**Building a Better Profanity Detection Library with scikit-learn**

Why existing libraries are uninspiring and how I built a better one.

[Victor Zhou](https://medium.com/@victorczhou?source=post_page-----3638b2f2c4c2--------------------------------)

[Victor Zhou](https://medium.com/@victorczhou?source=post_page-----3638b2f2c4c2--------------------------------)

[Feb 4, 2019·6 min read](https://towardsdatascience.com/building-a-better-profanity-detection-library-with-scikit-learn-3638b2f2c4c2?source=post_page-----3638b2f2c4c2--------------------------------)

A few months ago, I needed a way to detect profanity in user-submitted text strings:

Image for post



This shouldn’t be that hard, right?

I ended up building and releasing my own library for this purpose called [profanity-check](https://github.com/vzhou842/profanity-check):

**[vzhou842/profanity-check](https://github.com/vzhou842/profanity-check" \t "_blank)**

[A fast, robust Python library to check for offensive language in strings. - vzhou842/profanity-check](https://github.com/vzhou842/profanity-check" \t "_blank)

[github.com](https://github.com/vzhou842/profanity-check" \t "_blank)

Of course, before I did that, I looked in the [Python Package Index](https://pypi.org/) (PyPI) for any existing libraries that could do this for me. The only half decent results for the search query “profanity” were:

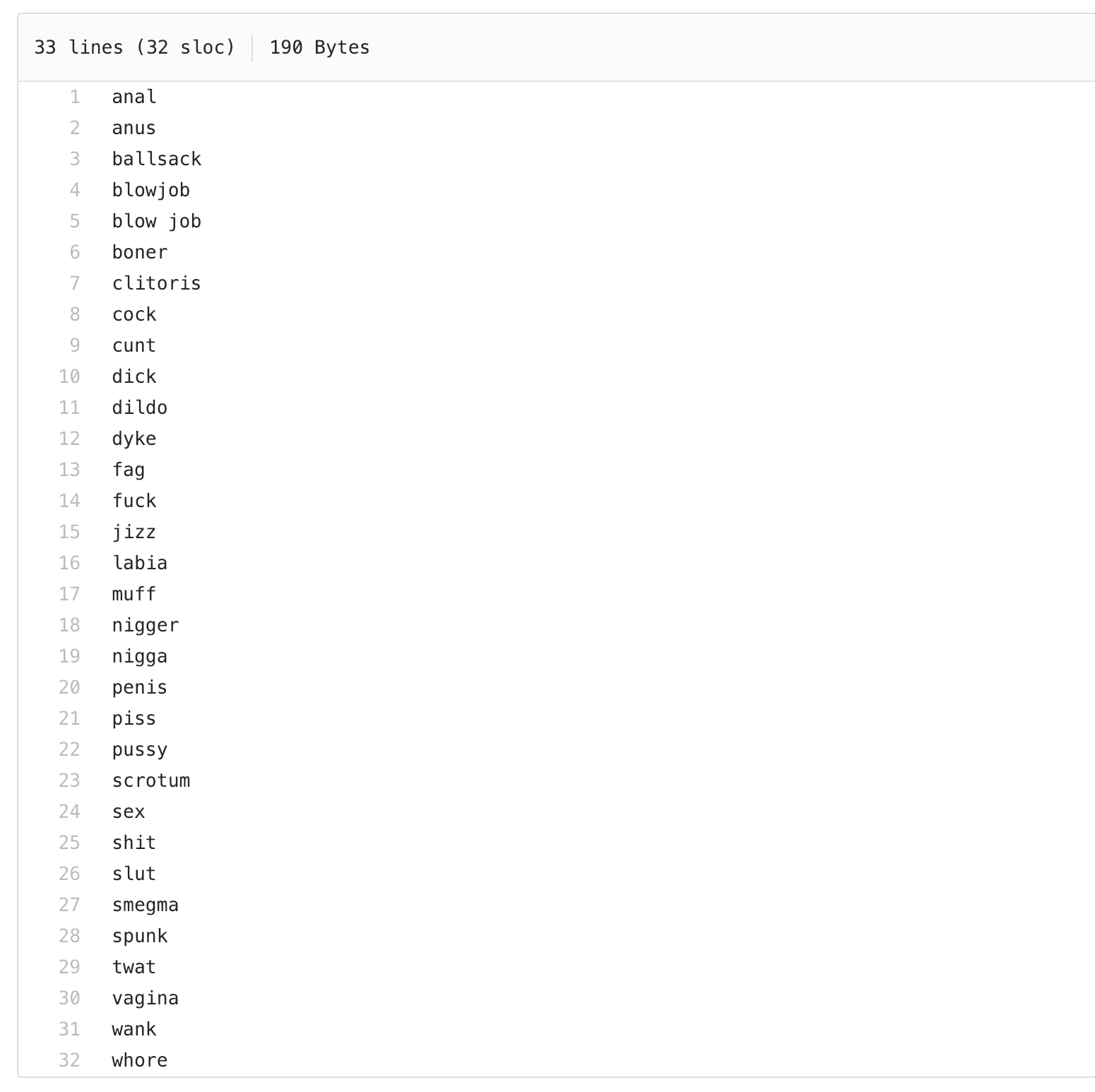
* [profanity](https://pypi.org/project/profanity/) (the ideal package name)
* [better-profanity](https://pypi.org/project/better-profanity/): *“Inspired from package*[*profanity*](https://github.com/ben174/profanity)*of*[*Ben Friedland*](https://github.com/ben174)*, this library is much faster than the original one.”*
* [profanityfilter](https://pypi.org/project/profanityfilter/) (has 31 Github stars, which is 30 more than most of the other results have)
* [profanity-filter](https://pypi.org/project/profanity-filter/) (uses Machine Learning, enough said?!)

