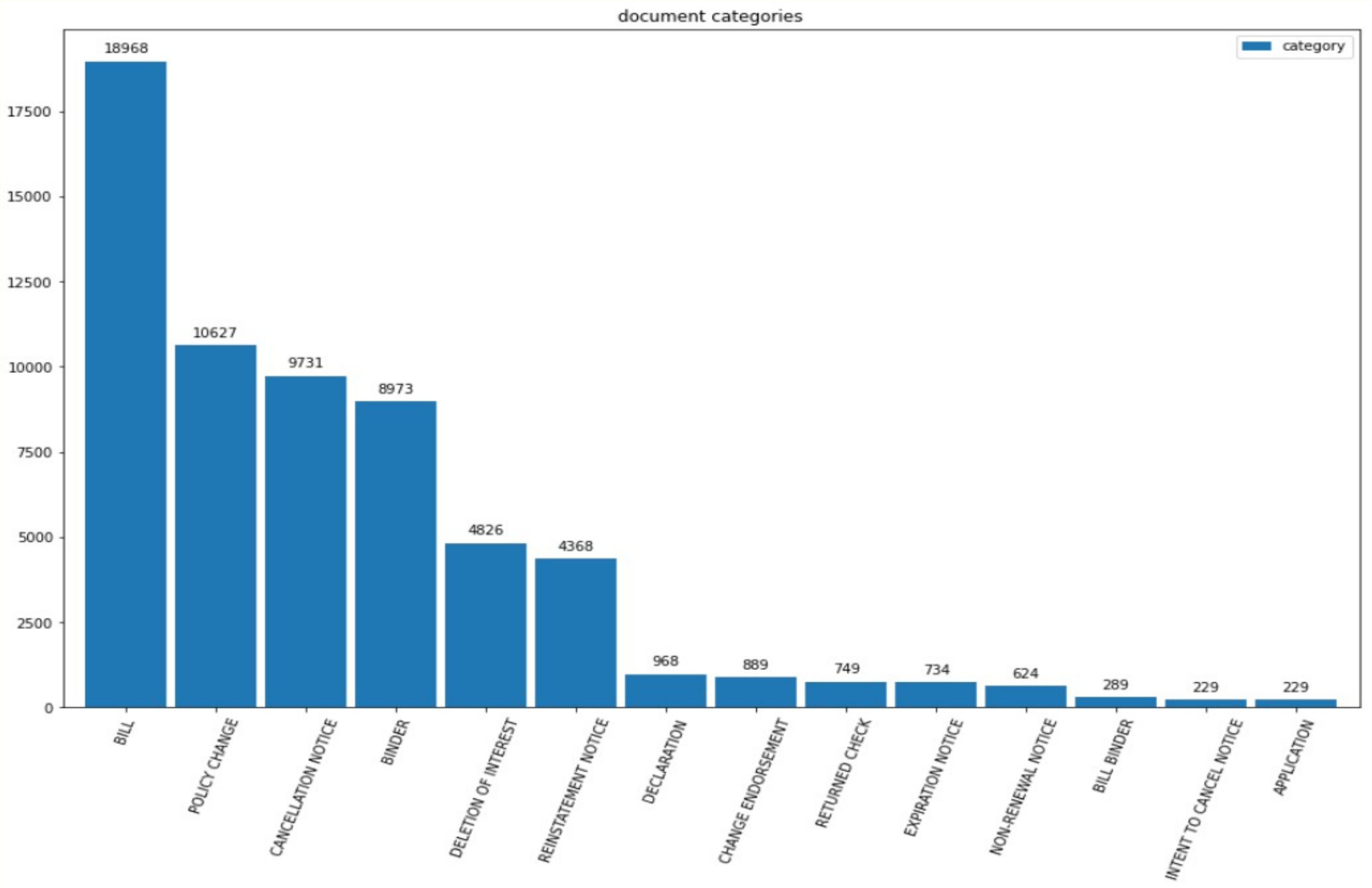
HeavyWater Machine Learning Problem

Solution by Mark Wilber

What we are dealing with

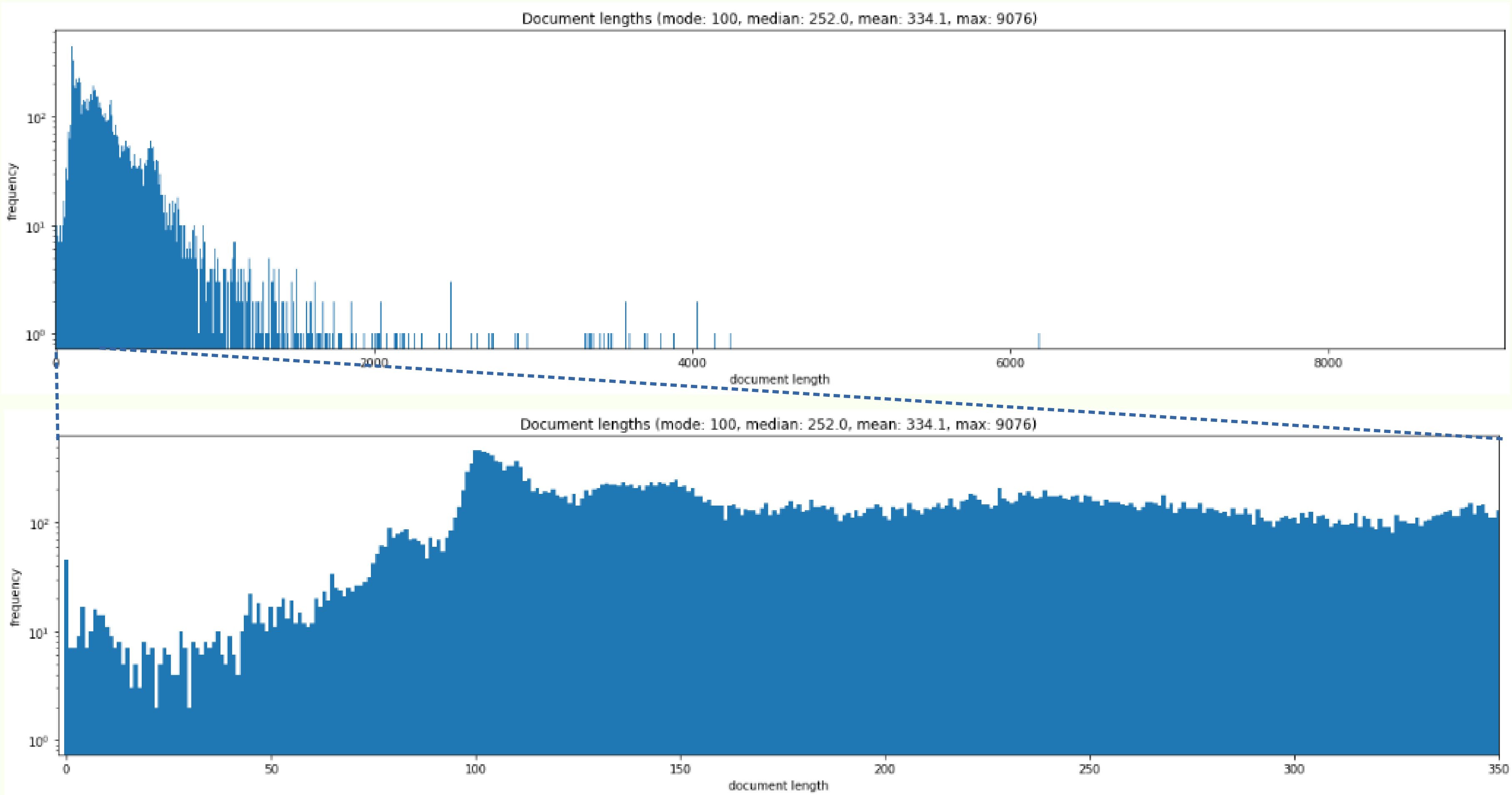
62,204 documents, 14 categories

Unbalanced classes, spanning nearly 2 orders of magnitude



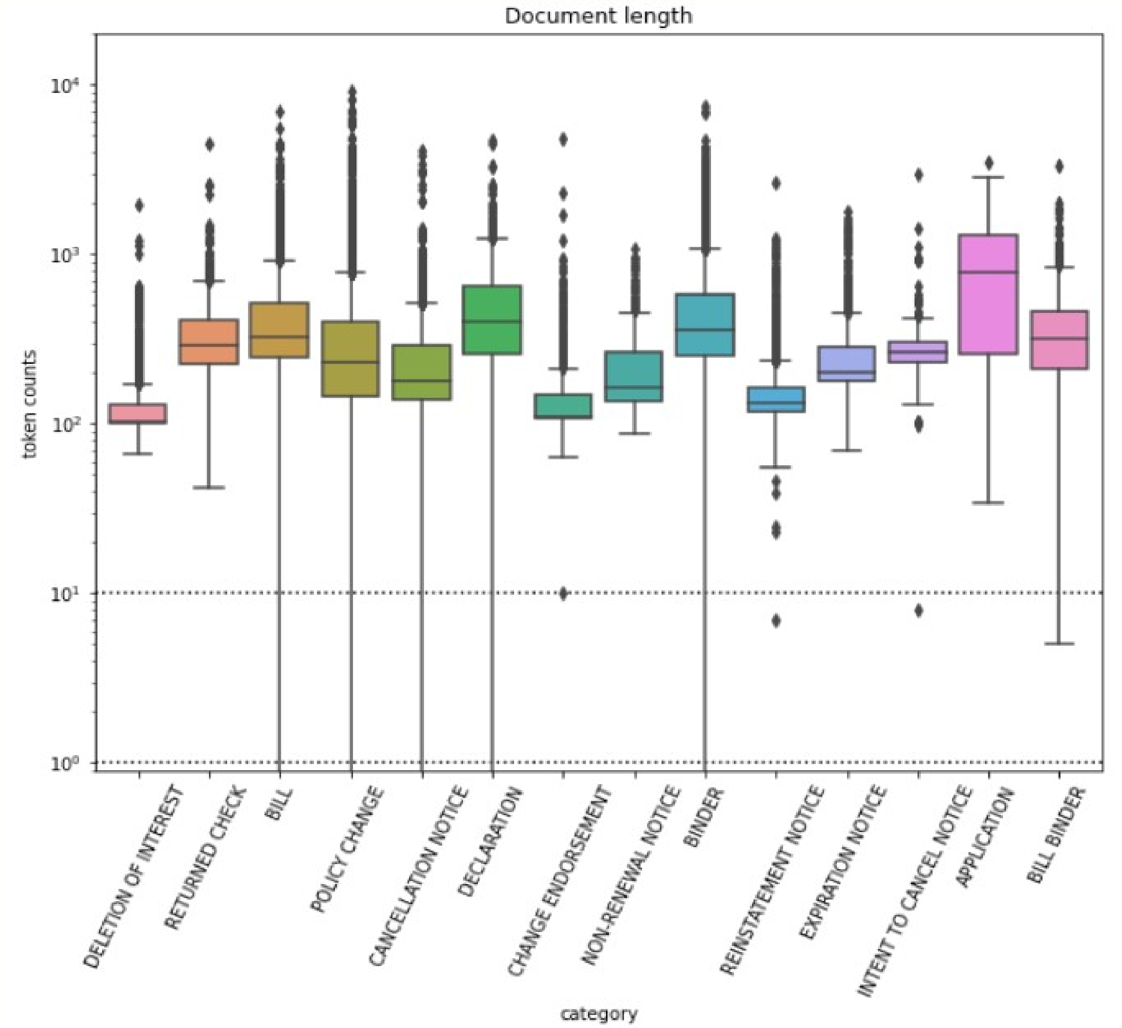
What we are dealing with

Document lengths spanning 0–9076 tokens (mode: 100, median: 252, mean: 334.1)



What we are dealing with

Document lengths vary widely by category, but few are shorter than 10 tokens



What we are dealing with

1,037,933 unique tokens, which contrasts with: Entire English language

Problem vocabulary exceeds that of OED:

> Oxford Dictionary has 273,000 headwords; 171,476 of them being in current use, 47,156 being obsolete words and around 9,500 derivative words included as subentries. The dictionary contains 157,000 combinations and derivatives in bold type, and 169,000 phrases and combinations in bold italic type, making a total of over 600,000 word-forms. There is one count that puts the English vocabulary at about 1 million words — but that count presumably includes words such as Latin species names, prefixed and suffixed words, scientific terminology, jargon, foreign words of extremely limited English use and technical acronyms.

It seems exceedingly unlikely that there is so much variation in the lexicon of mortgages and loans

tf	rank	# ≥ rank	frac ≥ rank
6	77189	960745	0.925632
5	88316	949618	0.914912
4	103088	934846	0.900680
3	128487	909447	0.876209
2	172658	865276	0.833652
1	300995	736939	0.710006

