✦ Member-only story

Custom NLP Approaches to Data Anonymization

Practical ways for de-identifying real-world private data

Omri Mendels · Follow

Published in Towards Data Science

8 min read · Jan 8, 2020

As internet services become ubiquitous, the aspiration for internet privacy continues to grow. In recent years, different laws such like GDPR which standardize the way services collect private information came into play. This brought the privacy aspects to the attention of every company and increased the investment in handling and anonymizing private data.

My job at the Commercial Software Engineering (CSE) team in Microsoft is to collaborate with Microsoft's most strategic customers. Together, we develop new workloads in the fields of AI, data at scale, IoT and more. While engaging with these customers, we've realized that the PII (Personally Identifiable Information) issue is a recurring topic and blocker for many companies who wish to extend their set of solutions, either on-premises or in the cloud.

Therefore, we've decided to create Presidio, a production ready open-source service, available for free, for anyone who wishes to address the data privacy problem.

Presidio allows any user to create standard and transparent processes for anonymizing PII entities on structured and unstructured data. To do so, it exposes a set of predefined PII recognizers (for common entities like names, credit card numbers and phone numbers), and tools for extending it with new logic for identifying more specific PII entities. In this blog post, we'll focus on how Natural Language Processing could be leveraged to identify the different types of private entities.

The PII detection process

Presidio leverages a set of recognizers, each capable of detecting one or more PII entities in one or more languages. The process, described in figure 1, is generally comprised of 8 different steps :

1. Get a request for anonymization from the user
2. Pass request to Presidio-Analyzer for PII entities identification
3. Extract NLP features (lemmas, named entities, keywords, part-of-speech etc.), to be used by the various recognizers
4. Fetch all PII recognizers (predefined + custom from the Recognizer Store service)
5. Run all recognizers
6. Aggregate results
7. Pass to Presidio-Anonymizer for de-dentification
8. Return de-identified text to caller

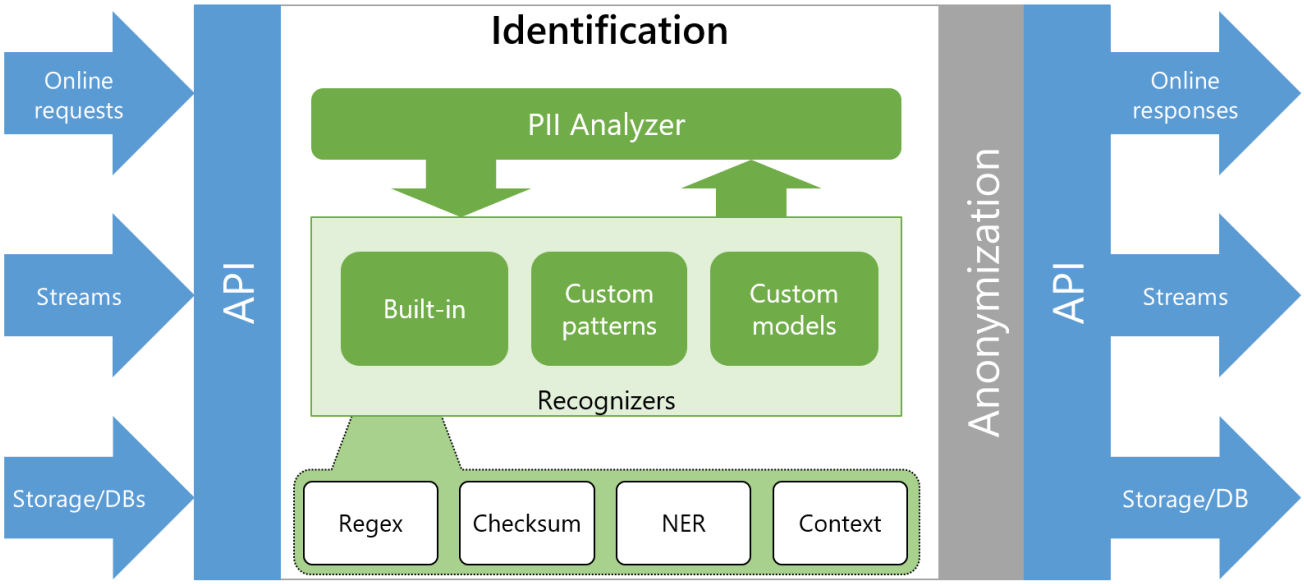


Figure 1 — Animation of the identification process in Presidio

The following animation in figure 2 illustrates the same process on one specific example. Notice that when we change the context from “phone number” to “pool card number” the confidence of the phone number recognizer decreases. This is a screenshot of the Presidio Demo. Check it out.

