Contextualized Topic Models Documentation

Release 2.5.0

Federico Bianchi

Apr 24, 2023

CONTENTS:

CHAPTER

ONE

CONTEXTUALIZED TOPIC MODELS

Contextualized Topic Models (CTM) are a family of topic models that use pre-trained representations of language (e.g., BERT) to support topic modeling. See the papers for details:

- Bianchi, F., Terragni, S., & Hovy, D. (2021). *Pre-training is a Hot Topic: Contextualized Document Embeddings Improve Topic Coherence*. ACL. https://aclanthology.org/2021.acl-short.96/
- Bianchi, F., Terragni, S., Hovy, D., Nozza, D., & Fersini, E. (2021). *Cross-lingual Contextualized Topic Models with Zero-shot Learning*. EACL. https://www.aclweb.org/anthology/2021.eacl-main.143/

1.1 Topic Modeling with Contextualized Embeddings

Our new topic modeling family supports many different languages (i.e., the one supported by HuggingFace models) and comes in two versions: **CombinedTM** combines contextual embeddings with the good old bag of words to make more coherent topics; **ZeroShotTM** is the perfect topic model for task in which you might have missing words in the test data and also, if trained with muliglingual embeddings, inherits the property of being a multilingual topic model!

The big advantage is that you can use different embeddings for CTMs. Thus, when a new embedding method comes out you can use it in the code and improve your results. We are not limited by the BoW anymore.

We also have kitty! a new submodule that can be used to quickly create an human in the loop classifier to quickly classify your documents and create named clusters.

1.2 Tutorials

You can look at our medium blog post or start from one of our Colab Tutorials:

Name	Link
Combined TM on Wikipedia Data (Preproc+Saving+Viz) (stable **v2.2.0**)	
Zero-Shot Cross-lingual Topic Modeling (Preproc+Viz) (stable **v2.2.0**)	
Kitty: Human in the loop Classifier (High-level usage) (stable **v2.2.0**)	
SuperCTM and -CTM (High-level usage) (stable **v2.2.0**)	

1.3 Overview

1.3.1 TL;DR

- In CTMs we have two models. CombinedTM and ZeroShotTM, which have different use cases.
- CTMs work better when the size of the bag of words **has been restricted to a number of terms** that does not go over **2000 elements**. This is because we have a neural model that reconstructs the input bag of word, Moreover, in CombinedTM we project the contextualized embedding to the vocab space, the bigger the vocab the more parameters you get, with the training being more difficult and prone to bad fitting. This is **NOT** a strict limit, however, consider preprocessing your dataset. We have a preprocessing pipeline that can help you in dealing with this.
- Check the contextual model you are using, the **multilingual model one used on English data might not give results that are as good** as the pure English trained one.
- **Preprocessing is key**. If you give a contextual model like BERT preprocessed text, it might be difficult to get out a good representation. What we usually do is use the preprocessed text for the bag of word creating and use the NOT preprocessed text for BERT embeddings. Our preprocessing class can take care of this for you.

1.3.2 Features

An important aspect to take into account is which network you want to use: the one that combines contextualized embeddings and the BoW (CombinedTM) or the one that just uses contextualized embeddings (ZeroShotTM)

But remember that you can do zero-shot cross-lingual topic modeling only with the `ZeroShotTM` model. See cross-lingual-topic-modeling

We also have `Kitty`: a utility you can use to do a simpler human in the loop classification of your documents. This can be very useful to do document filtering. It also works in cross-lingual setting and thus you might be able to filter documents in a language you don't know!

