# Neural Markov Logic Networks (Supplementary material)

**Giuseppe Marra**[1] **Ondřej Kuželka**[2]

[1]Department of Computer Science, KU Leuven, Leuven, Belgium
[2]Faculty of Electrical Engineering, Czech Technical University in Prague, Prague, Czech Republic

## A GRADIENT-BASED OPTIMIZATION

The maximization of the log-likelihood is carried out by a gradient-based optimization scheme. The gradients of the log-likelihood w.r.t. both the parameters $w_{i,j}$, where $w_{i,j}$ denotes the $j$-th component of $\mathbf{w}_i$, and $\beta_i$ are:

$$\frac{\partial \log(P_{\widehat{\omega}})}{\partial w_{i,j}} = \beta_i \left( \frac{\partial \Phi_i(\widehat{\omega}; \mathbf{w}_i)}{\partial w_{i,j}} - \mathbb{E}_{\omega \sim P} \left[ \frac{\partial \Phi(\omega; \mathbf{w}_i)}{\partial w_{i,j}} \right] \right) \tag{1}$$

$$\frac{\partial \log(P_{\widehat{\omega}})}{\partial \beta_i} = \left( \Phi_i(\widehat{\omega}; w_i) - \mathbb{E}_{\omega \sim P}[\Phi_i(\omega; w_i)] \right) \tag{2}$$

At a stationary point, Eq. 2 recovers the initial constraint on statistics imposed in the maximization of the entropy. However, the minimization of the entropy is mapped to a new requirement: at stationary conditions, the expected value of the gradients of the $\Phi_i$ under the distribution must match the gradients of the $\Phi_i$ evaluated at the data points.

## B TRANSLATING MARKOV LOGIC NETWORKS TO NEURAL MARKOV LOGIC NETWORKS

In this section we show that any Markov logic network (MLN) without quantifiers can be represented as an NMLN.

Kuželka et al. [2018] studies two maximum-entropy models, Model A, which is close to the model that we study in this paper, and Model B, which is the same as MLNs. Syntactically, both models are encoded as sets of quantifier-free weighted first order logic formulas, e.g. $\Phi = \{(\alpha_1, w_1), \ldots, (\alpha_M, w_m)\}$. In particular, given a positive integer $k$, Model A defines the following distribution:

$$p_A(\omega) = \frac{1}{Z} \exp\left( \sum_{(\alpha,w) \in \Phi_A} w \cdot \#_k(\alpha, \omega) \right)$$

where $Z$ is the normalization constant and $\#(\alpha, \omega)$ is the fraction of size-$k$ subsets $\mathscr{S}$ of constants in the possible world $\omega$ for which $\omega\langle\mathscr{S}\rangle \models \alpha$ (i.e. the formula $\alpha$ is classically true in the fragment of $\omega$ induced by $\mathscr{S}$). Let us first define

$$\phi_{\alpha,w}(\gamma) = \begin{cases} w & \gamma \models \alpha \\ 0 & \gamma \not\models \alpha \end{cases}$$

It is then easy to see that the distribution $p_A(\omega)$ can also easily be encoded as an NMLN by selecting the potential function $\phi(\gamma) = \sum_{(\alpha,w) \in \Phi_A} \phi_{\alpha,w}(\gamma)$ and by carefully selecting the weights $\beta_i$ in the NMLN.

Next we show that all distributions in Model B can be translated to distributions in Model A. First we will assume that the formulas $\alpha_i$ do not contain any constants.

*Accepted for the 37th Conference on Uncertainty in Artificial Intelligence* (UAI 2021).

Model B is given by

$$p_B(\omega) = \frac{1}{Z} \exp\left( \sum_{(\beta,v)\in\Phi_B} v \cdot n(\beta,\omega) \right)$$

where $n(\beta,\omega)$ is the number[1] of true injective groundings of the formula $\beta$ in the possible world $\omega$. Hence, Model B is exactly the same as Markov logic networks up to the requirement on injectivity of the groundings. However, as shown in [Buchman and Poole, 2015], any Markov logic network can be translated into such modified Markov logic network with the injectivity requirement on the groundings.

Let $k$ be an integer greater or equal to the number of variables in any formula in $\Phi_B$. Now, let $\Gamma$ be the set of all size-$k$ fragments. For every formula $\beta$ in $\Phi_B$, we introduce a partition $\mathscr{P}$ on $\Gamma$ induced by the equivalence relation $\sim_\beta$ defined by: $\gamma \sim_\beta \gamma'$ iff $n(\beta,\gamma) = n(\beta,\gamma')$. Since $\beta$ is assumed to not contain any constants, we can capture each of these equivalence classes $C$ by a (possibly quite big) first-order logic sentence without constants $\beta_C$. Let $C_i$ be the equivalence class that contains fragments $\gamma$ such that $n(\beta,\gamma) = i$. Let $m(\beta,\omega) = \sum_{C_i\in\mathscr{P}} \sum_{\gamma\in\Gamma_k(\omega)} i \cdot \mathbb{1}(\gamma \models \beta_C)$. By construction, it holds $m(\beta,\omega) = \sum_{\gamma\in\Gamma_k(\omega)} n(\beta,\gamma)$. Every true injective grounding of the formula $\beta$, having $l$ variables, is contained in $\binom{n-l}{k-l}$ different size-$k$ fragments of $\omega$, each of which gives rise to $k!$ anonymized fragments in the multi-set $\Gamma_k(\omega)$. So $m(\beta,\omega)$ is over-counting the number of true groundings $n(\beta,\omega)$ by a constant factor. It follows that, by carefully selecting the weights of the formulas $\beta_C$ we can encode the distribution $p_B(\omega)$ also in Model A. Although this particular transformation that we have just sketched is not very efficient, it does show that NMLNs with potential functions of width $k$ can express all distributions that can be expressed by MLNs containing formulas with at most $k$ variables and no existential quantifiers.

