dscomp2019 project list

June 6th, 2019

1 Evaluate different types of distributional representations on a semantic composition task

Pre-processed German and English corpora (specified in Section 2) were used to train different types of word representations. The resulting word representations can be downloaded from:

`http://shaw.sfs.uni-tuebingen.de:7070/embeddings/`.

Evaluate the representations: (i) *intrinsically*, on word analogy/word similarity tasks, (ii) *extrinsically*, on noun compound composition. To make the experiments comparable (per language), some parameters are fixed: the size of the word vectors: 300; the context size: 10, symmetric (20 words around the target word); the minimum count of a word: 50.

Both English and German word representations are trained on *word forms*. For German, the embedding vocabulary contains 300,433 words, and there are 646M words in the training file. The English corpus was additionally preprocessed to include the noun compounds in the composition dataset (specified in Section 2), resulting in a vocabulary of 364,592 words (and 1,832M words in the training file).

All the word representations are saved in the `word2vec` binary format. If you need to convert between the binary and the text format, you can use the `convertvec` utility[1]. In Python you can load both formats using the `gensim` library[2]. The intrinsic evaluation should be done using `gensim` - methods `evaluate_word_pairs` and `evaluate_word_analogies` from `gensim.models.KeyedVectors`. The datasets for intrinsic evaluation are available in the `projects`[3] GitHub repository.

[1] `https://github.com/marekrei/convertvec`
[2] `https://radimrehurek.com/gensim/models/keyedvectors.html`
[3] `https://github.com/dscomp2019/projects`

Project 1: Test word2vec Skip-Gram negative sampling (SG-NEG) with and without subsampling for German.

Project aim: quantify the impact of subsampling when creating word representations with `word2vec`. Subsamplig gives the model access to a different context - since it downsamples the frequent words, and upsamples the less frequent ones. Word representations train significantly faster when trained *with* subsampling - but are the resulting representations better or worse when applied to a downstream task like composition?

Word representations:

- `de.wiki.sg.ng5.no_subs.bin`
- `de.wiki.sg.ng5.1e-5_subs.bin`

Both word spaces were trained using `skip-gram` with 5 negative samples. The difference is that `de.wiki.sg.ng5.no_subs.bin` was trained with no subsampling, while `de.wiki.sg.ng5.1e-5_subs.bin` was trained with subsampling using $t = 1e - 5$.

You should also pick a third file from the available embedding spaces and compare to it as well (e.g. pick a GloVe or a fastText embedding set).

Evaluate the word representations intrinsically, on the datasets proposed by Köper et al. [2015], `https://www.ims.uni-stuttgart.de/forschung/ressourcen/lexika/analogies_ims/analogies.en.html` for:

- word similarity (`de_re-rated_Schm280_tabs.txt`)
- word analogy (`de_trans_Google_analogies.txt`, `de_sem-para_SemRel.txt`)

Evaluate the representations also extrinsically, on the noun compound composition task. You should evaluate on the `deu-nn` dataset provided by Dima et al. [2019], `https://talar.sfb833.uni-tuebingen.de:8443/erdora/cmdi/SFB833/A03/TACL_datasets/deu-nn`.

Evaluate on at least four composition models: `addition`, `matrix`, `fulllex` and `transweight`. Use the Python/Tensorflow implementations provided in `commix`, `https://github.com/sfb833-a3/commix`.

Project 2: Test word2vec Skip-Gram hierarchical softmax (SG-HS) with and without subsampling for German.

See project aim from Project 1. Word representations:

- `de.wiki.sg.hs.no_subs.bin`
- `de.wiki.sg.hs.1e-5_subs.bin`

Both word spaces were trained using `skip-gram` with hierarchical softmax. The difference is that `de.wiki.sg.hs.no_subs.bin` was trained with no subsampling, while `de.wiki.sg.hs.1e-5_subs.bin` was trained with subsampling using $t = 1e - 5$.

You should also pick a third file from the available embedding spaces and compare to it as well (e.g. pick a GloVe or a fastText embedding set).

Evaluate the word representations intrinsically, on the datasets proposed by Köper et al. [2015], `https://www.ims.uni-stuttgart.de/forschung/ressourcen/lexika/analogies_ims/analogies.en.html` for:

- word similarity (`de_re-rated_Schm280_tabs.txt`)
- word analogy (`de_trans_Google_analogies.txt`, `de_sem-para_SemRel.txt`)

Evaluate the representations also extrinsically, on the noun compound composition task. You should evaluate on the `deu-nn` dataset provided by Dima et al. [2019], `https://talar.sfb833.uni-tuebingen.de:8443/erdora/cmdi/SFB833/A03/TACL_datasets/deu-nn`.

Evaluate on at least four composition models: `addition`, `matrix`, `fulllex` and `transweight`. Use the Python/Tensorflow implementations provided in `commix`, `https://github.com/sfb833-a3/commix`.

Project 3: Test word2vec CBOW negative sampling (CBOW-NEG) with and without subsampling for German.

See project aim from Project 1. Word representations:

- `de.wiki.cbow.ng5.no_subs.bin`
- `de.wiki.cbow.ng5.1e-5_subs.bin`

Both word spaces were trained using `cbow` with 5 negative samples. The difference is that `de.wiki.cbow.ng5.no_subs.bin` was trained with no subsampling, while `de.wiki.cbow.ng5.1e-5_subs.bin` was trained with subsampling using $t = 1e - 5$.

You should also pick a third file from the available embedding spaces and compare to it as well (e.g. pick a GloVe or a fastText embedding set).

Evaluate the word representations intrinsically, on the datasets proposed by Köper et al. [2015], `https://www.ims.uni-stuttgart.de/forschung/ressourcen/lexika/analogies_ims/analogies.en.html` for:

- word similarity (`de_re-rated_Schm280_tabs.txt`)
- word analogy (`de_trans_Google_analogies.txt`, `de_sem-para_SemRel.txt`)

Evaluate the representations also extrinsically, on the noun compound composition task. You should evaluate on the `deu-nn` dataset provided by Dima et al. [2019], `https://talar.sfb833.uni-tuebingen.de:8443/erdora/cmdi/SFB833/A03/TACL_datasets/deu-nn`.

Evaluate on at least four composition models: `addition`, `matrix`, `fulllex` and `transweight`. Use the Python/Tensorflow implementations provided in `commix`, `https://github.com/sfb833-a3/commix`.

Project 4: Test word2vec CBOW hierarchical softmax (CBOW-HS) with and without subsampling for German.