1.4 References

If you find this useful you can cite the following papers :)

ZeroShotTM

```
@inproceedings{bianchi-etal-2021-cross,
    title = "Cross-lingual Contextualized Topic Models with Zero-shot Learning",
    author = "Bianchi, Federico and Terragni, Silvia and Hovy, Dirk  and
      Nozza, Debora and Fersini, Elisabetta",
    booktitle = "Proceedings of the 16th Conference of the European Chapter of the␣
↪Association for Computational Linguistics: Main Volume",
    month = apr,
    year = "2021",
    address = "Online",
    publisher = "Association for Computational Linguistics",
    url = "https://www.aclweb.org/anthology/2021.eacl-main.143",
    pages = "1676--1683",
}
```

CombinedTM

```
@inproceedings{bianchi-etal-2021-pre,
    title = "Pre-training is a Hot Topic: Contextualized Document Embeddings Improve␣
↪Topic Coherence",
    author = "Bianchi, Federico  and
      Terragni, Silvia  and
      Hovy, Dirk",
    booktitle = "Proceedings of the 59th Annual Meeting of the Association for␣
↪Computational Linguistics and the 11th International Joint Conference on Natural␣
↪Language Processing (Volume 2: Short Papers)",
    month = aug,
    year = "2021",
    address = "Online",
    publisher = "Association for Computational Linguistics",
    url = "https://aclanthology.org/2021.acl-short.96",
    doi = "10.18653/v1/2021.acl-short.96",
    pages = "759--766",
}
```

1.5 Development Team

- Federico Bianchi <f.bianchi@unibocconi.it> Bocconi University
- Silvia Terragni <s.terragni4@campus.unimib.it> University of Milan-Bicocca
- Dirk Hovy <dirk.hovy@unibocconi.it> Bocconi University

1.6 Software Details

- Free software: MIT license
- Documentation: https://contextualized-topic-models.readthedocs.io.
- Super big shout-out to Stephen Carrow for creating the awesome https://github.com/estebandito22/PyTorchAVITM package from which we constructed the foundations of this package. We are happy to redistribute this software again under the MIT License.

1.7 Credits

This package was created with Cookiecutter and the audreyr/cookiecutter-pypackage project template. To ease the use of the library we have also included the rbo package, all the rights reserved to the author of that package.

1.8 Note

Remember that this is a research tool :)

CHAPTER

TWO

DATA PREPARATION

One of the most fundamental pieces in CTM is the data preparation. This allows us to generate embeddings and to use them to train the variational autoencoder.

2.1 Introduction

This entire process is regulated by the TopicModelDataPreparation class.

```
from contextualized_topic_models.utils.data_preparation import TopicModelDataPreparation

# we first need to get an embedding model. This is based on distilroberta and trained on␣
↪paraphrase
# data.
qt = TopicModelDataPreparation("paraphrase-distilroberta-base-v2")
```

Next, we need some data. This is just an example of the data we can pass to CTM:

```
text_for_contextual = [
    "hello, this is unpreprocessed text you can give to the model",
    "have fun with our topic model",
]

text_for_bow = [
    "hello unpreprocessed give model",
    "fun topic model",
]
```

You see that we have two lists: the first one contains the original documents while the second one contains the bag of words representation that is going to be used to generate our topic words. The original documents are the ones that are passed to the embedding model to create the contextualized representations.

To generate the embeddings you can simply run

```
training_dataset = qt.fit(text_for_contextual=list_of_unpreprocessed_documents,
text_for_bow=list_of_preprocessed_documents)
```

2.2 Using Custom Embeddings

Note that you can also use your own Custom Embeddings if you want. You just need to change the way you fit the TopicModelDataPreparation object. Setting custom_embeddings to an array will skip the use of the contextual model to generate the embeddings and the embedding you will pass will be used.

```
def fit(self, text_for_contextual, text_for_bow, labels=None, custom_embeddings=None):
```

2.3 SBERT

Underneath we are using SBERT, you should now that some SBERT models truncate your document to a max length. You should check this if you want to know more.

Nonetheless, changing the max length in CTM is easy:

```
from contextualized_topic_models.utils.data_preparation import TopicModelDataPreparation

# we first need to get an embedding model. This is based on distilroberta and trained on␣
↪paraphrase
# data.
qt = TopicModelDataPreparation("paraphrase-distilroberta-base-v2", max_seq_length=200)
```

CHAPTER

THREE

COMBINEDTM: COHERENT TOPIC MODELS

Combined TM combines the BoW with SBERT, a process that seems to increase the coherence of the predicted topics (https://aclanthology.org/2021.acl-short.96/).