Third-party libraries can sometimes be sketchy, though, so I did my due diligence on these 4 results.

profanity, better-profanity, and profanityfilter

After a quick dig through the profanity repository, I found a file named [wordlist.txt](https://github.com/ben174/profanity/blob/master/profanity/data/wordlist.txt):

Image for post



Sorry this image of profanities is so big…

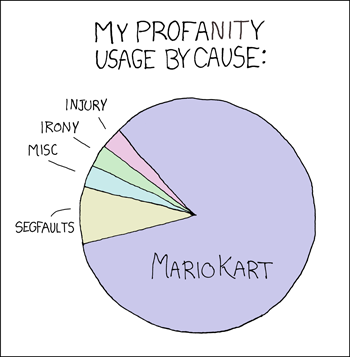
The entire profanity library is just a wrapper over this list of 32 words! profanity detects profanity simply by looking for one of these words.

To my dismay, better-profanity and profanityfilter both took the same approach:

* better-profanity uses [a 140-word wordlist](https://github.com/snguyenthanh/better_profanity/blob/master/better_profanity/profanity_wordlist.txt)
* profanityfilter uses [a 418-word wordlist](https://github.com/areebbeigh/profanityfilter/blob/master/profanityfilter/data/badwords.txt)

This is bad because **profanity detection libraries based on wordlists are extremely subjective.**For example, better-profanity's wordlist includes the word “suck.” Are you willing to say that any sentence containing the word “suck” is profane? Furthermore, any hard-coded list of bad words will inevitably be incomplete — do you think profanity's 32 bad words are the only ones out there?

Image for post



Fucking Blue Shells. source: [xkcd](https://xkcd.com/290/)

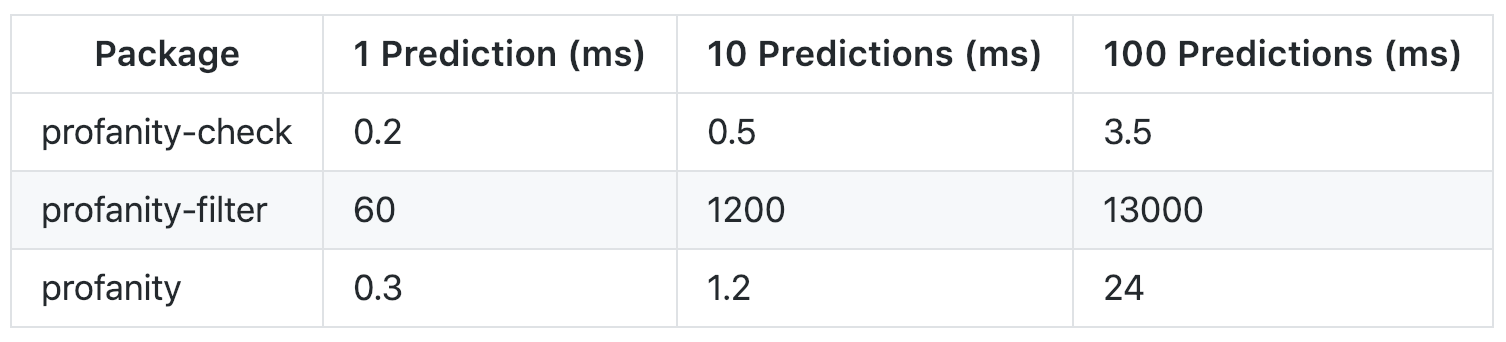
Having already ruled out 3 libraries, I put my hopes on the 4th and final one: profanity-filter.