First-order logic formulas defining MLNs may also contain constants. In NMLNs we may represent constants using vector-space embeddings as described in the main text. One can then easily extend the argument sketched above to the case covering MLNs with constants.

## C THE MIN-MAX ENTROPY PROBLEM

As we show in this section, NMLNs naturally emerge from a principle of min-max entropy. For simplicity, we assume that we have one training example $\widehat{\omega}$ (the same arguments extend easily to the case of multiple training examples). The example $\widehat{\omega}$ may be, for instance, a knowledge graph (represented in first-order logic). We want to learn a distribution based on this example.[2] For that we exploit the principle of min-max entropy.

**Maximizing Entropy** Let us first assume that the potentials $\Phi_i(\omega;\mathbf{w}_i,\mathbf{W}_e)$ and the parameter vectors $\mathbf{w}_1, \ldots, \mathbf{w}_m$, and $\mathbf{W}_e$ are fixed. Using standard duality arguments for convex problems [Boyd and Vandenberghe, 2004, Wainwright et al., 2008], one can show that the solution of the following convex optimization problem is an NMLN:

$$\max_{P_\omega} -\sum_\omega P_\omega \log P_\omega \tag{3}$$

$$\text{s.t.} \quad \sum_\omega P_\omega = 1,\ \forall\omega\colon\ p_\omega \geq 0 \tag{4}$$

$$\forall i\colon\ \mathbb{E}_{P_\omega}[\Phi_i(\omega;\mathbf{w}_i,\mathbf{W}_e)] = \Phi_i(\widehat{\omega};\mathbf{w}_i,\mathbf{W}_e) \tag{5}$$

Here, the probability distribution is represented through the variables $P_\omega$, one for each possible world. This problem asks to find a maximum-entropy distribution such that the expected values of the potentials $\Phi_i$ are equal to the values of the same potentials on the training examples $\widehat{\omega}$, i.e. $\mathbb{E}_{P_\omega}[\Phi_i(\omega;\mathbf{w}_i,\mathbf{W}_e)] = \Phi_i(\widehat{\omega};\mathbf{w}_i,\mathbf{W}_e)$. One can use Lagrangian duality to obtain the solution in the form of an NMLN: $P_\omega = \frac{1}{Z}\exp\left(\sum_i \beta_i \Phi(\omega;\mathbf{w}_i,\mathbf{W}_e)\right))$. Here, the parameters $\beta_i$ are solutions of the dual problem $\max_{\beta_i}\{\sum_{i=1}^M \beta_i \Phi_i(\widehat{\omega};\mathbf{w}_i,\mathbf{W}_e)) - \log Z\}$, which coincides with maximum-likelihood when the domain size of the training example $\widehat{\omega}$ and the domain size of the modelled distribution are equal.[3] For details we refer to the discussion of

[1] In [Kuželka et al., 2018], Model B is defined using *fractions* of true grounding substitutions instead of *numbers* of true grounding substitutions. However, these two definitions are equivalent up to normalizations and both work for our purposes but the latter one is a bit more convenient here. Hence we choose the latter one here.

[2] Regarding consistency of learning SRL models from a single example, we refer e.g. to [Kuželka and Davis, 2019].

[3] We note that the derivation of the dual problem follows easily from the derivations in [Kuželka et al., 2018], which in turn rely on standard convex programming derivations from [Boyd and Vandenberghe, 2004, Wainwright et al., 2008]. Throughout this section we assume that a positive solution exists, which is needed for the strong duality to hold; this is later guaranteed by adding noise during learning.

model A in [Kuželka et al., 2018].

**Minimizing Entropy** Now we lift the assumption that the weights $\mathbf{w}$ and $\mathbf{W}_e$ are fixed. We learn them by minimizing the entropy of the max-entropy distribution. Let us denote by $H(\beta_1,\ldots,\beta_m,\mathbf{w}_1,\ldots,\mathbf{w}_m,\mathbf{W}_e)$ the entropy of the distribution $P_\omega$. We can write the resulting *min-max* entropy problem as:

$$\min_{\mathbf{w}_i,\mathbf{W}_e} \max_{\beta_i} H(\beta_1,\ldots,\beta_M,\mathbf{w}_1,\ldots,\mathbf{w}_m,\mathbf{W}_e) =$$
$$-\max_{\mathbf{w}_i,\mathbf{W}_e} \min_{\beta_i} -H(\beta_1,\ldots,\beta_M,\mathbf{w}_1,\ldots,\mathbf{w}_m,\mathbf{W}_e)$$

subject to the constraints (4) and (5). Plugging in the dual problem and using strong duality, we obtain the following unconstrained optimization problem which is equivalent to the maximization of log-likelihood when the domain size of the training example $\widehat{\omega}$ and that of the modelled distribution are the same, i.e. the optimization problem is:

$$\max_{\mathbf{w}_i,\mathbf{W}_e,\beta_i} \left\{ \sum_{i=1}^{m} \beta_i \Phi_i(\widehat{\omega};\mathbf{w}_i,\mathbf{W}_e) - \log Z \right\}. \tag{6}$$

The maximization of the log-likelihood will be carried out by a gradient-based method (see Appendix A).