3.1 Usage

Here is how you can use the CombinedTM. This is a standard topic model that also uses contextualized embeddings. The good thing about CombinedTM is that it makes your topic much more coherent (see the paper https://arxiv.org/abs/2004.03974). n_components=50 specifies the number of topics.

```
from contextualized_topic_models.models.ctm import CombinedTM
from contextualized_topic_models.utils.data_preparation import TopicModelDataPreparation
from contextualized_topic_models.utils.data_preparation import bert_embeddings_from_file

qt = TopicModelDataPreparation("paraphrase-distilroberta-base-v2")

training_dataset = qt.fit(text_for_contextual=list_of_unpreprocessed_documents, text_for_
↪bow=list_of_preprocessed_documents)

ctm = CombinedTM(bow_size=len(qt.vocab), contextual_size=768, n_components=50) # 50␣
↪topics

ctm.fit(training_dataset) # run the model

ctm.get_topics()
```

Once the model is trained, it is very easy to get the topics!

```
ctm.get_topics()
```

3.2 Creating the Test Set

The **transform** method will take care of most things for you, for example the generation of a corresponding BoW by considering only the words that the model has seen in training.

If you use **CombinedTM** you need to include the test text for the BOW:

```
testing_dataset = qt.transform(text_for_contextual=testing_text_for_contextual, text_for_
↪bow=testing_text_for_bow)

# n_sample how many times to sample the distribution (see the doc)
ctm.get_doc_topic_distribution(testing_dataset, n_samples=20) # returns a (n_documents,␣
↪n_topics) matrix with the topic distribution of each document
```

3.2.1 Warning

Note that the way we use the transform method here is different from what we do for ZeroShotTM! This is very important!

3.2.2 Tutorial

You can find a tutorial here: it will show you how you can use CombinedTM.

CHAPTER

FOUR

ZEROSHOTTM: TOPIC MODELING WITH MISSING WORDS AND CROSS-LINGUAL CAPABILITIES

Our ZeroShotTM can be used for zero-shot topic modeling. This is because the entire document is encoded into an embedding using a contextualized model. Thus, we are not limited by the usual problems you might encounter with bag of words: at test time, words that are missing from the training set will be encoded using the contextualized model, thus providing a reliable topic model even in sparse context!

More interestingly, this model can be used for cross-lingual topic modeling (See next sections)! You can also read paper (https://www.aclweb.org/anthology/2021.eacl-main.143)

4.1 Usage

```
from contextualized_topic_models.models.ctm import ZeroShotTM
from contextualized_topic_models.utils.data_preparation import TopicModelDataPreparation
from contextualized_topic_models.utils.data_preparation import bert_embeddings_from_file

text_for_contextual = [
    "hello, this is unpreprocessed text you can give to the model",
    "have fun with our topic model",
]

text_for_bow = [
    "hello unpreprocessed give model",
    "fun topic model",
]

qt = TopicModelDataPreparation("paraphrase-multilingual-mpnet-base-v2")

training_dataset = qt.fit(text_for_contextual=text_for_contextual, text_for_bow=text_for_
↪bow)

ctm = ZeroShotTM(bow_size=len(qt.vocab), contextual_size=768, n_components=50)

ctm.fit(training_dataset) # run the model

ctm.get_topics(2)
```

As you can see, the high-level API to handle the text is pretty easy to use; **text_for_bert** should be used to pass to the model a list of documents that are not preprocessed. Instead, to **text_for_bow** you should pass the preprocessed text used to build the BoW.

Once the model is trained, it is very easy to get the topics!

```
ctm.get_topics()
```

4.2 Creating the Test Set

The **transform** method will take care of most things for you, for example the generation of a corresponding BoW by considering only the words that the model has seen in training. However, this comes with some bumps when dealing with the ZeroShotTM, as we will se in the next section.

If you use **ZeroShotTM** you do not need to use the *testing_text_for_bow* because if you are using a different set of test documents, this will create a BoW of a different size. Thus, the best way to do this is to pass just the text that is going to be given in input to the contexual model:

```
testing_dataset = qt.transform(text_for_contextual=testing_text_for_contextual)

# n_sample how many times to sample the distribution (see the doc)
ctm.get_doc_topic_distribution(testing_dataset, n_samples=20)
```

4.2.1 Warning

Note that the way we use the transform method here is different from what we do for CombinedTM! This is very important!

