses 3 bytes “AB”, “A” and “A”.

An optimal solution uses only 2 bytes, “A” and “BAA”.

Dynamic Programming

- The computer science solution[1] to such 1D tiling problems is a two pass algorithm called "Dynamic Programming".
- For each prefix, the optimal length is the shortest sub-prefix before each valid suffix multigram.

To Encode the string "ABAA" in terms of **A**, **B**, **AB** and **BAA**, look at what n-grams each prefix ends with:

encode("A"): **A** = 1
encode("AB"): **AB**, or "A" + **B** = min(1, encode("A")+1) = min(1, 1+1) = 1
encode("ABA"): "AB" + **A** = encode("AB")+1 = 1+1 = 2
encode("ABAA"): "ABA" + **A**, or "A" + **BAA** = min(encode("ABA")+1, encode("A")+1) =
min(2+1, 1+1) = 2

[1] Wagner, *Communications of the ACM*, **1974**, *16*, 148.

Which n-grams to use?

- Simple - just choose the n-grams that will give maximal compression on the data that will be compressed

- If you don't know the exact data, then choose a representative training set
  - We will look at sensitivity to the training set later

- More of a problem is that you cannot try every possible combination of ~190 n-grams to find the best set
  - For example, in a set of 1000 SMILES strings from ChEMBL the number of n-grams of length 2 to 60 that occur in at least two strings is ~71000 (for 10K SMILES, this number is almost 1M)
  - 71000 choose 190 is[1] 4.398×10^{569}

[1] https://www.hackmath.net/en/calculator/n-choose-k

N-gram selection algorithm

- This implementation uses a greedy approach:
  - 1. Iterate through a list of n-grams prioritised by estimated (or previously measured) score * frequency
    - In slow settings, at least 80 previously measured n-grams must be tested and at least 100 in total
  - 2. Choose the one that gives the best compression on a training set when added to the current set of chosen n-grams
  - 3. Update n-gram scores (see below)
  - 4. Back to Step 1, until 256 n-grams have been chosen
- Potential improvement: occasional backwards elimination
- Scores are a measure of the additional compression obtained if that n-gram were chosen
- Initial values are assigned as the length of the n-gram – 1
  - E.g. c1ccccc1 would be assigned 7 (as 8 characters would be replaced by one n-gram)
- Note that these values become over-optimistic as the selection procedure progresses
  - A partial correction is to re-estimate the score for any n-gram that contains the latest chosen n-gram
    - E.g. if cc is chosen, then the value of c1ccccc1 will be re-estimated. This is done by encoding c1ccccc1 using the chosen n-grams, in this case as c,1,cc,cc,c,1 – i.e. 6 n-grams, giving a score of 5.
- For any n-gram chosen in (1) above to be tested, its estimated score is replaced by a measured one
  - Measured score = the additional compression on the training set divided by the number of occurrences of that token
  - Measured n-gram scores are not re-estimated subsequently

Training set for identification of n-grams

- A set of 1.2M randomly sorted SMILES strings should be provided
- The first 10K are retained as a holdout test set to evaluate the size after compression