**Justification** Selecting the potential in such a way as to decrease the (maximum) entropy of MaxEnt models can be shown to decrease the KL-divergence between the true data distribution and the model distribution [Zhu et al., 1997]. Of course, this coincides with maximum-likelihood estimation, but only when the domain size of the training example $\widehat{\omega}$ and that of the modelled distribution are the same; we refer to [Kuželka et al., 2018] for a more thorough discussion of this distinction for max-entropy models. Importantly, the min-max entropy perspective allows us to see the different roles that are played by the parameters $\beta_i$ on the one hand and $\mathbf{w}_i$ and $\mathbf{W}_e$ on the other.

# D GENERATING MOLECULES

## D.1 MOLECULES FIRST-ORDER-LOGIC REPRESENTATION

Even though molecules can be described with a high level of precision, using both spatial features (i.e. atoms distances, bond length etc.) and chemical features (i.e. atom charge, atom mass, hybridization), in this work, we focused only on structural symbolic descriptions of molecules.

In particular, we described a molecule using three sets of FOL predicates:

- *Atom-type unary predicates*: these are `C`, `N`, `O`, `S`, `Cl`, `F`, `P`.
- *Bond-type binary predicate*: these are `SINGLE` and `DOUBLE`.
- an auxiliary binary predicate `SKIPBOND` (see later).

An example of a molecule FOL description can be:

```
O(0), C(1), C(2), C(3), N(4), C(5), C(6), C(7), O(8), O(9), SINGLE(0,1), SINGLE(1,0), SINGLE(1,2),
SINGLE(2,1), SINGLE(2,3), SINGLE(3,2), SINGLE(3,4), SINGLE(4,3), SINGLE(4,5), SINGLE(5,4),
SINGLE(5,6), SINGLE(6,5), SINGLE(5,7), SINGLE(7,5), DOUBLE(7,8), DOUBLE(8,7), SINGLE(7,9),
SINGLE(9,7),SINGLE(6,1),SINGLE(1,6)
```

To help NMLNs capture long-range dependencies in molecules even with not so large fragments (e.g. $k = 3,4,5$), we added "skip-bond" atoms. For any three distinct $x$, $y$, $z$, such that there is a bond connecting $x$ and $y$ and a bond connecting $y$ and $z$, we add `SKIPBOND(x,z)`. This forces NMLN to learn the "definition" of skip-bonds and allows them, for instance, to implicitly capture the presence of a six-ring of atoms with a fragment of size 4 (whereas without skip-bonds we would need fragments of size 6 for that).

## D.2 FASTER INFERENCE WITH CHEMICAL CONSTRAINTS.

To speed up the convergence, we exploited a slightly modified Gibbs Sampling procedure, again inspired by the blocking technique. In particular, given a constant or a pair of constants, we sample all its relations in parallel. Since we known, that a

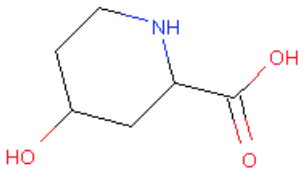



Figure 1: An example of molecule

constant should belong to *exactly one* unary relation, i.e. atom type, and a pair of constants should belong to *at most one* bond type, we can shorten the length of the chain by avoiding sampling inconsistent states. Inconsistent states are simply defined in NMLN as a set of constraints over the chain intermediate states. We also executed the same generation experiment without any constraint and there were no sensible differences.

We show a random sample of the training data (Figure 2) and the most frequent molecules sampled by GS (Figure 3).

### D.3 MLN COMPARISON

In order to compare with Markov Logic Network (MLN) , we first learned the structure of a MLN using Alchemy. Next, we used the trained MLN in using the same Gibbs Sampler we used for Neural MLNs, for a fairer comparison. This is the reason why MLNs are capable not to predict more than one type per atom/bond. Training the following rules required 11 hours, moreover we left Gibbs Sampling to burn for one hour before using sample for evaluation. On the contrary, training a NMLN required 1 hour (we trained and sample at the same time, but we started using samples for evaluation after one hour of training).