4.3 Cross-Lingual Topic Modeling

Once you have trained the ZeroShotTM model with multilingual embeddings, you can use this simple pipeline to predict the topics for documents in a different language (as long as this language is covered by **paraphrase-multilingual-mpnet-base-v2**).

```
# here we have a Spanish document
testing_text_for_contextual = [
    "hola, bienvenido",
]

# since we are doing multilingual topic modeling, we do not need the BoW in
# ZeroShotTM when doing cross-lingual experiments (it does not make sense, since we␣
↪trained with an english Bow
# to use the spanish BoW)
testing_dataset = qt.transform(testing_text_for_contextual)

# n_sample how many times to sample the distribution (see the doc)
ctm.get_doc_topic_distribution(testing_dataset, n_samples=20) # returns a (n_documents,␣
↪n_topics) matrix with the topic distribution of each document
```

Advanced Notes: We do not need to pass the Spanish bag of word: the bag of words of the two languages will not be comparable! We are passing it to the model for compatibility reasons, but you cannot get the output of the model (i.e., the predicted BoW of the trained language) and compare it with the testing language one.

CHAPTER

FIVE

KITTY: HUMAN-IN-THE-LOOP CLASSIFICATION WITH CONTEXTUALIZED TOPIC MODELS

Kitty is a utility to generate a simple topic classifier from a topic model. It first runs a CTM instance on the data for you and you can then select and label a set of topics of interest. Once this is done, you can apply this selection to a wider range of documents.

Please cite the following papers if you use Kitty:

- Bianchi, F., Terragni, S., & Hovy, D. (2021). *Pre-training is a Hot Topic: Contextualized Document Embeddings Improve Topic Coherence*. ACL. https://aclanthology.org/2021.acl-short.96/
- Bianchi, F., Terragni, S., Hovy, D., Nozza, D., & Fersini, E. (2021). *Cross-lingual Contextualized Topic Models with Zero-shot Learning*. EACL. https://www.aclweb.org/anthology/2021.eacl-main.143/

To simply put it, Kitty makes use of ZeroShotTM to extract topic from your data. Kitty runs some very simple preprocessing on your data to remove words that might not be too useful. We also have a google colab tutorial.

5.1 Usage

In this example we use an english embedding model, however you might need langauge-specific models to do this, check out the related section of the documentation

```
from contextualized_topic_models.models.kitty_classifier import Kitty

# read the training data
training_set = list(map(lambda x : x.strip(), open("train_data").readlines()))

kt = Kitty()
kt.train(training, topics=5, epochs=1, stopwords_list=["stop", "words"], embedding_model=
↪"paraphrase-distilroberta-base-v2")

print(kt.pretty_print_word_classes())
```

This could probably output topics like these ones:

```
0   family, plant, types, type, moth
1   district, mi, area, village, west
2   released, series, television, album, film
3   school, station, historic, public, states
4   born, football, team, played, season
```

Now, you can then use a simple dictionary to assign the topics to some labels. For example, topic 0 seems to be describing nature related things.

```
kt.assigned_classes = {0 : "nature", 1 : "location",
                       2 : "entertainment", 3 : "shop/offices", 4: "sport"}

kt.predict(["the village of Puza is a very nice village in Italy"])

>> location

kt.predict(["Pussetto is a soccer player that currently plays for Udiense Calcio"])

>> sport
```

If you are using a jupyter notebook, you can use the widget to fill in the labels.

```
kt.widget_annotation()
```

5.2 Cross-Lingual Support

A nice feature of Kitty is that it can be used to filter documents in different languages. Assume you have access to a large corpus of Italian documents and a smaller corpus of English documents. You can run Kitty on the English documents, map the labels and apply Kitty on the Italian documents. It is enough to change the embedding model.

```
from contextualized_topic_models.models.kitty_classifier import Kitty

# read the training data
training = list(map(lambda x : x.strip(), open("train_data").readlines()))

# define kitty with a multilingual embedding model
kt = Kitty(embedding_model="paraphrase-multilingual-mpnet-base-v2",  contextual_size=768)

kt.train(training, 5, stopwords_list=["stopwords"]) # train a topic model with 5 topics

print(kt.pretty_print_word_classes())
```

You can then apply the mapping as we did before and predict in different languages:

```
kt.predict(["Pussetto è un calciatore che attualmente gioca per l'Udinese Calcio"])

>> sport
```

You should refer to SBERT Pretrained Models to know if the languages you want to use are supported by SBERT.

