Enterprise Semantic Search with Python Large Language Models

PyData Miami 2022

September 22, 2022

Nelson Correa, Andinum, Inc

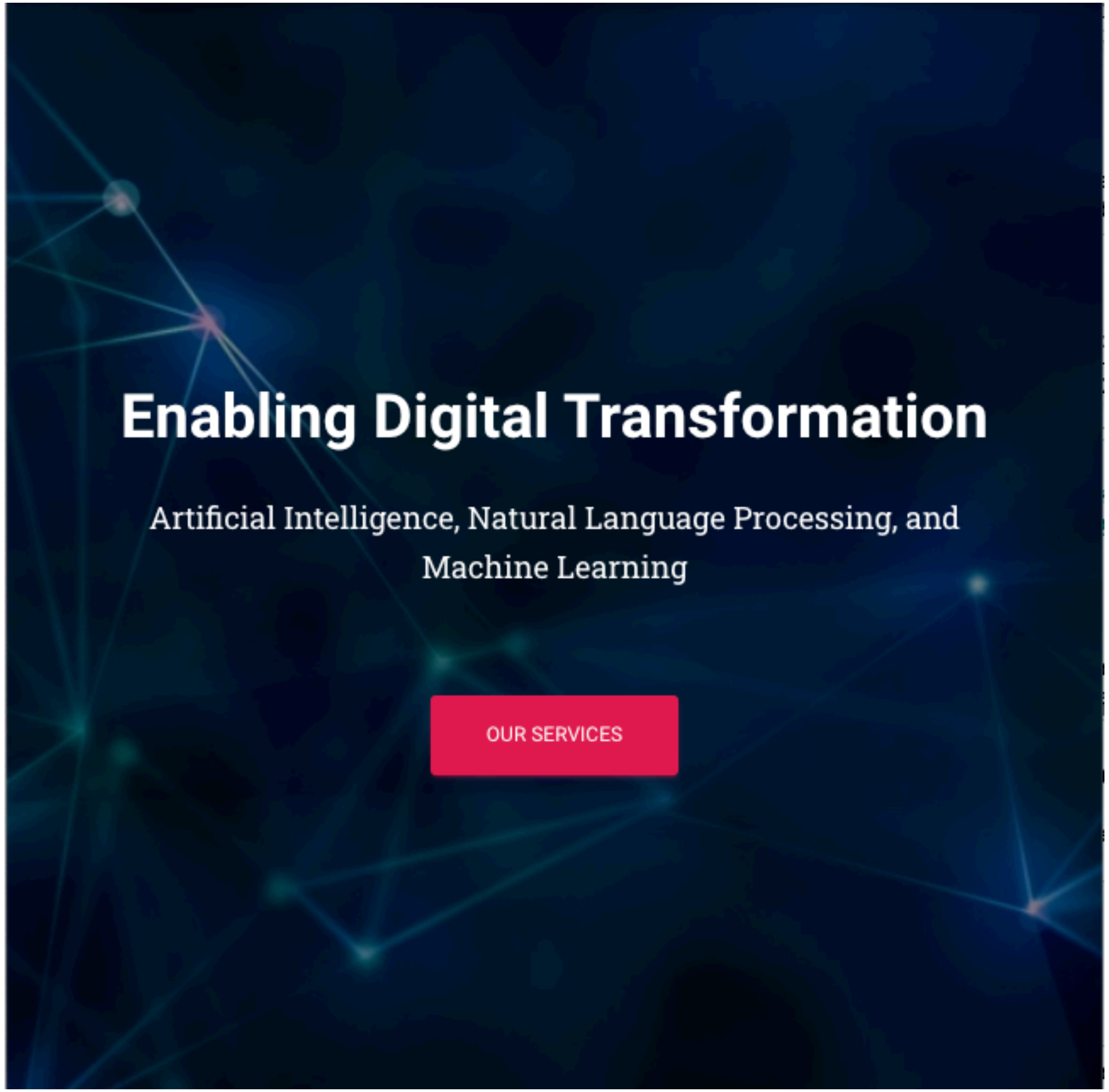
ANDINUM

Who we are

- Document Understanding
- Digital Transformation
- Compliance

Nelson Correa, Ph.D.