The trained MLN is showed below:

```
-0.000134051;c(a1)
-0.000134051;n(a1)
-0.000134051;o(a1)
-0.000134051;s(a1)
-9.65167e-05;p(a1)
-9.65165e-05;cl(a1)
-9.65166e-05;f(a1)
-0.000780653;double(a1,a2)
-0.000824121;single(a1,a2)
-0.000780779;skipBond(a1,a2)
-0.000206012;double(a1,a1)
-0.000199457;single(a1,a1)
-3.78726e-05;skipBond(a1,a1)
-0.0376156;p(a1) v cl(a1) v f(a2) v double(a3,a1) v single(a2,a1) v skipBond(a3,a1)
-0.0377639;p(a1) v cl(a1) v f(a2) v single(a1,a3) v single(a3,a2) v skipBond(a1,a2)
-0.0373712;double(a1,a2) v double(a2,a3) v double(a3,a1) v single(a1,a2) v
           skipBond(a1,a2) v skipBond(a3,a1)
-0.0377255;double(a1,a2) v double(a2,a3) v single(a2,a1) v single(a3,a2) v
           skipBond(a1,a2) v skipBond(a3,a1)
-0.0471552;c(a1) v n(a1) v o(a2) v s(a3) v double(a2,a1) v single(a1,a3)
-0.0375403;double(a1,a2) v double(a3,a1) v single(a3,a2) v skipBond(a1,a2) v
           skipBond(a2,a3) v skipBond(a3,a1)
```

```
-0.0378946;double(a1,a2) v single(a1,a3) v single(a3,a2) v skipBond(a1,a2) v
           skipBond(a2,a3) v skipBond(a3,a1)
-0.0373712;double(a1,a2) v double(a2,a3) v double(a3,a1) v single(a1,a3) v
           skipBond(a1,a2) v skipBond(a2,a3)
-0.0376156;p(a1) v cl(a1) v f(a2) v double(a2,a1) v single(a2,a1) v skipBond(a1,a3)
-0.0373712;double(a1,a2) v double(a2,a3) v double(a3,a1) v single(a3,a1) v
           skipBond(a1,a2) v skipBond(a2,a3)
-0.0377255;double(a1,a2) v double(a2,a3) v single(a1,a3) v single(a2,a1) v
           skipBond(a1,a2) v skipBond(a3,a1)
-0.0471552;c(a1) v n(a1) v o(a2) v s(a2) v double(a3,a1) v single(a1,a2)
-0.0471552;c(a1) v n(a2) v o(a3) v s(a2) v double(a3,a1) v single(a1,a2)
-0.0377255;double(a1,a2) v double(a2,a3) v single(a1,a3) v single(a3,a2) v
           skipBond(a2,a3) v skipBond(a3,a1)
-0.0377255;double(a1,a2) v double(a2,a3) v single(a1,a2) v single(a2,a3) v
           skipBond(a1,a2) v skipBond(a2,a3)
-0.0377255;double(a1,a2) v double(a3,a1) v single(a1,a3) v single(a3,a2) v
           skipBond(a1,a2) v skipBond(a2,a3)
-0.0377255;double(a1,a2) v double(a3,a1) v single(a1,a2) v single(a3,a1) v
           skipBond(a1,a2) v skipBond(a2,a3)
-0.0377255;double(a1,a2) v double(a3,a1) v single(a1,a3) v single(a3,a2) v
           skipBond(a1,a2) v skipBond(a3,a1)
-0.0376156;p(a1) v cl(a1) v f(a2) v double(a2,a1) v single(a1,a3) v skipBond(a3,a1)
-0.0375403;double(a1,a2) v double(a3,a1) v single(a2,a3) v skipBond(a1,a2) v
           skipBond(a2,a3) v skipBond(a3,a1)
-0.0377255;double(a1,a2) v double(a2,a3) v single(a2,a3) v single(a3,a1) v
           skipBond(a1,a2) v skipBond(a3,a1)
-0.0471552;c(a1) v n(a2) v o(a2) v s(a2) v double(a3,a1) v single(a1,a2)
-0.0375403;double(a1,a2) v double(a2,a3) v single(a3,a1) v skipBond(a1,a2) v
           skipBond(a2,a3) v skipBond(a3,a1)
-0.0380798;double(a1,a2) v single(a1,a3) v single(a2,a1) v single(a3,a2) v
           skipBond(a2,a3) v skipBond(a3,a1)
-0.0471552;c(a1) v n(a2) v o(a1) v s(a3) v double(a1,a2) v single(a2,a3)
-0.0378946;double(a1,a2) v single(a2,a1) v single(a3,a2) v skipBond(a1,a2) v
           skipBond(a2,a3) v skipBond(a3,a1)
-0.0377255;double(a1,a2) v double(a2,a3) v single(a1,a3) v single(a2,a1) v
           skipBond(a2,a3) v skipBond(a3,a1)
-0.0377255;double(a1,a2) v double(a2,a3) v single(a2,a1) v single(a3,a2) v
           skipBond(a2,a3) v skipBond(a3,a1)
-0.0339415;p(a1) v cl(a1) v f(a2) v single(a1,a3) v single(a3,a2) v skipBond(a1,a1)
-0.0471552;c(a1) v n(a2) v o(a1) v s(a2) v double(a1,a3) v single(a3,a2)
-0.0377255;double(a1,a2) v double(a3,a1) v single(a1,a3) v single(a2,a1) v
           skipBond(a1,a2) v skipBond(a3,a1)
-0.0377255;double(a1,a2) v double(a3,a1) v single(a2,a1) v single(a3,a2) v
           skipBond(a1,a2) v skipBond(a2,a3)
-0.0375403;double(a1,a2) v double(a2,a3) v single(a3,a2) v skipBond(a1,a2) v
           skipBond(a2,a3) v skipBond(a3,a1)
-0.0376156;p(a1) v cl(a1) v f(a2) v double(a2,a1) v single(a3,a2) v skipBond(a3,a1)
-0.0471552;c(a1) v n(a1) v o(a1) v s(a2) v double(a3,a1) v single(a1,a2)
-0.0377255;double(a1,a2) v double(a2,a3) v single(a1,a3) v single(a2,a1) v
           skipBond(a1,a2) v skipBond(a2,a3)
-0.0377255;double(a1,a2) v double(a2,a3) v single(a1,a3) v single(a3,a2) v
           skipBond(a1,a2) v skipBond(a3,a1)
-0.0376156;p(a1) v cl(a1) v f(a2) v double(a2,a1) v single(a3,a2) v skipBond(a3,a2)
-0.0376156;p(a1) v cl(a1) v f(a2) v double(a2,a3) v single(a3,a2) v skipBond(a3,a1)
-0.0377639;p(a1) v cl(a1) v f(a2) v single(a1,a3) v single(a2,a1) v skipBond(a1,a3)
```

```
-0.0377255;double(a1,a2) v double(a2,a3) v single(a1,a3) v single(a2,a1) v
           skipBond(a2,a1) v skipBond(a3,a2)
```

# E IMPLEMENTATION DETAILS

We provide details about hyperparameters exploited in the experiments described in the main text.