5.3 Using Custom Embeddings with Kitty

Do you have custom embeddings and want to use them for faster results? Just give them to Kitty!

```
from contextualized_topic_models.models.kitty_classifier import Kitty
import numpy as np

# read the training data
training_data = list(map(lambda x : x.strip(), open("train_data").readlines()))
customer_embeddings = np.load('customer_embeddings.npy')

kt = Kitty()
kt.train(training_data, custom_embeddings=customer_embeddings, stopwords_list=["stopwords
↪"])

print(kt.pretty_print_word_classes())
```

Note: Custom embeddings must be numpy.arrays.

5.4 What Makes Kitty Different from Other Topic Models?

Nothing! It just offers a user-friendly utility that makes use of the ZeroShotTM model in the backend.

CHAPTER

SIX

EXTENSIONS: SUPERCTM AND -CTM

6.1 SuperCTM

Inspiration for SuperCTM has been taken directly from the work by Card et al., 2018 (you can read this as "we essentially implemented their approach in our architecture"). SuperCTM should give better representations of the documents - this is somewhat expected, since we are using the labels to give more information to the model - and in theory should also make the model able to find topics more coherent with respect to the labels. The model is super easy to use and requires minor modifications to the already implemented pipeline:

```
from contextualized_topic_models.models.ctm import ZeroShotTM
from contextualized_topic_models.utils.data_preparation import TopicModelDataPreparation

text_for_contextual = [
    "hello, this is unpreprocessed text you can give to the model",
    "have fun with our topic model",
]

text_for_bow = [
    "hello unpreprocessed give model",
    "fun topic model",
]

labels = [0, 1] # we need to have  a label for each document

qt = TopicModelDataPreparation("paraphrase-multilingual-mpnet-base-v2")

# training dataset should contain the labels
training_dataset = qt.fit(text_for_contextual=text_for_contextual, text_for_bow=text_for_
↪bow, labels=labels)

# model should know the label size in advance
ctm = CombinedTM(bow_size=len(qt.vocab), contextual_size=768, n_components=50, label_
↪size=len(set(labels)))

ctm.fit(training_dataset) # run the model

ctm.get_topics(2)
```

6.2 -CTM

We also implemented the intuition found in the work by Higgins et al., 2018, where a weight is applied to the KL loss function. The idea is that giving more weight to the KL part of the loss function helps in creating disentangled representations by forcing independence in the components. Again, the model should be straightforward to use:

```
ctm = CombinedTM(bow_size=len(qt.vocab), contextual_size=768, n_components=50, loss_
↪weights={"beta" : 3})
```

CHAPTER

SEVEN

EVALUATION

We have also included some of the metrics normally used in the evaluation of topic models, for example you can compute the coherence of your topics using NPMI using our simple and high-level API.

7.1 Metrics Covered

The metrics we cover are the one we also describe in our papers.

- Coherence (e.g., NPMI, Word Embeddings)
- Topic Diversity (e.g., Inversed RBO)
- Matches
- Centroid Distance

You can take a look at the evaluation measures file to get a better idea on how to add them to your code.

7.2 Example

If you want to compute the NPMI coherence you can simply do as follows:

```
from contextualized_topic_models.evaluation.measures import CoherenceNPMI

with open('preprocessed_documents.txt', "r") as fr:
    texts = [doc.split() for doc in fr.read().splitlines()] # load text for NPMI

npmi = CoherenceNPMI(texts=texts, topics=ctm.get_topic_lists(10))
npmi.score()
```

CHAPTER

EIGHT

MONO AND MULTI-LINGUAL EMBEDDINGS

8.1 Multilingual

Some of the examples below use a multilingual embedding model `paraphrase-multilingual-mpnet-base-v2`. This means that the representations you are going to use are mutlilinguals. However you might need a broader coverage of languages. In that case, you can check SBERT to find a model you can use.