- Developer and R&D in AI/ML/NLP
- ex-IBM, ex-Academic, Entrepreneur



Enterprise Semantic Search

- Enterprise Search
- Semantic Search

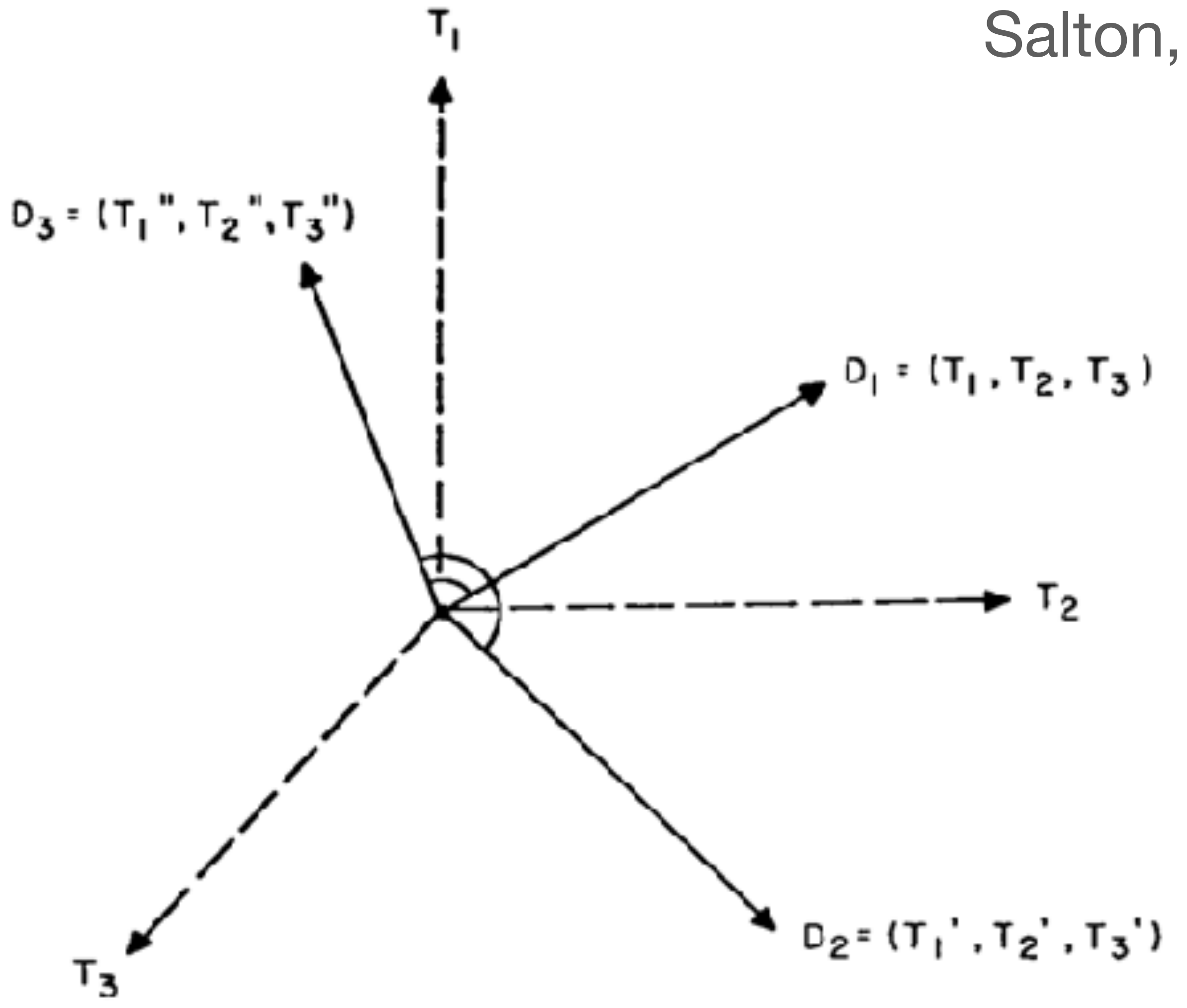
Enterprise and Semantic Search

Exploring term and dense vector indexing

- **Enterprise Search**
  - Multiple enterprise data sources
  - Data, media and documents
  - Technology: Relational, no-SQL, Other
  - Structured and unstructured (text & media)

- **Semantic Search**
  - Data semantics: tokens vs. “*token meanings*”
  - Similarity: discrete symbols vs. dense vectors
  - Semantic Web: URIs, Relations, Schemas

Fig. 1. Vector representation of document space.