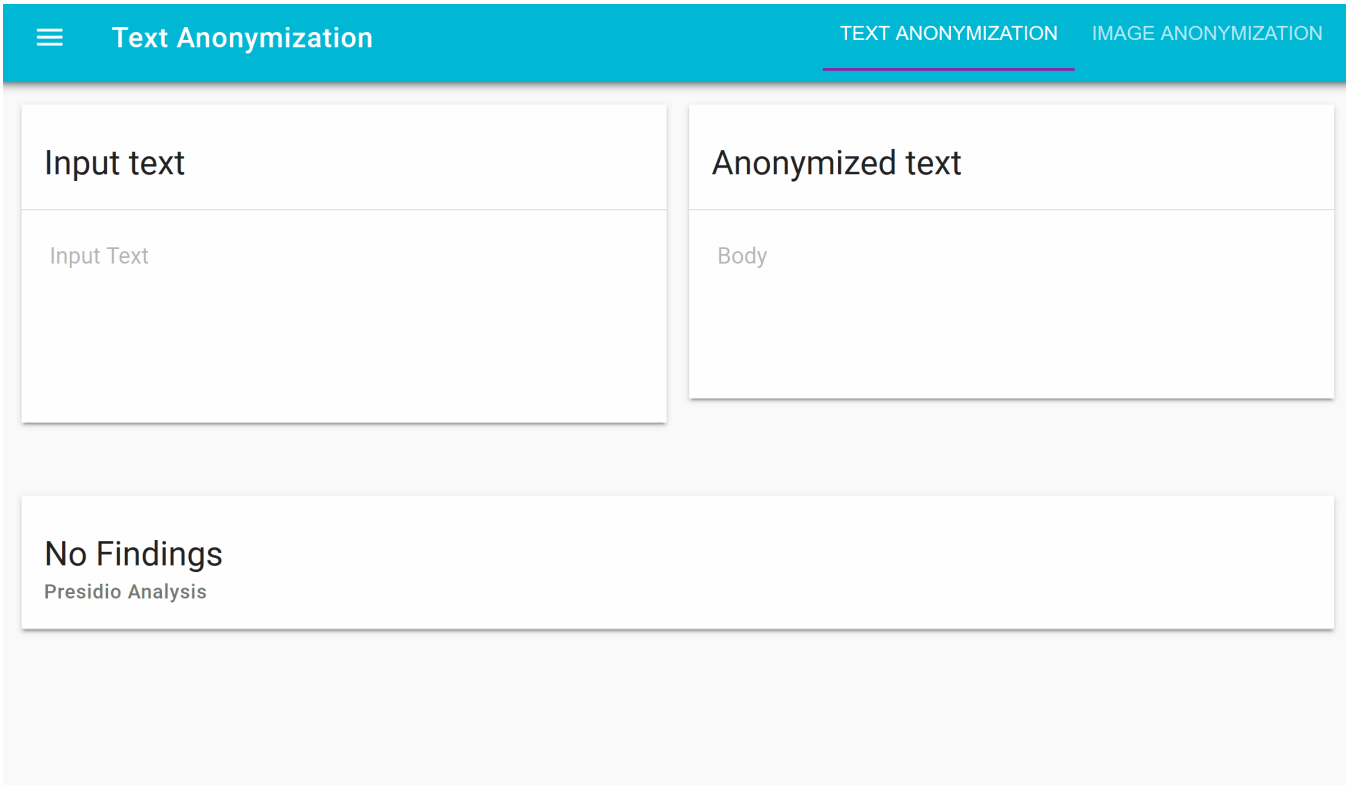


Figure 2 — Example input and output

NLP for data anonymization

PII recognizers are required to detect different types of entities in free text. Different NLP approaches come to mind for such task:

- For entities which share a pattern, we could leverage **Regular Expressions**, **validation** (e.g. checksum) and **context** of surrounding words. This logic could be used, for example, to detect a credit card number or phone number.
- For a finite list of options, we could use a **blacklist.** It could either be a static black-list (e.g. all the titles: *Mr., Ms., Mrs. Miss, Dr., Prof., …*) or dynamic (i.e. connected to a database and queries all possible options).
- For entities which can be identified using specific logic, we can write **rule-based** recognizers.
- For entities that require natural language understanding of the input, we could train **Machine Learning** models, specifically for **Named Entity Recognition** (NER), or use pre-trained models.

