mol2vec Documentation

Release 0.1

Samo Turk, Sabrina Jaeger, Simone Fulle

Jun 23, 2018
## Contents

1 Installation &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; 3
2 Usage &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; 5
3 How to cite? &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; 7
4 API documentation &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; 9
5 Indices and tables &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; 15
Mol2vec - an unsupervised machine learning approach to learn vector representations of molecular substructures

Contents

- Mol2vec documentation
  - Installation
    * Requirements
    * Building the documentation
  - Usage
    * As python module
    * Command line application
  - How to cite?
  - API documentation
    * Features - Main Mol2vec Module
    * Helpers - Mostly plotting functions
  - Indices and tables
CHAPTER 1

Installation

```
$ pip install git+https://github.com/samoturk/mol2vec
```

**Note:** RDKit has to be installed manually and is not automatically installed by pip as a dependency.

### 1.1 Requirements

- Python 3 (Python 2.x is not supported)
- NumPy
- matplotlib
- seaborn
- pandas
- IPython
- RDKit
- scikit-learn
- gensim
- tqdm
- joblib

### 1.2 Building the documentation

To build the documentation install *sphinx, numpydoc* and *sphinx_rtd_theme* and then run in the docs directory:
make html
2.1 As python module

```python
from mol2vec import features
from mol2vec import helpers
```

First line imports functions to generate “sentences” from molecules and train the model, and second line imports functions useful for depictions. Check examples directory for more details and Mol2vec notebooks repository for visualisations made to easily run in Binder.

2.2 Command line application

Command line application has subcommands to prepare a corpus from molecular data (SDF or SMILES), train Mol2vec model and featurize new samples. To get help from Mol2vec command line application:

```bash
mol2vec --help
```

For more detail on individual subcommands run:

```bash
mol2vec $sub-command --help
```

2.2.1 Subcommand ‘corpus’

Generates corpus to train Mol2vec model. It generates morgan identifiers (up to selected radius) which represent words (molecules are sentences). Words are ordered in the sentence according to atom order in canonical SMILES (generated when generating corpus) and at each atom starting by identifier at radius 0. Corpus subcommand also optionally replaces rare identifiers with selected string (e.g. UNK) which can be later used to represent completely new substructures (i.e. at featurization step). NOTE: It saves the corpus with replaced uncommon identifiers in separate file with ending “_{selected string to replace uncommon}”’. Since this is unsupervised method we recommend using as much molecules as possible (e.g. complete ZINC database).
Note: Corpus generation using 20M compounds with replacement of uncommon identifiers takes 6 hours on 4 cores.

To prepare a corpus using radius 1, 4 cores, replace uncommon identifiers that appear <= 3 times with ‘UNK’ run:

```
mol2vec corpus -i mols.smi -o mols.cp -r 1 -j 4 --uncommon UNK --threshold 3
```

### 2.2.2 Subcommand ‘train’

Trains Mol2vec model using previously prepared corpus.

**Note:** Training the model on 20M sentences takes ~2 hours on 4 cores.

To train a Mol2vec model on corpus with replaced uncommon identifiers using Skip-gram, window size 10, generating 300 dimensional vectors and using 4 cores run:

```
mol2vec train -i mols.cp_UNK -o model.pkl -d 300 -w 10 -m skip-gram --threshold 3 -j 4
```

### 2.2.3 Subcommand ‘featurize’

Featurizes new samples using pre-trained Mol2vec model. It saves the result in CSV file with columns for molecule identifiers, canonical SMILES (generated during featurization) and all potential SD fields from input SDF file and finally followed by mol2vec-{0 to n-1} where n is dimensionality of embeddings in the model.

To featurize new samples using pre-trained embeddings and using vector trained on uncommon samples to represent new substructures:

```
mol2vec featurize -i new.smi -o new.csv -m model.pkl -r 1 --uncommon UNK
```
How to cite?

@article{doi:10.1021/acs.jcim.7b00616,
    author = {Jaeger, Sabrina and Fulle, Simone and Turk, Samo},
    title = {Mol2vec: Unsupervised Machine Learning Approach with Chemical Intuition},
    journal = {Journal of Chemical Information and Modeling},
    volume = {0},
    number = {ja},
    pages = {null},
    year = {0},
    doi = {10.1021/acs.jcim.7b00616},
    URL = {http://dx.doi.org/10.1021/acs.jcim.7b00616},
    eprint = {http://dx.doi.org/10.1021/acs.jcim.7b00616}
}
4.1 Features - Main Mol2vec Module

```python
class mol2vec.features.DfVec(vec)
    Helper class to store vectors in a pandas DataFrame

    Parameters
    vec: np.array

class mol2vec.features.MolSentence(sentence)
    Class for storing mol sentences in pandas DataFrame

    Methods

    contains(word)
        Contains (and __contains__) method enables usage of ‘Word’ in MolSentence

    contains(word)
        Contains (and __contains__) method enables usage of ‘Word’ in MolSentence

mol2vec.features.featureize(in_file, out_file, model_path, r, uncommon=None)
    Featurize mols in SDF, SMI. SMILES are regenerated with RDKit to get canonical SMILES without chirality information.

    Parameters
    in_file [str] Input SDF, SMI, ISM (or GZ)
    out_file [str] Output csv
    model_path [str] File path to pre-trained Gensim word2vec model
    r [int] Radius of morgan fingerprint
```
uncommon [str] String to used to replace uncommon words/identifiers while training. Vector obtained for ‘uncommon’ will be used to encode new (unseen) identifiers