See project aim from Project 1. Word representations:

- `de.wiki.cbow.hs.no_subs.bin`
- `de.wiki.cbow.hs.1e-5_subs.bin`

Both word spaces were trained using `cbow` with hierarchical softmax. The difference is that `de.wiki.cbow.hs.no_subs.bin` was trained without subsampling, while `de.wiki.cbow.hs.1e-5_subs.bin` was trained with subsampling using $t = 1e - 5$.

You should also pick a third file from the available embedding spaces and compare to it as well (e.g. pick a GloVe or a fastText embedding set).

Evaluate the word representations intrinsically, on the datasets proposed by Köper et al. [2015], `https://www.ims.uni-stuttgart.de/forschung/ressourcen/lexika/analogies_ims/analogies.en.html` for:

- word similarity (`de_re-rated_Schm280_tabs.txt`)
- word analogy (`de_trans_Google_analogies.txt`, `de_sem-para_SemRel.txt`)

Evaluate the representations also extrinsically, on the noun compound composition task. You should evaluate on the `deu-nn` dataset provided by Dima et al. [2019], `https://talar.sfb833.uni-tuebingen.de:8443/erdora/cmdi/SFB833/A03/TACL_datasets/deu-nn`.

Evaluate on at least four composition models: `addition`, `matrix`, `fulllex` and `transweight`. Use the Python/Tensorflow implementations provided in `commix`, `https://github.com/sfb833-a3/commix`.

Project 5: Test word2vec Skip-Gram negative sampling (SG-NEG) with and without subsampling for English.

See project aim from Project 1. Word representations:

- `en.wiki-tacl-compounds.sg.ng5.no_subs.bin`
- `en.wiki-tacl-compounds.sg.ng5.1e-5_subs.bin`

Both word spaces were trained using `skip-gram` with 5 negative samples. `en.wiki-tacl-compounds.sg.ng5.no_subs.bin` was trained without subsampling. `en.wiki-tacl-compounds.sg.ng5.1e-5_subs.bin` was trained with subsampling using $t = 1e - 5$.

You should also pick a third file from the available embedding spaces and compare to it as well (e.g. pick a GloVe or a fastText embedding set).

Evaluate the word representations intrinsically on

1. word similarity
   - the WordSim353 dataset by Finkelstein et al. [2001], `ws353.txt`
   - the Rare Words dataset by Luong et al. [2013], `luong_rare.txt`
2. word analogy
   - the Google analogy dataset by Mikolov et al. [2013], `mikolov_analogies.txt`
   - the paradigmatic relations dataset by Köper et al. [2015], `en_sem-para_SemRel.txt`

You are encouraged (optional) to also report results on the BATS dataset introduced by Gladkova et al. [2016], or on the additional datasets available in `hyperwords`[4] by Levy et al. [2015].

Evaluate the representations also extrinsically, on the noun compound composition task. You should evaluate on the `eng-nn` dataset provided by Dima et al. [2019], `https://talar.sfb833.uni-tuebingen.de:8443/erdora/cmdi/SFB833/A03/TACL_datasets/eng-nn`.

Evaluate on at least four composition models: `addition`, `matrix`, `fulllex` and `transweight`. Use the Python/Tensorflow implementations provided in `commix`, `https://github.com/sfb833-a3/commix`.

Project 6: Test word2vec Skip-Gram hierarchical softmax (SG-HS) with and without subsampling for English.

See project aim from Project 1. Word representations:

- `en.wiki-tacl-compounds.sg.hs.no_subs.bin`
- `en.wiki-tacl-compounds.sg.hs.1e-5_subs.bin`

Both word spaces were trained using `skip-gram` with hierarchical softmax. The difference is that `en.wiki-tacl-compounds.sg.hs.no_subs.bin` was trained without subsampling, while `en.wiki-tacl-compounds.sg.hs.1e-5_subs.bin` was trained with subsampling using $t = 1e - 5$.

You should also pick a third file from the available embedding spaces and compare to it as well (e.g. pick a GloVe or a fastText embedding set).

[4] `https://bitbucket.org/omerlevy/hyperwords/src/default/testsets/`

Evaluate the word representations intrinsically on

1. word similarity
   - the WordSim353 dataset by Finkelstein et al. [2001], `ws353.txt`
   - the Rare Words dataset by Luong et al. [2013], `luong_rare.txt`
2. word analogy
   - the Google analogy dataset by Mikolov et al. [2013], `mikolov_analogies.txt`
   - the paradigmatic relations dataset by Köper et al. [2015], `en_sem-para_SemRel.txt`

You are encouraged (optional) to also report results on the BATS dataset introduced by Gladkova et al. [2016], or on the additional datasets available in `hyperwords`[5] by Levy et al. [2015].

Evaluate the representations also extrinsically, on the noun compound composition task. You should evaluate on the `eng-nn` dataset provided by Dima et al. [2019], `https://talar.sfb833.uni-tuebingen.de:8443/erdora/cmdi/SFB833/A03/TACL_datasets/eng-nn`.

Evaluate on at least four composition models: `addition`, `matrix`, `fulllex` and `transweight`. Use the Python/Tensorflow implementations provided in `commix`, `https://github.com/sfb833-a3/commix`.

Project 7: Test word2vec CBOW negative sampling (CBOW-NEG) with and without subsampling for English.

See project aim from Project 1. Word representations:

- `en.wiki-tacl-compounds.cbow.ng5.no_subs.bin`
- `en.wiki-tacl-compounds.cbow.ng5.1e-5_subs.bin`

Both word spaces were trained using `cbow` with 5 negative samples. The difference is that `de.wiki.cbow.ng5.no_subs.bin` was trained without subsampling, while `de.wiki.cbow.ng5.1e-5_subs.bin` was trained with subsampling using $t = 1e - 5$.

You should also pick a third file from the available embedding spaces and compare to it as well (e.g. pick a GloVe or a fastText embedding set).

Evaluate the word representations intrinsically on

1. word similarity
   - the WordSim353 dataset by Finkelstein et al. [2001], `ws353.txt`

[5] `https://bitbucket.org/omerlevy/hyperwords/src/default/testsets/`

- the Rare Words dataset by Luong et al. [2013], `luong_rare.txt`

2. word analogy
   - the Google analogy dataset by Mikolov et al. [2013], `mikolov_analogies.txt`
   - the paradigmatic relations dataset by Köper et al. [2015], `en_sem-para_SemRel.txt`

You are encouraged (optional) to also report results on the BATS dataset introduced by Gladkova et al. [2016], or on the additional datasets available in `hyperwords`[6] by Levy et al. [2015].