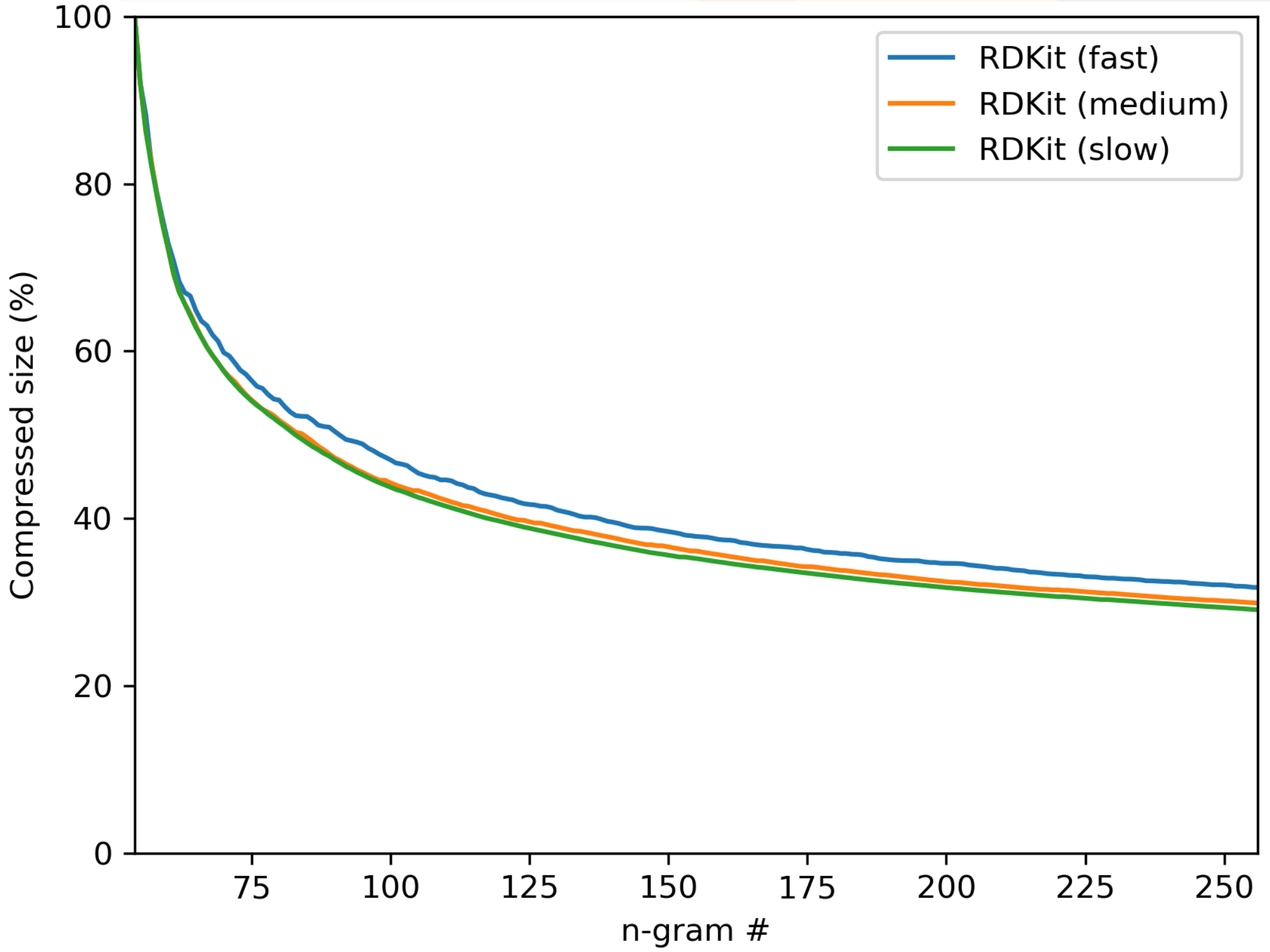
- Note that finding the initial n-gram that yields maximal compression can be done with a small training set
  - …but finding the 200th n-gram that maximally improves compression requires a larger training set
- With this in mind, n-grams are selected from a set of 1000 SMILES initially*, a size that increases by 45* for each additional n-gram, arriving at ~10K for the final n-gram
  - To avoid overfitting, training sets do not overlap but instead are read as the next N entries in the original file

- Example: the final n-gram chosen for the RDKit canonical SMILES compression model was /C=N/
  - It occurred only 94 times in a set of 10045 SMILES
  - Maximum possible reduction in characters is therefore 4*94 = 376 characters
  - Actual reduction in training set was 281
  - Reduction in character count for holdout set was 249 fewer characters (0.09% of original size)