All the functions $\phi$ were implemented using feed-forward neural networks. No regularization technique is exploited on the weights or activations of the network (e.g. L2, dropout), even though, as highlighted in the main text, the addition of noise has a regularizing effect apart from avoiding NMLN to focus on deterministic rules.

When we searched over grids (grid-search), we list all the values and we show in bold the selected ones.

**NMLN (Smokers)**

- Network architecture: 1 hidden layer with 30 sigmoidal neurons.
- Number of parallel chains: 10
- $\pi_n$: 0.
- learning rates: [0.1, **0.01**, 0.001]

**NMLN-K3 (Nations)**

- Network architecture: 2 hidden layer with [150,**100**,75] and [**50**,25] [**ReLU**,sigmoid] neurons.
- Number of parallel chains: 10
- $\pi_n$: [**0.01**,0.02,0.03]
- learning rates: [$10^{-4}$,$\mathbf{10^{-5}}$,$10^{-6}$]

**NMLN-K3 (Kinship, UMLS)** We just run them on the best performing configuration on Nations.

**NMLN-K2E (Nations, Kinship, UMLS)**

- Network architecture: 2 hidden layer with 75 and 50 ReLU neurons.
- Number of parallel chains: 10
- $\pi_n$: [0.01,0.02,0.03]
- learning rates: [0.1, 0.01, 0.001]
- Embedding size: [10,20,30]
- Number of disconnected fragments per connected one: 2

For $\pi_n$, learning rate and embedding size the selected configurations are, respectively for each datasets: Nations (**0.02, 0.001, 20**), Kinship (**0.02, 0.1, 10**), UMLS (**0.02, 0.01, 30**)

**NMLN-K2E (WordNet, Freebase)**

- Network architecture: 3 hidden layer with 50,50 and 50 ReLU neurons.
- Number of parallel chains: 10
- $\pi_n$: [0.01,0.03, 0.04, 0.05, 0.1]
- learning rates: [0.1, 0.01, 0.001]
- Embedding size: [5,10,30]
- Number of disconnected fragments per connected one: [2,4,10]

For $\pi_n$, learning rate, embedding size and the number of disconnected fragments per connected one, the selected configurations are, respectively for each datasets: Wordnet (**0.05, 0.01, 10, 2**), Freebase (**0.04, 0.01, 10, 2**)

**Molecules generation**

- Network architecture: 2 hidden layer with 150 and 50 ReLU neurons.
- Number of parallel chains: **10**
- $\pi_n$: [0., **0.01**]
- learning rates: [**0.0001**]