Evaluate the representations also extrinsically, on the noun compound composition task. You should evaluate on the `eng-nn` dataset provided by Dima et al. [2019], `https://talar.sfb833.uni-tuebingen.de:8443/erdora/cmdi/SFB833/A03/TACL_datasets/eng-nn`.

Evaluate on at least four composition models: `addition`, `matrix`, `fulllex` and `transweight`. Use the Python/Tensorflow implementations provided in `commix`, `https://github.com/sfb833-a3/commix`.

Project 8: Test word2vec CBOW hierarchical softmax (CBOW-HS) with and without subsampling for English.

See project aim from Project 1. Word representations:

- `en.wiki-tacl-compounds.cbow.hs.no_subs.bin`
- `en.wiki-tacl-compounds.cbow.hs.1e-5_subs.bin`

Both word spaces were trained using `cbow` with hierarchical softmax. The difference is that `en.wiki-tacl-compounds.cbow.hs.no_subs.bin` was trained without subsampling, while `en.wiki-tacl-compounds.cbow.hs.1e-5_subs.bin` was trained with subsampling using $t = 1e-5$.

You should also pick a third file from the available embedding spaces and compare to it as well (e.g. pick a GloVe or a fastText embedding set).

Evaluate the word representations intrinsically on

1. word similarity
   - the WordSim353 dataset by Finkelstein et al. [2001], `ws353.txt`
   - the Rare Words dataset by Luong et al. [2013], `luong_rare.txt`
2. word analogy
   - the Google analogy dataset by Mikolov et al. [2013], `mikolov_analogies.txt`

[6] `https://bitbucket.org/omerlevy/hyperwords/src/default/testsets/`

- the paradigmatic relations dataset by Köper et al. [2015], `en_sem-para_SemRel.txt`

You are encouraged (optional) to also report results on the BATS dataset introduced by Gladkova et al. [2016], or on the additional datasets available in `hyperwords`[7] by Levy et al. [2015].

Evaluate the representations also extrinsically, on the noun compound composition task. You should evaluate on the `eng-nn` dataset provided by Dima et al. [2019], `https://talar.sfb833.uni-tuebingen.de:8443/erdora/cmdi/SFB833/A03/TACL_datasets/eng-nn`.

Evaluate on at least four composition models: `addition`, `matrix`, `fulllex` and `transweight`. Use the Python/Tensorflow implementations provided in `commix`, `https://github.com/sfb833-a3/commix`.

Project 9: Test GloVe word, context and word + context embeddings for English.

The goal of the project is to quantify the impact of using the word or context vectors on the word similarity/analogy tasks and on the compound composition task for English. By default, GloVe will add the resulting word and context vectors ($w + c$). However, one can train GloVe and instruct it to specifically save a concatenation of the word and context vectors by setting the `-model` parameter to 0.

Test the 3 types of word representations:

- the word representations,
  `en.wiki-tacl-compounds.glove.ctx10.d300.min50.w_only.bin`
- the context representations,
  `en.wiki-tacl-compounds.glove.ctx10.d300.min50.c_only.bin`
- the summed word and context representations,
  `en.wiki-tacl-compounds.glove.ctx10.d300.min50.w_plus_c.bin`

Evaluate the word representations intrinsically on

1. word similarity
   - the WordSim353 dataset by Finkelstein et al. [2001], `ws353.txt`
   - the Rare Words dataset by Luong et al. [2013], `luong_rare.txt`
2. word analogy
   - the Google analogy dataset by Mikolov et al. [2013], `mikolov_analogies.txt`

[7] `https://bitbucket.org/omerlevy/hyperwords/src/default/testsets/`

- the paradigmatic relations dataset by Köper et al. [2015], `en_sem-para_SemRel.txt`

You are encouraged (optional) to also report results on the BATS dataset introduced by Gladkova et al. [2016], or on the additional datasets available in `hyperwords`[8] by Levy et al. [2015].

Evaluate the representations also extrinsically, on the noun compound composition task. You should evaluate on the `eng-nn` dataset provided by Dima et al. [2019], `https://talar.sfb833.uni-tuebingen.de:8443/erdora/cmdi/SFB833/A03/TACL_datasets/eng-nn`.

Evaluate on at least four composition models: `addition`, `matrix`, `fulllex` and `transweight`. Use the Python/Tensorflow implementations provided in `commix`, `https://github.com/sfb833-a3/commix`.

Project 10: Test GloVe word, context and word + context embeddings for German.

The goal of the project is to quantify the impact of using the word or context vectors on the word similarity/analogy tasks and on the compound composition task for German. By default, GloVe will add the resulting word and context vectors ($w + c$). However, one can train GloVe and instruct it to specifically save a concatenation of the word and context vectors by setting the `-model` parameter to 0.

Test the 3 types of word representations:

- the word representations, `de.wiki.glove.ctx10.d300.min50.w_only.bin`
- the context representations, `de.wiki.glove.ctx10.d300.min50.c_only.bin`
- the summed word and context representations, `de.wiki.glove.ctx10.d300.min50.w_plus_c.bin`

Evaluate the word representations intrinsically, on the datasets proposed by Köper et al. [2015], `https://www.ims.uni-stuttgart.de/forschung/ressourcen/lexika/analogies_ims/analogies.en.html` for:

- word similarity (`de_re-rated_Schm280_tabs.txt`)
- word analogy (`de_trans_Google_analogies.txt`, `de_sem-para_SemRel.txt`)

[8] `https://bitbucket.org/omerlevy/hyperwords/src/default/testsets/`

Evaluate the representations also extrinsically, on the noun compound composition task. You should evaluate on the `deu-nn` dataset provided by Dima et al. [2019], `https://talar.sfb833.uni-tuebingen.de:8443/erdora/cmdi/SFB833/A03/TACL_datasets/deu-nn`.

Evaluate on at least four composition models: `addition`, `matrix`, `fulllex` and `transweight`. Use the Python/Tensorflow implementations provided in `commix`, `https://github.com/sfb833-a3/commix`.

Project 11: Test fastText embeddings using 3-6 n-grams and 5-6 n-grams for English.

The goal of the project is to quantify the impact of adding subword information on the word similarity/analogy tasks and on the compound composition task for English. According to Bojanowski et al. [2017] longer n-grams are particularly beneficial when dealing with semantic tasks.