* Slow settings: 1000/45 Medium settings: 250/12 Fast settings: 100/4.5

SmiZip in practice

RDKit canonical SMILES: Compressed size of ChEMBL test set

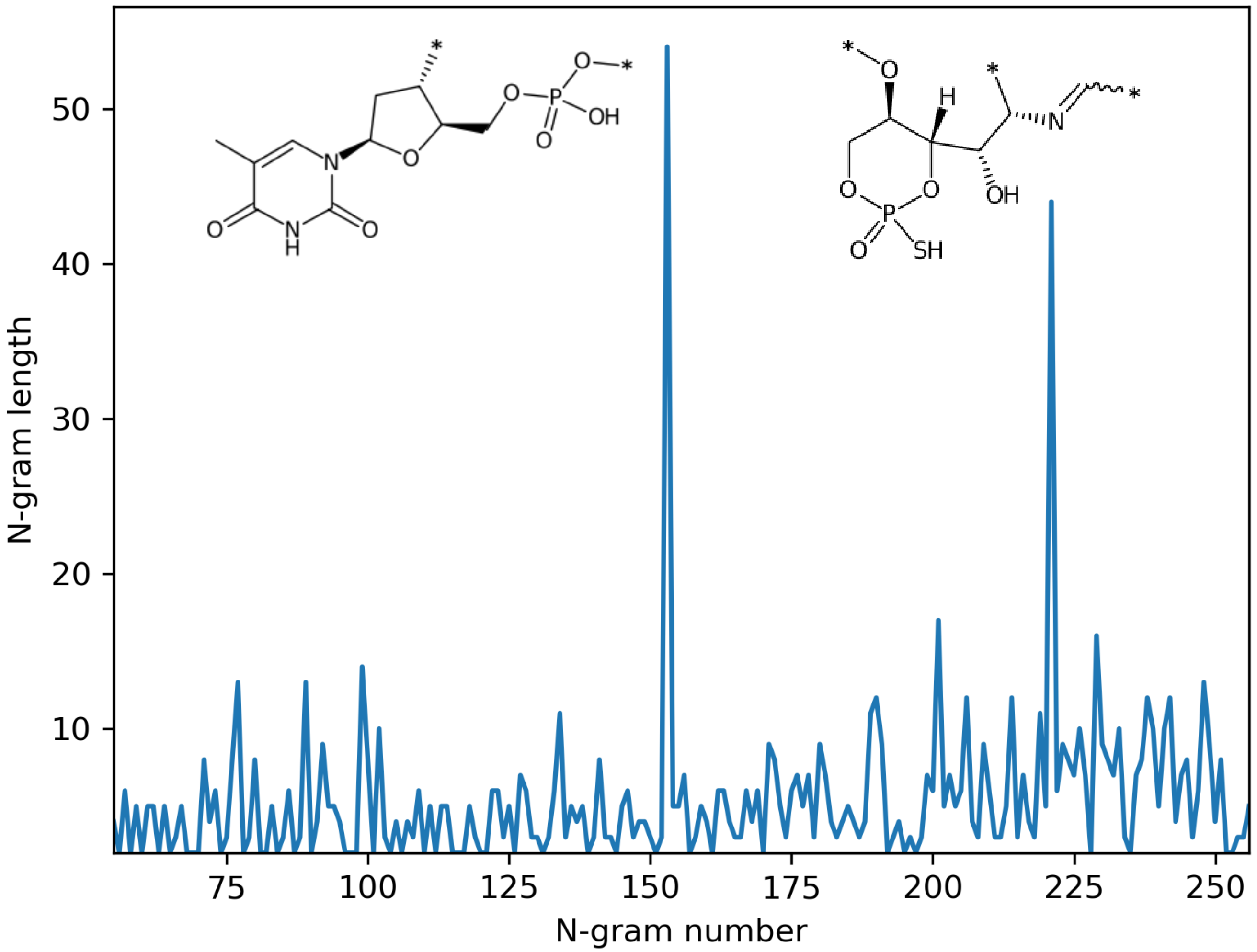


Speed setting	Time	Size after compression
Fast	15m	31.7%
Medium	1h 30m	29.9%
Slow	5h 8m	29.1%

N-grams 1-6	7-12	13-18
(=O)	c1ccc	c2
cc	c2ccc	O)
[C@@H]	)c	c(
CC	C(=O)	C(F)(F)F
[C@H]	c1	[nH]
(C	c3ccc	C(=O)N

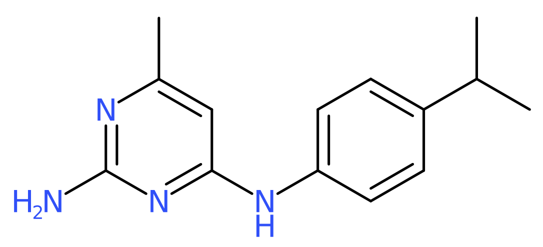
What length of N-gram needs to be supported?

- My implementation supports n-grams of up to 60 characters

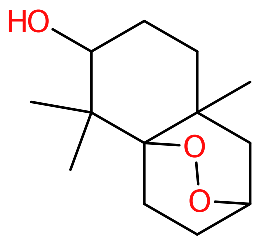


Effect of ring renumbering for different toolkits

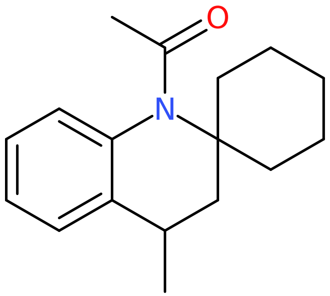
- OEChem:
  - Avoids reusing any digit until it gets to 10
  - **Cc1cc(nc(n1)N)Nc2ccc(cc2)C(C)C**
  - **Cc1cc(nc(n1)N)Nc1ccc(cc1)C(C)C**
  - Large effect: 28.2% -> 24.8%



- RDKit:
  - Avoids reusing the same digit on an atom
  - **CC12CCC(O)C(C)(C)C13CCC(C2)OO3**
  - **CC12CCC(O)C(C)(C)C11CCC(C2)OO1**
  - Negligible: 29.07% -> 29.00%