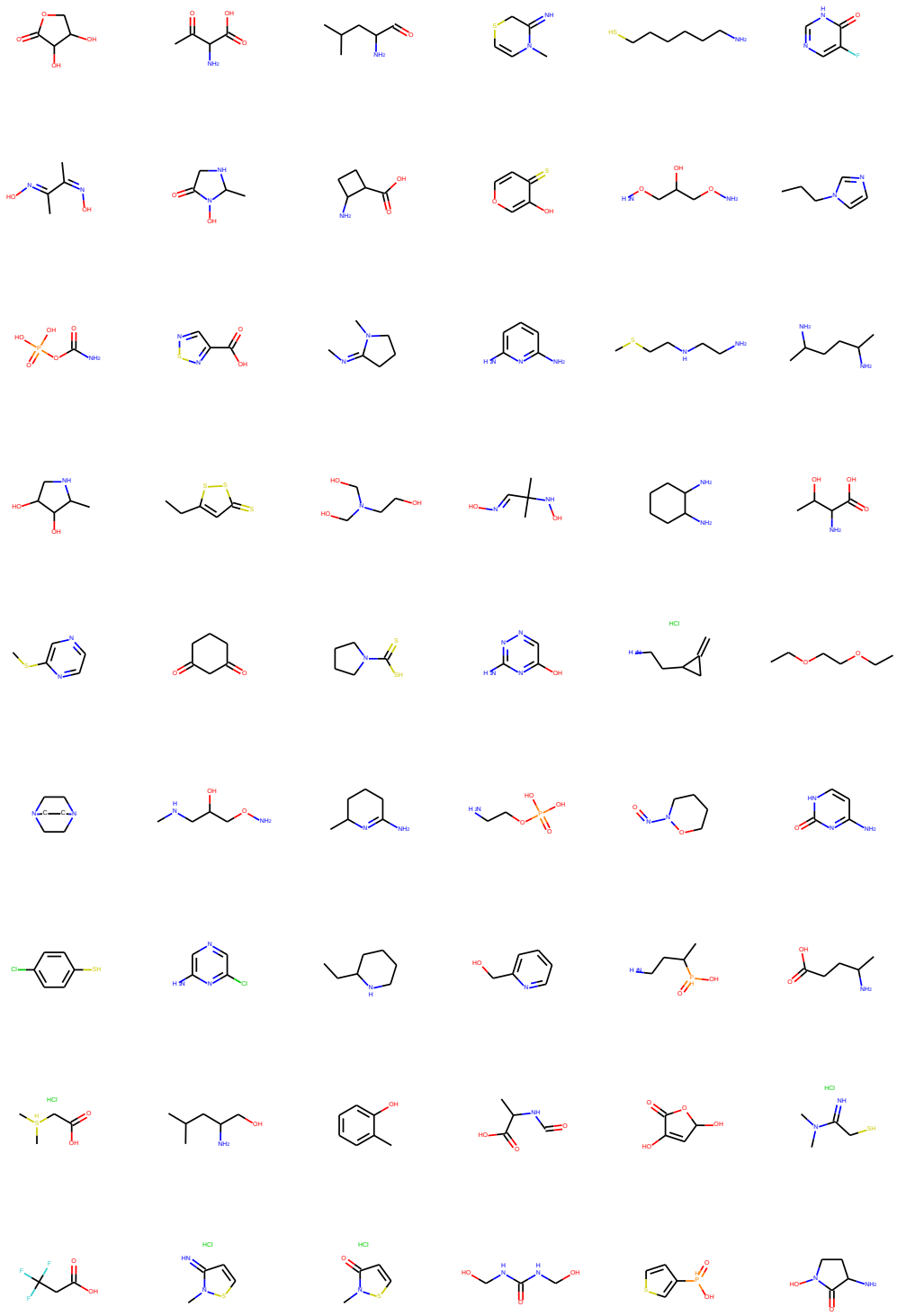


Figure 2: Molecules from the training data.

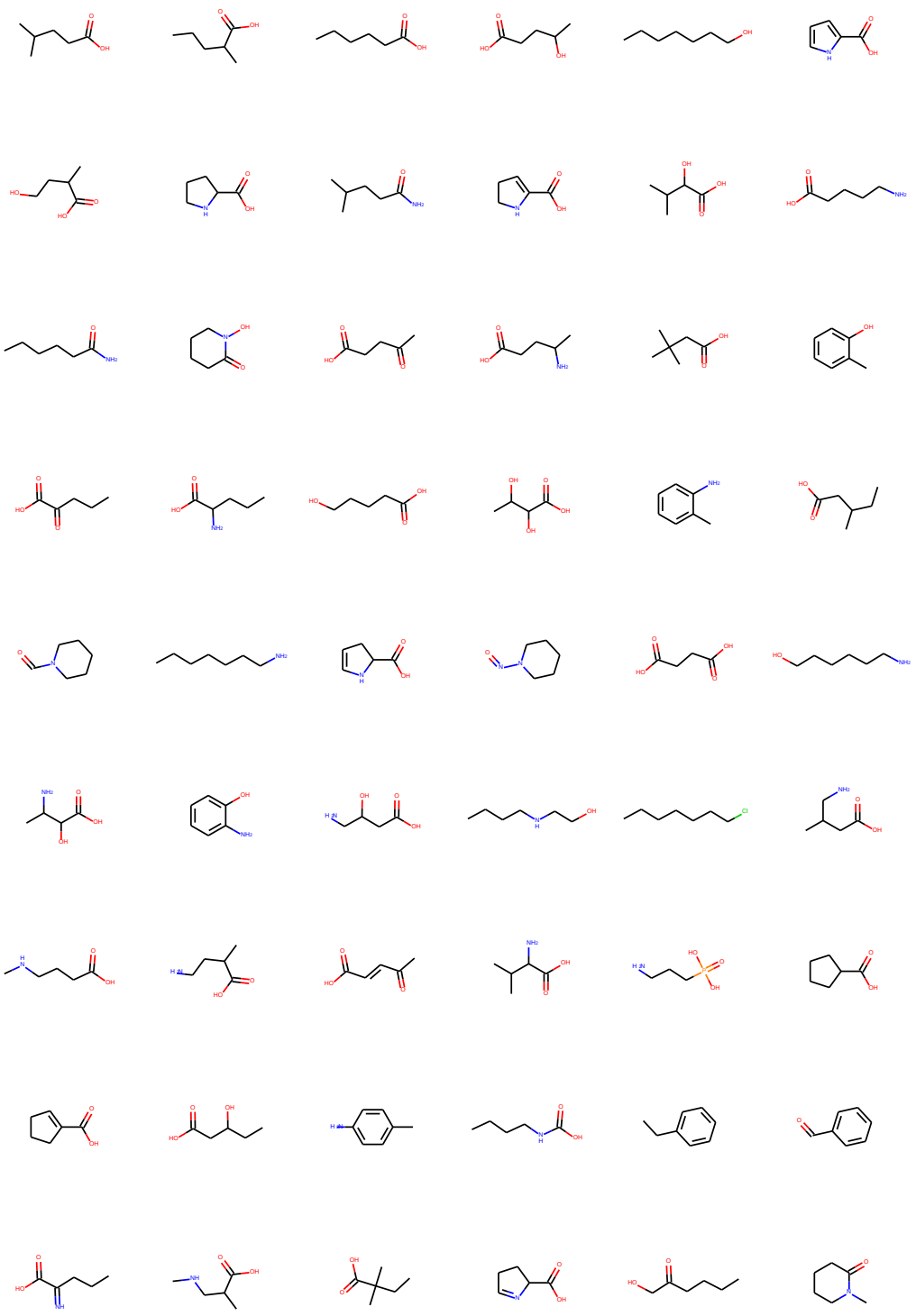


Figure 3: Most frequent generated molecules.