In the next section, we will focus on the work we did around improving the Named Entity Recognition rates for person names, locations and organizations.

NER for person names, locations and organizations

To improve our detection rates on these three entities, we experimented with different models. The following section describes the datasets used, different models evaluated and results. The code for running this process can be found on our GitHub repo for research.

Datasets

While some labeled datasets are available, we wanted to increase the coverage of names, organizations, and locations. Thus we started with a labeled dataset (e.g., OntoNotes or CoNLL-2003), and processed it to extract templates. These examples were later used to generate new sentences with a wider variety of entity values (names, organizations and locations) than the original dataset.

For example, from the sentence "Thank you ***George***!", where George was manually labeled as person, we extracted the following template: "Thank you [PERSON]!". Figure 3 provides an additional example:

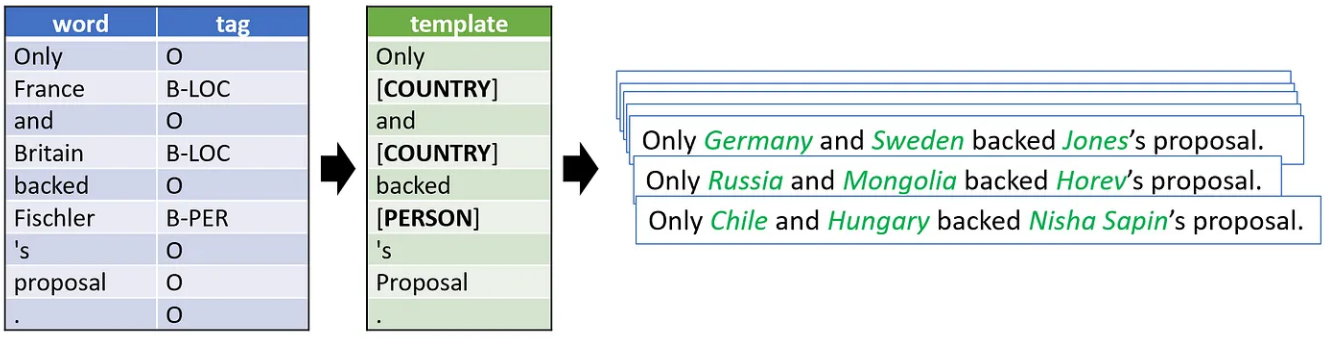


Figure 3 — Data augmentation example

We used a fake PII dataset and multiple fake PII generators to sample entities and create new sentences. These sentences were automatically labeled during the generation process, so training new NER models was easily applicable.

However, this process required us to do some pre-processing of the dataset and come up with creative solutions to different problems. To name a few:

- How to treat *country* vs. *nationality*? In many cases, a "location" entity refers to a nationality (or nation man or nation woman). For example, In the sentence "*The El-Salvadorian [LOC] boxer won the world championship*", we can't replace "*El-Salvadorian*" with "*Ethiopia*" as this would make the sentence incorrect. Therefore, we have created new intermediate entities for Country, Nationality, Nation-man and Nation-woman.

- How should *gender* be handled? Some sentences originally talked about a male or female, but during the data generation process we might replace a name with a name from the opposite gender.
- Should real-life knowledge matter? If the original sentence describes a conflict between two countries, should we just replace these countries with arbitrary country names?
- Some person names in the dataset are the name of an institution or an organization, like the "Allen AI Institute" or "the Trump administration". In this case, should we replace "Trump" with an arbitrary name?

From 8,000 unique templates, we extracted 80,000 different labeled sentences. We also obtained a cleaner dataset than the previous one, as many unlabeled entities were handled during this process. The new dataset was split to train/test/validation sets, with no samples from the same template appearing in more than one set. In addition, we made 10% of the samples lower-case, as this generally represents the ratio of lowercase texts we might meet when the model is deployed. **To the best of our knowledge, this the biggest PII dataset available to this day.**

Models

Different approaches for modeling were evaluated. Specifically, we looked at Conditional Random Fields, spaCy based models and Flair based models.

spaCy: spaCy is a production grade NLP library for tokenization, part-of-speech tagging, entity extraction, text classification and more. It contains a convolutional neural network model which is considered to be the fastest deep learning NLP model. While other models have higher accuracy on public datasets, it's likely that they suffer from longer training and inference times. spaCy also provides fast tokenization and lemmatization, which is used in the context analysis module in Presidio. We evaluated different flavors of spaCy: First, we looked at results from the pretrained spaCy models (both 2.1.0 and 2.2.0), Then, we looked at fine-tuning a pretrained spaCy model, and finally, at training a spaCy model from scratch while leveraging pretrained word embeddings (FastText).