8.2 English

If you are doing topic modeling in English, **you SHOULD use an English sentence-bert model**, for example *paraphrase-distilroberta-base-v2*. In that case, it's really easy to update the code to support monolingual English topic modeling. If you need other models you can check SBERT for other models.

```
qt = TopicModelDataPreparation("paraphrase-distilroberta-base-v2")
```

8.3 Language-Specific

In general, our package should be able to support all the models described in the sentence transformer package and in HuggingFace. You need to take a look at HuggingFace models and find which is the one for your language. For example, for Italian, you can use UmBERTo. How to use this in the model, you ask? well, just use the name of the model you want instead of the english/multilingual one:

```
qt = TopicModelDataPreparation("Musixmatch/umberto-commoncrawl-cased-v1")
```

CHAPTER

NINE

PUBLISHED PAPERS ON CTM

Contextualized Topic Models (CTM) are a family of topic models that use pre-trained representations of language (e.g., BERT) to support topic modeling. See the papers for details:

- Bianchi, F., Terragni, S., & Hovy, D. (2021). *Pre-training is a Hot Topic: Contextualized Document Embeddings Improve Topic Coherence*. ACL. https://aclanthology.org/2021.acl-short.96/
- Bianchi, F., Terragni, S., Hovy, D., Nozza, D., & Fersini, E. (2021). *Cross-lingual Contextualized Topic Models with Zero-shot Learning*. EACL. https://www.aclweb.org/anthology/2021.eacl-main.143/

If you want to replicate our results, you can use our code. You will find the W1 dataset in the colab and here: https://github.com/vinid/data, if you need the W2 dataset, send us an email (it is a bit bigger than W1 and we could not upload it on github).

Note: Thanks to Camille DeJarnett (Stanford/CMU), Xinyi Wang (CMU), and Graham Neubig (CMU) we found out that with the current version of the package and all the dependencies (e.g., the sentence transformers embedding model, CUDA version, PyTorch version), results with the model *distiluse-base-multilingual-cased* are lower than what appears in the paper. We suggest to use *paraphrase-multilingual-mpnet-base-v2* which is a newer multilingual model that has results that are higher than those in the paper.

See for example the results on the matches metric for Italian in the following table.

Model Name	Matches
paraphrase-multilingual-mpnet-base-v2	**0.67**
distiluse-base-multilingual-cased	0.57
paper	0.62

Thus, if you use ZeroShotTM for a multilingual task, we suggest the use of *paraphrase-multilingual-mpnet-base-v2*.

CHAPTER

TEN

VISUALIZATION

We offer two main plots to show your topics. The first one is based on PyLDAvis, a great package to analyze how your topic model is behaving. You can take a look to the PyLDAvis Github Page to get a better idea on how to interpret the different components.

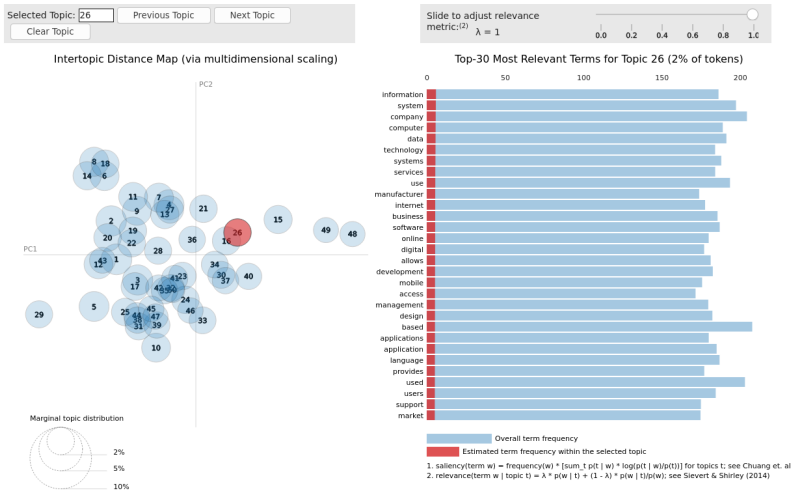
The second visualization is a very simple topic visualization with a wordcloud.