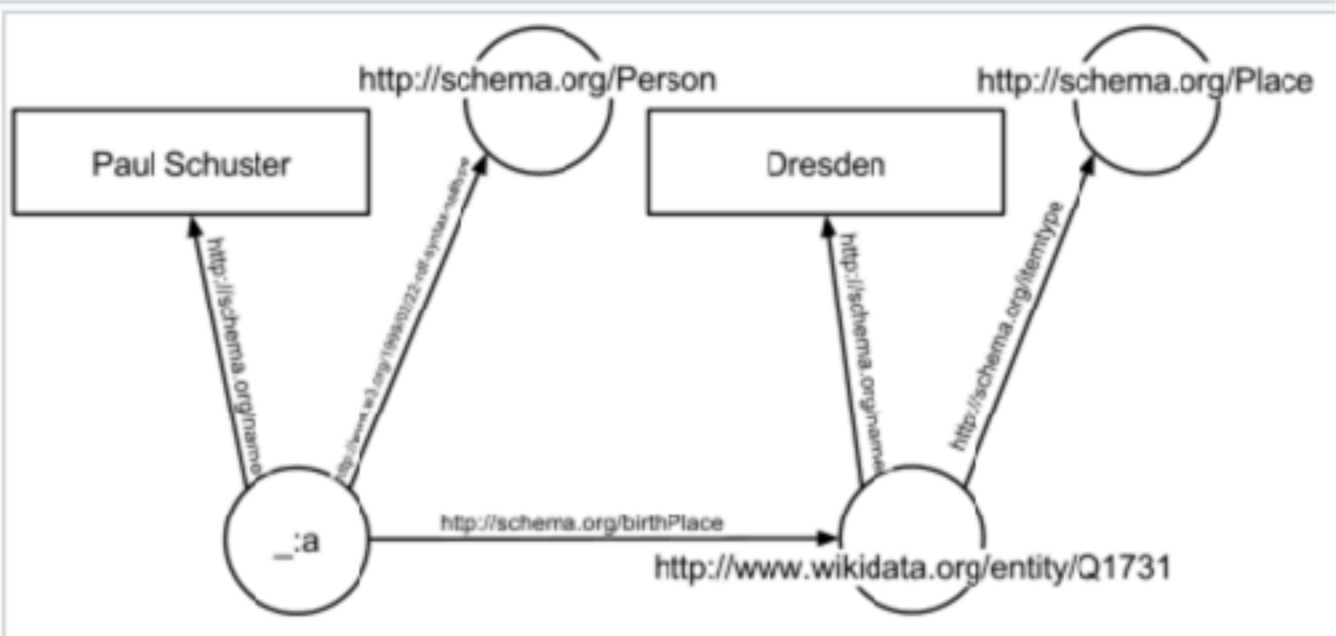


Document vector space model
Each term is a dimension of the space.
Salton, 1975

Neural document embeddings
arbitrary ML-learned dimensions

```
<div vocab="https://schema.org/" typeof="Person">
  <span property="name">Paul Schuster</span> was born in
  <span property="birthPlace" typeof="Place"
href="https://www.wikidata.org/entity/Q1731">
    <span property="name">Dresden</span>.
  </span>
</div>
```

Semantic web
RDFa and RDF Graph
Wikipedia



Graph resulting from the RDFa example

Agenda

Information retrieval, NLP, deep learning and AI models

1. Introduction: Enterprise search and Semantic search
2. Information Retrieval: Traditional and neural
3. NLP and Large Language Models
4. Financial semantic search for CFPB consumer complaints
5. Data visualization in dense vector spaces: UMAP
6. Evaluation, metrics, model risk and ethics

Questions

Modern Information Retrieval (IR)

Search in large document collections

Information Retrieval

Traditional vector space model

- Documents, queries, tokens, document collections
- Vocabulary (*V*): set of tokens in a document collection (|*V*| range 10^4 to $>10^6$)
- Indexing:
  - SMART vector space model (VSM)
  - Each *Document* and *Query* are |*V*|-dim vectors
  - Binary (one-hot), Count, Weighted (e.g., TF-IDF)
- Search: Similarity (*dot-product* or *cosine*) of *Query* and *Document Collection*
- Each token is unrelated to every other one; lexical gap
- Indexing improvements: Stop words, stemming, lemmatizing, query expansion

```
# Sklearn TfidfVectorizer
# Document collection
dc = {
    "d1":"Simple is better than complex.",
    "d2":"Complex is better than complicated.",
    "d3":"Flat is better than nested.",
    "d4":"Sparse is better than dense."
}

from sklearn.feature_extraction.text import TfidfVectorizer
vectorizer = sk.feature_extraction.text.TfidfVectorizer()

vectorizer.fit(dc.values())
vec_features = vectorizer.get_feature_names()
print(f"Vectorizer features: {len(vec_features)} tokens\n{', '.join(vec_features)}")
```

```
Vectorizer features: 10 tokens
better, complex, complicated, dense, flat, is, nested, simple, sparse, than
```

```
# Vectorize documents
DC = vectorizer.transform(dc.values()) # Sparse matrix
pd.DataFrame(DC.toarray(), columns=vec_features)
```

	better	complex	complicated	dense	flat	is	nested	simple	sparse	than
0	0.334174	0.504879	0.000000	0.000000	0.000000	0.334174	0.000000	0.640375	0.000000	0.334174
1	0.334174	0.504879	0.640375	0.000000	0.000000	0.334174	0.000000	0.000000	0.000000	0.334174
2	0.310920	0.000000	0.000000	0.000000	0.595813	0.310920	0.595813	0.000000	0.000000	0.310920
3	0.310920	0.000000	0.000000	0.595813	0.000000	0.310920	0.000000	0.000000	0.595813	0.310920

Vocabulary size = 10
Four documents d1, …, d4 indexed as 10-dimensional vectors

Information Retrieval

Dense vector encoder (BERT)

- Traditional VSM is *high-dimensional* (|*V*|) and *sparse*, and tokens are *discrete* (symbolic).
- Dense vector space representations (50 to +1000-dim)
    - LSA (SVD), LDA dimension reduction
    - word2vec, doc2vec, FastText, neural methods
    - Transformer models (BERT, dim = 768)
- Methods: self-supervised ML
    - corpus-based statistics and tasks (e.g., MLM), non-contextual/contextual word representations
    - Bi-directional or auto-regressive models
- ADVANTAGE: *Word*, *sentence* and *document* similarity, via vector similarity (cosine or dot-product)

```
# from transformers import AutoTokenizer, TFAutoModel
model_ckpt = "ProsusAI/finbert"
tokenizer = hf.AutoTokenizer.from_pretrained(model_ckpt)
model = hf.TFAutoModel.from_pretrained(model_ckpt, from_pt=True)

def cls_pooling(model_output):
    return model_output.last_hidden_state[:, 0]

def get_embeddings(text_list):
    encoded_input = tokenizer(text_list, padding=True,
                              truncation=True, return_tensors="tf")
    encoded_input = {k: v for k, v in encoded_input.items()}
    model_output = model(**encoded_input)
    return cls_pooling(model_output)

model.summary()
```

```
Model: "tf_bert_model_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 bert (TFBertMainLayer)      multiple                  109482240

=================================================================
Total params: 109,482,240
Trainable params: 109,482,240
Non-trainable params: 0
```

```
# FinBERT: Vectorize documents
embeds_text = list(dc.values())
embedding = get_embeddings(embeds_text)
pd.DataFrame(embedding, columns=None)
```