Compare two types of subword-aware representations obtained using fastText to the word2vec counterpart:

- a fastText model using 3, 4, 5 and 6-grams,
  `en.wiki-tacl-compounds.fasttext.3-6ngrams.neg5.1e-5_subs.bin`
- a fastText model using 5 and 6-grams,
  `en.wiki-tacl-compounds.fasttext.5-6ngrams.neg5.1e-5_subs.bin`
- a word2vec skip-gram with negative sampling and subsampling model,
  `en.wiki-tacl-compounds.sg.ng5.1e-5_subs.bin`

To represent the out-of-vocabulary (OOV) words the corresponding fastText binary files are also available:
`en.wiki-tacl-compounds.fasttext.3-6ngrams.neg5.1e-5_subs.ft_bin`,
`en.wiki-tacl-compounds.fasttext.5-6ngrams.neg5.1e-5_subs.ft_bin`.

Evaluate the word representations intrinsically on

1. word similarity
   - the WordSim353 dataset by Finkelstein et al. [2001], `ws353.txt`
   - the Rare Words dataset by Luong et al. [2013], `luong_rare.txt`
2. word analogy
   - the Google analogy dataset by Mikolov et al. [2013], `mikolov_analogies.txt`
   - the paradigmatic relations dataset by Köper et al. [2015], `en_sem-para_SemRel.txt`

You are encouraged (optional) to also report results on the BATS dataset introduced by Gladkova et al. [2016], or on the additional datasets available in `hyperwords`[9] by Levy et al. [2015].

[9] `https://bitbucket.org/omerlevy/hyperwords/src/default/testsets/`

Evaluate the representations also extrinsically, on the noun compound composition task. You should evaluate on the `eng-nn` dataset provided by Dima et al. [2019], `https://talar.sfb833.uni-tuebingen.de:8443/erdora/cmdi/SFB833/A03/TACL_datasets/eng-nn`.

Evaluate on at least four composition models: `addition`, `matrix`, `fulllex` and `transweight`. Use the Python/Tensorflow implementations provided in `commix`, `https://github.com/sfb833-a3/commix`.

Project 12: Test fastText embeddings using 3-6 n-grams and 5-6 n-grams for German.

The goal of the project is to quantify the impact of adding subword information on the word similarity/analogy tasks and on the compound composition task for German. According to Bojanowski et al. [2017] longer n-grams are particularly beneficial when dealing with semantic tasks.

Compare two types of subword-aware representations obtained using fastText to the word2vec counterpart:

- a fastText model using 3, 4, 5 and 6-grams,

  `de.wiki.fasttext.3-6ngrams.neg5.1e-5_subs.bin`
- a fastText model using 5 and 6-grams,

  `de.wiki.fasttext.5-6ngrams.neg5.1e-5_subs.bin`
- a word2vec skip-gram with negative sampling and subsampling model,

  `de.wiki.sg.ng5.1e-5_subs.bin`

To represent the out-of-vocabulary (OOV) words the corresponding fastText binary files are also available:
`de.wiki.fasttext.3-6ngrams.neg5.1e-5_subs.ft_bin`,
`de.wiki.fasttext.5-6ngrams.neg5.1e-5_subs.ft_bin`.

Evaluate the word representations intrinsically, on the datasets proposed by Köper et al. [2015], `https://www.ims.uni-stuttgart.de/forschung/ressourcen/lexika/analogies_ims/analogies.en.html` for:

- word similarity (`de_re-rated_Schm280_tabs.txt`)
- word analogy (`de_trans_Google_analogies.txt`,

  `de_sem-para_SemRel.txt`)

Evaluate the representations also extrinsically, on the noun compound composition task. You should evaluate on the `deu-nn` dataset provided by Dima et al. [2019], `https://talar.sfb833.uni-tuebingen.de:8443/erdora/cmdi/SFB833/A03/TACL_datasets/deu-nn`.

Evaluate on at least four composition models: `addition`, `matrix`, `fulllex` and `transweight`. Use the Python/Tensorflow implementations provided in `commix`, `https://github.com/sfb833-a3/commix`.

Project 13: Test word2vec skip-gram retrofitted embeddings for English.

The goal of the project is to investigate the impact of retraining word representations to take into account information contained in semantic lexicons such as the WordNet, FrameNet or the Paraphrase Database for English. The retrofitting procedure is fast and can be run on top of pre-trained word representations. Using the code and semantic lexicons provided by Faruqui et al. [2015][10] retrofit the following word representations:

- skip-gram with negative sampling and 1e-5 subsampling, `en.wiki-tacl-compounds.sg.ng5.1e-5_subs.bin`
- skip-gram with negative sampling, no subsampling, `en.wiki-tacl-compounds.sg.ng5.no_subs.bin`
- skip-gram with hierarchical softmax and 1e-5 subsampling, `en.wiki-tacl-compounds.sg.hs.1e-5_subs.bin`
- skip-gram with hierarchical softmax, no subsampling, `en.wiki-tacl-compounds.sg.hs.no_subs.bin`

Evaluate the word representations intrinsically on

1. word similarity
   - the WordSim353 dataset by Finkelstein et al. [2001], `ws353.txt`
   - the Rare Words dataset by Luong et al. [2013], `luong_rare.txt`
2. word analogy
   - the Google analogy dataset by Mikolov et al. [2013], `mikolov_analogies.txt`
   - the paradigmatic relations dataset by Köper et al. [2015], `en_sem-para_SemRel.txt`

You are encouraged (optional) to also report results on the BATS dataset introduced by Gladkova et al. [2016], or on the additional datasets available in `hyperwords`[11] by Levy et al. [2015].

Evaluate the representations also extrinsically, on the noun compound composition task. You should evaluate on the `eng-nn` dataset provided by Dima et al. [2019], `https://talar.sfb833.uni-tuebingen.de:8443/erdora/cmdi/SFB833/A03/TACL_datasets/eng-nn`.

Evaluate on at least four composition models: `addition`, `matrix`, `fulllex` and `transweight`. Use the Python/Tensorflow implementations provided in `commix`, `https://github.com/sfb833-a3/commix`.

[10] `https://github.com/mfaruqui/retrofitting`

[11] `https://bitbucket.org/omerlevy/hyperwords/src/default/testsets/`

Project 14: Test word2vec CBOW retrofitted embeddings for English.

