

# Building a Text Classification model with spaCy 3.x

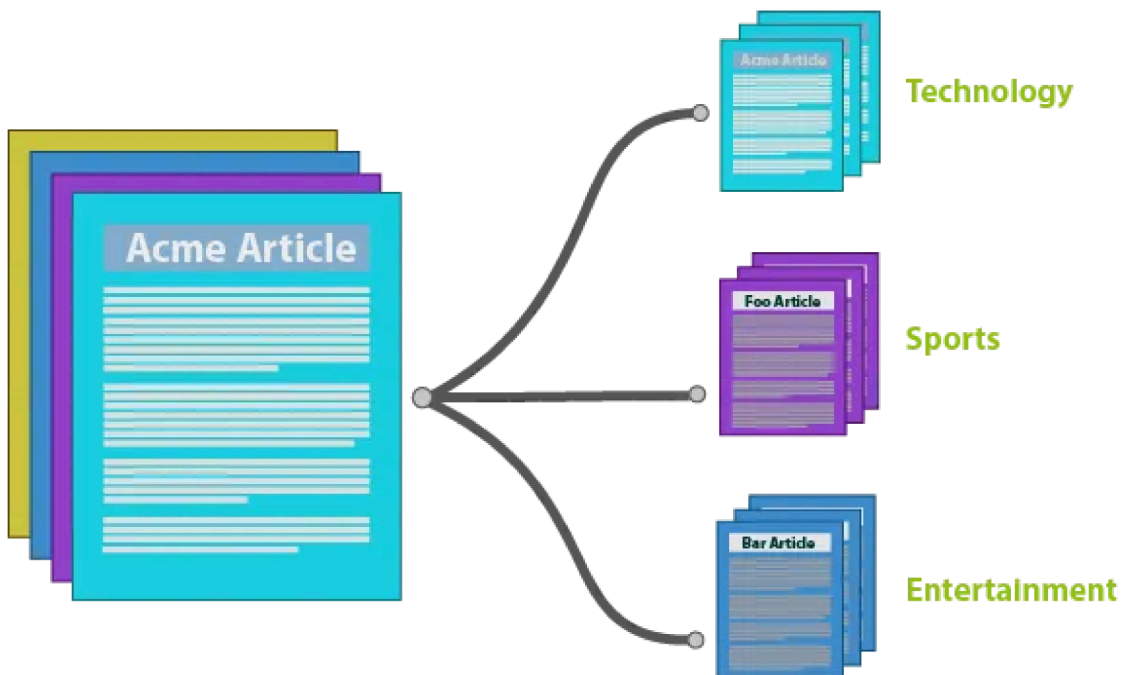


Johni Douglas Marangon · [Follow](#)

6 min read · Nov 7, 2022



90



In this post our goal is to demonstrate a modern approach to build a binary text classification in spaCy 3.x using our custom [TextCategorizer](#) component.

Text classification is the task of predicting categories of a text or documents and can be applied to a multitude of applications.

We will use a [spaCy CLI tool](#) to prototype a text classifier pipeline. You can find the complete notebook [here](#).

---

*Keep in your mind the KISS design principle. It is very important when you are building an AI solution.*

---

## How it works

TextCategorizer is a pipeline component for text classification. This is one of many other [components available in spaCy](#). The TextCategorizer predicts categories for a whole document.

There are two ways to configure this pipeline component: `textcat` and `textcat_multilabel`. When you need to predict exactly one true label per document, use the `textcat`. If you want to perform multi-label classification and predict zero, one or more true labels per document, use `textcat_multilabel`. For a binary text classification, you can use `textcat` with two labels or `textcat_multilabel` with one label. In this post we will use `textcat` with two labels.

spaCy has its own deep learning library called [thinc](#) used under the hood for different NLP models. Behind the scenes spaCy uses deep learning to train the pipelines components. All components are powered by statistical models.

The predictions are based on weight values that are calculated during training processes and are gradually changing. The weight values are

estimated based on examples the model has seen during training.