	0	1	2	3	4	5	6	7	8	9	...	758
0	0.190899	0.874197	-1.032465	-0.137693	-0.306911	-0.725966	0.436320	0.340217	1.060956	-1.084645	...	0.058592
1	0.107703	0.896653	-1.145805	-0.081705	-0.417182	-0.801632	0.639703	0.530051	1.056912	-0.929148	...	0.237219
2	0.271673	0.214038	-0.597655	-0.412695	-0.234643	-0.437923	-0.073856	0.258040	0.904815	-0.660722	...	0.194288
3	0.025600	0.475642	-0.886289	-0.326687	-0.135318	-1.126632	0.167423	0.529638	0.543420	-0.991395	...	0.336393

4 rows × 768 columns

BERT output embedding dimension = 768
Four documents d1, …, d4 indexed as 768-dimensional vectors

Financial semantic search of CFPB consumer complaints

Financial semantic search

Consumer financial complaints

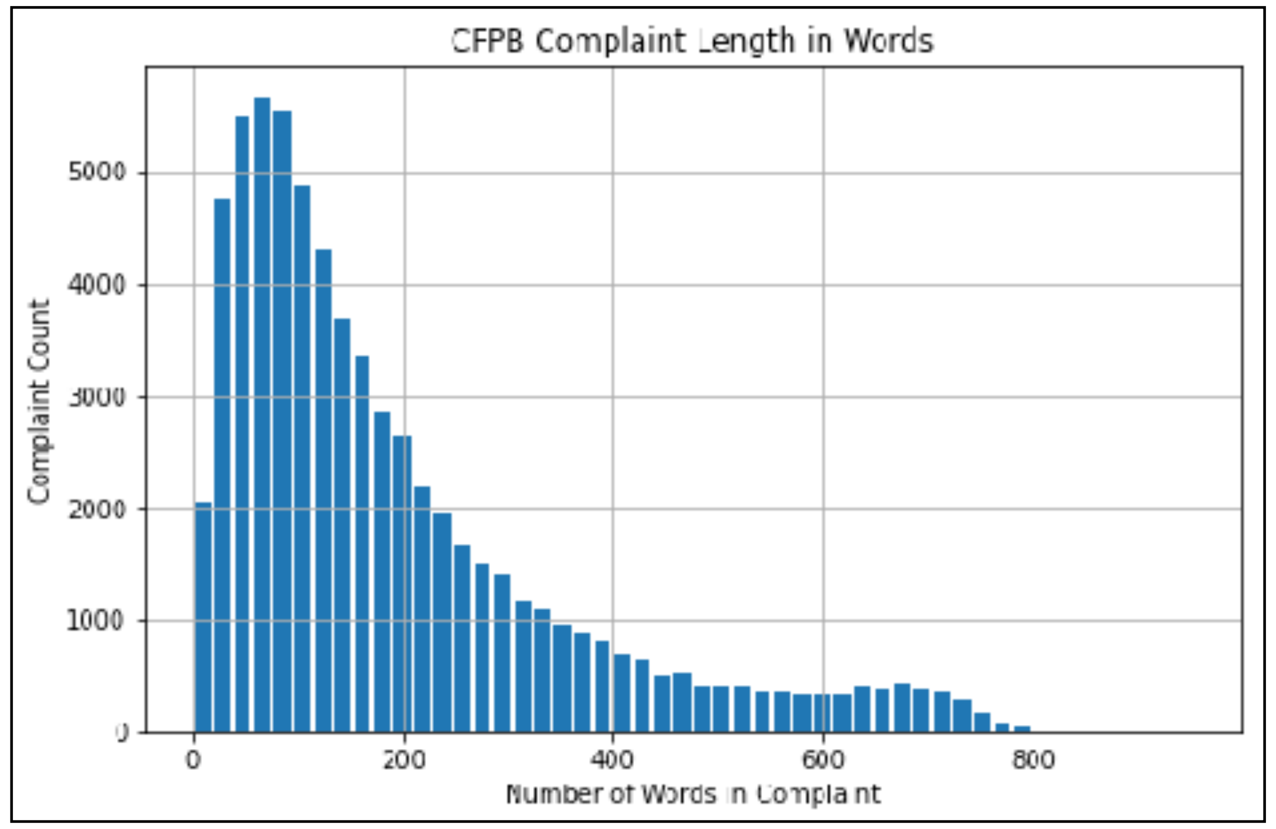
- FinTEC and FinNLP
  - Financial data and document analysis; structured and unstructured data
  - Extraction, classification, search, prediction, …
- Consumer Financial Protection Bureau (CFPB), 2010
- CFPB consumer complaints database (CCD)
  - Collected since 2011, over 2,600,000 complaints (994,000 in 2021)
  - “*complaint_what_happend*” (narrative text)
  - Label fields: Company, Product, Issue
  - Other fields: Date, State, ZIP code