The goal of the project is to investigate the impact of retraining word representations to take into account information contained in semantic lexicons such as the WordNet, FrameNet or the Paraphrase Database for English. The retrofitting procedure is fast and can be run on top of pre-trained word representations. Using the code and semantic lexicons provided by Faruqui et al. [2015][12] retrofit the following word representations:

- CBOW with negative sampling and 1e-5 subsampling,
  `en.wiki-tacl-compounds.cbow.ng5.1e-5_subs.bin`
- CBOW with negative sampling, no subsampling,
  `en.wiki-tacl-compounds.cbow.ng5.no_subs.bin`
- CBOW with hierarchical softmax and 1e-5 subsampling,
  `en.wiki-tacl-compounds.cbow.hs.1e-5_subs.bin`
- CBOW with hierarchical softmax, no subsampling,
  `en.wiki-tacl-compounds.cbow.hs.no_subs.bin`

Evaluate the word representations intrinsically on

1. word similarity
   - the WordSim353 dataset by Finkelstein et al. [2001], `ws353.txt`
   - the Rare Words dataset by Luong et al. [2013], `luong_rare.txt`
2. word analogy
   - the Google analogy dataset by Mikolov et al. [2013], `mikolov_analogies.txt`
   - the paradigmatic relations dataset by Köper et al. [2015], `en_sem-para_SemRel.txt`

You are encouraged (optional) to also report results on the BATS dataset introduced by Gladkova et al. [2016], or on the additional datasets available in `hyperwords`[13] by Levy et al. [2015].

Evaluate the representations also extrinsically, on the noun compound composition task. You should evaluate on the `eng-nn` dataset provided by Dima et al. [2019], `https://talar.sfb833.uni-tuebingen.de:8443/erdora/cmdi/SFB833/A03/TACL_datasets/eng-nn`.

Evaluate on at least four composition models: `addition`, `matrix`, `fulllex` and `transweight`. Use the Python/Tensorflow implementations provided in `commix`, `https://github.com/sfb833-a3/commix`.

[12] `https://github.com/mfaruqui/retrofitting`
[13] `https://bitbucket.org/omerlevy/hyperwords/src/default/testsets/`

Project 15: Test fastText retrofitted embeddings for English.

The goal of the project is to investigate the impact of retraining word representations to take into account information contained in semantic lexicons such as the WordNet, FrameNet or the Paraphrase Database for English. The retrofitting procedure is fast and can be run on top of pre-trained word representations. Using the code and semantic lexicons provided by Faruqui et al. [2015][14] retrofit the following word representations:

- a fastText model using 3, 4, 5 and 6-grams,
  `en.wiki-tacl-compounds.fasttext.3-6ngrams.neg5.1e-5_subs.bin`
- a fastText model using 5 and 6-grams,
  `en.wiki-tacl-compounds.fasttext.5-6ngrams.neg5.1e-5_subs.bin`
- a word2vec skip-gram with negative sampling and subsampling model,
  `en.wiki-tacl-compounds.sg.ng5.1e-5_subs.bin`
- a word2vec CBOW with negative sampling and subsampling model,
  `en.wiki-tacl-compounds.cbow.ng5.1e-5_subs.bin`

Evaluate the word representations intrinsically on

1. word similarity
   - the WordSim353 dataset by Finkelstein et al. [2001], `ws353.txt`
   - the Rare Words dataset by Luong et al. [2013], `luong_rare.txt`
2. word analogy
   - the Google analogy dataset by Mikolov et al. [2013], `mikolov_analogies.txt`
   - the paradigmatic relations dataset by Köper et al. [2015], `en_sem-para_SemRel.txt`

You are encouraged (optional) to also report results on the BATS dataset introduced by Gladkova et al. [2016], or on the additional datasets available in `hyperwords`[15] by Levy et al. [2015].

Evaluate the representations also extrinsically, on the noun compound composition task. You should evaluate on the `eng-nn` dataset provided by Dima et al. [2019], `https://talar.sfb833.uni-tuebingen.de:8443/erdora/cmdi/SFB833/A03/TACL_datasets/eng-nn`.

Evaluate on at least four composition models: `addition`, `matrix`, `fulllex` and `transweight`. Use the Python/Tensorflow implementations provided in `commix`, `https://github.com/sfb833-a3/commix`.

[14] `https://github.com/mfaruqui/retrofitting`

[15] `https://bitbucket.org/omerlevy/hyperwords/src/default/testsets/`

Project 16: Test word2vec skip-gram retrofitted embeddings for German.

The goal of the project is to investigate the impact of retraining word representations to take into account information contained in semantic lexicons such as GermaNet for German. GermaNet[16] is the German equivalent of the English WordNet, and is developed at the University of Tübingen. Create four semantic lexicons including (a) only synonyms; (b) only hypernyms/hyponyms (c) only meronyms/holonyms; (d) all relations - synonyms, hyper/hyponyms, meronyms/hononyms.

The retrofitting procedure is fast and can be run on top of pre-trained word representations. Using the code and semantic lexicons provided by Faruqui et al. [2015][17] retrofit the following word representations:

- skip-gram with negative sampling and 1e-5 subsampling,
  `de.wiki.sg.ng5.1e-5_subs.bin`
- skip-gram with negative sampling, no subsampling,
  `de.wiki.sg.ng5.no_subs.bin`
- skip-gram with hierarchical softmax and 1e-5 subsampling,
  `de.wiki.sg.hs.1e-5_subs.bin`
- skip-gram with hierarchical softmax, no subsampling,
  `de.wiki.sg.hs.no_subs.bin`

Evaluate the word representations intrinsically, on the datasets proposed by Köper et al. [2015], `https://www.ims.uni-stuttgart.de/forschung/ressourcen/lexika/analogies_ims/analogies.en.html` for:

- word similarity (`de_re-rated_Schm280_tabs.txt`)
- word analogy (`de_trans_Google_analogies.txt`,
  `de_sem-para_SemRel.txt`)

Evaluate the representations also extrinsically, on the noun compound composition task. You should evaluate on the `deu-nn` dataset provided by Dima et al. [2019], `https://talar.sfb833.uni-tuebingen.de:8443/erdora/cmdi/SFB833/A03/TACL_datasets/deu-nn`.

Evaluate on at least four composition models: `addition`, `matrix`, `fulllex` and `transweight`. Use the Python/Tensorflow implementations provided in `commix`, `https://github.com/sfb833-a3/commix`.

