Lbl2Vec

Release 1.0.2

Tim Schopf

Jan 18, 2023

USER GUIDE:

1 Lbl2Vec **3**

2 Benefits **5**

3 Table of Contents **7**

4 How does it work? **9**
4.1 The Algorithm 9

5 Installation **13**

6 Usage **15**
6.1 Model Training 15

7 Citation information **19**

8 `Lbl2Vec` **21**

9 `Lbl2TransformerVec` **25**

10 Indices and tables **29**

Python Module Index **31**

Index **33**

CHAPTER

ONE

LBL2VEC

Lbl2Vec is an algorithm for **unsupervised document classification** and **unsupervised document retrieval.** It automatically generates jointly embedded label, document and word vectors and returns documents of topics modeled by manually predefined keywords. This package includes **two different model types**. The plain ***Lbl2Vec*** **model uses Doc2Vec**, whereas ***Lbl2TransformerVec*** **uses transformer-based language models** to create the embeddings. Once you train a model you can:

- Classify documents as related to one of the predefined topics.
- Get similarity scores for documents to each predefined topic.
- Get most similar predefined topic of documents.

See the papers presenting Lbl2Vec [1,2] and Lbl2TransformerVec [3] for more details on how it works.

Corresponding Medium post describing the use of Lbl2Vec for unsupervised text classification can be found here.

A Medium post evaluating Lbl2Vec and Lbl2TransformerVec against zero-shot approaches can be found here.

CHAPTER TWO

BENEFITS

1. No need to label the whole document dataset for classification.
2. No stop word lists required.
3. No need for stemming/lemmatization.
4. Works on short text.
5. Creates jointly embedded label, document, and word vectors.

CHAPTER

THREE

TABLE OF CONTENTS

1. *How does it work?*
2. *Installation*
3. *Usage*
    1. *Model Training*
        1. *Lbl2Vec*
        2. *Lbl2TransformerVec*
    2. *Document prediction*
    3. *Save and load models*
4. *Citation information*

CHAPTER

FOUR

HOW DOES IT WORK?

Back to Table of Contents

The key idea of the algorithm is that many semantically similar keywords can represent a topic. In the first step, the algorithm creates a joint embedding of document and word vectors. Once documents and words are embedded in a vector space, the goal of the algorithm is to learn label vectors from previously manually defined keywords representing a topic. Finally, the algorithm can predict the affiliation of documents to topics from *document vector* <-> *label vector* similarities.

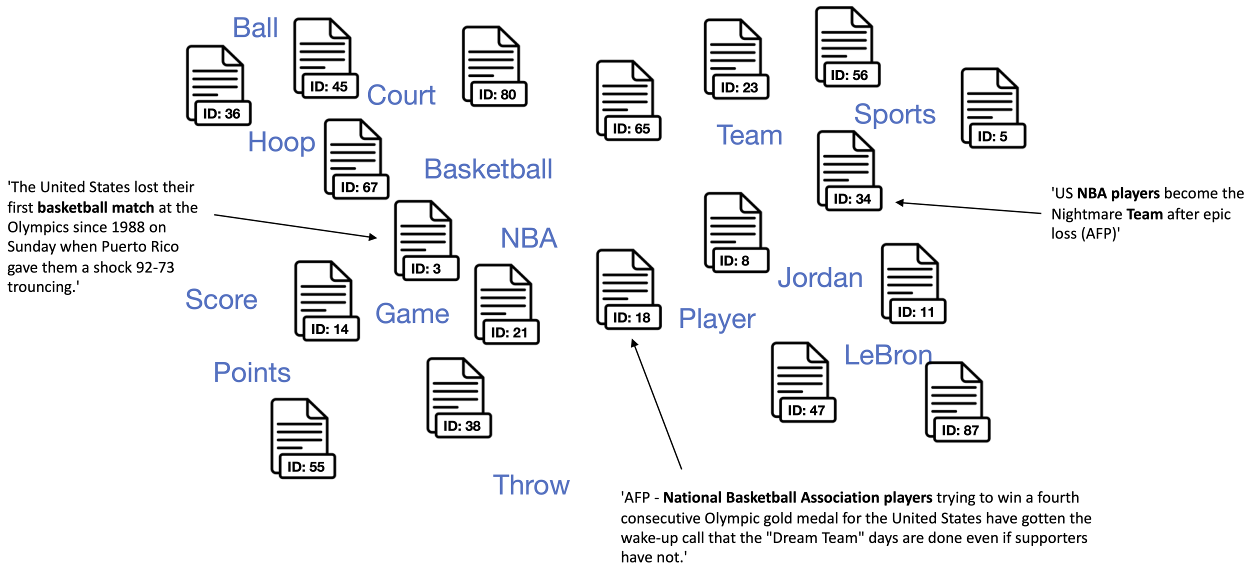
4.1 The Algorithm

0. Use the manually defined keywords for each topic of interest.

Domain knowledge is needed to define keywords that describe topics and are semantically similar to each other within the topics.