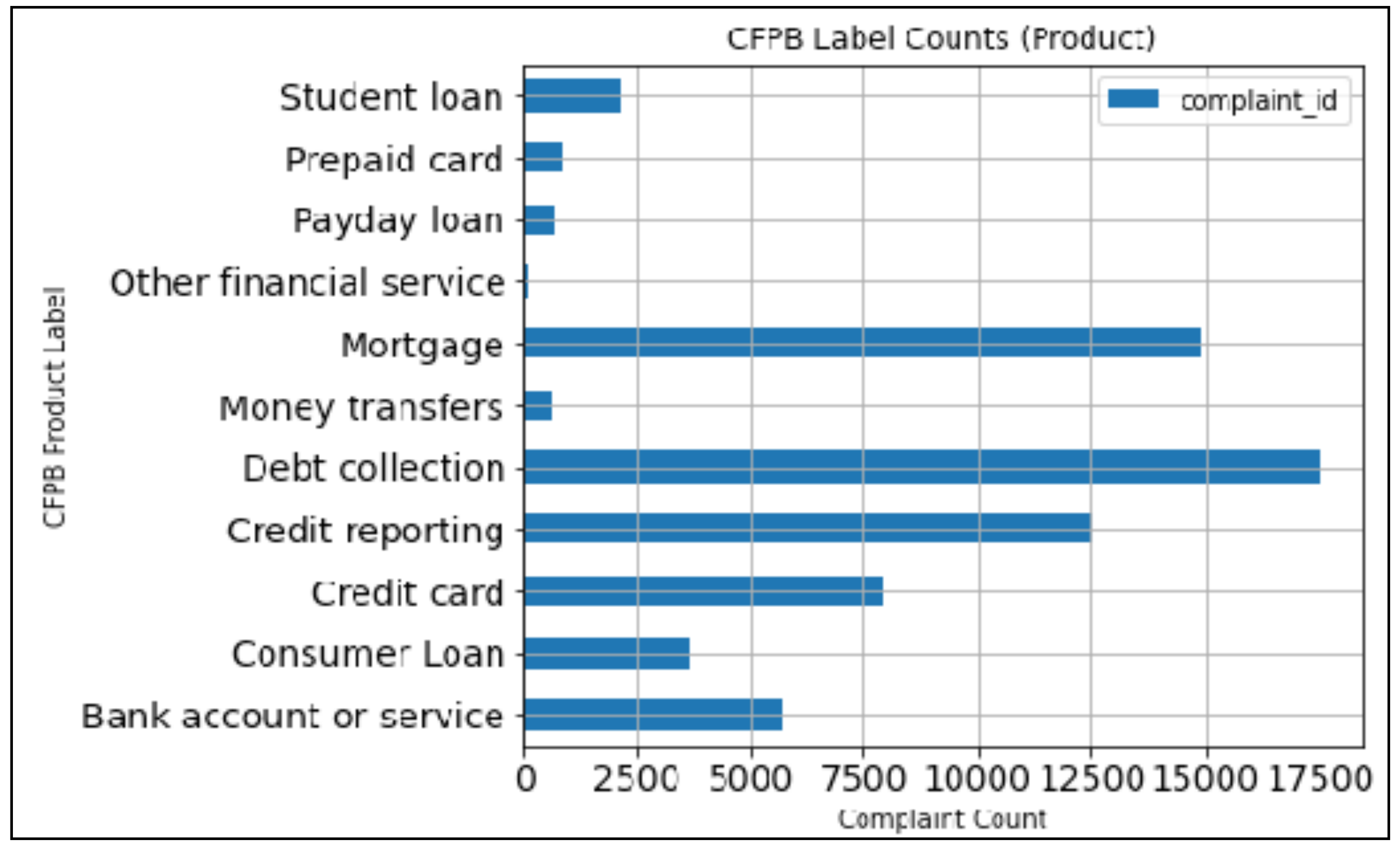
```
# CSV or JSON files
cfpb_base = "/home/nelson/dataset/regulator/cfpb/"                    # Ubuntu
cfpb_base = "/Users/nelson/dev/datasets/nlp/cfpb/cfpb_kaggle/"        # MBP macOS,
cfpb_kaggle_csv_fn = "cfpbk-consumer_complaints.csv"

print(f"READ path: {cfpb_base}")
print(f"READING ... cfpb_kaggle_csv_fn: {cfpb_kaggle_csv_fn}")
ccd_df = pd.read_csv(cfpb_base+cfpb_kaggle_csv_fn)

print(f"ccd_df.shape: {ccd_df.shape}")
print(f"ccd_df.columns: {ccd_df.columns.to_list()}\n")
```

	date_received	product	sub_product	issue	sub_issue	consumer_complaint_narrative	company_public_response
553086	02/11/2016	Payday loan	Payday loan	Charged fees or interest I didn't expect	Charged fees or interest I didn't expect	I have been paying {$180.00} a month through d...	NaN
553090	03/30/2016	Mortgage	Conventional fixed mortgage	Application, originator, mortgage broker	NaN	I recently became aware that Amerisave Mortgag...	Company believes it acted appropriately as aut...
553096	02/12/2016	Mortgage	Conventional fixed mortgage	Application, originator, mortgage broker	NaN	Bank of America has demonstrated an on-going l...	Company has responded to the consumer and the ...