[16] `http://www.sfs.uni-tuebingen.de/GermaNet/`
[17] `https://github.com/mfaruqui/retrofitting`

Project 17: Test word2vec CBOW retrofitted embeddings for German.

The goal of the project is to investigate the impact of retraining word representations to take into account information contained in semantic lexicons such as GermaNet for German. GermaNet[18] is the German equivalent of the English WordNet, and is developed at the University of Tübingen. Create four semantic lexicons including (a) only synonyms; (b) only hypernyms/hyponyms (c) only meronyms/holonyms; (d) all relations - synonyms, hyper/hyponyms, meronyms/hononyms.

The retrofitting procedure is fast and can be run on top of pre-trained word representations. Using the code and semantic lexicons provided by Faruqui et al. [2015][19] retrofit the following word representations:

- CBOW with negative sampling and 1e-5 subsampling,

  `de.wiki.cbow.ng5.1e-5_subs.bin`
- CBOW with negative sampling, no subsampling,

  `de.wiki.cbow.ng5.no_subs.bin`
- CBOW with hierarchical softmax and 1e-5 subsampling,

  `de.wiki.cbow.hs.1e-5_subs.bin`
- CBOW with hierarchical softmax, no subsampling,

  `de.wiki.cbow.hs.no_subs.bin`

Evaluate the word representations intrinsically, on the datasets proposed by Köper et al. [2015], `https://www.ims.uni-stuttgart.de/forschung/ressourcen/lexika/analogies_ims/analogies.en.html` for:

- word similarity (`de_re-rated_Schm280_tabs.txt`)
- word analogy (`de_trans_Google_analogies.txt`,

  `de_sem-para_SemRel.txt`)

Evaluate the representations also extrinsically, on the noun compound composition task. You should evaluate on the `deu-nn` dataset provided by Dima et al. [2019], `https://talar.sfb833.uni-tuebingen.de:8443/erdora/cmdi/SFB833/A03/TACL_datasets/deu-nn`.

Evaluate on at least four composition models: `addition`, `matrix`, `fulllex` and `transweight`. Use the Python/Tensorflow implementations provided in `commix`, `https://github.com/sfb833-a3/commix`.

[18] `http://www.sfs.uni-tuebingen.de/GermaNet/`

[19] `https://github.com/mfaruqui/retrofitting`

Project 18: Test fastText retrofitted embeddings for German.

The goal of the project is to investigate the impact of retraining word representations to take into account information contained in semantic lexicons such as GermaNet for German. GermaNet[20] is the German equivalent of the English WordNet, and is developed at the University of Tübingen. Create four semantic lexicons including (a) only synonyms; (b) only hypernyms/hyponyms (c) only meronyms/holonyms; (d) all relations - synonyms, hyper/hyponyms, meronyms/hononyms.

The retrofitting procedure is fast and can be run on top of pre-trained word representations. Using the code and semantic lexicons provided by Faruqui et al. [2015][21] retrofit the following word representations:

- a fastText model using 3, 4, 5 and 6-grams,
  `de.wiki.fasttext.3-6ngrams.neg5.1e-5_subs.bin`
- a fastText model using 5 and 6-grams,
  `de.wiki.fasttext.5-6ngrams.neg5.1e-5_subs.bin`
- skip-gram with negative sampling and 1e-5 subsampling,
  `de.wiki.sg.ng5.1e-5_subs.bin`
- CBOW with negative sampling and 1e-5 subsampling,
  `de.wiki.cbow.ng5.1e-5_subs.bin`

Evaluate the word representations intrinsically, on the datasets proposed by Köper et al. [2015], `https://www.ims.uni-stuttgart.de/forschung/ressourcen/lexika/analogies_ims/analogies.en.html` for:

- word similarity (`de_re-rated_Schm280_tabs.txt`)
- word analogy (`de_trans_Google_analogies.txt`,
  `de_sem-para_SemRel.txt`)

Evaluate the representations also extrinsically, on the noun compound composition task. You should evaluate on the `deu-nn` dataset provided by Dima et al. [2019], `https://talar.sfb833.uni-tuebingen.de:8443/erdora/cmdi/SFB833/A03/TACL_datasets/deu-nn`.

Evaluate on at least four composition models: `addition`, `matrix`, `fulllex` and `transweight`. Use the Python/Tensorflow implementations provided in `commix`, `https://github.com/sfb833-a3/commix`.

[20] `http://www.sfs.uni-tuebingen.de/GermaNet/`
[21] `https://github.com/mfaruqui/retrofitting`

Project 19: Test fastText with different sizes of the corpus for German.

The goal of the project is to explore the limits of the fastText model when using much less training data on German. You are provided with the fastText embeddings when using 100% of the data.

- a fastText model using 3, 4, 5 and 6-grams,
  `de.wiki.fasttext.3-6ngrams.neg5.1e-5_subs.bin`
- a fastText model using 5 and 6-grams,
  `de.wiki.fasttext.5-6ngrams.neg5.1e-5_subs.bin`

Choose the n-gram size - either 3-6 or 5-6 ngrams. Train additional fastText embeddings using 1%, 5%, 25% and 50% of the data, and test the representations you obtain against the representations trained on the full-sized corpus.

Evaluate the word representations intrinsically, on the datasets proposed by Köper et al. [2015], `https://www.ims.uni-stuttgart.de/forschung/ressourcen/lexika/analogies_ims/analogies.en.html` for:

- word similarity (`de_re-rated_Schm280_tabs.txt`)
- word analogy (`de_trans_Google_analogies.txt`,
  `de_sem-para_SemRel.txt`)

Evaluate the representations also extrinsically, on the noun compound composition task. You should evaluate on the `deu-nn` dataset provided by Dima et al. [2019], `https://talar.sfb833.uni-tuebingen.de:8443/erdora/cmdi/SFB833/A03/TACL_datasets/deu-nn`.

Evaluate on at least four composition models: `addition`, `matrix`, `fulllex` and `transweight`. Use the Python/Tensorflow implementations provided in `commix`, `https://github.com/sfb833-a3/commix`.

Project 20: Test fastText with different sizes of the corpus for English.

The goal of the project is to explore the limits of the fastText model when using much less training data on English. You are provided with the fastText embeddings when using 100% of the data.

- a fastText model using 3, 4, 5 and 6-grams,
  `en.wiki-tacl-compounds.fasttext.3-6ngrams.neg5.1e-5_subs.bin`
- a fastText model using 5 and 6-grams,
  `en.wiki-tacl-compounds.fasttext.5-6ngrams.neg5.1e-5_subs.bin`

Choose the n-gram size - either 3-6 or 5-6 ngrams. Train additional fastText embeddings using 1%, 5%, 25% and 50% of the data, and test the representations you obtain against the representations trained on the full-sized corpus.