Flair: Flair is a deep learning NLP toolkit with promising results on public datasets. It is built on top of PyTorch, and features both special embedding techniques (called Flair embeddings) and prediction models. Furthermore, it provides easy integration with other embedding models (like BERT, ELMo) and stacking of embeddings from

different models and sources. We evaluated two different Flair based models: A Flair model with BERT embeddings, and a Flair model with a stacking of Flair embeddings and GloVe embeddings.

Conditional Random Fields (CRF): CRFs are a class of methods for sequence tagging. These discriminative graphical models learn the dependencies between predictions and are a natural fit for Named Entity Recognition tasks. Prior to the introduction of recurrent and convolutional neural networks, CRFs achieved state of the art performance on NER tasks on public datasets. They are much faster in training and prediction than neural-network-based models and provide relatively interpretable results. We evaluated only the vanilla CRF model using L-BGFS optimization, using the sklearn-crfsuite Python package.

To conclude, here are the models we experimented with:

1. Evaluate using default pre-trained models:

- **spaCy** 2.1.0
- **spaCy** 2.2.0 (which has better support for lower-case entities)

2. Train from scratch:

- **Conditional Random Fields** (CRF)

3. Train using pretrained embeddings:

- **spaCy** with **FastText** embeddings
- **Flair** with **BERT** embeddings
- **Flair** with **GloVe** and **Flair** embeddings

4. Fine-tune an existing trained model:

- **spaCy** version 2.2.0

Metrics

We focused on F2 (recall is more important than precision) of the PII/no-PII binary decision. We also looked at specific classes F2 and computational performance. Lastly, interpretability was an additional factor to consider.

Results

As can be seen in figure 4, all models achieved decent results, but Flair based models were superior.

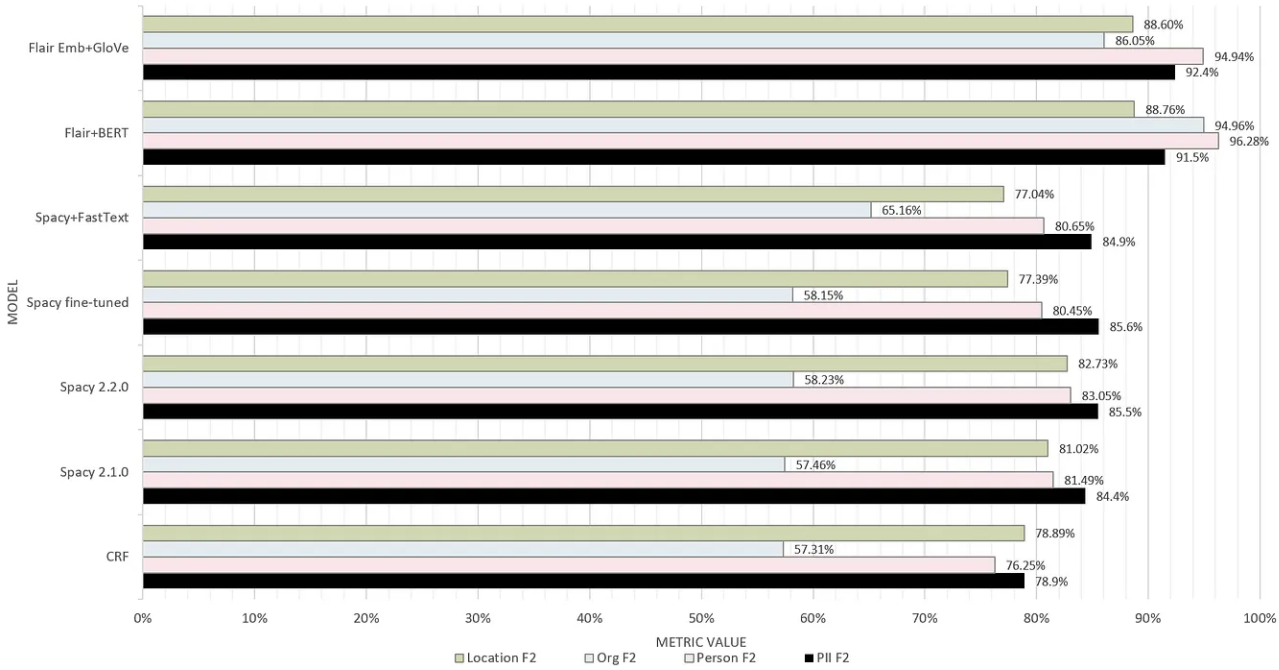


Figure 4 — F2 results on various models and entities

Some insights from these results:

- Flair based models achieved higher results than spaCy based models, which achieved higher results on all entities than CRF.
- We could see that spaCy 2.2.0 provides better results than 2.1.0 as expected, but not by a large amount.
- spaCy and CRF models struggled with the Org entity, which is often confused with person names. Flair models achieve much higher results on Org which might hint on overfitting to the generated fake PII entities for Org.
- Training spaCy models or fine-tuning them fail to improve the model's F2 score. This might be due to the nature of the training set, which was derived from the same dataset spaCy was originally trained on (OntoNotes). It might be that other sources of data would benefit from fine-tuning or training these models.

Computational performance

Although Flair models achieve higher F2 scores, they are also much slower to train and predict. Table 1 shows approximated training and inference time using the various methods evaluated. The analysis was made on one GPU machine[1].

Model	**Inference response time** (milliseconds)	**Training time** (hours)
Flair	111.11	~120
Spacy	9.62	~18
CRF	0.45	~0.5

Table 1 — Inference and training time for the different models

[1] NC6 instance on Azure: 6 vCPU, 56GiB Memory, one half NVIDIA Tesla K80, GPU memory 12GiB

Conclusion

In this work, we evaluated various models for better detection rates in Presidio. We considered the trade-off between detection rates and computational performance, which is crucial in many use cases. For the datasets we evaluated, we see that there isn't a practical reason to replace the current spaCy model we use in Presidio. It is, however, possible to apply the data augmentation framework and the different models evaluated to new data, and tailor Presidio towards more domain-specific datasets. We also see that by having a CRF model instead of spaCy, we can potentially improve Presidio's run time. If performance is not an issue, for example for offline jobs, we should consider using a Flair based approach, potentially with Flair embeddings + GloVe, to improve the detection rates in Presidio.

Presidio is fully open source and available for free, for anyone who wishes to address the data privacy problem. We also welcome contributors, pull requests or any kind of feedback. Click here to get started.

For more information

1. Presidio's GitHub repo
2. General Presidio Documentation
3. Research, dataset creation and modeling code

Other references

1. spaCy
2. Flair
3. Fake Name Generator
4. Faker

About the author

Omri Mendels is a Principal Data Scientist @ Microsoft. You can connect with him on LinkedIn, GitHub or Quora.