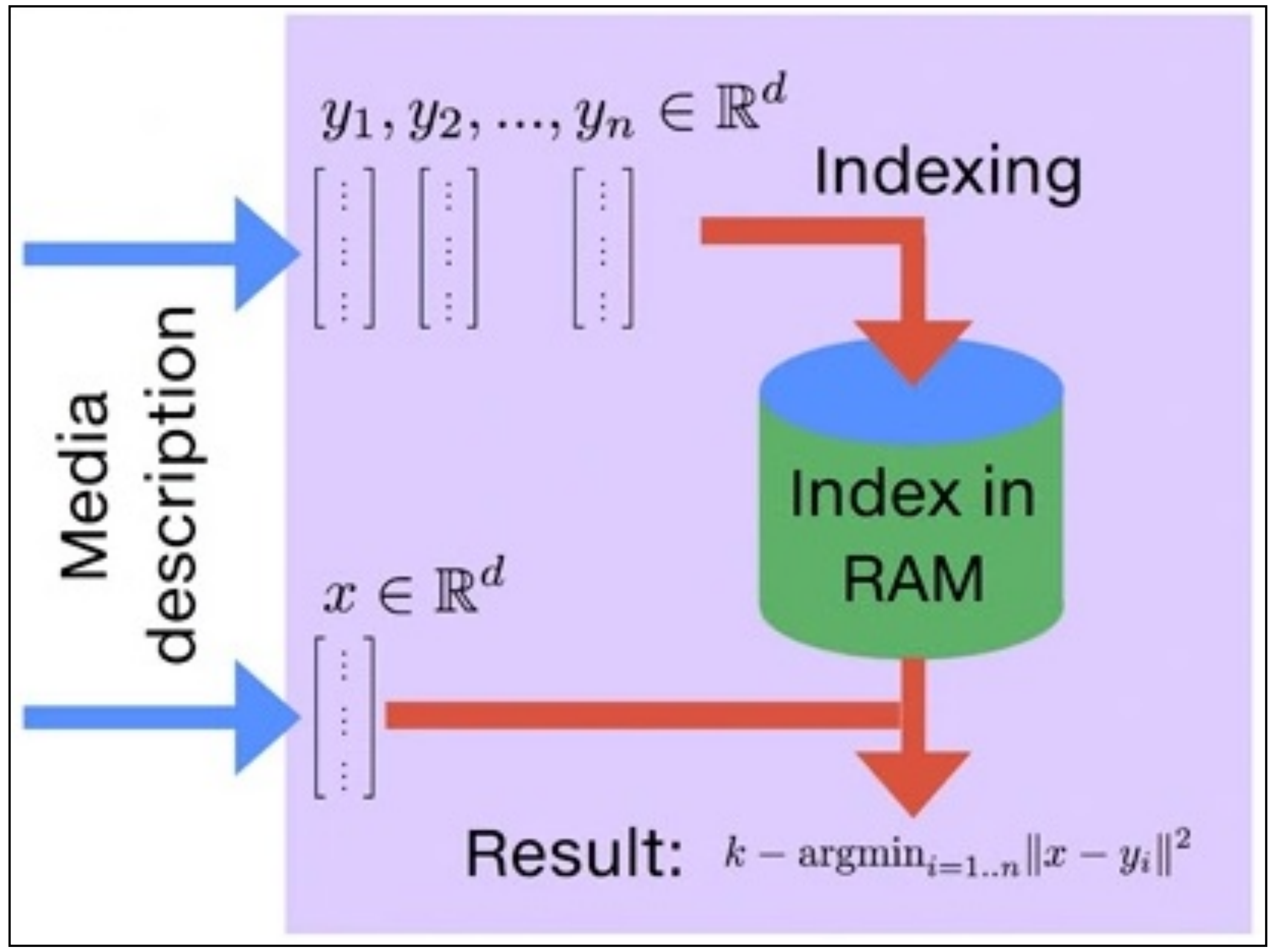


CFPB Semantic search

FinBERT and FAISS

- Document embeddings: FinBERT (FIN custom BERT)
    - https://arxiv.org/abs/1908.10063
- ANN: FAISS indexing and search (*Meta*)
    - Billion-scale data sets (k-NN graph)
    - Original use case: Image search
    - Input: object dense embedding (image, text, …)
    - https://arxiv.org/abs/1702.08734 (FAISS)
- Available in *HuggingFace transformers library* and several db engines (Elasticsearch, PostgreSQL)
- CFPB Semantic search application (notebook): FinBERT, FAISS, Elastic/Postgres, Flask/Node

FAISS - source: *Meta*




```
# Add FAISS embeddings
embeddings_dataset.add_faiss_index(column="embeddings")
embeddings_dataset
```

100% 49/49 [00:00<00:00, 139.38it/s]

```
# Sample query embedding
query = "there is incorrect information on my credit report"
query_embedding = get_embeddings([query]).numpy()

# Score and return top-k = 20
scores, samples = embeddings_dataset.get_nearest_examples(
    "embeddings", query_embedding, k=20
)
```

	date_received	product	consumer_complaint_narrative	complaint_id	embeddings	score
19	06/01/2015	Credit reporting	Attached to this complaint are XXXX pages:1. S...	1399498	[0.10054400563240051, -0.17099657654762268, -0...	327.929016
18	08/13/2015	Money transfers	I am filing this complaint regarding Pay Pal. ...	1518475	[0.27614957094192505, 0.31134259700775146, -0....	327.904968
17	09/03/2015	Credit card	i got the protection insurance on this account...	1550194	[-0.04275791347026825, -0.1680782437324524, -0...	327.895844

CFPB search examples

Sample queries & Demo

- Example queries (information needs)
  - Use cases (IR, QA …); granularity query/document
  - Queries about product, issue, company, sentiment
  - Meta-queries (queries about the collection)
- Comparison to other models and approaches
  - Alternate transformer models: Multi-QA-MPNet; DistilBERT MS-MARCO
  - TfidfVectorizer/BM25: add_faiss_index vs. add_elasticsearch_index
- Speed (Intel i7-860 Processor; 66,000 records)
  - Query embedding: 150 ms
  - FAISS k-NN search: 20 ms

***Query*: there is incorrect information on my credit report**

[1] - complaint_id: 1636731, date: 11/03/2015, score: 20.44

I an writing in regards to my credit balance with XXXX XXXX XXXX. I currently have a maximum credit limit available to me of {$800.00}, which I have owed over {$790.00}. However, I have paid this credit card down to {$280.00} and this amount has not been reported on my credit report. Therefore, my credit card utilization is incorrect and reflects inaccurate information ... which has reduced my credit score.

company: *Equifax*, product: *Credit reporting*, issue: *Incorrect information on credit report*