Evaluate the word representations intrinsically on

1. word similarity
   - the WordSim353 dataset by Finkelstein et al. [2001], `ws353.txt`
   - the Rare Words dataset by Luong et al. [2013], `luong_rare.txt`
2. word analogy
   - the Google analogy dataset by Mikolov et al. [2013], `mikolov_analogies.txt`
   - the paradigmatic relations dataset by Köper et al. [2015], `en_sem-para_SemRel.txt`

You are encouraged (optional) to also report results on the BATS dataset introduced by Gladkova et al. [2016], or on the additional datasets available in `hyperwords`[22] by Levy et al. [2015].

Evaluate the representations also extrinsically, on the noun compound composition task. You should evaluate on the `eng-nn` dataset provided by Dima et al. [2019], `https://talar.sfb833.uni-tuebingen.de:8443/erdora/cmdi/SFB833/A03/TACL_datasets/eng-nn`.

Evaluate on at least four composition models: `addition`, `matrix`, `fulllex` and `transweight`. Use the Python/Tensorflow implementations provided in `commix`, `https://github.com/sfb833-a3/commix`.

1.1 Writing the Paper Report

You should use LaTeX and the official ACL 2019 styles[23] to write your paper. Your paper should have between 4 and 6 pages of content (excluding references).

Please use the GitHub repository you've created for your registration to upload the .pdf of your paper once it's ready. You should name the file
`family-name_paper_report.pdf`. I will not take into consideration paper report that are submitted per email. The deadline for submitting the final version of your paper report is July 27th.

If your assignment is based on code/scripts that you wrote, or produces additional files (e.g. the semantic lexicons for the German retrofitting projects) please upload them as well to your repository.

Your paper should clearly state it's aim - i.e. what is the aspect you are trying to find out more about. It should provide brief descriptions of the resources that you used (data/existing code), and of your experimental setup. The results should present a fair comparison between different models.

Make sure that when you report intrinsic evaluation numbers (i.e., on the analogies datasets) you also mention the coverage (how many of the total number of analogies in the dataset could be answered).

[22] `https://bitbucket.org/omerlevy/hyperwords/src/default/testsets/`
[23] `http://www.acl2019.org/medias/340-acl2019-latex.zip`

Similarly, when you report extrinsic evaluation numbers (i.e. on the composition datasets) you might run into cases where some of the training/dev/test words/phrases are not in your vocabulary. Make sure to report the coverage together with the results.

And remember, write the paper first[24]!

2 Resources

2.1 Corpora

- preprocessed English Wikipedia dump[25] made available by Müller and Schütze [2015]
- preprocessed German Wikipedia dump[26] made available by Müller and Schütze [2015]

2.2 Software for training embeddings

- `word2vec`, `https://github.com/tmikolov/word2vec`
- `GloVe`, `https://nlp.stanford.edu/projects/glove/`, v1.2
- `fastText`,
  `https://github.com/facebookresearch/fastText`, release 0.2.0

2.3 Pre-trained embeddings in different languages

- fastText embeddings for 157 languages by Bojanowski et al. [2017],
  `https://fasttext.cc/docs/en/crawl-vectors.html`

References

Maximilian Köper, Christian Scheible, and Sabine Schulte im Walde. Multilingual reliability and "semantic" structure of continuous word spaces. In *Proceedings of the 11th International Conference on Computational Semantics*, pages 40–45, London, UK, April 2015. Association for Computational Linguistics. URL `https://www.aclweb.org/anthology/W15-0105`.

Corina Dima, Daniël de Kok, Neele Witte, and Erhard Hinrichs. No Word is an Island — A Transformation Weighting Model for Semantic Composition. *Transactions of the Association for Computational Linguistics*, 2019.

[24] `https://www.cs.jhu.edu/~jason/advice/write-the-paper-first.html`

[25] `http://cistern.cis.lmu.de/marmot/naacl2015/en.wikidump.bz2`

[26] `http://cistern.cis.lmu.de/marmot/naacl2015/de.wikidump.bz2`

Lev Finkelstein, Evgeniy Gabrilovich, Yossi Matias, Ehud Rivlin, Zach Solan, Gadi Wolfman, and Eytan Ruppin. Placing search in context: The concept revisited. In *Proceedings of the 10th International Conference on World Wide Web*, WWW '01, pages 406–414, New York, NY, USA, 2001. ACM. ISBN 1-58113-348-0. doi: 10.1145/371920.372094. URL `http://doi.acm.org/10.1145/371920.372094`.

Thang Luong, Richard Socher, and Christopher Manning. Better word representations with recursive neural networks for morphology. In *Proceedings of the Seventeenth Conference on Computational Natural Language Learning*, pages 104–113, Sofia, Bulgaria, August 2013. Association for Computational Linguistics. URL `https://www.aclweb.org/anthology/W13-3512`.

Tomas Mikolov, Kai Chen, Greg S. Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space, 2013. URL `http://arxiv.org/abs/1301.3781`.

Anna Gladkova, Aleksandr Drozd, and Satoshi Matsuoka. Analogy-based detection of morphological and semantic relations with word embeddings: what works and what doesn't. In *Proceedings of the NAACL Student Research Workshop*, pages 8–15, San Diego, California, June 2016. Association for Computational Linguistics. doi: 10.18653/v1/N16-2002. URL `https://www.aclweb.org/anthology/N16-2002`.

Omer Levy, Yoav Goldberg, and Ido Dagan. Improving distributional similarity with lessons learned from word embeddings. *Transactions of the Association for Computational Linguistics*, 3:211–225, 2015. ISSN 2307-387X. URL `https://transacl.org/ojs/index.php/tacl/article/view/570`.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5:135–146, 2017. doi: 10.1162/tacl_a_00051. URL `https://doi.org/10.1162/tacl_a_00051`.

Manaal Faruqui, Jesse Dodge, Sujay K. Jauhar, Chris Dyer, Eduard Hovy, and Noah A. Smith. Retrofitting word vectors to semantic lexicons. In *Proceedings of NAACL*, 2015.

Thomas Müller and Hinrich Schütze. Robust morphological tagging with word representations. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 526–536, Denver, Colorado, May–June 2015. Association for Computational Linguistics. doi: 10.3115/v1/N15-1055. URL `https://www.aclweb.org/anthology/N15-1055`.