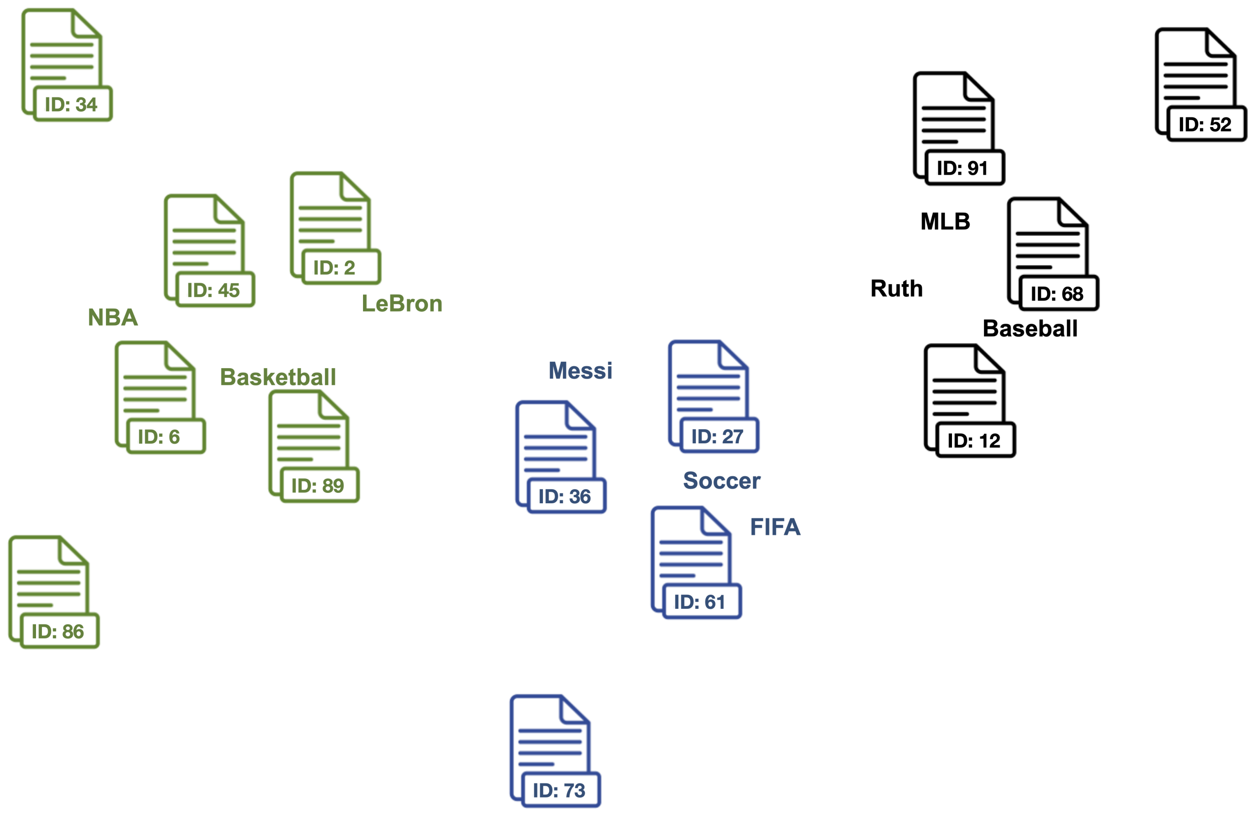
1. Create jointly embedded document and word vectors using Doc2Vec , Sentence-Transformers, or SimCSE.

Documents will be placed close to other similar documents and close to the most distinguishing words.


Ball
ID: 36
ID: 45
Court
ID: 80
ID: 65
ID: 23
ID: 56
Sports
ID: 5
Team
Hoop
ID: 67
Basketball
ID: 34
'The United States lost their first basketball match at the Olympics since 1988 on Sunday when Puerto Rico gave them a shock 92-73 trouncing.'
ID: 3
NBA
'US NBA players become the Nightmare Team after epic loss (AFP)'
ID: 8
Jordan
ID: 11
Score
ID: 14
Game
ID: 21
ID: 18
Player
LeBron
Points
ID: 38
ID: 47
ID: 87
ID: 55
Throw
'AFP - National Basketball Association players trying to win a fourth consecutive Olympic gold medal for the United States have gotten the wake-up call that the "Dream Team" days are done even if supporters have not.'