[2] - complaint_id: 1548154, date: 09/02/2015, score: 20.40

This account was paid Contacted XXXX XXXX twice for payments arrangements they ignored my requests before it went to collections Contacted collection agency and informed them that I had contacted XXXX XXXX twice to set up payments and was ignored and I would be sending payments directly to XXXX XXXX I started making payments XX/XX/XXXX and paid off the account in XX/XX/XXXX I paid {$50.00} monthly This account should have never been a collection/charge off I made a good faith offer to pay this bill prior to them charging it off I became unemployed but still tried to pay this bill when I fell behind in payments XXXX XXXX ignored my requests I wrote to them and was ignored

company: *Equifax*, product: *Credit reporting*, issue: *Incorrect information on credit report*

[3] - complaint_id: 1538562, date: 08/26/2015, score: 20.38

I submitted a fax claiming that it was fraud and they never got back to me.

company: *Equifax*, product: *Credit reporting*, issue: *Incorrect information on credit report*

Evaluation, visualization, risk

Evaluation

Benchmarks, systems, metrics

- Use dense embeddings with caution, per use case
- Evaluation benchmark
  - Document collection
  - Queries (information needs, use cases)
  - Document relevance judgements
- Systems: Models & FAISS vs. e.g., BM25 (Elasticsearch)
  - BM25 is a robust and competitive baseline
- Metrics: Precision, recall, F1, MAP, MRR, Recall at K
- Existing benchmarks:
  - REUTERS, NIST MUC, TREC, CLIR,
    LETOR Learning to Rank, MS-MARCO, MRPC
  - BEIR (UKP-TUDA): https://github.com/UKPLab/beir

Model (→)	Lexical	Sparse			Dense			
Dataset (↓)	BM25	DeepCT	SPARTA	docT5query	DPR	ANCE	TAS-B	GenQ
MS MARCO	0.228	0.296‡	0.351‡	0.338‡	0.177	0.388‡	0.408‡	0.408‡
TREC-COVID	0.656	0.406	0.538	0.713	0.332	0.654	0.481	0.619
BioASQ	0.465	0.407	0.351	0.431	0.127	0.306	0.383	0.398
NFCorpus	0.325	0.283	0.301	0.328	0.189	0.237	0.319	0.319
NQ	0.329	0.188	0.398	0.399	0.474‡	0.446	0.463	0.358
HotpotQA	0.603	0.503	0.492	0.580	0.391	0.456	0.584	0.534
FiQA-2018	0.236	0.191	0.198	0.291	0.112	0.295	0.300	0.308
Signal-1M (RT)	0.330	0.269	0.252	0.307	0.155	0.249	0.289	0.281
TREC-NEWS	0.398	0.220	0.258	0.420	0.161	0.382	0.377	0.396
Robust04	0.408	0.287	0.276	0.437	0.252	0.392	0.427	0.362
ArguAna	0.315	0.309	0.279	0.349	0.175	0.415	0.429	**0.493**
Touché-2020	**0.367**	0.156	0.175	0.347	0.131	0.240	0.162	0.182
CQADupStack	0.299	0.268	0.257	0.325	0.153	0.296	0.314	0.347
Quora	0.789	0.691	0.630	0.802	0.248	0.852	0.835	0.830
DBPedia	0.313	0.177	0.314	0.331	0.263	0.281	0.384	0.328
SCIDOCS	0.158	0.124	0.126	0.162	0.077	0.122	0.149	0.143
FEVER	0.753	0.353	0.596	0.714	0.562	0.669	0.700	0.669
Climate-FEVER	0.213	0.066	0.082	0.201	0.148	0.198	0.228	0.175
SciFact	0.665	0.630	0.582	0.675	0.318	0.507	0.643	0.644
Avg. Performance vs. BM25		**- 27.9%**	**- 20.3%**	**+ 1.6%**	**- 47.7%**	**- 7.4%**	**- 2.8%**	**- 3.6%**

BEIR benchmark zero-shot system performance (https://arxiv.org/abs/2104.08663)
Source: Ubiquitous Knowledge Processing Lab, Technische Universität Darmstadt