profanity-filter

profanity-filter uses Machine Learning! Sweet!

Turns out, it’s ***really***slow. Here’s a benchmark I ran in December 2018 comparing (1) profanity-filter, (2) my library profanity-check, and (3) profanity (the one with the list of 32 words):

Image for post

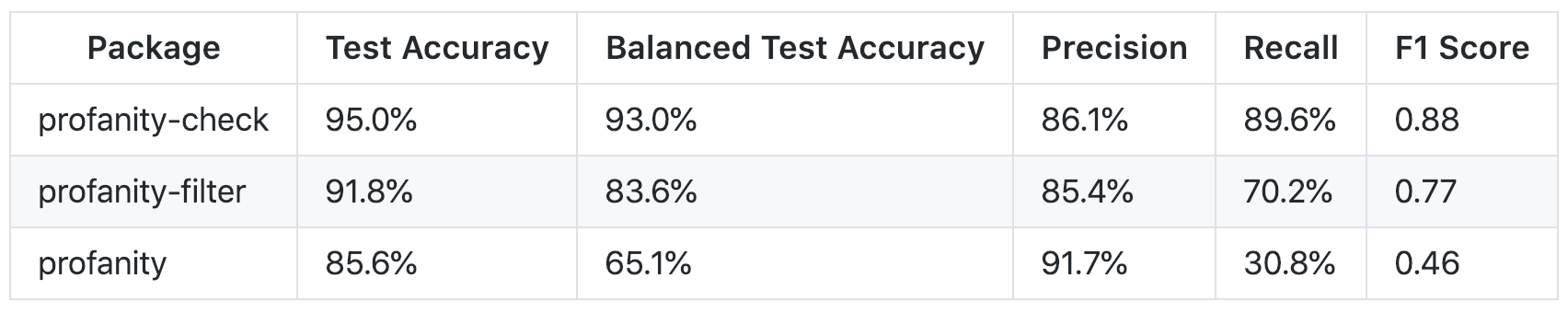


A human could probably do this faster than profanity-filter can

I needed to be able to perform many predictions in real time, and profanity-filter was not even close to being fast enough. But hey, maybe this is a classic tradeoff of accuracy for speed, right?

Nope.

Image for post



At least profanity-filter is not dead last this time

None of the libraries I’d found on PyPI met my needs, so I built my own.

Building profanity-check, Part 1: Data

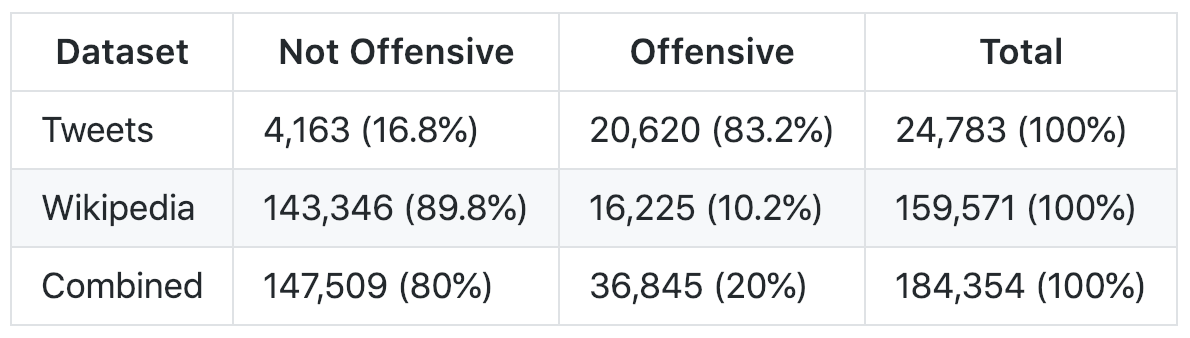
I knew that I wanted profanity-check to base its classifications on data to avoid being subjective *(read: to be able to say I used Machine Learning)*. I put together a combined dataset from two publicly-available sources:

* the “Twitter” dataset from [t-davidson/hate-speech-and-offensive-language](https://github.com/t-davidson/hate-speech-and-offensive-language/tree/master/data), which contains tweets scraped from Twitter.
* the “Wikipedia” dataset from [this Kaggle competition](https://www.kaggle.com/c/jigsaw-toxic-comment-classification-challenge) published by Alphabet’s [Conversation AI](https://conversationai.github.io/) team, which contains comments from Wikipedia’s talk page edits.

Each of these datasets contains text samples hand-labeled by humans through crowdsourcing sites like [Figure Eight](https://www.figure-eight.com/).

Here’s what my dataset ended up looking like:

Image for post



Combined = Tweets + Wikipedia

*The Twitter dataset has a column named class that’s 0 if the tweet contains hate speech, 1 if it contains offensive language, and 2 if it contains neither. I classified any tweet with a class of 2 as “Not Offensive” and all other tweets as “Offensive.”*

*The Wikipedia dataset has several binary columns (e.g. toxic or threat) that represent whether or not that text contains that type of toxicity. I classified any text that contained*any *of the types of toxicity as “Offensive” and all other texts as “Not Offensive.”*

Building profanity-check, Part 2: Training

Now armed with a cleaned, combined dataset (which you can [download here](https://github.com/vzhou842/profanity-check/blob/master/profanity_check/data/clean_data.csv)), I was ready to train the model!

*I’m skipping over how I cleaned the dataset because, honestly, it’s pretty boring— if you’re interested in learning more about preprocessing text datasets check out*[*this*](https://machinelearningmastery.com/clean-text-machine-learning-python/)*or*[*this*](https://medium.com/@datamonsters/text-preprocessing-in-python-steps-tools-and-examples-bf025f872908)*.*

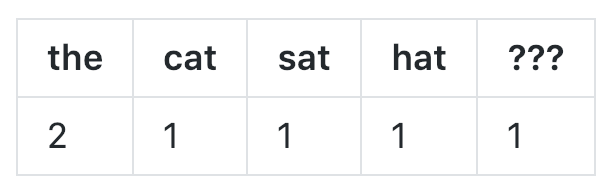
Are you also surprised the code is so short? Apparently [scikit-learn](https://scikit-learn.org/) does everything.

Two major steps are happening here: (1) vectorization and (2) training.

Vectorization: Bag of Words

I used scikit-learn's [CountVectorizer](https://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.CountVectorizer.html) class, which basically turns any text string into a vector by counting how many times each given word appears. This is known as a [Bag of Words](https://victorzhou.com/blog/bag-of-words/) (BOW) representation. For example, if the only words in the English language were the, cat, sat, and hat, a possible vectorization of the sentence the cat sat in the hat might be:

Image for post



“the cat sat in the hat” -> [2, 1, 1, 1, 1]

The ??? represents any unknown word, which for this sentence is in. Any sentence can be represented in this way as counts of the, cat, sat, hat, and ???!

Image for post



A handy reference table for the next time you need to vectorize “cat cat cat cat cat”

Of course, there are far more words in the English language, so in the code above I use the fit\_transform() method, which does 2 things:

* **Fit:**learns a vocabulary by looking at all words that appear in the dataset.
* **Transform**: turns each text string in the dataset into its vector form.

Training: Linear SVM

The model I decided to use was a Linear Support Vector Machine (SVM), which is implemented by scikit-learn's [LinearSVC](https://scikit-learn.org/stable/modules/generated/sklearn.svm.LinearSVC.html) class. [This](https://medium.com/machine-learning-101/chapter-2-svm-support-vector-machine-theory-f0812effc72) and [this](https://www.svm-tutorial.com/2014/11/svm-understanding-math-part-1/) are good introductions if you don’t know what SVMs are.