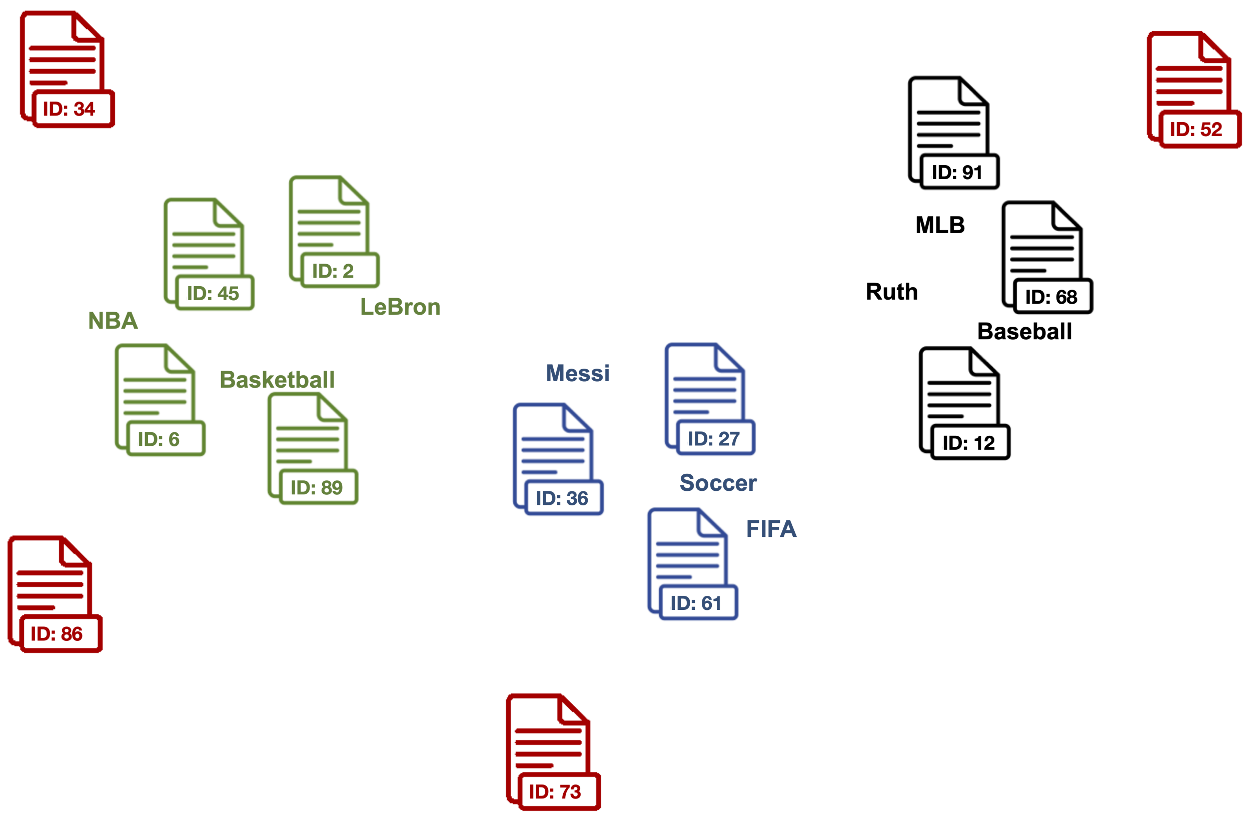
2. Find document vectors that are similar to the keyword vectors of each topic.

Each color represents a different topic described by the respective keywords.



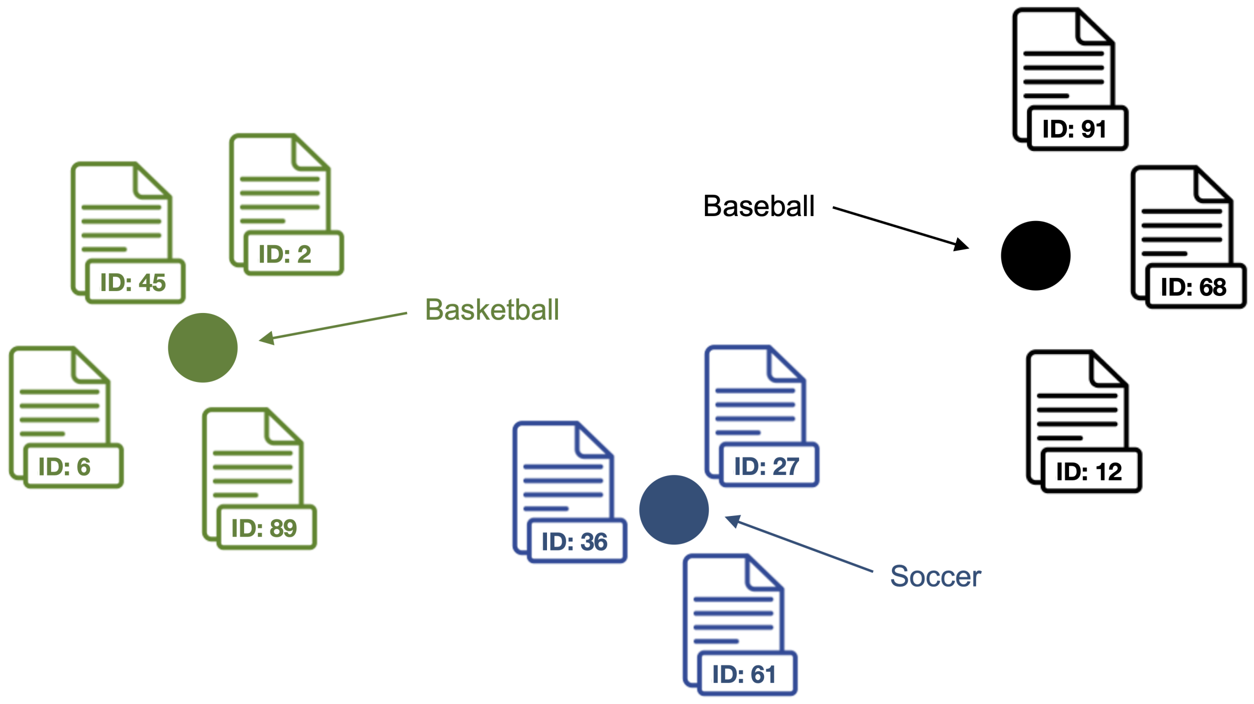
3. Clean outlier document vectors for each topic.

Red documents are outlier vectors that are removed and do not get used for calculating the label vector.