- OpenBabel:
  - Like RDKit, avoids reusing the same digit on an atom but does this by placing ring openings first before ring closures
    - => Not possible to renumber rings without changing the order (with corresponding inversion of any tetstereo)
  - **CC(=O)N1c2ccccc2C(CC21CCCCC2)C**
  - **CC(=O)N1c2ccccc2C(CC11CCCCC1)C**
  - Not implemented, so SMILES remain at 25.7%



sosei HEPTARES

Compressing using suboptimal tokens

- What if we use the tokens chosen from training on Toolkit X to compress canonical SMILES from Toolkit Y?
  - A measure of the extent to which the generated SMILES are different

…using tokens chosen from training on…

Compress SMILES from…

	RDKit	OEChem	Open Babel	Combined set
RDKit	29.1%	35.9%	34.9%	30.6%
OEChem	32.4%	24.8% *28.2%	26.5%	25.8%
Open Babel	32.2%	27.5%	25.7%	26.6%
Combined set				27.6%

* Without ring renumbering

- OEChem and Open Babel can interchange tokens without a large loss of compression, but the RDKit tokens don't transfer to those or vice versa.
- Using tokens from a combined set (1/3 from each toolkit) yields a happy medium
  - Could be used as a general purpose compressor for SMILES from any source

Separating the effects of canonical ordering and traversal order

- How much of the difference between toolkit SMILES compression is due to the order in which the canonicalization procedure writes out atoms?
- What if we read the canonical SMILES from Toolkit X and write it out (without canonicalization) with Toolkit Y?
  - Any differences will be mostly due to traversal order (e.g. favour double bonds over single bonds)

Generate SMILES using… / …from canonical SMILES from…	RDKit	OEChem	Open Babel
RDKit	29.1%	28.3%	28.9%
OEChem (w/o ring renum)	25.1% (28.8%)	24.8% (28.2%)	25.1% (28.8%)
Open Babel	28.7%	24.9%	25.7%

- The SMILES written by OEChem have the highest compression regardless of the original atom order (Comparing rows, e.g. 25.1% < 29.1%/28.7%)
  - Likely explanation is that this is due to the preference in favouring ring atoms over non-ring atoms:
    - E.g. `FC1CC(I)C(O)C1` is written as `FC1CC(C(C1)I)O`
    - Rings with similar substitution patterns will share the same n-gram, `C1CC(C(C1)`, which is not the case where the ring is broken up by the substituent atoms

Separating the effects of canonical ordering and traversal order

- How much of the difference between toolkit SMILES compression is due to the order in which the canonicalization procedure writes out atoms?
- What if we read the canonical SMILES from Toolkit X and write it out (without canonicalization) with Toolkit Y?
  - Any differences will be mostly due to traversal order (e.g. favour double bonds over single bonds)

…from canonical SMILES from…

Generate SMILES using…

	RDKit	OEChem	Open Babel
RDKit	29.1%	28.3%	28.9%
OEChem (w/o ring renum)	25.1% (28.8%)	24.8% (28.2%)	25.1% (28.8%)
Open Babel	28.7%	24.9%	25.7%

- The canonical ordering used by OEChem leads to the highest compression no matter which toolkit writes out the SMILES string (comparing columns, e.g. 28.3% < 29.1%/28.9%)
  - At least in the case of Open Babel, the improvement from 25.7% to 24.9% is likely due to the fact that the atom order is largely preserved from the original

Cmprssng evn frthr

- We have already seen the effects on compression of
  - (a) Ring renumbering for OEChem SMILES in particular
  - (b) A traversal order that favours ring atoms

Cmprssng evn frthr

- We have already seen the effects on compression of
  - (a) Ring renumbering for OEChem SMILES in particular
  - (b) A traversal order that favours ring atoms
- Let's repurpose an earlier idea of mine: DeepSMILES[1]
  - If we use just the ring rewriting features of DeepSMILES, then c1ccccc1 and c2ccccc2 are both written as cccccc6
  - This is (a) shorter (see 95.1% below), and (b) more compressible (30.4% vs 28.5%)

	.smi	.smiz	.smiz.gz [-9]
SMILES	100%	30.4%	23.7%
DeepSMILES	95.1% (100%)	27.1% (28.5%)	21.3% (22.4%)

[1] 6th RDKit UGM 2018, https://www.slideshare.net/NextMoveSoftware/deepsmiles, https://github.com/baoilleach/deepsmiles

Comparison to alternatives

Comparison to Gzip

- While the use-cases are different, it’s still interesting to compare SmiZip vs Gzip
- Note that my SmiZip implementation is in Python, but uses pyahocorasick (written in C) for string matching

- The following comparison is on a file containing just SMILES from ChEMBL 31 (2304875 lines)

	Size after compression	Time to compress (s)	Time to decompress (s)
gzip	27.8%	9.7	2.3
gzip -9	26.6%	31.7	2.3
SmiZip	30.4%	69.0	3.7
SmiZip, then gzip -9	23.7%		

What about general alternatives for short string compression?