The first step in the training process is to estimate the gradient of the loss comparing given labels and predicted ones. Then, this estimation is backpropagated to calculate the gradient of the weights. These are the values that help to update the model to make better predictions. After the iteration is completed, the model is ready to be saved and used.

In short, training is an iterative process in which the model's predictions are compared against the reference annotations in order to estimate the

Open in app ↗

Sign up

Sign in



Search

Write



## Data Preparation

In this post we're going to use an already-labelled dataset available [here](#). The dataset is labeled for a tweet sentiment analysis with two categories, positive and negative in Brazilian Portuguese. Run the code below to load the dataset into a Pandas structure data.

```
import pandas as pd

df =
pd.read_csv("https://gist.githubusercontent.com/johnidm/582cfeadd2bf4
18df4539c9422f824d2/raw/twitter-sentiment-pt-BR-md-2-l.csv")
df.head()
```

Two tasks should be done in this step: apply preprocessing techniques in the text and transform the text and label to binary spaCy format using [Doc Objects](#).

```

import string
import re
import spacy
from spacy.lang.pt.stop_words import STOP_WORDS

nlp = spacy.blank("pt")

REGX_USERNAME = r"@[A-Za-z0-9$-_.&+]+"
REGX_URL = r"https?://[A-Za-z0-9./]+"

def preprocessing(text):
    text = text.lower()

    text = re.sub(REGX_USERNAME, ' ', text)
    text = re.sub(REGX_URL, ' ', text)

    emojis = {
        ':)': 'emocaopositiva',
        ':(': 'emocaonegativa'
    }

    for e in emojis:
        text = text.replace(e, emojis[e])

    tokens = [token.text for token in nlp(text)]

    tokens = [t for t in tokens if
               t not in STOP_WORDS and
               t not in string.punctuation and
               len(t) > 3]

    tokens = [t for t in tokens if not t.isdigit()]

    return " ".join(tokens)

df["tweet_text_clean"] = df["tweet_text"].apply(preprocessing)

df.head()

```

In the preprocessing step, it is common to remove empty lines, nonsense words, stop words, urls, punctuation, remove accentuation, make all text lowercase or use more advanced techniques such as stemming or lematization. See what I wrote about it in my [last publication](#).

We will shuffle the whole dataset and split into three parts:

- 75% — train set: used to train the pipeline.
- 15% — dev set: used to evaluate the pipeline during the training interactions.
- 10% — test set: used to evaluate the final model.

```
dataset = list(df[["tweet_text_clean",
"sentiment"]].sample(frac=1).itertuples(index=False, name=None))

train_data = dataset[:15000]
dev_data = dataset[15000:18000]
test_data = dataset[18000:]

print(f"Total: {len(dataset)} - Train: {len(train_data)} - Dev:
{len(dev_data)} - Test: {len(test_data)}")
```

Now we need to transform the data and store it as a binary file on disk. We are performing the data using the DocBin structure, which makes data manipulations in spaCy more efficient.

```
def convert(data, outfile):
    db = spacy.tokens.DocBin()
    docs = []
    for doc, label in nlp.pipe(data, as_tuples=True):
        doc.cats["POS"] = label == 1
        doc.cats["NEG"] = label == 0
        db.add(doc)

    db.to_disk(outfile)

convert(train_data, "./train.spacy")
convert(dev_data, "./dev.spacy")
convert(test_data, "./test.spacy")
```

As you can see the label was saved in the `doc.cats` property with bool value. When you are predicting the text categories you can access the result with

score in this property.

Finally we are able to start to train the pipeline.

## Training a Pipeline

### Create a configuration file

Training config files include all settings and hyperparameters for training your pipeline instead of providing lots of arguments on the command line or in a source code.

Now, we need to create a configuration file that tells spaCy what settings will be used to learn from our data.

```
!python -m spacy init config --lang pt --pipeline textcat --optimize  
efficiency --force config.cfg
```

The above command creates a configuration file with a `textcat` pipeline component, Portuguese language and optimized efficiency. The other parameters will be filled automatically by spaCy with the default values. For now, we will just stick with default, this will work fine for most cases.