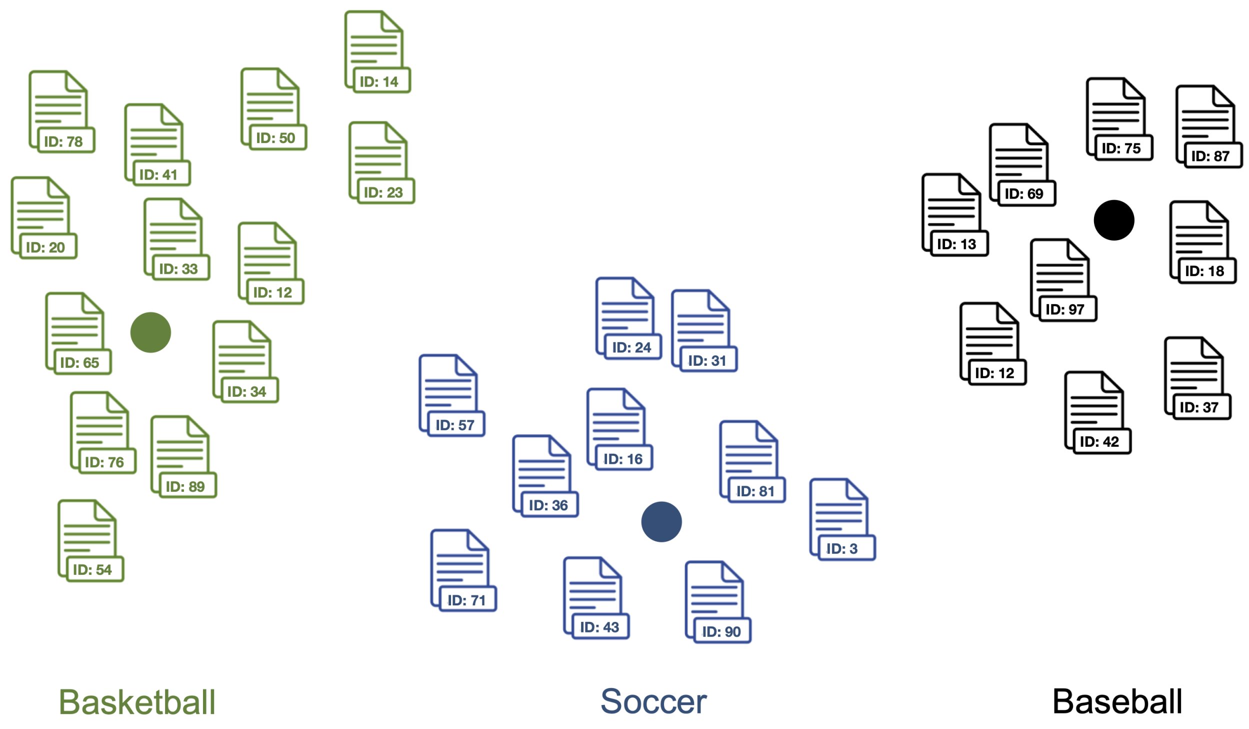
4. **Compute the centroid of the outlier cleaned document vectors as label vector for each topic.**

   Points represent the label vectors of the respective topics.



5. **Compute *label vector* <-> *document vector* similarities for each label vector and document vector in the dataset.**

   Documents are classified as topic with the highest *label vector* <-> *document vector* similarity.


ID: 14
ID: 78
ID: 50
ID: 41
ID: 23
ID: 20
ID: 33
ID: 12
ID: 65
ID: 34
ID: 76
ID: 89
ID: 54
ID: 24
ID: 31
ID: 57
ID: 16
ID: 36
ID: 81
ID: 3
ID: 71
ID: 43
ID: 90
ID: 75
ID: 87
ID: 69
ID: 13
ID: 18
ID: 97
ID: 12
ID: 37
ID: 42
Basketball
Soccer
Baseball

CHAPTER

FIVE

INSTALLATION

Back to Table of Contents

```
pip install lbl2vec
```

CHAPTER SIX

USAGE

Back to Table of Contents

For detailed information visit the API Guide and the examples.

6.1 Model Training

6.1.1 Lbl2Vec model

Back to Table of Contents

Lbl2Vec learns word vectors, document vectors and label vectors using Doc2Vec during training.

Train new Lbl2Vec model from scratch using Doc2Vec

```
from lbl2vec import Lbl2Vec

# init model
model = Lbl2Vec(keywords_list=descriptive_keywords, tagged_documents=tagged_docs)

# train model
model.fit()
```

Important parameters:

- `keywords_list`: iterable list of lists with descriptive keywords of type str. For each label at least one descriptive keyword has to be added as list of str.
- `tagged_documents`: iterable list of gensim.models.doc2vec.TaggedDocument elements. If you wish to train a new Doc2Vec model this parameter can not be None, whereas the `doc2vec_model` parameter must be None. If you use a pretrained Doc2Vec model this parameter has to be None. Input corpus, can be simply a list of elements, but for larger corpora, consider an iterable that streams the documents directly from disk/network.

Use word and document vectors from pretrained Doc2Vec model

Uses word vectors and document vectors from a pretrained Doc2Vec model to learn label vectors during Lbl2Vec model training.

```
from lbl2vec import Lbl2Vec

# init model
model = Lbl2Vec(keywords_list=descriptive_keywords, doc2vec_model=pretrained_d2v_model)

# train model
model.fit()
```

Important parameters:

- `keywords_list`: iterable list of lists with descriptive keywords of type str. For each label at least one descriptive keyword has to be added as list of str.
- `doc2vec_model`: pretrained gensim.models.doc2vec.Doc2Vec model. If given a pretrained Doc2Vec model, Lbl2Vec uses the pre-trained Doc2Vec model from this parameter. If this parameter is defined, `tagged_documents` parameter has to be None. In order to get optimal Lbl2Vec results the given Doc2Vec model should be trained with the parameters "dbow_words=1" and "dm=0".

