README.md

embedding-as-service

One-Stop Solution to encode sentence to fixed length vectors from various embedding techniques
• Inspired from bert-as-service

stars 138 downloads 41k pypi v2.0.1 issues 11 open license MIT all contributors 9

What is it • Installation • Getting Started • Supported Embeddings • API •

What is it

Encoding/Embedding is a upstream task of encoding any inputs in the form of text, image, audio, video, transactional data to fixed length vector. Embeddings are quite popular in the field of NLP, there has been various Embeddings models being proposed in recent years by researchers, some of the famous one are bert, xlnet, word2vec etc. The goal of this repo is to build one stop solution for all embeddings techniques available, here we are starting with popular text embeddings for now and later on we aim to add as much technique for image, audio, video inputs also.

`embedding-as-service` help you to encode any given text to fixed length vector from supported embeddings and models.

💾 Installation

▲ Back to top

Here we have given the capability to use `embedding-as-service` like a module or you can run it as a server and handle queries by installing client package `embedding-as-service-client`

Using `embedding-as-service` as module

Install the embedding-as-servive via `pip`.

```
$ pip install embedding-as-service
```

Note that the code MUST be running on **Python >= 3.6**. Again module does not support Python 2!

Using `embedding-as-service` as a server

Here you also need to install a client module `embedding-as-service-client`

```
$ pip install embedding-as-service # server
$ pip install embedding-as-service-client # client
```

Client module need not to be on Python 3.6, it supports both Python2 and Python3

⚡ Getting Started

▲ Back to top

1. Intialise encoder using supported embedding and models from here

If using `embedding-as-service` **as a module**

```
>>> from embedding_as_service.text.encode import Encoder
>>> en = Encoder(embedding='bert', model='bert_base_cased', max_seq_length=256)
```

If using `embedding-as-service` **as a server**

```
# start the server by proving embedding, model, port, max_seq_length[default=25
$ embedding-as-service-start --embedding bert --model bert_base_cased --port 80
```

```
>>> from embedding_as_service_client import EmbeddingClient
>>> en = EmbeddingClient(host=<host_server_ip>, port=<host_port>)
```

2. Get sentences tokens embedding

```
>>> vecs = en.encode(texts=['hello aman', 'how are you?'])
>>> vecs
array([[[ 1.7049843 ,  0.         ,  1.3486509 , ..., -1.3647075 ,
 0.6958289 ,  1.8013777 ], ... [ 0.4913215 ,  0.60877025,  0.73050433, ..., -0.
>>> vecs.shape
(2, 128, 768) # batch x max_sequence_length x embedding_size
```

3. Using pooling strategy, click here for more.

▶ *Supported Pooling Methods*

```
>>> vecs = en.encode(texts=['hello aman', 'how are you?'], pooling='reduce_mean
>>> vecs
array([[-0.33547154,  0.34566957,  1.1954105 , ...,  0.33702594,
 1.0317835 , -0.785943  ], [-0.3439088 ,  0.36881036,  1.0612687 , ...,  0.2885

>>> vecs.shape
(2, 768) # batch x embedding_size
```

4. Show embedding Tokens

```
>>> en.tokenize(texts=['hello aman', 'how are you?'])
[['_hello', '_aman'], ['_how', '_are', '_you', '?']]
```

5. Using your own tokenizer

```
>>> texts = ['hello aman!', 'how are you']

# a naive whitespace tokenizer
>>> tokens = [s.split() for s in texts]
>>> vecs = en.encode(tokens, is_tokenized=True)
```

📋 API

▴ Back to top

1. **class** `embedding_as_service.text.encoder.Encoder`

Argument	Type	Default	Description
`embedding`	str	*Required*	embedding method to be used, check `Embedding` column here
`model`	str	*Required*	Model to be used for mentioned embedding, check `Model` column here
`max_seq_length`	int	128	Maximum Sequence Length, default is 128

2. **def** `embedding_as_service.text.encoder.Encoder.encode`

Argument	Type	Default	Description
`Texts`	List[str] or List[List[str]]	*Required*	List of sentences or list of list of sentence tokens in case of `is_tokenized=True`
`pooling`	str	(Optional)	Pooling methods to apply, here is available methods
`is_tokenized`	bool	`False`	set as True in case of tokens are passed for encoding
`batch_size`	int	`128`	maximum number of sequences handled by encoder, larger batch will be partitioned into small batches.

3. **def** `embedding_as_service.text.encoder.Encoder.tokenize`

Argument	Type	Default	Description
`Texts`	List[str]	*Required*	List of sentences

✅ Supported Embeddings and Models

▲ Back to top

Here are the list of supported embeddings and their respective models.

	Embedding	Model	Embedding dimensions	Paper
1️⃣	**albert**	`albert_base`	768	Read Paper 🔖
		`albert_large`	1024	
		`albert_xlarge`	2048	
		`albert_xxlarge`	4096	

	Embedding	Model	Embedding dimensions	Paper
2️⃣	xlnet	`xlnet_large_cased`	1024	Read Paper 📝
		`xlnet_base_cased`	768	
3️⃣	bert	`bert_base_uncased`	768	Read Paper 📝
		`bert_base_cased`	768	
		`bert_multi_cased`	768	
		`bert_large_uncased`	1024	
		`bert_large_cased`	1024	
4️⃣	elmo	`elmo_bi_lm`	512	Read Paper 📝
5️⃣	ulmfit	`ulmfit_forward`	300	Read Paper 📝
		`ulmfit_backward`	300	
6️⃣	use	`use_dan`	512	Read Paper 📝
		`use_transformer_large`	512	
		`use_transformer_lite`	512	
7️⃣	word2vec	`google_news_300`	300	Read Paper 📝
8️⃣	fasttext	`wiki_news_300`	300	Read Paper 📝
		`wiki_news_300_sub`	300	
		`common_crawl_300`	300	
		`common_crawl_300_sub`	300	
9️⃣	glove	`twitter_200`	200	Read Paper 📝
		`twitter_100`	100	
		`twitter_50`	50	
		`twitter_25`	25	
		`wiki_300`	300	
		`wiki_200`	200	
		`wiki_100`	100	

Embedding	Model	Embedding dimensions	Paper
	`wiki_50`	50	
	`crawl_42B_300`	300	
	`crawl_840B_300`	300	

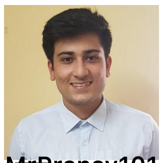
Credits

This software uses the following open source packages:

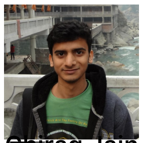
- XLnet
- tensorflow-hub

Contributors ✨

Thanks goes to these wonderful people (emoji key):


MrPranav101
💻 📖 🚇

Aman Srivastava
💻 📖 🚇


Chirag Jain
💻 📖 🚇

Ashutosh Singh
💻 📖 🚇

Dhaval Taunk
💻 📖 🚇

Alec Koumjian
🐛

Pradeesh
🐛

This project follows the all-contributors specification. Contributions of any kind welcome!

Please read the contribution guidelines first.

Citing

▲ Back to top

If you use embedding-as-service in a scientific publication, we would appreciate references to the following BibTex entry:

```
@misc{aman2019embeddingservice,
  title={embedding-as-service},
  author={Srivastava, Aman},
  howpublished={\url{https://github.com/amansrivastava17/embedding-as-service}}
  year={2019}
}
```