Rami Al-Rfou, Bryan Perozzi, and Steven Skiena. Polyglot: Distributed word representations for multilingual NLP. In *Proceedings of the Seventeenth Conference on Computational Natural Language Learning*, pages 183–192, Sofia, Bulgaria, August 2013. Association for Computational Linguistics. URL https://www.aclweb.org/anthology/W13-3520.

A Training Word Representations

A.1 Training `word2vec` embeddings

Train `word2vec` embeddings using the original implementation of `word2vec` (see Section 2.2). The following command will train `word2vec` on the corpus in text_file.txt using the skip-gram model with hierarchical softmax and a subsampling threshold t of $1e-5$.

```
word2vec -train text_file.txt -output vectors.bin -cbow 0 -size 300 -window
10 -negative 0 -hs 1 -sample 1e-5 -threads 4 -binary 1 -min-count 50 -iter 15
```

`word2vec` uses the following hyperparameters:

- `-train`: the input data, text only, one sentence per line
- `-output`: the file where the word vectors will be stored
- `-cbow`: whether to train a CBOW model (1) or a Skip-Gram model (0)
- `-size`: the dimensionality of the resulting word representations
- `-window`: the size of the context window (symmetric)
- `-negative`: the number of negative samples to use - this should be set to 0 when not training with negative samples
- `-hs`: whether to train the model using hierarchical softmax (1) or not (0)
- `-sample`: the subsampling threshold (a good value is 1e-5); 0 if subsampling should not be used)
- `-threads`: the number of threads to use when training
- `-binary`: whether to save the vectors in the `word2vec` binary format (1) or in a text format (0)
- `-min-count`: the minimum frequency a word should have to be added to the vocabulary (50 is a good value)
- `-iter`: the number of iterations to train the model for (15 is a good choice)

A.2 Training GloVe embeddings

Train GloVe embeddings using the original implementation mentioned in Section 2.2. Training GloVe embeddings involves four steps:

1. Building a vocabulary:

   ```
   vocab_count -verbose 2 < text_file.txt > vocabulary.txt -min-count 50
   ```

   The vocabulary is built from the corpus file *text_file.txt* and will be saved in `vocabulary.txt`. Word vectors will be trained only for the words with a minimum frequency of 50 in *text_file.txt*.

2. Construct word-word co-occurrence statistics:

   ```
   co-occur -memory 8 -vocab-file vocabulary/txt -verbose 2 -window-size
   10 < text_file.txt > cooccurrences.txt
   ```

   The co-occurences are counted using a symmetric context window of size 10 (10 to the left and 10 to the right).

3. Shuffling the co-occurrences:

   ```
   shuffle -memory 8 -verbose 2 < cooccurrences.txt > cooc_shuffled.txt
   ```

4. Training the GloVe vectors on the shuffled co-occurences:

   ```
   glove -save-file glove_vectors.txt -threads 4
   -input-file cooc_shuffled.txt -vocab-file vocabulary.txt -iter 15
   -vector-size 300 -binary 0 -write-header 1 -model 2
   ```

   The trained vectors will be saved in *glove_vectors.txt*. The training will do 15 iterations (`-iter`) over the co-occurence matrix. The `-binary` flag set to 0 tells GloVe to save the word vectors in a text format. The option `-write-header` set to 1 instructs GloVe to write a `word2vec` style header to the word vectors file: a line containing the size of the vocabulary and the size of the word representations separated by a space. The option `-model` 2 is the default setting - GloVe will produce as output the added word + context vectors. Setting `-model` to 1 will save the word vectors only. Setting it to 0 will save both the word and the context vectors (and their biases).

A.3 Training fastText embeddings

Train fastText embeddings using the original implementation mentioned in Section 2.2.

The following command will train word representations on the input file `text_file.txt` using the skipgram with negative sampling training method with 5 negative samples, and using n-grams in the range 3 to 6.

```
fasttext skipgram -input text_file.txt -output fasttext_vectors -lr 0.05
-dim 300 -ws 10 -neg 5 -loss ns -thread 4 -minCount 50 -epoch 5 -t 1e-5
-minn 3 -maxn 6
```

- `skipgram, cbow`: whether to train a CBOW model or a Skip-Gram model
- `-output`: the prefix of the file where the word vectors will be stored; two files will be saved - a .vec file, which contains word vectors in .txt format, and a .bin file that can be used to retrieve the word representations for out of vocabulary words
- `-dim`: the dimensionality of the resulting word representations
- `-ws`: the size of the context window (symmetric)
- `-neg`: the number of negative samples to use - this should be set to 0 when not training with negative samples
- `-loss`: whether to train the model using negative sampling, hierarchical softmax or softmax
- `-t`: the subsampling threshold (a good value is 1e-5); 0 if subsampling should not be used)
- `-thread`: the number of threads to use when training
- `-minCount`: the minimum frequency a word should have to be added to the vocabulary (50 is a good value)
- `-epoch`: the number of iterations to train the model for (5 is a good choice)
- `minn`: the minimum length of the n-grams to be trained
- `maxn`: the maximum length of the n-grams to be trained
- `lr`: the learning rate

B Training Composition Models

Composition models can be trained using the `commix` package. The `train/dev/test` splits of each dataset contain triples of the form

telecom industry telecom_industry
waste water waste_water
glass fiber glass_fiber

Before you train a composition model, make sure to filter out of the `train` set the triples that contain unknown words (i.e. the embedding vocabulary might not cover all the words). You should not remove the unknown words from the `dev` and `test` sets.

Train the `addition` model:

```
python3 training.py embeddings.bin dataset_directory
--composition_model=addition --batch_size 100
```

Train the `matrix` model:

```
python3 training.py embeddings.bin dataset_directory
--composition_model=matrix --dropout 0 --nonlinearity identity
```

Train the `fulllex` model:

```
python3 training.py embeddings.bin dataset_directory
--composition_model=fulllex --dropout 0.6 --nonlinearity identity --use_nn
```

Train the `transweight` model:

```
python3 training.py embeddings.bin dataset_directory
--composition_model=trans_weight --dropout 0.8
--nonlinearity relu --transforms 100
```

Keep the other parameters to their default values (e.g. learning rate 0.01, batch size 100, patience 5 epochs, etc.). The `training.py` script will evaluate by default on the `dev` split of the dataset. To evaluate also on the `test` script you should set the `--eval_on_test` flag to `True`.