mol2vec.features.generate_corpus (in_file, out_file, r, sentence_type='alt', n_jobs=1)
Generates corpus file from sdf

Parameters

in_file [str] Input sdf
out_file [str] Outfile name prefix, suffix is either _r0, _r1, etc. or _alt_r1 (max radius in alt sentence)
r [int] Radius of morgan fingerprint
sentence_type [str]
   Options: ‘all’ - generates all corpus files for all types of sentences, ‘alt’ - generates a corpus file with only combined alternating sentence, ‘individual’ - generates corpus files for each radius
n_jobs [int] Number of cores to use (only ‘alt’ sentence type is parallelized)

mol2vec.features.insert_unk (corpus, out_corpus, threshold=3, uncommon='UNK')
Handling of uncommon “words” (i.e. identifiers). It finds all least common identifiers (defined by threshold) and replaces them by ‘uncommon’ string.

Parameters

corpus [str] Input corpus file
out_corpus [str] Outfile corpus file
threshold [int] Number of identifier occurrences to consider it uncommon
uncommon [str] String to use to replace uncommon words/identifiers

mol2vec.features.mol2alt_sentence (mol, radius)
Same as mol2sentence() expect it only returns the alternating sentence Calculates ECFP (Morgan fingerprint) and returns identifiers of substructures as ‘sentence’ (string). Returns a tuple with 1) a list with sentence for each radius and 2) a sentence with identifiers from all radii combined. NOTE: Words are ALWAYS reordered according to atom order in the input mol object. NOTE: Due to the way how Morgan FPs are generated, number of identifiers at each radius is smaller

Parameters

mol [rdkit.Chem.rdchem.Mol]
radius [float] Fingerprint radius

Returns

list alternating sentence
combined

mol2vec.features.mol2sentence (mol, radius)
Calculates ECFP (Morgan fingerprint) and returns identifiers of substructures as ‘sentence’ (string). Returns a tuple with 1) a list with sentence for each radius and 2) a sentence with identifiers from all radii combined. NOTE: Words are ALWAYS reordered according to atom order in the input mol object. NOTE: Due to the way how Morgan FPs are generated, number of identifiers at each radius is smaller

Parameters

mol [rdkit.Chem.rdchem.Mol]
radius [float] Fingerprint radius

Returns

identifier sentence List with sentences for each radius

alternating sentence Sentence (list) with identifiers from all radii combined

mol2vec.features.remove_salts_solvents(smiles, hac=3)

Remove solvents and ions have max ‘hac’ heavy atoms. This function removes any fragment in molecule that has number of heavy atoms <= “hac” and it might not be an actual solvent or salt

Parameters

smiles [str] SMILES

ahac [int] Max number of heavy atoms

Returns

str smiles

mol2vec.features.sentences2vec(sentences, model, unseen=None)

Generate vectors for each sentence (list) in a list of sentences. Vector is simply a sum of vectors for individual words.

Parameters

sentences [list, array] List with sentences

model [word2vec.Word2Vec] Gensim word2vec model


Returns

np.array

mol2vec.features.train_word2vec_model(infile_name, outfile_name=None, vector_size=100, window=10, min_count=3, n_jobs=1, method='skip-gram', **kwargs)

Trains word2vec (Mol2vec, ProtVec) model on corpus file extracted from molecule/protein sequences. The corpus file is treated as LineSentence corpus (one sentence = one line, words separated by whitespaces)

Parameters

infile_name [str] Corpus file, e.g. proteins split in n-grams or compound identifier

outfile_name [str] Name of output file where word2vec model should be saved

vector_size [int] Number of dimensions of vector

window [int] Number of words considered as context

min_count [int] Number of occurrences a word should have to be considered in training

n_jobs [int] Number of cpu cores used for calculation

method [str] Method to use in model training. Options cbow and skip-gram, default: skip-gram

Returns

word2vec.Word2Vec
4.2 Helpers - Mostly plotting functions

mol2vec.helpers.depict_atoms(mol, atom_ids, radii, molSize=(300, 300), atm_color=(0, 1, 0), oth_color=(0.8, 1, 0))

Get a depiction of molecular substructure. Useful for depicting bits in fingerprints.
Inspired by: http://rdkit.blogspot.ch/2016/02/morgan-fingerprint-bit-statistics.html

Parameters
- mol [rdkit.Chem.rdchem.Mol] RDKit molecule
- atom_ids [list] List of atoms to depict
- radii [list] List of radii - how many atoms around each atom with atom_id to highlight
- molSize [tuple]
- atm_color, oth_color [tuple] Colors of central atoms and surrounding atoms and bonds

Returns
- IPython.display.SVG

mol2vec.helpers.depict_identifier(mol, identifier, radius, useFeatures=False, **kwargs)

Depict an identifier in Morgan fingerprint.