- Smaz - https://github.com/antirez/smaz (2009)
  - Very popular (1.1K stars on GitHub) – ported to Go, Rust, Java, JavaScript, TypeScript, C#...
  - This is SmiZip except that two values (of the 256) allow the subsequent character(s) to be passed through unchanged
  - Uses greedy matching of strings rather than an optimal encoding
  - Does not use the Aho-Corasick algorithm (1975) for string matching, but instead some sort of hashing
  - Implemented for SMILES by Andrew Dalke as smilez – https://hg.sr.ht/~dalke/smilez - compressed SMILES to 38%
    – Python 3 port (David Lorenzana, https://github.com/davidlorenzana/smilez)
- FemtoZip – https://github.com/gtoubassi/femtozip (2012)
  - [1]Andrew Dalke used it to compress SMILES to 40%
- Shoco – http://ed-von-schleck.github.io/shoco/ (2014)
  - [1]Andrew Dalke trained it on 1.5M SMILES and was able to compress them to 47%
- Unishox2- https://github.com/siara-cc/Unishox2 (2019)
- AIMCS - An Artificial Intelligence based Method for Compression of Short Strings (2020)
  - https://ieeexplore.ieee.org/document/9108719
  - https://github.com/MasoudAbedi/AIMCS-an-artificial-intelligence-based-method-for-compression-of-short-strings

[1] https://news.ycombinator.com/item?id=10060018

sosei HEPTARES

What if we used SmiZip for general text compression?

- Rather than consider how well Shoco, etc. work on SMILES, let's train SmiZip on English text and see how it compares to the others
- From https://ed-von-schleck.github.io/shoco/#comparisons-with-other-compressors

> Performance-wise, shoco is typically faster by at least a factor of 2. As an example, compressing and decompressing all words in `/usr/dict/share/words` with smaz takes around 0.325s on my computer and compresses on average by 28%, while shoco has a compression average of 33% (with the standard model; an optimized model will be even better) and takes around 0.145s. shoco is *especially* fast at decompression.

- So shoco (standard model) compresses to 67%, while smaz compresses to 72%
- SmiZip compresses to 55.6% (or 59.9% if we include the CR)
  - Note: trained on Centos 7 /usr/share/dict/words, and only supporting the characters found there:
    - !&',-./0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz
- For the text in the quote above[2] shoco compresses to 71%, while SmiZip gives 65%

[1] https://news.ycombinator.com/item?id=10060018
[2] I removed the parentheses and semicolon, and replaced % by pc.

Applications beyond compression

SmallWorld

- A system for nearest neighbour search in graph edit distance space
- Uses a graph database with 1 trillion nodes, 11 trillion edges
- Each node is a SMILES string written as an anonymous graph
  - All atoms replaced with *
  - All bonds as single bonds
- These compress really well as fewer characters are used

SMIZIP

SMILES entries can be compressed with a set of anonymous SMILES specific multi-grams

- Replaces multi-character substrings with a single byte
- Maintain random access to records

https://www.daylight.com/meetings/mug01/Sayle/SmiZip/index.htm

DOCK Meeting, UCSF, 12th Aug 2023

From https://www.nextmovesoftware.com/talks/Mayfield_ManagingAndSearchingOneTrillionCompounds_DOCK_230812.pdf

Matched n-gram pairs

- Apply SmiZip[1] to ChEMBL and look for examples where two compressed representations differ by replacement of an n-gram
    - Method repurposed[2] from 8th RDKit UGM
- Note that the results are biased by the n-grams used in the optimal encoding
    - A more direct method to find such replacements could be implemented directly on SMILES strings (without involving SmiZip)
- In theory could be used to generate additional structures by replacement of tokens
    - Perhaps additional context would be required