6.1.2 Lbl2TransformerVec model

Back to Table of Contents

Lbl2TransformerVec learns word vectors, document vectors and label vectors using transformer-based language models during training. Using state-of-the-art transformer embeddings may not only yield to better predictions but also eliminates the issue of unknown keywords during model training. While the Doc2Vec-based model can only use keywords that Lbl2Vec has seen during training, the transformer-based Lbl2TransformerVec model can learn label vectors from any set of keywords. That is because transformer vocabularies consist of individual characters, subwords, and words, allowing transformers to effectively represent every word in a sentence. This eliminates the out-of-vocabulary scenario. However, using transformers instead of Doc2Vec is much more computationally expensive, especially if no GPU is available.

Train new Lbl2TransformerVec model from scratch using the default transformer-embedding model

```
from lbl2vec import Lbl2TransformerVec

# init model using the default transformer-embedding model ("sentence-transformers/all-
↪MiniLM-L6-v2")
model = Lbl2TransformerVec(keywords_list=listdescriptive_keywords, documents=document_
↪list)

# train model
model.fit()
```

Train Lbl2TransformerVec model using an arbitrary Sentence-Transformers embedding model

```
from lbl2vec import Lbl2TransformerVec
from sentence_transformers import SentenceTransformer

# select sentence-tranformers model
transformer_model = SentenceTransformer("all-mpnet-base-v2")

# init model
model = Lbl2TransformerVec(transformer_model=transformer_model, keywords_
↪list=listdescriptive_keywords,
                           documents=document_list)

# train model
model.fit()
```

Train Lbl2TransformerVec model using an arbitrary SimCSE embedding model

```
from lbl2vec import Lbl2TransformerVec
from transformers import AutoModel

# select SimCSE model
transformer_model = AutoModel.from_pretrained("princeton-nlp/sup-simcse-roberta-base")

# init model
model = Lbl2TransformerVec(transformer_model=transformer_model, keywords_
↪list=listdescriptive_keywords,
                           documents=document_list)

# train model
model.fit()
```

Important parameters:

- `keywords_list`: iterable list of lists with descriptive keywords of type str. For each label at least one descriptive keyword has to be added as list of str.
- `documents`: iterable list of text document elements (strings).
- `transformer_model`: Transformer model used to embed the labels, documents and keywords. The embedding models must be either of type sentence_transformers.SentenceTransformer or transformers.AutoModel

6.1.3 Document prediction

Back to Table of Contents

The prediction API calls are the same for Lbl2Vec and Lbl2TransformerVec.

Predict label similarities for documents used for training

Computes the similarity scores for each document vector stored in the model to each of the label vectors.

```
# get similarity scores from trained model
model.predict_model_docs()
```

Important parameters:

- `doc_keys`: list of document keys (optional). If None: return the similarity scores for all documents that are used to train the Lbl2Vec model. Else: only return the similarity scores of training documents with the given keys.

Predict label similarities for new documents that are not used for training

Computes the similarity scores for each given and previously unknown document vector to each of the label vectors from the model.

```
# get similarity scores for each new document from trained model
model.predict_new_docs(tagged_docs=tagged_docs)
```

Important parameters:

- `tagged_docs`: iterable list of gensim.models.doc2vec.TaggedDocument elements

6.1.4 Save and load models

Back to Table of Contents

The save and load API calls are the same for Lbl2Vec and Lbl2TransformerVec.

Save model to disk

```
model.save('model_name')
```

Load model from disk

```
model = Lbl2Vec.load('model_name')
```

CITATION INFORMATION

Back to Table of Contents

When citing Lbl2Vec [1,2] or Lbl2TransformerVec [3] in academic papers and theses, please use the following BibTeX entries:

```
@conference{schopf_etal_webist21,
author={Tim Schopf and Daniel Braun and Florian Matthes},
title={Lbl2Vec: An Embedding-based Approach for Unsupervised Document Retrieval on␣
↪Predefined Topics},
booktitle={Proceedings of the 17th International Conference on Web Information Systems␣
↪and Technologies - WEBIST,},
year={2021},
pages={124-132},
publisher={SciTePress},
organization={INSTICC},
doi={10.5220/0010710300003058},
isbn={978-989-758-536-4},
issn={2184-3252},
}
```

```
@InProceedings{10.1007/978-3-031-24197-0_4,
author="Schopf, Tim
and Braun, Daniel
and Matthes, Florian",
editor="Marchiori, Massimo
and Dom{\'i}nguez Mayo, Francisco Jos{\'e}
and Filipe, Joaquim",
title="Semantic Label Representations with Lbl2Vec: A Similarity-Based Approach␣
↪for Unsupervised Text Classification",
booktitle="Web Information Systems and Technologies",
year="2023",
publisher="Springer International Publishing",
address="Cham",
pages="59--73",
abstract="In this paper, we evaluate the Lbl2Vec approach for unsupervised text document␣
↪classification. Lbl2Vec requires only a small number of keywords describing the␣
↪respective classes to create semantic label representations. For classification,␣
↪Lbl2Vec uses cosine similarities between label and document representations, but no␣
↪annotation information. We show that Lbl2Vec significantly outperforms common␣
↪unsupervised text classification approaches and a widely used zero-shot text␣
↪classification approach. Furthermore, we show that using more precise keywords can␣
```