*The [CalibratedClassifierCV](https://scikit-learn.org/stable/modules/generated/sklearn.calibration.CalibratedClassifierCV.html) in the code above exists as a wrapper to give me the predict\_proba() method, which returns a probability for each class instead of just a classification. You can pretty much just ignore it if that last sentence made no sense to you, though.*

Here’s one (simplified) way you could think about why the Linear SVM works: during the training process, the model learns which words are “bad” and how “bad” they are because those words appear more often in offensive texts. **It’s as if the training process is picking out the “bad” words for me**, which is much better than using a wordlist I write myself!

A Linear SVM combines the best aspects of the other profanity detection libraries I found: it’s fast enough to run in real-time yet robust enough to handle many different kinds of profanity.

Caveats

That being said, profanity-check is far from perfect. Let me be clear: take predictions from profanity-check with a grain of salt because**it makes mistakes.**For example, its not good at picking up less common variants of profanities like “f4ck you” or “you b1tch” because they don’t appear often enough in the training data. You’ll never be able to detect *all*profanity (people will come up with new ways to evade filters), but profanity-checkdoes a good job at finding most.

profanity-check

profanity-check is open source and available on PyPI! To use it, simply

$ pip install profanity-check

How could profanity-check be even better? Feel free to reach out or comment with any thoughts or suggestions!

**[vzhou842/profanity-check](https://github.com/vzhou842/profanity-check" \t "_blank)**

[A fast, robust Python library to check for offensive language in strings. - vzhou842/profanity-check](https://github.com/vzhou842/profanity-check" \t "_blank)

[github.com](https://github.com/vzhou842/profanity-check" \t "_blank)

*Also posted on*[*victorzhou.com*](https://victorzhou.com/blog/better-profanity-detection-with-scikit-learn/)*.*

[Victor Zhou](https://medium.com/@victorczhou?source=post_sidebar--------------------------post_sidebar-----------)

CS @ Princeton University. I write about web development, machine learning, and more at [https://victorzhou.com](https://victorzhou.com/).

[Follow](https://medium.com/m/signin?actionUrl=%2F_%2Fsubscribe%2Fuser%2Fdd190d205cab%2F3638b2f2c4c2&operation=register&redirect=https%3A%2F%2Ftowardsdatascience.com%2Fbuilding-a-better-profanity-detection-library-with-scikit-learn-3638b2f2c4c2&source=post_sidebar-dd190d205cab-------------------------follow_card-----------)

304

8

Sign up for The Variable

By Towards Data Science

Every Thursday, the Variable delivers the very best of Towards Data Science: from hands-on tutorials and cutting-edge research to original features you don't want to miss. [Take a look.](https://medium.com/towards-data-science/newsletters/the-variable?source=newsletter_v3_promo--------------------------newsletter_v3_promo-----------)



Get this newsletter

By signing up, you will create a Medium account if you don’t already have one. Review our [Privacy Policy](https://policy.medium.com/medium-privacy-policy-f03bf92035c9?source=newsletter_v3_promo--------------------------newsletter_v3_promo-----------) for more information about our privacy practices.

304

8

* [Machine Learning](https://towardsdatascience.com/tagged/machine-learning)
* [Python](https://towardsdatascience.com/tagged/python)
* [Scikit Learn](https://towardsdatascience.com/tagged/scikit-learn)
* [Support Vector Machine](https://towardsdatascience.com/tagged/support-vector-machine)
* [NLP](https://towardsdatascience.com/tagged/nlp)

[More from Towards Data Science](https://towardsdatascience.com/?source=follow_footer-------------------------------------)

[Follow](https://medium.com/m/signin?actionUrl=%2F_%2Fsubscribe%2Fcollection%2Ftowards-data-science%2F3638b2f2c4c2&operation=register&redirect=https%3A%2F%2Ftowardsdatascience.com%2Fbuilding-a-better-profanity-detection-library-with-scikit-learn-3638b2f2c4c2&source=follow_footer--------------------------follow_footer-----------)

Your home for data science. A Medium publication sharing concepts, ideas and codes.