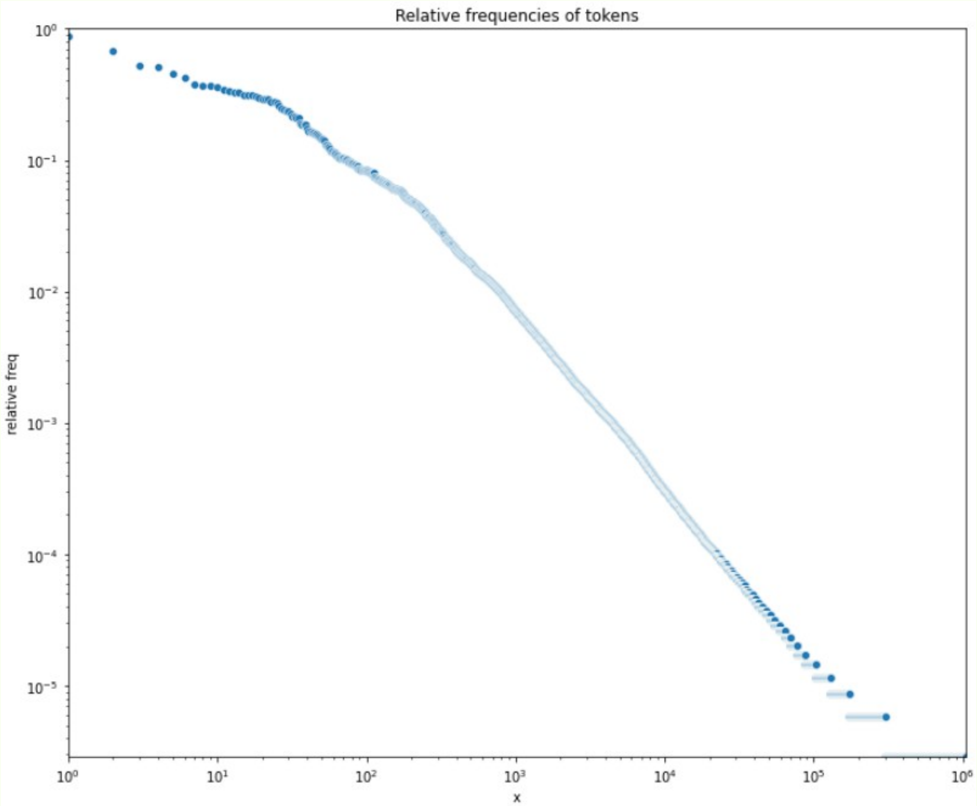
Explanation: most of the tokens are “uninformative” (garbage)

71% of tokens only appear once, 92.6% occur 6 × or fewer

- A small fraction are names (of humans, businesses), special codes
- *Speculation:* bogus terms due to scan / OCR noise ⇨ smudges create nonsense terms

What we are dealing with

Most frequent terms don't follow Zipf's relation



- First ~25 tokens frequency declines weakly vs Zipf
- After 750th ranked token, looks OK

Is this corpus actually this unusual?

Handling Data

Problem with stop words:

- Can’t use curated lists for stop words, as we only have word hashes
- Test with `sklearn.feature_extraction.text.TfidfVectorizer` shows: `max_df=0.80` eliminates 9 tokens, but it’s anybody’s guess what they are. *Probably* stop words ...
- Given time and justification, could use statistical bases for identifying stop words, e.g.:

  Gerlach, M., Shi, H. & Amaral, L.A.N. A universal information theoretic approach to the identification of stopwords. Nat Mach Intell 1, 606–612 (2019). https://doi.org/10.1038/s42256-019-0112-6

Handling Data

Trouble with small classes:

- With smallest class sizes $\mathcal{O}(200)$, even train-test split yields ~10% uncertainty in test stats
  ⇨ Can’t be sure cross-validation picks best model
- ⇨ Can’t trust relative scores between techniques

Many documents seem too short.

Test-train split

- 1st removed documents of length < 10 (still retaining very short examples)
- 50-50 test-train split to retain plausible stats on results, at some cost to performance …
- stratified sampling
- after model selection, can train on full data set (but won’t know how much better results may be)

Handling Data

tf-idf features

- `min_df=5:` ⇨ Eliminates most vocabulary
- `ngram_range=(1, 2):` ⇨ 283 k vocabulary
- `sublinear_tf=True`
- `max_df=0.8: ⇨ option (not taken) for ‘stop word’ removal`

Handling Data

tf-idf features

- `min_df=5`: ⇨ Eliminates most vocabulary
- `ngram_range=(1, 2)`: ⇨ 283 k vocabulary
- `sublinear_tf=True`
- ~~`max_df=0.8`~~: ⇨ `option (not taken) for ‘stop word’ removal`

Modeling

`f1_scorer`**: ⇨ when doing grid search, optimize for** f_1

- Used `average="weighted"`, but `average="macro"` would yield better results on small classes

Complement Naive Bayes

- Default settings for baseline
- Followed by grid search
- best with `alpha=0.0139 and norm=False` yielded substantial improvements
  ⇨ model *3* × *larger*

Model Results

Model	Macro Averaged			Weighted Average			Model size (MB)
	precision	recall	f_1	precision	recall	f_1	
Naive Bayes default (baseline)	0.75	0.50	0.53	0.79	0.78	0.76	63
Naive Bayes best	0.80	0.58	0.62	0.81	0.81	0.80	272
Random Forest default	0.80	0.65	0.70	0.84	0.85	0.84	451
Random Forest best	0.80	0.75	0.77	0.87	0.87	0.87	273
Gradient Boosting default	0.81	0.64	0.69	0.81	0.81	0.80	1.2
XGBoost default	0.79	0.65	0.70	0.82	0.82	0.82	5.4
XGBoost best	0.82	0.73	0.76	0.87	0.87	0.87	21
CNN							
Bidirectional LSTM							