Machine Learning | Naturallanguageprocessing | Pii | Privacy | NLP

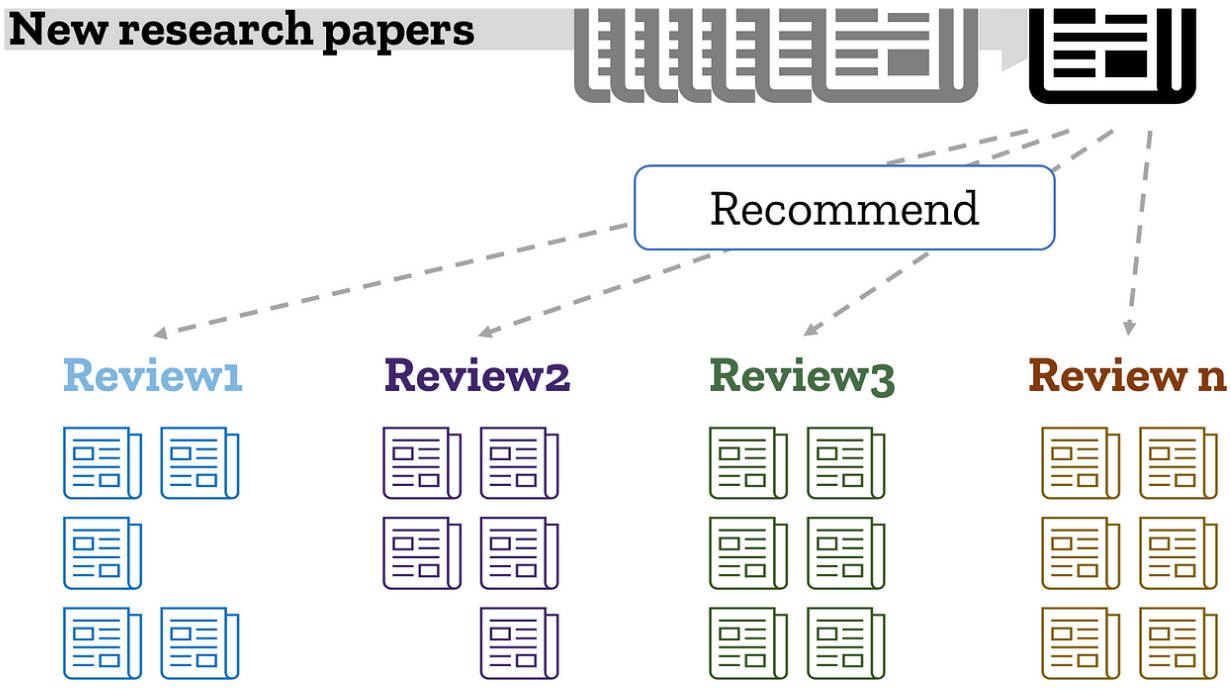
Follow

Written by Omri Mendels

161 Followers · Writer for Towards Data Science

Principal Data Scientist at Microsoft. Loves data, coding and bringing ML to life

More from Omri Mendels and Towards Data Science



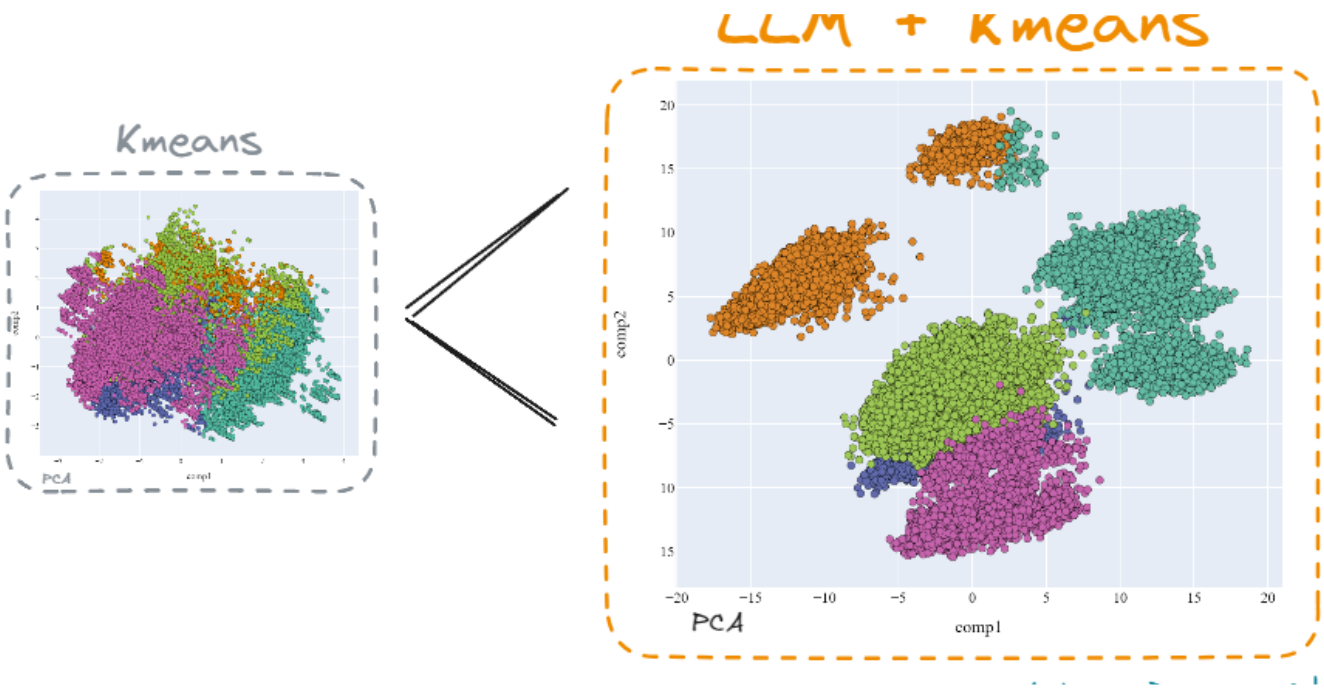
Omri Mendels in Data Science at Microsoft

How we reduced our text similarity runtime by 99.96%

In many NLP pipelines, we wish to compare a query to a set of text documents. The process usually involves some encoding or embedding of...

7 min read · Mar 23, 2021

743 3



Damian Gil in Towards Data Science

Mastering Customer Segmentation with LLM

Unlock advanced customer segmentation techniques using LLMs, and improve your clustering models with advanced techniques

23 min read · Sep 26

3.1K 25

Khouloud El Alami in Towards Data Science

Don’t Start Your Data Science Journey Without These 5 Must-Do Steps From a Spotify Data Scientist

A complete guide to everything I wish I’d done before starting my Data Science journey, here’s to acing your first year with data

18 min read · Sep 24

2.4K 23

Omri Mendels in Towards Data Science

What is it with NLP models and biblical names?