[Paras Chopra](https://medium.com/@paraschopra?source=follow_footer---------0----------------------------)

[·Feb 4, 2019](https://towardsdatascience.com/making-deep-neural-networks-paint-to-understand-how-they-work-4be0901582ee?source=follow_footer---------0----------------------------)

[**Making deep neural networks paint to understand how they work**](https://towardsdatascience.com/making-deep-neural-networks-paint-to-understand-how-they-work-4be0901582ee?source=follow_footer---------0----------------------------)

It’s a mystery that deep learning works so well. Even though there are [several hints](https://www.youtube.com/watch?v=Y-WgVcWQYs4) about [why deep neural networks are so effective](https://www.reddit.com/r/MachineLearning/comments/abj1mc/d_notes_on_why_deep_neural_networks_are_able_to/), the truth is that nobody is entirely sure and theoretical understanding of deep learning is very much an active area of research.

In this tutorial, we’ll scratch a tiny aspect of the problem in an unusual manner. We will make neural networks paint abstract images for us, and then we will interpret those images to develop a better intuition on what might be happening under the hood. Also, as a bonus, by the end of the tutorial…

[Read more · 10 min read](https://towardsdatascience.com/making-deep-neural-networks-paint-to-understand-how-they-work-4be0901582ee?readmore=1&source=follow_footer---------0----------------------------)

1K

[8](https://towardsdatascience.com/making-deep-neural-networks-paint-to-understand-how-they-work-4be0901582ee?responsesOpen=true&source=follow_footer---------0----------------------------)

[Jeff Hale](https://jeffhale.medium.com/?source=follow_footer---------1----------------------------)

[·Feb 4, 2019](https://towardsdatascience.com/learn-enough-python-to-be-useful-argparse-e482e1764e05?source=follow_footer---------1----------------------------)

[**Learn Enough Python to be Useful: argparse**](https://towardsdatascience.com/learn-enough-python-to-be-useful-argparse-e482e1764e05?source=follow_footer---------1----------------------------)

How to Get Command Line Arguments Into Your Scripts

If you plan to be a software developer with Python, you’ll want to be able to use argparse for your scripting needs. If you’re a data scientist, you’ll likely find yourself needing to port your code from a Jupyter Notebook to a reproducible script. For many newer data scientists this can be a step from a comfortable, happy place into scary land. This guide is designed to make the leap less scary.

[[Image for post](https://towardsdatascience.com/learn-enough-python-to-be-useful-argparse-e482e1764e05?source=follow_footer---------1----------------------------)](https://towardsdatascience.com/learn-enough-python-to-be-useful-argparse-e482e1764e05?source=follow_footer---------1----------------------------)

[[](https://towardsdatascience.com/learn-enough-python-to-be-useful-argparse-e482e1764e05?source=follow_footer---------1----------------------------)](https://towardsdatascience.com/learn-enough-python-to-be-useful-argparse-e482e1764e05?source=follow_footer---------1----------------------------)

Like a mountain through the clouds, argparse brings arguments from the command line. 😏

[argparse](https://docs.python.org/3/howto/argparse.html#id1) is the “recommended command-line parsing module in the Python standard library.” It’s what you use to get command line arguments into your program.

I couldn’t find a…

[Read more · 6 min read](https://towardsdatascience.com/learn-enough-python-to-be-useful-argparse-e482e1764e05?readmore=1&source=follow_footer---------1----------------------------)

3.1K

[22](https://towardsdatascience.com/learn-enough-python-to-be-useful-argparse-e482e1764e05?responsesOpen=true&source=follow_footer---------1----------------------------)

[Rodrigo Nader](https://medium.com/@rodrigonader?source=follow_footer---------2----------------------------)

[·Feb 4, 2019](https://towardsdatascience.com/creating-a-custom-classifier-for-text-cleaning-a2a1fc818935?source=follow_footer---------2----------------------------)

[**Creating a Custom Classifier for Text Cleaning**](https://towardsdatascience.com/creating-a-custom-classifier-for-text-cleaning-a2a1fc818935?source=follow_footer---------2----------------------------)