10.1 PyLdaVis Visualization

If you already know how to use that, you'll see that using PyLDAvis is very simple. First of all you need to install pyldavis.

We support pyLDA visualizations with few lines of code!

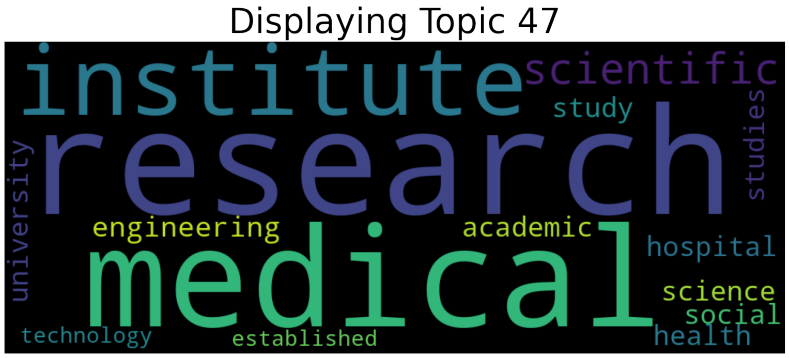
```
import pyLDAvis as vis

lda_vis_data = ctm.get_ldavis_data_format(tp.vocab, training_dataset, n_samples=10)

ctm_pd = vis.prepare(**lda_vis_data)
vis.display(ctm_pd)
```



10.2 Showing The Topic Word Cloud

You can also create a word cloud of the topic!

```
ctm.get_wordcloud(topic_id=47, n_words=15)
```



CHAPTER

ELEVEN

FREQUENTLY ASKED QUESTIONS

11.1 I am getting very poor results. What can I do?

There are many elements that can influence the final results in a topic model. A good preprocessing is fundamental for obtaining meaningful topics. On this link you can find some suggestions on how to preprocess your data, but be careful because each dataset may have their pecularities. If you still get poor results, don't hesitate to contact us! We would be happy to help you :)

11.2 Am I forced to use the SBERT Embeddings?

Not at all! you can actually the embedding you like most. Keep in mind to check that the matrix you create as the same number of documents (first dimension) as the Bag of Word representation and you are good to go! Check the code here to see how we create the representations.

11.3 Is the BoW needed even for the ZeroShotTM?

Yes, it is. The BoW is necessary in the reconstruction phase, without that we would lose the symbolic information that allows us to get the topics.

11.4 ZeroShotTM or CombinedTM? Which one should I use?

ZeroShotTm and CombinedTM can be basically used for the same tasks. ZeroShotTM has two main pros:

1) it can handle unseen words in the test phase. This makes it very useful to be used in our Kitty module, for example.

2) If your objective is to do cross-lingual topic modeling (i.e. train a topic model on a dataset in one language and predict the topics for data in other languages), then ZeroShotTM is the model for you.

If you just aim at extracting topics from a corpus, you can use either the CombinedTM or the ZeroShotTM. We have designed the CombinedTM for the purpose of obtaining more coherent topics, so we suggest you use this for more general topic extraction. Yet, as you can read in this paper, the ZeroShotTM model still gets results that are very similar to the ones of the CombinedTM.