Table 1: Novel Molecules existing in ChemSpider

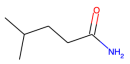
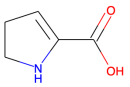
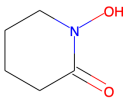
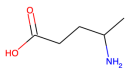
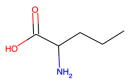
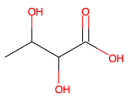
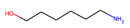
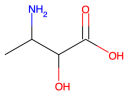
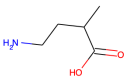
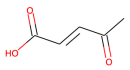
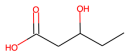
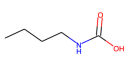
| Smile code | Molecule | Name |
|---|---|---|
| CC(C)CCC(N)=O |  | 4-Methylpentanamide |
| O=C(O)C1=CCCN1 |  | Multiple search results |
| O=C1CCCCN1O |  | 1-Hydroxy-2-piperidinone |
| CC(N)CCC(=O)O |  | 4-AMINOVALERIC ACID |
| CCCC(N)C(=O)O |  | DL-Norvaline |
| CC(O)C(O)C(=O)O |  | 4-DEOXYTETRONIC ACID |
| NCCCCCCO |  | MO8840000 |
| CC(N)C(O)C(=O)O |  | 3-amino-2-hydroxybutanoic acid |
| CC(CCN)C(=O)O |  | 4-Amino-2-methylbutanoic acid |
| CC(=O)C=CC(=O)O |  | 4-Oxo-2-pentenoic acid |
| CCC(O)CC(=O)O |  | 3-Hydroxyvaleric acid |
| CCCCNC(=O)O |  | Butylcarbamic acid |

Table 1 – *Continued from previous page*

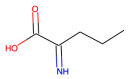
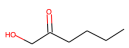
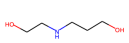
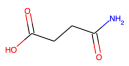
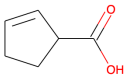
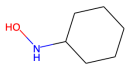
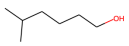
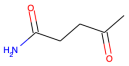
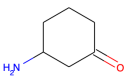
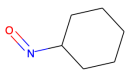
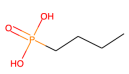
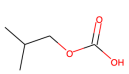
| Smile code | Molecule | Name |
|---|---|---|
| CCCC(=N)C(=O)O |  | Multiple search results |
| CCCCC(=O)CO |  | 1-Hydroxy-2-hexanone |
| OCCCNCCO |  | 3-((2-Hydroxyethyl)amino)propanol |
| NC(=O)CCC(=O)O |  | 4-Amino-4-oxobutanoic acid |
| O=C(O)C1C=CCC1 |  | 2-Cyclopentenecarboxylic acid |
| ONC1CCCCC1 |  | NC3410400 |
| CC(C)CCCCO |  | 5-Methyl-1-hexanol |
| CC(=O)CCC(N)=O |  | 4-Oxopentanamide |
| NC1CCCC(=O)C1 |  | 3-Aminocyclohexanone |
| O=NC1CCCCC1 |  | Nitrosocyclohexane |
| CCCCP(=O)(O)O |  | Butylphosphonate |
| CC(C)COC(=O)O |  | Isobutyl hydrogen carbonate |

Table 1 – *Continued from previous page*

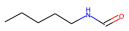
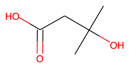
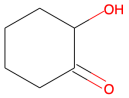
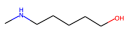
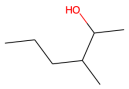
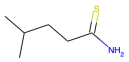
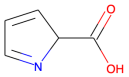
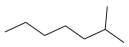
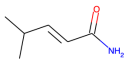
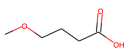
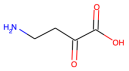
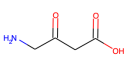
| Smile code | Molecule | Name |
|---|---|---|
| CCCCCNC=O |  | MFCD07784338 |
| CC(C)(O)CC(=O)O |  | 3-OH-isovaleric acid |
| O=C1CCCCC1O |  | Adipoin |
| CNCCCCCO |  | MFCD16696454 |
| CCCC(C)C(C)O |  | MFCD00021889 |
| CC(C)CCC(N)=S |  | 4-Methylpentanethioamide |
| O=C(O)C1C=CC=N1 |  | Multiple search results |
| CCCCCC(C)C |  | 2-Methylheptane |
| CC(C)C=CC(N)=O |  | 4-Methyl-2-pentenamide |
| COCCCC(=O)O |  | 4-Methoxybutanoic acid |
| NCCC(=O)C(=O)O |  | 4-Amino-2-oxobutanoic acid |
| NCC(=O)CC(=O)O |  | 4-Amino-3-oxobutanoic acid |