N-gram A	N-gram B	Freq	
F	Cl	21532	
c1ccc(cc1)	c1cccc(c1)	19671	Para/meta
CC	C	16416	
c1ccccc1	c1cccc(c1)	15960	Ortho/meta
CC	CCC	14388	
Cl	Br	13153	
Cl)	F)	12056	
CCC	CCCC	11580	
c1ccc(cc1)	c1ccccc1	11327	Para/ortho
C[C@@H](	C[C@H](	10757	
CCN	CCCN	9702	
Cc1ccc(cc1)	Cc1ccccc1	9040	Para/ortho
c1ccc(cc1)	COc1ccc(cc1)	8974	
c1ccc(cc1)	Cc1ccc(cc1)	8899	
Cc1ccc(cc1)	COc1ccc(cc1)	8813	
CO	CCO	8315	
F	Br	8298	
[C@H]1	[C@@H]1	8017	
o	s	7952	
C(F)(F)F	Cl	7599	
[C@@H](	[C@H](	7175	
c1ccc(c(c1)	c1cc(ccc1	7169	Meta+para/Meta
[N+](=O)[O-]	Cl	7076	
C(F)(F)F	F	6946	

[1] Here we use OEChem as it preserves ring structures in the SMILES (as discussed earlier)
[2] https://baoilleach.blogspot.com/2020/10/finding-matched-pairs-of-peptide-at.html

Hypothesis: Compressibility is linked to structural complexity

- In information theory, Kolmogorov Complexity refers to the idea that the complexity of a string is the length of the shortest possible description of the string (in some universal description language)

- For SMILES strings of the same length, those that are shorter when compressed contain n-grams that are frequently observed
  - Complexity/lack of compressibility indicates that the SMILES string contains n-grams that are not common; this rarity of occurrence may indicate difficulty of synthesis or at minimum chemical reasonableness

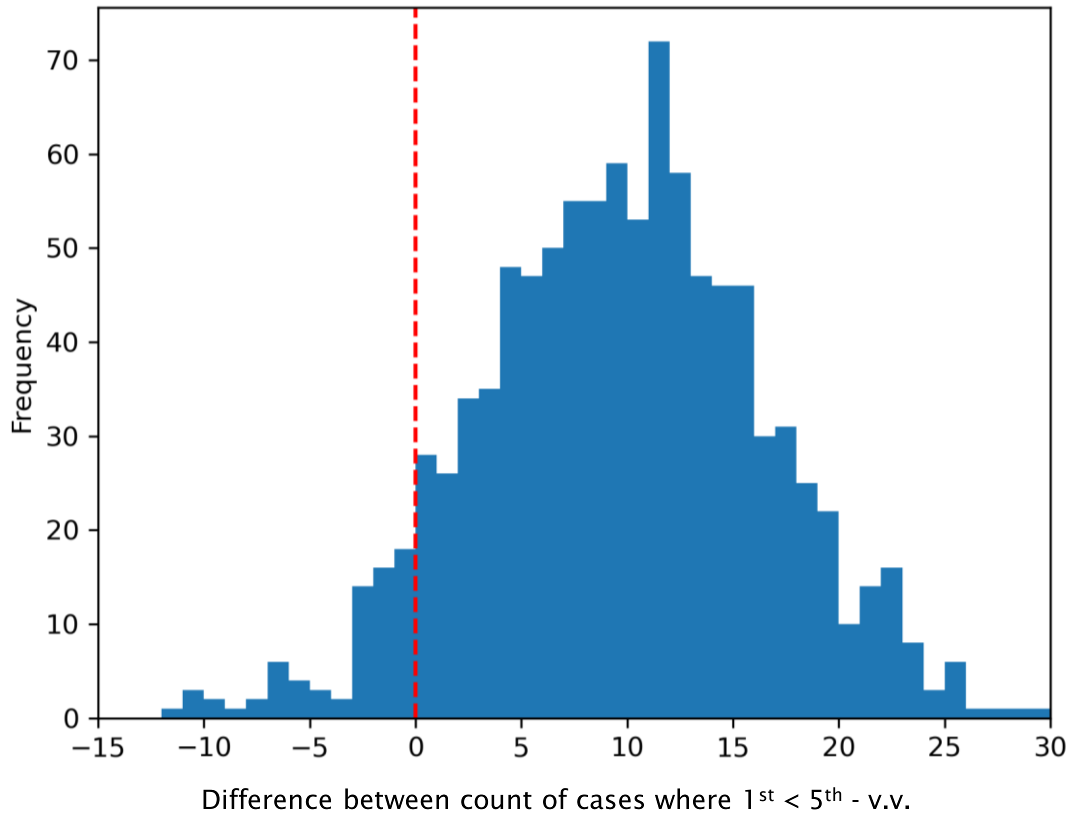
- How can we test this hypothesis?
  - We would need a benchmark set of molecules with different complexity…

- What about comparing the molecules made early in a project vs those made later?
  - Assumption 1: That this is linked to complexity

- Let's use data from ChEMBL assays
  - Assumption 2: Less potent molecules were made earlier; more potent molecules, later