Machine Learning for Sentence Classification

[[Image for post](https://towardsdatascience.com/creating-a-custom-classifier-for-text-cleaning-a2a1fc818935?source=follow_footer---------2----------------------------)](https://towardsdatascience.com/creating-a-custom-classifier-for-text-cleaning-a2a1fc818935?source=follow_footer---------2----------------------------)

[[](https://towardsdatascience.com/creating-a-custom-classifier-for-text-cleaning-a2a1fc818935?source=follow_footer---------2----------------------------)](https://towardsdatascience.com/creating-a-custom-classifier-for-text-cleaning-a2a1fc818935?source=follow_footer---------2----------------------------)

Recently I've been studying NLP more than other data science fields, and one challenge that I face more often than not is the cleaning part of the process. Building NLP models require many pre-processing steps, and if the data is not properly treated, it could result in poor models, which is necessarily what we want to avoid.

In this article, we're going to focus on PDF documents. The goal here is to open a PDF file, convert it to plain text, understand the need for data cleaning and build a machine learning model for that purpose.

In this post we…

[Read more · 8 min read](https://towardsdatascience.com/creating-a-custom-classifier-for-text-cleaning-a2a1fc818935?readmore=1&source=follow_footer---------2----------------------------)

367

[3](https://towardsdatascience.com/creating-a-custom-classifier-for-text-cleaning-a2a1fc818935?responsesOpen=true&source=follow_footer---------2----------------------------)

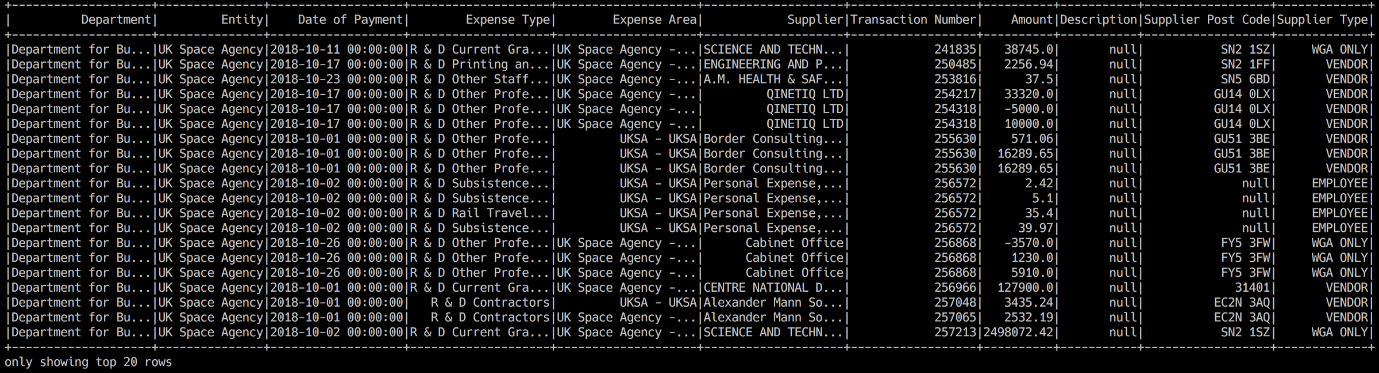
[Ashley Broadley](https://medium.com/@ls12styler?source=follow_footer---------3----------------------------)

[·Feb 4, 2019](https://towardsdatascience.com/a-journey-into-big-data-with-apache-spark-part-2-4511aa19a900?source=follow_footer---------3----------------------------)

[**A Journey Into Big Data with Apache Spark — Part 2**](https://towardsdatascience.com/a-journey-into-big-data-with-apache-spark-part-2-4511aa19a900?source=follow_footer---------3----------------------------)

An introduction to building your first Apache Spark application with Scala to read from CSV files.

[[Image for post](https://towardsdatascience.com/a-journey-into-big-data-with-apache-spark-part-2-4511aa19a900?source=follow_footer---------3----------------------------)](https://towardsdatascience.com/a-journey-into-big-data-with-apache-spark-part-2-4511aa19a900?source=follow_footer---------3----------------------------)

[[](https://towardsdatascience.com/a-journey-into-big-data-with-apache-spark-part-2-4511aa19a900?source=follow_footer---------3----------------------------)](https://towardsdatascience.com/a-journey-into-big-data-with-apache-spark-part-2-4511aa19a900?source=follow_footer---------3----------------------------)

A tabular view of a DataFrame from a CSV file in Apache Spark

Welcome back for the second part of (what I hope to be) a series of posts about getting to know [Apache Spark](https://spark.apache.org/).