Table 1 – *Continued from previous page*

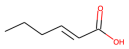
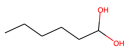
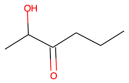
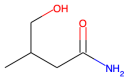
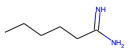
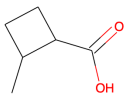
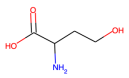
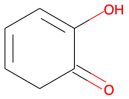
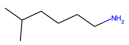
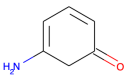
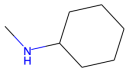
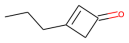
| Smile code | Molecule | Name |
|---|---|---|
| CCCC=CC(=O)O |  | 2-Hexenoic acid |
| CCCCCC(O)O |  | Hexanediol |
| CCCC(=O)C(C)O |  | 2-Hydroxy-3-hexanone |
| CC(CO)CC(N)=O |  | 4-Hydroxy-3-methylbutanamide |
| CCCCCC(=N)N |  | Hexanamidine |
| CC1CCC1C(=O)O |  | 2-Methylcyclobutanecarboxylic acid |
| NC(CCO)C(=O)O |  | Homoserine |
| O=C1CC=CC=C1O |  | Multiple search results |
| CC(C)CCCCN |  | 5-Methyl-1-hexanamine |
| NC1=CC=CC(=O)C1 |  | Multiple search results |
| CNC1CCCCC1 |  | GX1529000 |
| CCCC1=CC(=O)C1 |  | 3-Propyl-2-cyclobuten-1-one |

Table 1 – *Continued from previous page*

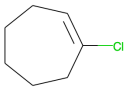
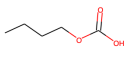
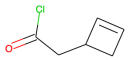
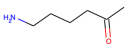
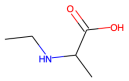
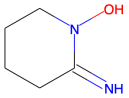
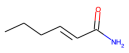
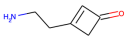
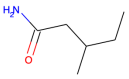
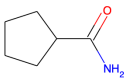
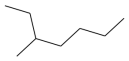
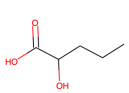
| Smile code | Molecule | Name |
|---|---|---|
| ClC1=CCCCCC1 |  | 1-Chlorocycloheptene |
| CCCCOC(=O)O |  | monobutyl carbonate |
| O=C(Cl)CC1C=CC1 |  | Cyclobutylideneacetyl chloride |
| CC(=O)CCCCN |  | 6-Amino-2-hexanone |
| CCNC(C)C(=O)O |  | N-Ethylalanine |
| N=C1CCCCN1O |  | 2-Imino-1-piperidinol |
| CCCC=CC(N)=O |  | (2E)-2-Hexenamide |
| NCCC1=CC(=O)C1 |  | Multiple search results |
| CCC(C)CC(N)=O |  | 3-Methylpentanamide |
| NC(=O)C1CCCC1 |  | Cyclopentanecarboxamide |
| CCCCC(C)CC |  | 3-Methylheptane |
| CCCC(O)C(=O)O |  | 2-hydroxyvaleric acid |

Table 1 – *Continued from previous page*

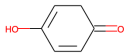
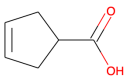
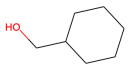
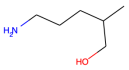
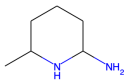
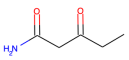
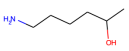
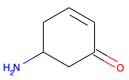
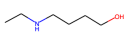
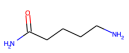
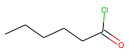
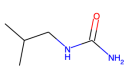
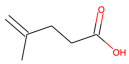
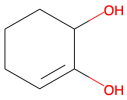
| Smile code | Molecule | Name |
|---|---|---|
| O=C1C=CC(O)=CC1 |  | 4-(2H)Hydroxy-2,4-cyclohexadien-1-one |
| O=C(O)C1CC=CC1 |  | 3-Cyclopentenecarboxylic Acid |
| OCC1CCCCC1 |  | Cyclohexylmethanol |
| CC(CO)CCCN |  | 5-Amino-2-methyl-1-pentanol |
| CC1CCCC(N)N1 |  | 6-Methyl-2-piperidinamine |
| CCC(=O)CC(N)=O |  | 3-Oxopentanamide |
| CC(O)CCCCN |  | 6-Amino-2-hexanol |
| NC1CC=CC(=O)C1 |  | Multiple search results |
| CCNCCCCO |  | 4-(Ethylamino)-1-butanol |
| NCCCCC(N)=O |  | 5-Aminopentanamide |
| CCCCCC(=O)Cl |  | 506332 |
| CC(C)CNC(N)=O |  | Isobutylurea |

Table 1 – *Continued from previous page*

| Smile code | Molecule | Name |
|---|---|---|
| C=C(C)CCC(=O)O |  | 4-Methyl-4-pentenoic acid |
| OC1=CCCCC1O |  | Multiple search results |

## References


Stephen Boyd and Lieven Vandenberghe. *Convex optimization*. Cambridge university press, 2004.

David Buchman and David Poole. Representing aggregators in relational probabilistic models. In *Twenty-Ninth AAAI Conference on Artificial Intelligence*, 2015.

Ondřej Kuželka and Jesse Davis. Markov logic networks for knowledge base completion: A theoretical analysis under the MCAR assumption. In *Proceedings of the Thirty-Fifth Conference on Uncertainty in Artificial Intelligence, UAI*, 2019.

Ondřej Kuželka, Yuyi Wang, Jesse Davis, and Steven Schockaert. Relational marginal problems: Theory and estimation. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

Martin J Wainwright, Michael I Jordan, et al. Graphical models, exponential families, and variational inference. *Foundations and Trends® in Machine Learning*, 1(1–2):1–305, 2008.

Song Chun Zhu, Ying Nian Wu, and David Mumford. Minimax entropy principle and its application to texture modeling. *Neural computation*, 9(8):1627–1660, 1997.