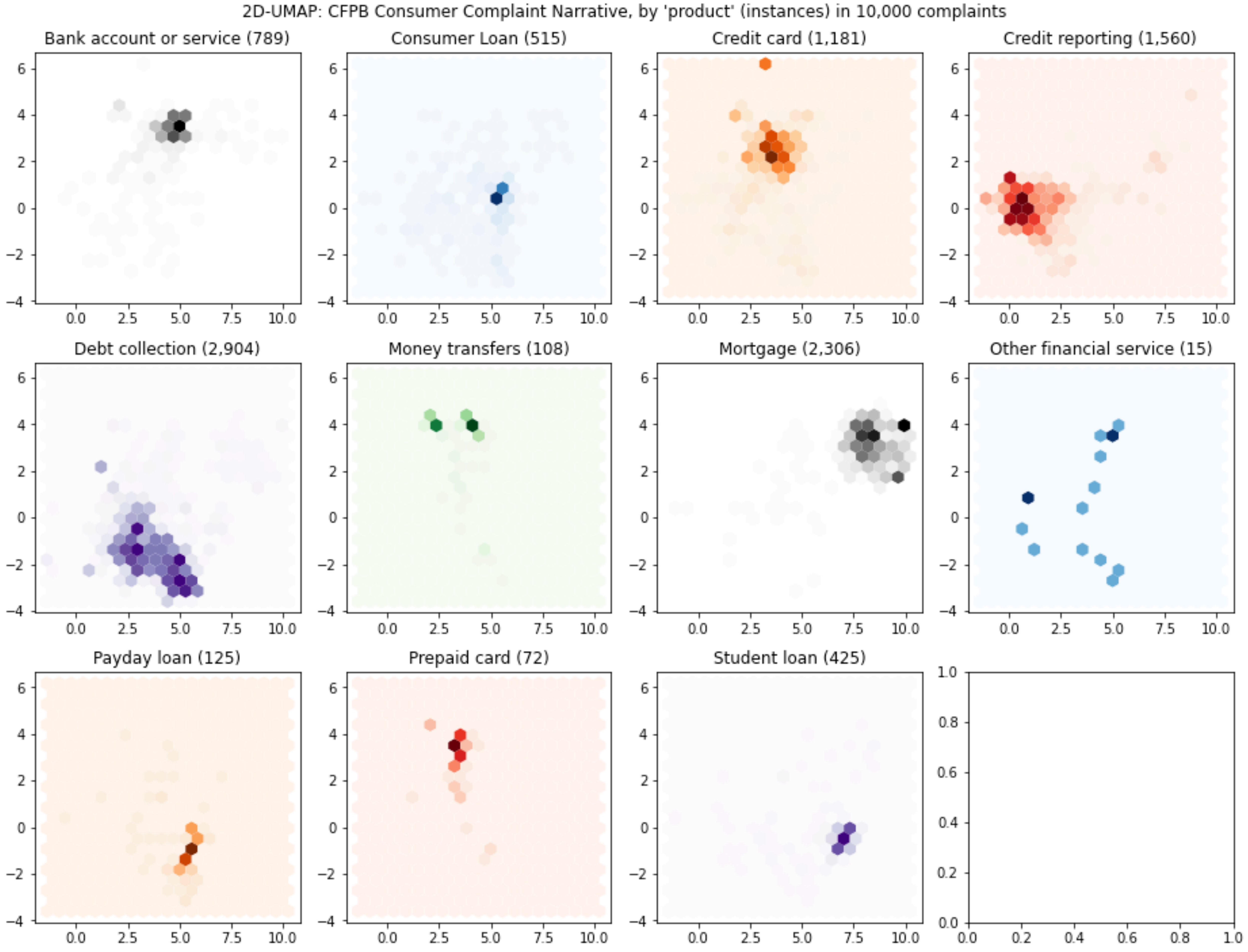
Data visualization

Understand your data clusters

- Space dimensions in ML-embeddings have no intrinsic meaning (unlike VSM)
- Dimensionality reduction maps dense high-dimensional spaces to 2D or 3D
- UMAP - Uniform Manifold Approximation and Projection for dimension reduction (alternative to t-SNE)
    - https://github.com/lmcinnes/umap
- Apply to 768-dim encodings to reduce to 2D for visualization

```python
from umap import UMAP
from sklearn.preprocessing import MinMaxScaler

X_normalized = sklearn.preprocessing.MinMaxScaler().fit_transform(X_viz)
umapper = UMAP(n_components=2, metric="cosine").fit(X_normalized)
X_viz_2d = pd.DataFrame(umapper.embedding_, columns=["X", "Y"])
X_viz_2d["Label"] = y_viz
```



Model risk and ethics

Models, data, use cases, evaluation

- Model risk
  - Vector search rankings vs. symbolic search
  - Semantics of vector spaces (clusters and labels)
  - Data risk: representation and bias, in training and evaluation
  - Interpretabiliy and explainability of machine learning models is critical
- Ethics for the use of AI/ML in "*high-impact tasks in areas such as law enforcement, medicine, education, and employment*."
- Model cards
  - Model details, Intended use, Factors, Metrics, Evaluation data, Training data, Analysis, Ethics, Caveats
  - https://arxiv.org/abs/1810.03993 (Mitchell et al., 2019)
- AI/ML governance and regulation (cf. GDPR)


European Commission

EN English

Search

Shaping Europe's digital future

Home Policies Activities News Library Funding Calendar Consultations

Home > Policies > A European approach to artificial intelligence

A European approach to artificial intelligence

The EU's approach to artificial intelligence centers on excellence and trust, aiming to boost research and industrial capacity while ensuring safety and fundamental rights.

The way we approach Artificial Intelligence (AI) will define the world we live in the future. To help building a resilient Europe for the Digital Decade, people and businesses should be able to enjoy the benefits of AI while feeling safe and protected.

The European AI Strategy aims at making the EU a world-class hub for AI and ensuring that AI is human-centric and trustworthy. Such an objective translates into the European approach to excellence and trust through concrete rules and actions.

© iStock by Getty Images - 1139760401 peshkov

https://digital-strategy.ec.europa.eu/en/policies/european-approach-artificial-intelligence

Conclusion

Conclusion

AI/ML enterprise semantic search

We presented enterprise semantic search on the CFPB consumer complaints database with recent results and PyData tools.

- Contrasted *high-dimensional term-based indexing* (traditional IR) to *dense vector document representations* for search. BM25 is a strong baseline.
- CFPB Semantic search with the HuggingFace transformers library
    - FinBERT & other transformer models (embedding)
    - FAISS fast indexing and search
- Model risk and ethics considerations, including use of model cards and AI/ML governance
- GitHub Jupiter notebook and slides https://nelscorrea.github.io/PyData_Miami_2022

ANDINUM

- Text Classification
- Document Automation
- Information Extraction
- Regulatory compliance

Contact: nelson@andinum.com

Twitter: @nelscorrea