Open the `config.cfg` file and find the `components.textcat.model` section to see the key `@architectures` has `spacy.TextCatBOW.v2` value. This configuration indicates that `textcat` component will use an n-gram bag-of-words model to predict the score, this are the default settings.

Model Architectures is an important concept of spaCy world. The Text classification architectures section describe the options of model

architecture we can use when working with `textcat` component.

If you want to understand more about how you can config other architectures in the `textcat` component read more about it at [config and implementation](#).

For a binary text classification problem, I created the configuration file for each architecture available in spaCy.

- [Ensemble](#)
- [CNN](#)
- [BoW](#)

Choose the architecture, download the configuration file and use it in a command line training. You can test with different architectures when you finish reading this article. You can find how to use it [in this notebook](#) in the **Evaluate Architectures** section.

For now, I recommend that you use the file generated via the command line from the previous step.

## **Train command**

Finally, we can fire up the training command with the parameters created in the prior step.

```
!python -m spacy train config.cfg --paths.train ./train.spacy --  
paths.dev ./dev.spacy --output model --verbose
```

For each training step, the output tell us loss and accuracy. Accuracy is how often the classification is correct and the loss is how big are the mistakes. See a piece of spaCy model training output.

E	#	LOSS	TEXTCAT	CATS_SCORE	SCORE
---	-----	-----	-----	-----	-----
0	0	0.25	82.68	0.83	
0	200	33.87	97.77	0.98	
0	400	14.45	98.50	0.98	
0	600	7.21	98.83	0.99	
1	800	4.10	98.90	0.99	
1	1000	2.94	98.80	0.99	
2	1200	2.02	98.87	0.99	
3	1400	1.70	98.87	0.99	
4	1600	1.37	98.87	0.99	
5	1800	1.19	98.83	0.99	

Once training is done two models are created in the `model` folder:

- `model-best`: model that got the highest score during the training iteration.
- `model-last`: model trained in the last training iteration.

The most common is to choose the `model-best`.

## Evaluation the Pipeline

spaCy has a build in `evaluate` command to evaluate the model. Let's take a look at how our model actually performs.

```
!python -m spacy evaluate ./model/model-best/ ./test.spacy
```



The command above calculates the performance of the model for each individual category in terms of AOC (area under the ROC curve), precision (  $P$  ), recall (  $R$  ) and F1 score (  $F$  ).

## Test the Pipeline

The trained models are saved into output folder `model` . You can load the model and check the predictions for new inputs. This way you can use it in your client application.

```
texts = [":)", "Estou muito triste hoje", "Estou muito feliz hoje"]
nlp = spacy.load("./model/model-best")
for text in texts:
    doc = nlp(preprocessing(text))
    print(doc.cats, "-", text)
```

The dataset was trained with Brazilian Portuguese texts, **feel free to use other texts any other language to train your model** and enter new texts to predict the labels. The steps you learned in this post should be used on another dataset with little or no change.

## Conclusion

In this article, we explore a powerful approach to creating a binary text classification model with spaCy 3.x to classify tweets as positive or negative sentiments. I recommend that you use another dataset to predict other labels using this training process.

As future improvement you can use the [Transformer](#). It is a newer architecture that include the context of a word into its embedding, [spaCy 3.x](#)

support transformers like BERT, we can explore more to improve the performance. Soon as possible I will write about that.

Happy coddling!

Spacy

NLP

Text Classification

Python

Data Science



**Written by Johni Douglas Marangon**

Follow


167 Followers

Connect with me on <https://www.linkedin.com/in/johnidouglas/> and see my technical skills in <https://github.com/johnidm>

---

**More from Johni Douglas Marangon**

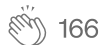


 Johni Douglas Marangon

## Using migrations in Python — SQLAlchemy with Alembic +...

Hello everyone, in this article, we are going to learn database migrations in Python.

8 min read · Apr 17, 2023



166



3



 Johni Douglas Marangon

## How to summarize text with OpenAI and LangChain

This is the first of three posts about my recent study of summarization using Large...