(continues on next page)

(continued from previous page)

```
↪significantly improve the classification results of similarity-based text␣
↪classification approaches.",
isbn="978-3-031-24197-0"
}
```

```
@inproceedings{schopf_etal_nlpir22,
author = {Schopf, Tim and Braun, Daniel and Matthes, Florian},
title = {Evaluating Unsupervised Text Classification: Zero-shot and Similarity-based␣
↪Approaches},
year = {2023},
publisher = {Association for Computing Machinery},
address = {New York, NY, USA},
booktitle = {2022 6th International Conference on Natural Language Processing and␣
↪Information Retrieval (NLPIR)},
keywords = {Natural Language Processing, Unsupervised Text Classification, Zero-shot␣
↪Text Classification},
location = {Bangkok, Thailand},
series = {NLPIR 2022}
}
```

CHAPTER EIGHT

LBL2VEC

class lbl2vec.lbl2vec.**Lbl2Vec**(*keywords_list: List[List[str]]*, *tagged_documents: Optional[List[TaggedDocument]] = None*, *label_names: Optional[List[str]] = None*, *epochs: int = 10*, *vector_size: int = 300*, *min_count: int = 50*, *window: int = 15*, *sample: float = 1e-05*, *negative: int = 5*, *workers: int = -1*, *doc2vec_model: Optional[Doc2Vec] = None*, *num_docs: Optional[int] = None*, *similarity_threshold: Optional[float] = None*, *similarity_threshold_offset: float = 0*, *min_num_docs: int = 1*, *clean_outliers: bool = False*, *verbose: bool = True*)

Creates jointly embedded label, document and word vectors. Once the model is trained it contains document and label vectors.

Parameters

- **keywords_list** (*iterable list of lists with descriptive keywords of type str.*) – For each label at least one descriptive keyword has to be added as list of str.
- **tagged_documents** (iterable list of gensim.models.doc2vec.TaggedDocument elements, optional) – If you wish to train word and document vectors from scratch this parameter can not be None, whereas the `doc2vec_model` parameter must be None. If you use a pretrained Doc2Vec model to load its learned word and document vectors this parameter has to be None. Input corpus, can be simply a list of elements, but for larger corpora, consider an iterable that streams the documents directly from disk/network.
- **label_names** (*iterable list of str, optional*) – Custom names can be defined for each label. Parameter values of label names and keywords of the same topic must have the same index. Default is to use generic label names.
- **epochs** (*int, optional*) – Number of iterations (epochs) over the corpus.
- **vector_size** (*int, optional*) – Dimensionality of the feature vectors.
- **min_count** (*int, optional*) – Ignores all words with total frequency lower than this.
- **window** (*int, optional*) – The maximum distance between the current and predicted word within a sentence.
- **sample** (*float, optional*) – The threshold for configuring which higher-frequency words are randomly downsampled, useful range is (0, 1e-5).
- **negative** (*int, optional*) – If > 0, negative sampling will be used, the int for negative specifies how many “noise words” should be drawn (usually between 5-20). If set to 0, no negative sampling is used.
- **workers** (*int, optional*) – The amount of worker threads to be used in training the model. Larger amount will lead to faster training. If set to -1, use all available worker threads of the machine.

- **`doc2vec_model`** (gensim.models.doc2vec.Doc2Vec, optional) – If given a pretrained Doc2Vec model, Lbl2Vec uses its word and document vectors to compute the label vectors. If this parameter is defined, `tagged_documents` has to be None. In order to get optimal Lbl2Vec results the given Doc2Vec model should be trained with the parameters "dbow_words=1" and "dm=0".
- **`num_docs`** (*`int, optional`*) – Maximum number of documents to calculate label embedding from. Default is all available documents.
- **`similarity_threshold`** (*`float, default=None`*) – Only documents with a higher similarity to the respective description keywords than this threshold are used to calculate the label embeddings.
- **`similarity_threshold_offset`** (*`float, default=0`*) – Sets similarity threshold to n-similarity_threshold_offset with n = (smiliarity of keyphrase_vector to most similar document_vector).
- **`min_num_docs`** (*`int, optional`*) – Minimum number of documents that are used to calculate the label embedding. Adds documents until requirement is fulfilled if simiarilty threshold is choosen too restrictive. This value should be chosen to be at least 1 in order to be able to calculate the label embedding. If this value is < 1 it can happen that no document is selected for label embedding calculation and therefore no label embedding is generated.
- **`clean_outliers`** (*`boolean, optional`*) – Whether to clean outlier candidate documents for label embedding calculation. Setting to False can shorten the training time. However, a loss of accuracy in the calculation of the label vectors may be possible.
- **`verbose`** (*`boolean, optional`*) – Whether to print status during training and prediction.

`add_lbl_thresholds`(*lbl_similarities_df: DataFrame*, *lbl_thresholds: List[Tuple[str, float]]*) → DataFrame

Adds threshold column with the threshold value of the most similar classification label.

Parameters

- **`lbl_similarities_df`** (pandas.DataFrame with first column of document keys, second column of most similar labels, third column of similarity scores of the document to the most similar label and the following columns with the respective labels and the similarity scores of the documents to the labels.) – This pandas.DataFrame type is returned by the *`predict_model_docs()`* and *`predict_new_docs()`* functions.
- **`lbl_thresholds`** (*`list of tuples`*) – First tuple element consists of the label name and the second tuple element of the threshold value.