Parameters
- mol [rdkit.Chem.rdchem.Mol] RDKit molecule
- identifier [int or str] Feature identifier from Morgan fingerprint
- radius [int] Radius of Morgan FP
- useFeatures [bool] Use feature-based Morgan FP

Returns
- IPython.display.SVG

mol2vec.helpers.mol_to_svg(mol, molSize=(300, 300), kekulize=True, drawer=None, font_size=0.8, **kwargs)

Generates a SVG from mol structure.
Inspired by: http://rdkit.blogspot.ch/2016/02/morgan-fingerprint-bit-statistics.html

Parameters
- mol [rdkit.Chem.rdchem.Mol]
- molSize [tuple]
- kekulize [bool]
- drawer [funct] Specify which drawing function to use (default: rdMolDraw2D.MolDraw2DSVG)
- font_size [float] Atom font size

Returns
- IPython.display.SVG
mol2vec.helpers.plot_2D_vectors(vectors, sumup=True, min_max_x=None, min_max_y=None, cmap=<MagicMock name='mock.pyplot.cm.viridis_r' id='140450684450688'>, colors=None, vector_labels=None, ax=None)

Plots 2d vectors by adding them in sequence and transposing them.

Parameters

- **vectors** [list] 2D vectors eg: [[0,1], [3,4]]
- **sumup** [bool] Show a vector that represents a sum of vectors
- **min_max_{x,y}** [tuple] min and max of {x,y} axis
- **cmap** [plt.cm] Default: plt.cm.viridis_r
- **colors** [list] List of matplotlib colors. Number of colors has to match number of vectors (including sum vector if sumup=True). Default=None selects colors from cmap
- **vector_labels** [list] Has to match number of vectors (including sum vector if sumup=True)
- **ax** [plt.ax] Name of axis to plot to

Returns

- **——**
- **plt.figure()**

mol2vec.helpers.plot_class_distribution(df, x_col, y_col, c_col, ratio=0.1, n=1, marker='o', alpha=1, x_label='auto', y_label='auto', cmap=<MagicMock name='mock.pyplot.cm.viridis' id='140450684442328'>, size=(8, 8), share_axes=False)

Scatter + histogram plots of x and y, e.g. after t-SNE dimensionality reduction. Colors are wrong in scatter plot if len(class) < 4. Open issue in matplotlib. (See: https://github.com/pandas-dev/pandas/issues/9724)

Parameters

- **df** [pd.DataFrame] Datframe with our data
- **{x,y}_col** [str] Name of a column with {x,y} values
- **c_col** [str] Name of a column with classes (basis for hue)
- **ratio** [float] Ratio to determine empty space of limits of x/y-axis
- **marker** [str] Marker in scatter plot
- **n** [float] Number of columns of legend
- **alpha** [float] Alpha for scatter plot
- **x_label** [str] Label of x-axis, default auto: x_col name
- **y_label** [str] Label of y-axis, default auto: y_col name
- **cmap** [matplotlib.colors.ListedColormap]
- **size** [tuple]
CHAPTER 5

Indices and tables

• genindex
• modindex
Index

C
contains() (mol2vec.features.MolSentence method), 9

D
depict_atoms() (in module mol2vec.helpers), 12
depict_identifier() (in module mol2vec.helpers), 12
DfVec (class in mol2vec.features), 9

F
featurize() (in module mol2vec.features), 9

G
generate_corpus() (in module mol2vec.features), 10

I
insert_unk() (in module mol2vec.features), 10

M
mol2alt_sentence() (in module mol2vec.features), 10
mol2sentence() (in module mol2vec.features), 10
mol2vec.features (module), 9
mol2vec.helpers (module), 11
mol_to_svg() (in module mol2vec.helpers), 12
MolSentence (class in mol2vec.features), 9

P
plot_2D_vectors() (in module mol2vec.helpers), 12
plot_class_distribution() (in module mol2vec.helpers), 13

R
remove_salts_solvents() (in module mol2vec.features), 11

S
sentences2vec() (in module mol2vec.features), 11

T
train_word2vec_model() (in module mol2vec.features), 11