In the [first episode](https://towardsdatascience.com/a-journey-into-big-data-with-apache-spark-part-1-5dfcc2bccdd2), we learned how to create and run an Apache Spark cluster using Docker. If you’ve not read that yet, you can do so [here](https://medium.com/@ls12styler/a-journey-into-big-data-with-apache-spark-part-1-5dfcc2bccdd2). I’ll be using that cluster as a cluster to run my Spark application against, so it will be quite useful for you to have it up and running.

I’ll be using Scala to build the application because I also want to get to know that. Scala isn’t something I’ve used…

[Read more · 15 min read](https://towardsdatascience.com/a-journey-into-big-data-with-apache-spark-part-2-4511aa19a900?readmore=1&source=follow_footer---------3----------------------------)

108

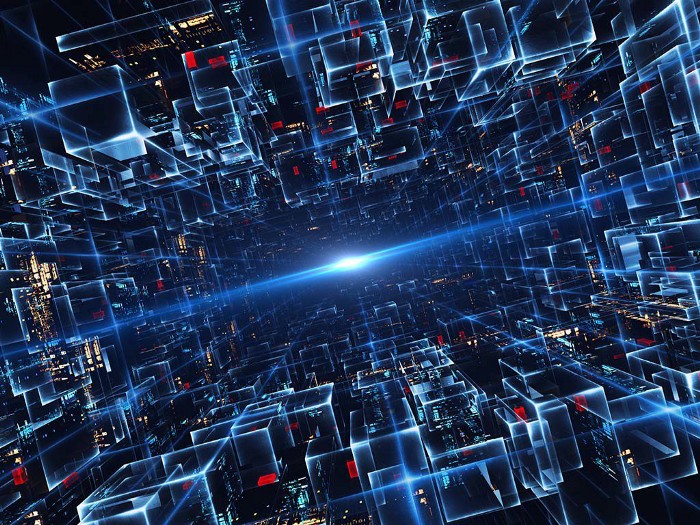
[6](https://towardsdatascience.com/a-journey-into-big-data-with-apache-spark-part-2-4511aa19a900?responsesOpen=true&source=follow_footer---------3----------------------------)

[Ted Gross](https://tedwgross.medium.com/?source=follow_footer---------4----------------------------)

[·Feb 4, 2019](https://towardsdatascience.com/the-drastic-mistake-of-using-mongoose-to-handle-your-big-data-a3c408e21a4c?source=follow_footer---------4----------------------------)

[**The Drastic Mistake Of Using Mongoose To Handle Your Big Data**](https://towardsdatascience.com/the-drastic-mistake-of-using-mongoose-to-handle-your-big-data-a3c408e21a4c?source=follow_footer---------4----------------------------)

[[Image for post](https://towardsdatascience.com/the-drastic-mistake-of-using-mongoose-to-handle-your-big-data-a3c408e21a4c?source=follow_footer---------4----------------------------)](https://towardsdatascience.com/the-drastic-mistake-of-using-mongoose-to-handle-your-big-data-a3c408e21a4c?source=follow_footer---------4----------------------------)

[[](https://towardsdatascience.com/the-drastic-mistake-of-using-mongoose-to-handle-your-big-data-a3c408e21a4c?source=follow_footer---------4----------------------------)](https://towardsdatascience.com/the-drastic-mistake-of-using-mongoose-to-handle-your-big-data-a3c408e21a4c?source=follow_footer---------4----------------------------)

Introduction

[Mongoose](https://www.npmjs.com/package/mongoose) is an incredibly popular and well-done library in the NPM universe. It is used extensively by many excellent programmers based upon its Model-Schema structure. Indeed, a cursory look in Google at many examples at creating any sort of stack with Data models that include MongoDB will show you that the authors mostly include Mongoose in their development structure. It is a respected, well-kept, and incredibly popular library. All the above is true, and the authors should be lauded for their excellent skills and in understanding the needs of the community.

The above is not a disclaimer nor a cynical…