Returns

lbl_similarities_df

Return type

pandas.DataFrame with first column of document keys, second column of most similar labels, third column of similarity scores of the document to the most similar label, fourth column of the label threshold values and the following columns with the respective labels and the similarity scores of the documents to the labels.

`fit`()

Trains the Lbl2Vec model which creates jointly embedded label, document and word vectors.

`classmethod load`(*filepath: str*) → object

Loads the Lbl2Vec model from disk.

Parameters

`filepath` (*`str`*) – Path of file.

Returns
lbl2vec_model

Return type
Lbl2Vec model loaded from disk.

predict_model_docs(*doc_keys: Optional[Union[List[int], List[str]]] = None*, *multiprocessing: bool = False*) → DataFrame

Computes similarity scores of documents that are used to train the Lbl2Vec model to each label.

Parameters

- **doc_keys** (`list of document keys, optional`) – If None: return the similarity scores for all documents that are used to train the Lbl2Vec model. Else: only return the similarity scores of training documents with the given keys.
- **multiprocessing** (`boolean, optional`) – Whether to use the ray multiprocessing library during prediction. If True, ray uses all available workers for prediction. If False, just use a single core for prediction.

Returns
labeled_docs

Return type
pandas.DataFrame with first column of document keys, second column of most similar labels, third column of similarity scores of the document to the most similar label and the following columns with the respective labels and the similarity scores of the documents to the labels. The similarity scores consist of cosine similarities and therefore have a value range of [-1,1].

predict_new_docs(*tagged_docs: List[TaggedDocument]*, *multiprocessing: bool = False*) → DataFrame

Computes similarity scores of given new documents that are not used to train the Lbl2Vec model to each label.

Parameters

- **tagged_docs** (iterable list of gensim.models.doc2vec.TaggedDocument elements) – New documents that are not used to train the model.
- **multiprocessing** (`boolean, optional`) – Whether to use the ray multiprocessing library during prediction. If True, ray uses all available workers for prediction. If False, just use a single core for prediction.

Returns
labeled_docs

Return type
pandas.DataFrame with first column of document keys, second column of most similar labels, third column of similarity scores of the document to the most similar label and the following columns with the respective labels and the similarity scores of the documents to the labels. The similarity scores consist of cosine similarities and therefore have a value range of [-1,1].

save(*filepath: str*)

Saves the Lbl2Vec model to disk.

Parameters
filepath (`str`) – Path of file.

CHAPTER

NINE

LBL2TRANSFORMERVEC

class lbl2vec.lbl2transformervec.**Lbl2TransformerVec**(*keywords_list: ~typing.List[~typing.List[str]], documents: ~typing.List[str], transformer_model: ~typing.Union[~sentence_transformers.SentenceTransformer.SentenceT ~transformers.models.auto.modeling_auto.AutoModel] = SentenceTransformer((0): Transformer({'max_seq_length': 256, 'do_lower_case': False}) with Transformer model: BertModel (1): Pooling({'word_embedding_dimension': 384, 'pooling_mode_cls_token': False, 'pooling_mode_mean_tokens': True, 'pooling_mode_max_tokens': False, 'pooling_mode_mean_sqrt_len_tokens': False}) (2): Normalize()), label_names: ~typing.Optional[~typing.List[str]] = None, similarity_threshold: ~typing.Optional[float] = None, similarity_threshold_offset: float = 0, min_num_docs: int = 1, max_num_docs: ~typing.Optional[int] = None, clean_outliers: bool = False, workers: int = -1, device: ~torch.device = device(type='cpu'), verbose: bool = True*)

Creates jointly embedded label and document vectors with transformer language models. Once the model is trained it contains document and label vectors.

Parameters

- **keywords_list** (*iterable list of lists with descriptive keywords of type str*) – For each label at least one descriptive keyword has to be added as list of str.
- **documents** (*iterable list of strings*) – Iterable list of text documents
- **transformer_model** (Union[SentenceTransformer, transformers.AutoModel], default=SentenceTransformer('all-MiniLM-L6-v2')) – Transformer model used to embed the labels, documents and keywords.
- **label_names** (*iterable list of str, default=None*) – Custom names can be defined for each label. Parameter values of label names and keywords of the same topic must have the same index. Default is to use generic label names.
- **similarity_threshold** (*float, default=None*) – Only documents with a higher similarity to the respective description keywords than this threshold are used to calculate the

label embeddings.

- **similarity_threshold_offset** (*float, default=0*) – Sets similarity threshold to n-similarity_threshold_offset with n = (smiliarity of keyphrase_vector to most similar document_vector).
- **min_num_docs** (*int, default=1*) – Minimum number of documents that are used to calculate the label embedding. Adds documents until requirement is fulfilled if simiarilty threshold is choosen too restrictive. This value should be chosen to be at least 1 in order to be able to calculate the label embedding. If this value is < 1 it can happen that no document is selected for label embedding calculation and therefore no label embedding is generated.
- **max_num_docs** (*int, default=None*) – Maximum number of documents to calculate label embedding from. Default is all available documents.
- **clean_outliers** (*boolean, default=False*) – Whether to clean outlier candidate documents for label embedding calculation. Setting to False can shorten the training time. However, a loss of accuracy in the calculation of the label vectors may be possible.
- **workers** (*int, default=-1*) – Use these many worker threads to train the model (=faster training with multicore machines). Setting this parameter to -1 uses all available cpu cores. If using GPU, this parameter is ignored.
- **device** (*torch.device, default=torch.device('cpu')*) – Specify the device that should be used for training the model. Default is to use the CPU device. To use CPU, set device to 'torch.device('cpu')'. To use GPU, you can e.g. specify 'torch.device('cuda:0')'.
- **verbose** (*boolean, default=True*) – Whether to print status during training and prediction.

add_lbl_thresholds(*lbl_similarities_df: DataFrame*, *lbl_thresholds: List[Tuple[str, float]]*) → DataFrame

Adds threshold column with the threshold value of the most similar classification label.