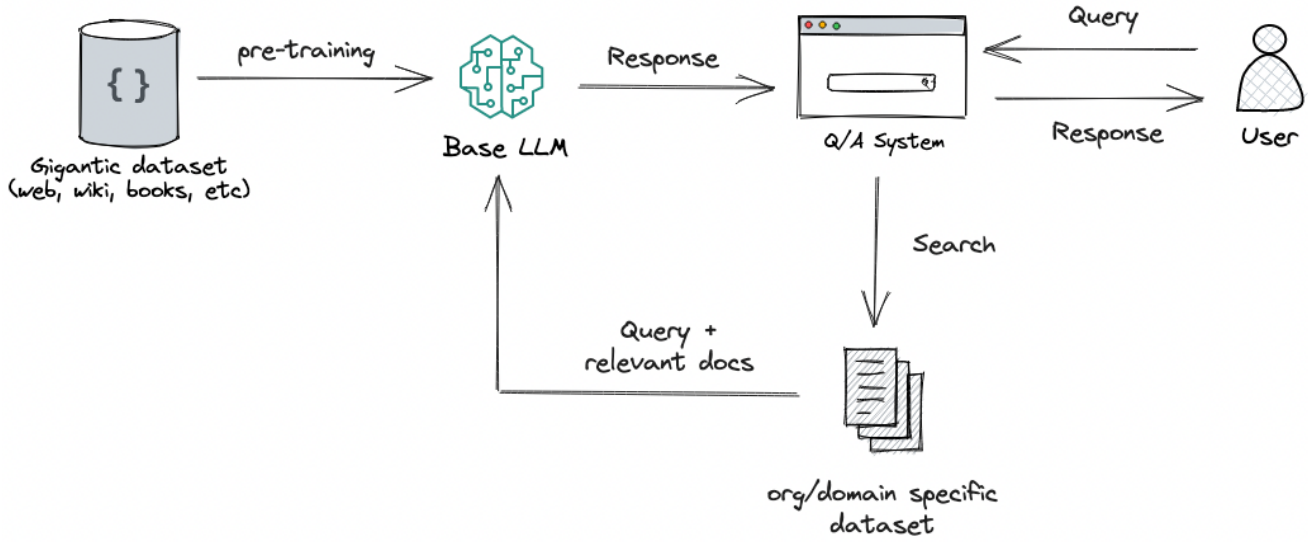
7 min read · May 8, 2021

70

See all from Omri Mendels

See all from Towards Data Science

Recommended from Medium



Heiko Hotz in Towards Data Science

RAG vs Finetuning — Which Is the Best Tool to Boost Your LLM Application?

The definitive guide for choosing the right method for your use case

· 19 min read · Aug 24

2.4K 18

Patrick Meyer in Towards AI

Entity Recognition with LLM: A Complete Evaluation

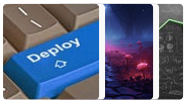
LLMs are capable of performing a wide range of NLP tasks, such as named entity recognition. In this study, I tested an open-source library...

· 10 min read · Sep 1

182

Lists

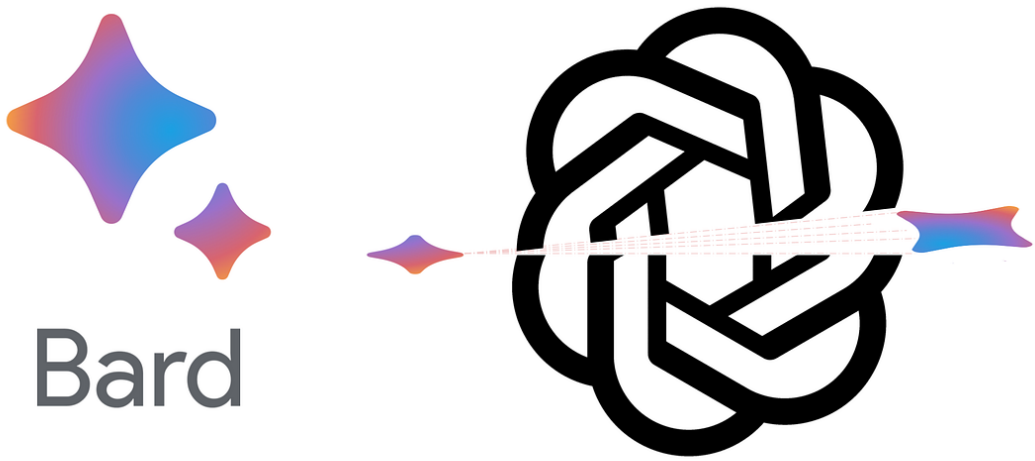
Natural Language Processing
709 stories · 314 saves