Test the hypothesis

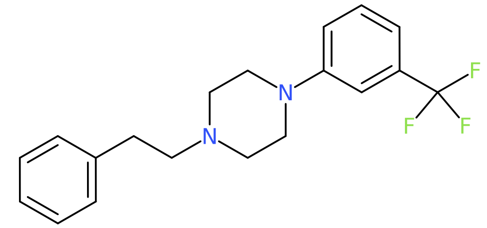
- We can use the Single Assay benchmark from my fingerprint comparison paper[1]
  - Sets of 5 molecules ordered by activity from the same ChEMBL assay
    - Adjacent molecules must be separated by ≥0.4 log units, excluded the lowest activity in the assay
  - Confounding molecules have been removed
    - Molecules that are have INNs or appear in Wikipedia, or that appear in multiple papers
  - Only assays with between 8 and 25 molecules
    - Smaller assays contain more dissimilar molecules; larger assays may contain several series
- For each of the 1000 datasets
  - For each of the 4.5K entries in the dataset, compress the first and last (5^{th}) SMILES strings if they are the same length *and* they have the same HAC
  - Count the no. of cases where the 1^{st} is shorter than the 5^{th}, and subtract the count of the no. of cases where it's v.v.
- Histogram of this number for 1000 datasets →

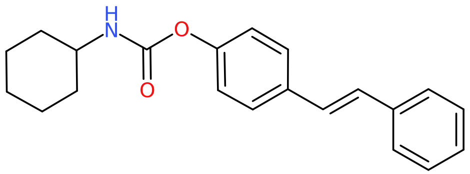


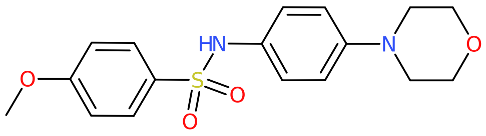
[1] NM O'Boyle, RA Sayle. Comparing structural fingerprints using a literature-based similarity benchmark. *J. Cheminf.* **2016**, *8*, 36.