Parameters

- **lbl_similarities_df** (pandas.DataFrame with first column of document keys, second column of most similar labels, third column of similarity scores of the document to the most similar label and the following columns with the respective labels and the similarity scores of the documents to the labels.) – This pandas.DataFrame type is returned by the *predict_model_docs()* and *predict_new_docs()* functions.
- **lbl_thresholds** (*list of tuples*) – First tuple element consists of the label name and the second tuple element of the threshold value.

Returns

lbl_similarities_df

Return type

pandas.DataFrame with first column of document keys, second column of most similar labels, third column of similarity scores of the document to the most similar label, fourth column of the label threshold values and the following columns with the respective labels and the similarity scores of the documents to the labels.

fit()

Trains the Lbl2TransformerVec model which creates jointly embedded label and document vectors.

classmethod load(*filepath: str*) → object

Loads the Lbl2Vec model from disk.

Parameters

filepath (*str*) – Path of file.

Returns
lbl2vec_model

Return type
Lbl2Vec model loaded from disk.

`predict_model_docs`(*doc_idxs: Optional[List[int]] = None*) → DataFrame

Computes similarity scores of documents that are used to train the Lbl2TransformerVec model to each label.

Parameters
`doc_idxs` (*list of document indices, default=None*) – If None: return the similarity scores for all documents that are used to train the Lbl2TransformerVec model. Else: only return the similarity scores of training documents with the given indices.

Returns
labeled_docs

Return type
pandas.DataFrame with first column of document texts, second column of most similar labels, third column of similarity scores of the document to the most similar label and the following columns with the respective labels and the similarity scores of the documents to the labels. The similarity scores consist of cosine similarities and therefore have a value range of [-1,1].

`predict_new_docs`(*documents: List[str]*, *workers: int = -1*, *device: device = device(type='cpu')*) → DataFrame

Computes similarity scores of given new documents that are not used to train the Lbl2TransformerVec model to each label.

Parameters

- `documents` (*iterable list of strings*) – New documents that are not used to train the model.
- `workers` (*int, default=-1*) – Use these many worker threads to train the model (=faster training with multicore machines). Setting this parameter to -1 uses all available cpu cores. If using GPU, this parameter is ignored.
- `device` (*torch.device, default=torch.device('cpu')*) – Specify the device that should be used for training the model. Default is to use the CPU device. To use CPU, set device to 'torch.device('cpu')'. To use GPU, you can e.g. specify 'torch.device('cuda:0')'.

Returns
labeled_docs

Return type
pandas.DataFrame with first column of document texts, second column of most similar labels, third column of similarity scores of the document to the most similar label and the following columns with the respective labels and the similarity scores of the documents to the labels. The similarity scores consist of cosine similarities and therefore have a value range of [-1,1].

`save`(*filepath: str*)

Saves the Lbl2Vec model to disk.

Parameters
`filepath` (*str*) – Path of file.

CHAPTER TEN

INDICES AND TABLES

- genindex
- modindex
- search

PYTHON MODULE INDEX

l

lbl2vec.lbl2transformervec, 25
lbl2vec.lbl2vec, 21

INDEX

A

`add_lbl_thresholds()` (*lbl2vec.lbl2transformervec.Lbl2TransformerVec method*), 26
`add_lbl_thresholds()` (*lbl2vec.lbl2vec.Lbl2Vec method*), 22

F

`fit()` (*lbl2vec.lbl2transformervec.Lbl2TransformerVec method*), 26
`fit()` (*lbl2vec.lbl2vec.Lbl2Vec method*), 22

L

`Lbl2TransformerVec` (*class in lbl2vec.lbl2transformervec*), 25
`Lbl2Vec` (*class in lbl2vec.lbl2vec*), 21
`lbl2vec.lbl2transformervec`
  `module`, 25
`lbl2vec.lbl2vec`
  `module`, 21
`load()` (*lbl2vec.lbl2transformervec.Lbl2TransformerVec class method*), 26
`load()` (*lbl2vec.lbl2vec.Lbl2Vec class method*), 22

M

`module`
  `lbl2vec.lbl2transformervec`, 25
  `lbl2vec.lbl2vec`, 21

P

`predict_model_docs()` (*lbl2vec.lbl2transformervec.Lbl2TransformerVec method*), 27
`predict_model_docs()` (*lbl2vec.lbl2vec.Lbl2Vec method*), 23
`predict_new_docs()` (*lbl2vec.lbl2transformervec.Lbl2TransformerVec method*), 27
`predict_new_docs()` (*lbl2vec.lbl2vec.Lbl2Vec method*), 23

S

`save()` (*lbl2vec.lbl2transformervec.Lbl2TransformerVec method*), 27
`save()` (*lbl2vec.lbl2vec.Lbl2Vec method*), 23