7 min read · Aug 17, 2023




87



3



 Johni Douglas Marangon

## Using `__call__` method to invoke class instance as a function

Introduction

3 min read · Aug 23, 2022



152



 Johni Douglas Marangon

## How to calculate the Word Error Rate in Python

In this article, we will look at the common metric to evaluate the performance of...

6 min read · Nov 16, 2023



26




2



See all from Johni Douglas Marangon

## Recommended from Medium



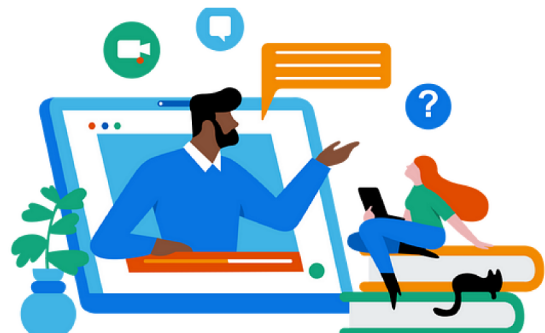
 Reyansh Bahl

### Token Classification in Python with HuggingFace

In this article, we will learn about token classification, its applications, and how it can...

9 min read · Nov 5, 2023

 4 



 Johni Douglas Marangon

### Building a custom Named Entity Recognition model using spaCy ...

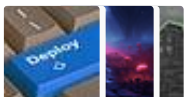
In today's post, we will learn how to train a NER. In the previous post we saw the...

4 min read · Nov 21, 2023

 60  1



## Lists



### Predictive Modeling w/ Python

20 stories · 1016 saves



### Coding & Development

11 stories · 518 saves



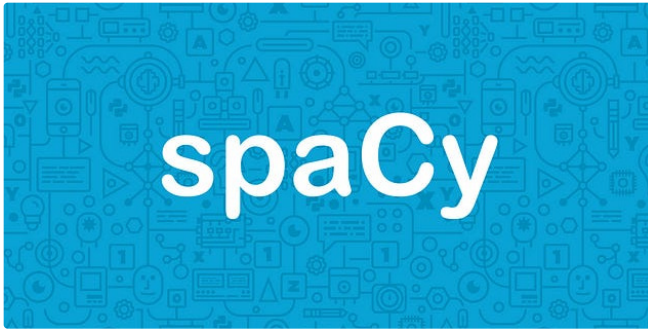
## Practical Guides to Machine Learning

10 stories · 1222 saves



## Natural Language Processing

1312 stories · 799 saves



Wiem Souai in UBI AI NLP

### Fine-Tuning SpaCy Models: Customizing Named Entity...

Exploring Named Entity Recognition (NER) with spaCy: A Guide to Fine-Tuning

7 min read · Feb 6, 2024



B-PER	O	O	B-LOC	I-LOC	O
John	lives	in	New	York	.



Ahmet Münir Kocaman

### Mastering Named Entity Recognition with BERT: A...

Introduction

11 min read · Oct 6, 2023

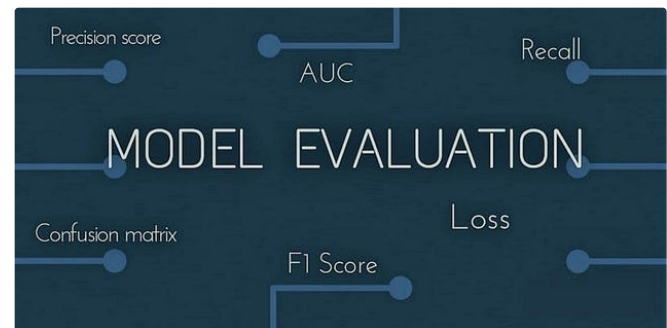


Krikor Postalian-Yrausquin in Towards AI

### NLP (doc2vec from scratch) & Clustering: Classification of news...

Using NLP (doc2vec), with deep and customized text cleaning, and then clusterin...

9 min read · Nov 14, 2023



Mohammed Farmaan in featurepreneur

### Understanding Named Entity Recognition Evaluation Metrics...

Introduction:

2 min read · Jan 3, 2024



18



---

See more recommendations