11.5 Can I load my own embeddings?

Sure, here is a snippet that can help you. You need to create the embeddings (for bow and contextualized) and you also need to have the vocab and an id2token dictionary (maps integers ids to words).

```
qt = TopicModelDataPreparation()

training_dataset = qt.load(contextualized_embeddings, bow_embeddings, id2token)
ctm = CombinedTM(bow_size=len(vocab), contextual_size=768, n_components=50)
ctm.fit(training_dataset) # run the model
ctm.get_topics()
```

You can give a look at the code we use in the TopicModelDataPreparation object to get an idea on how to create everything from scratch. For example:

```
vectorizer = CountVectorizer() #from sklearn

train_bow_embeddings = vectorizer.fit_transform(text_for_bow)
train_contextualized_embeddings = bert_embeddings_from_list(text_for_contextual, "chosen_
↪contextualized_model")
vocab = vectorizer.get_feature_names_out()
id2token = {k: v for k, v in zip(range(0, len(vocab)), vocab)}
```

11.6 How do I choose the correct number of topics?

The "n_component" parameter represents the number of topics for CTM. There is not a "right" answer about the choice of the number of topics. Usually, researchers try a different number of topics (10, 30, 50, etc, depending on the prior knowledge on the dataset) and select the number of topics that guarantees the highest average topic coherence. We also suggest you take into consideration the topic diversity.

CHAPTER

TWELVE

CREDITS

12.1 Development Lead

- Federico Bianchi <f.bianchi@unibocconi.it>
- Silvia Terragni <s.terragni4@campus.unimib.it>
- Dirk Hovy <dirk.hovy@unibocconi.it>

12.2 Contributors

None yet. Why not be the first?

CHAPTER

THIRTEEN

HISTORY

13.1 2.2.2 (2021-11-09)

- kitty now takes in input a stopword list instead of a language (from which it gathered the stopwords)
- solving a bug in the whitespace preprocessing function
- adding a new preprocessing function that supports passing the stopwords as a list
- deprecating whitespace preprocessing
- minor fixes to kitty API
- breaking change to kitty API, now uses WhiteSpacePreprocessingStopwords.

13.2 2.2.0 (2021-09-20)

- introducing kitty
- improving the documentation a lot

13.3 2.1.2 (2021-09-03)

- patching Issue 38
- improvements PR 80

13.4 2.1.0 (2021-07-16)

- new model introduced SuperCTM
- new model introduced -CTM

13.5 2.0.0 (2021-xx-xx)

- **warning, breaking changes were introduced:**
  - the order of the parameters in CTMDataset was changed (now first is contextual embeddings)
  - CTM takes in input bow_size, contextual_size instead of input_size and bert_size
  - changed the name of the parameters in the dataset
- introduced early stopping
- introduced visualization with pyldavis

13.6 1.8.2 (2021-02-08)

- removed constraint over pytorch version. This should solve problems for Windows users

13.7 1.8.0 (2021-01-11)

- novel way to handle text, we now allow for an easy usage of training and testing data
- better visualization of the training progress and of the sampling process
- removed old stuff from the documentation

13.8 1.7.1 (2020-12-17)

- some minor updates to the documentation
- adding a new method to visualize the topic using a wordcloud
- save and load will now generate a warning since the feature has not been tested

13.9 1.7.0 (2020-12-10)

- adding a new and much simpler way to handle text for topic modeling

13.10 1.6.0 (2020-11-03)

- introducing the two different classes for ZeroShotTM and CombinedTM
- depracating CTM class in favor of ZeroShotTM and CombinedTM

13.11 1.5.3 (2020-11-03)

- adding support for Windows encoding by defaulting file load to UTF-8

13.12 1.5.2 (2020-11-03)

- updated sentence-transformers version to 0.3.6
- beta support for model saving and loading
- new evaluation metrics based on coherence

13.13 1.5.0 (2020-09-14)

- Introduced a method to predict the topics for a set of documents (supports multiple sampling to reduce variation)
- Adding some features to bert embeddings creation like increased batch size and progress bar
- Supporting training directly from lists without the need to deal with files
- Adding a simple quick preprocessing pipeline

13.14 1.4.3 (2020-09-03)

- Updating sentence-transformers package to avoid errors

13.15 1.4.2 (2020-08-04)

- Changed the encoding on file load for the SBERT embedding function

13.16 1.4.1 (2020-08-04)

- Fixed bug over sparse matrices

13.17 1.4.0 (2020-08-01)

- New feature handling sparse bow for optimized processing
- New method to return topic distributions for words

13.18 1.0.0 (2020-04-05)

- Released models with the main features implemented

13.19 0.1.0 (2020-04-04)

- First release on PyPI.