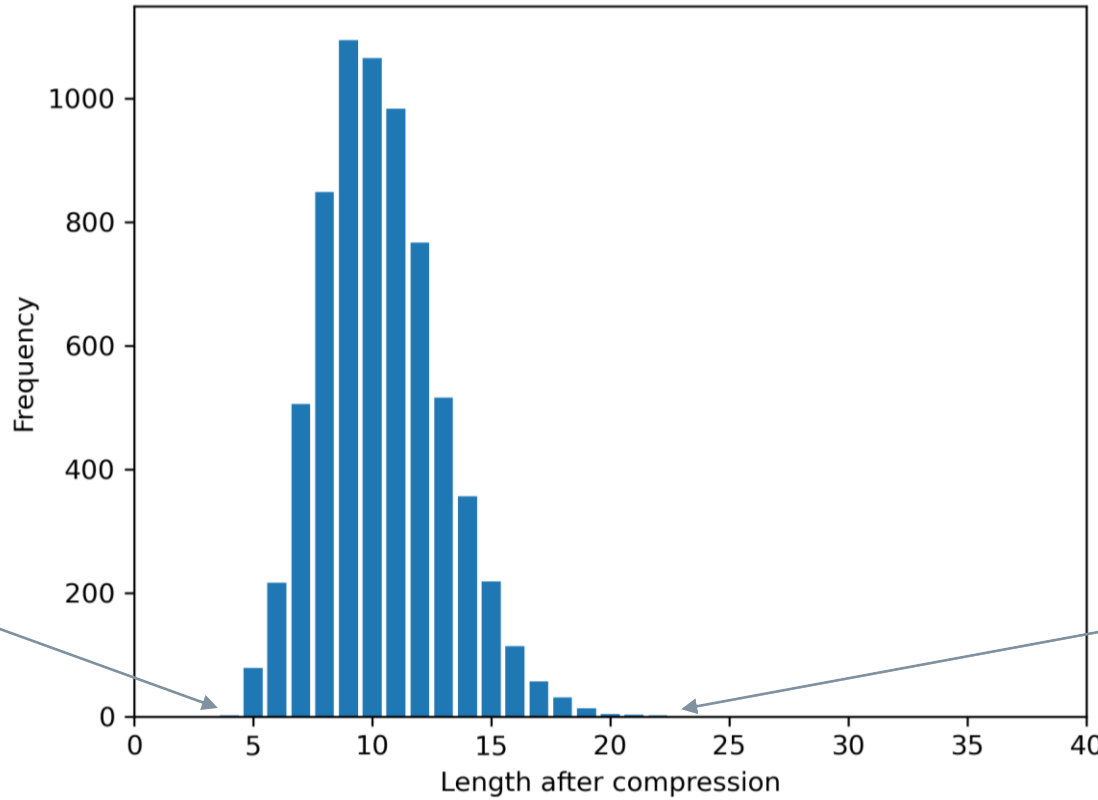
A measure of complexity based on SmiZip

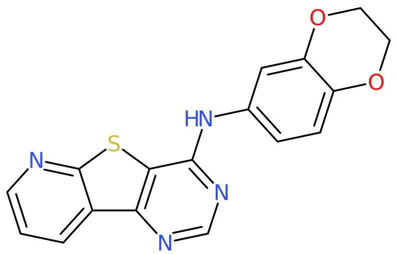
- How to exploit this to produce a measure of complexity?
- Could consider the frequency of occurrence of each token and sum the logs of their likelihoods
- A more empirical approach is use the distribution from (e.g.) ChEMBL to calculate a z-score
  - Here's the distribution for SMILES length of 40 and HAC of 24 (6878 cases in the ChEMBL 20 set)

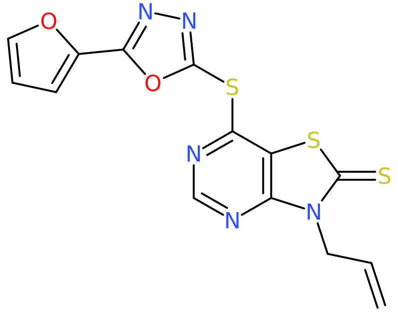
5


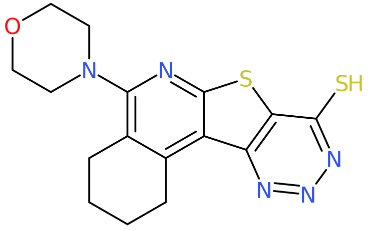
5


4





21


22


23

Use SmiZipped SMILES for a generative model

- Rather than training a generative model on SMILES, we can train on SmiZipped SMILES
- Potential advantages:
  - By learning from n-grams from the training data, the generated molecules will be more likely to be valid and be within the training set distribution
  - The requirement for the NN to learn that C in one context is different to C in another is lessened by capturing different environments as different n-grams
  - The frequency distribution of particular runs of characters is explicitly captured in n-grams rather than relying on the NN to learn it
  - There is less to get wrong: instead of having to generate 40 characters, only 10-12 are needed
- Potential disadvantages:
  - There are more tokens whose meaning the NN must learn
- Note: no need to stop at 256 tokens
  - See related work by Xinhao Li and Denis Fourches: "SMILES Pair Encoding: A Data-Driven Substructure Tokenization Algorithm for Deep Learning". *JCIM,* **2021**, *61*, 1560.
    - 3002 n-grams
- I've added support for SmiZip to https://github.com/MorganCThomas/SMILES-RNN
  - Can train a RNN, sample from it, and use fine-tuning or reinforcement learning to guide samples towards a goal
  - "--native" option can be used to show the n-grams being generated

Molecular similarity

- Similarity by Compression - JL Melville, JF Riley, JD Hirst. *JCIM.* **2007**, *47*, 1, 25-33.
  - Online demo: Zipotron! https://comp.chem.nottingham.ac.uk/download/zippity/zipotron.html
  - Found via an old blog post of mine: "Parsing SMILES with a frown?" - https://baoilleach.blogspot.com/2007/06/parsing-smiles-with-frown.html
  - Journal club review by Rajarshi Guha (rescued by Wayback Machine): "Gzip for Molecular Similarity" https://web.archive.org/web/20070831201130/http://cheminfoclub.blogspot.com/2006/12/gzip-for-molecular-similarity.html
  - Response to Rajarshi's review by Andrew Dalke: Re: "Gzip for Molecular Similarity" http://www.dalkescientific.com/writings/diary/archive/2007/07/26/gzip_for_molecular_similarity.html
- From Rajarshi:
  - "The method is based on the normalized compression distance. This metric evaluates the distance between two objects by considering their compressed sizes, when considered individually and when joined (concatenated) together. The premise is that if two objects are similar, then their compressed versions will also be similar, such that the concatenated version will compress very efficiently."
- Could be investigated with SmiZip and my similarity benchmark
  - Perhaps derive tokens from a dataset all possible orderings of A, B and then from a dataset containing both
  - Would not be particularly fast unfortunately…

Acknowledgements

Acknowledgements

- Sosei Heptares
  - Chris de Graaf
- NextMove Software
  - Roger Sayle
  - John Mayfield
- Dalke Scientific
  - Andrew Dalke
- Code
  - https://github.com/SoseiHeptares.com/smizip
- Slides
  - https://github.com/SoseiHeptares.com/presentations

sosei HEPTARES

Backup slides

Is the chosen n-gram always the top priority?