Predictive Modeling w/ Python
20 stories · 492 saves

Practical Guides to Machine Learning
10 stories · 562 saves

The New Chatbots: ChatGPT, Bard, and Beyond
13 stories · 143 saves



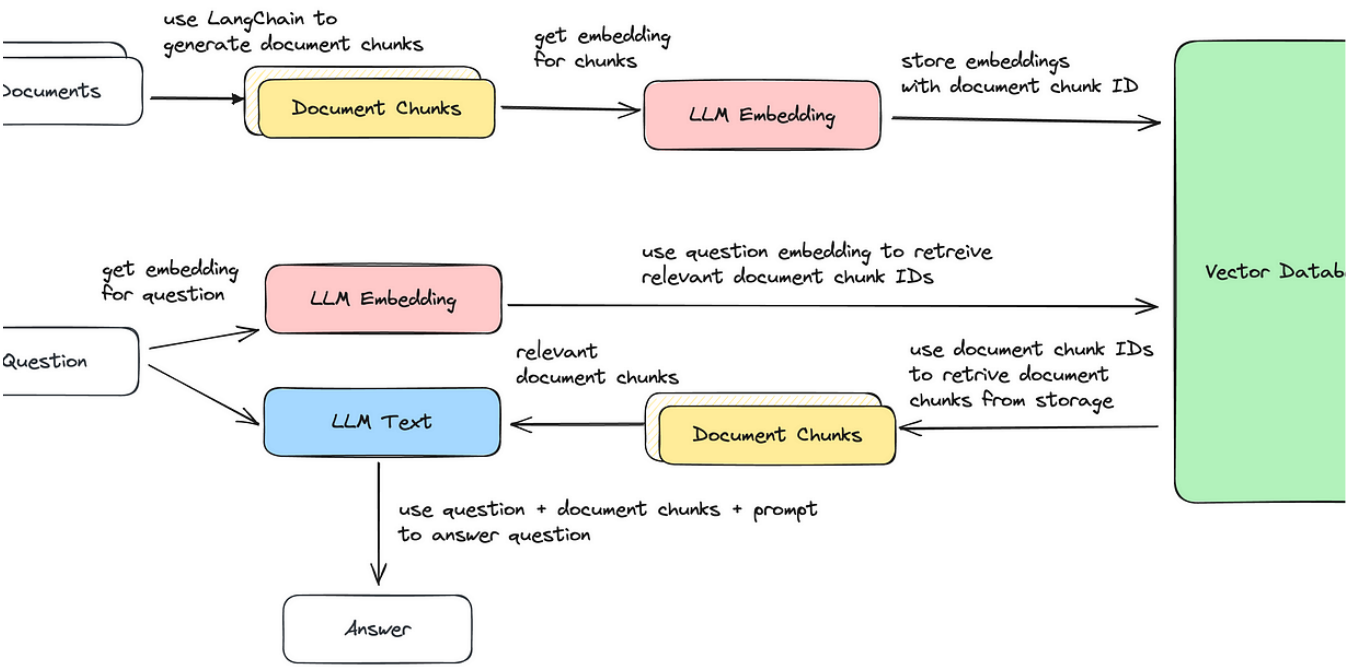
AL Anany

The ChatGPT Hype Is Over—Now Watch How Google Will Kill ChatGPT.

It never happens instantly. The business game is longer than you know.

✨ · 6 min read · Sep 1

14.1K 428



Sascha Heyer in Google Cloud - Community

Generative AI - Document Retrieval and Question Answering with LLMs

Apply LLMs to your domain-specific data

✨ · 7 min read · Jun 5

317 10

Qendel AI in GoPenAI

Bye Bye Llama-2, Mistral 7B is Taking Over: Get Started With Mistral 7B Instruct

Getting started with Mistral 7B and Langchain integration: A step-by-step guide.

✨ · 7 min read · Oct 1

212 3

Pankaj Pandey

spacy-llm: Integrating LLMs into structured NLP pipelines.

It is crucial for everyone, especially machine learning developers, to be familiar with Spacy, the NLP tool widely utilized in the field...

2 min read · May 21

 89  3

See more recommendations