Query Relevancy for Content

The challenge of a small data problem

SYSTEM1

Setting the stage

What we have:

- A consumer medical information website with ~3,500 articles

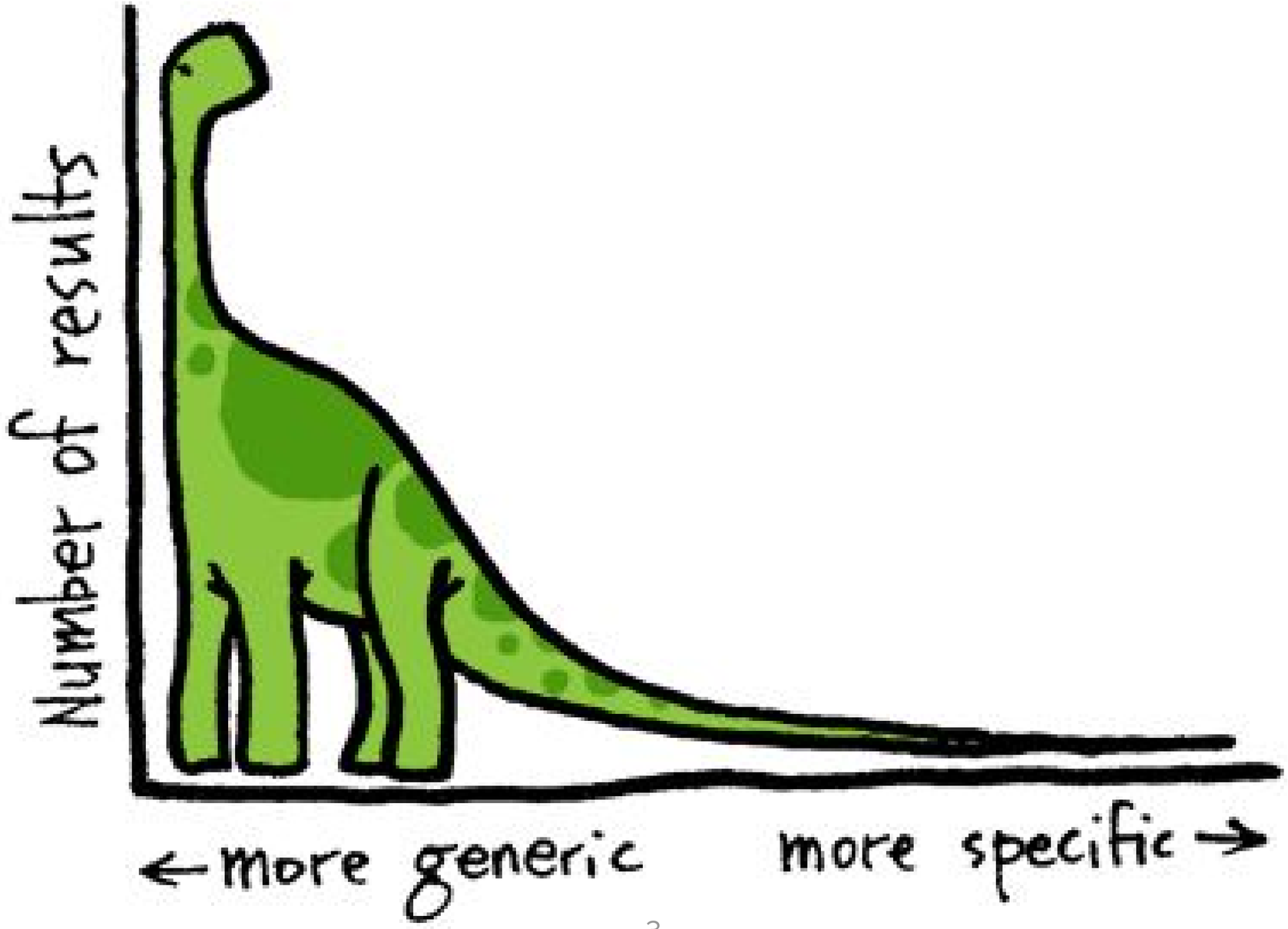
- A set of ~6,000 related keyword search phrases

What we want:

- Drive visitors to the website through paid search advertising, (SEM), expense $
- Monetize the website through search and display ads, revenue $
- revenue $ > expense $

SYSTEM1

In search, relevance reigns supreme



SYSTEM1

Step 1: Index the website content

Index the content using Elasticsearch. Easy, peasy right?

Elasticsearch gives you search but not relevance.

- how the content document fields are analyzed
- what query plans are used for the keyword phrases
- the breadth and depth of content (only ~3,500 articles)

Upshot: compare different ES analyzer/query combinations

Step 2: Score keyword results for relevancy

Which analyzer/query combination works best?

We can’t do it manually (3,500 articles * 6,000 keyword phrases)

We can relate keyword phrases to the content’s CMS taxonomy

We can run through all possible combinations and create a classification matrix (relevant/not-relevant)

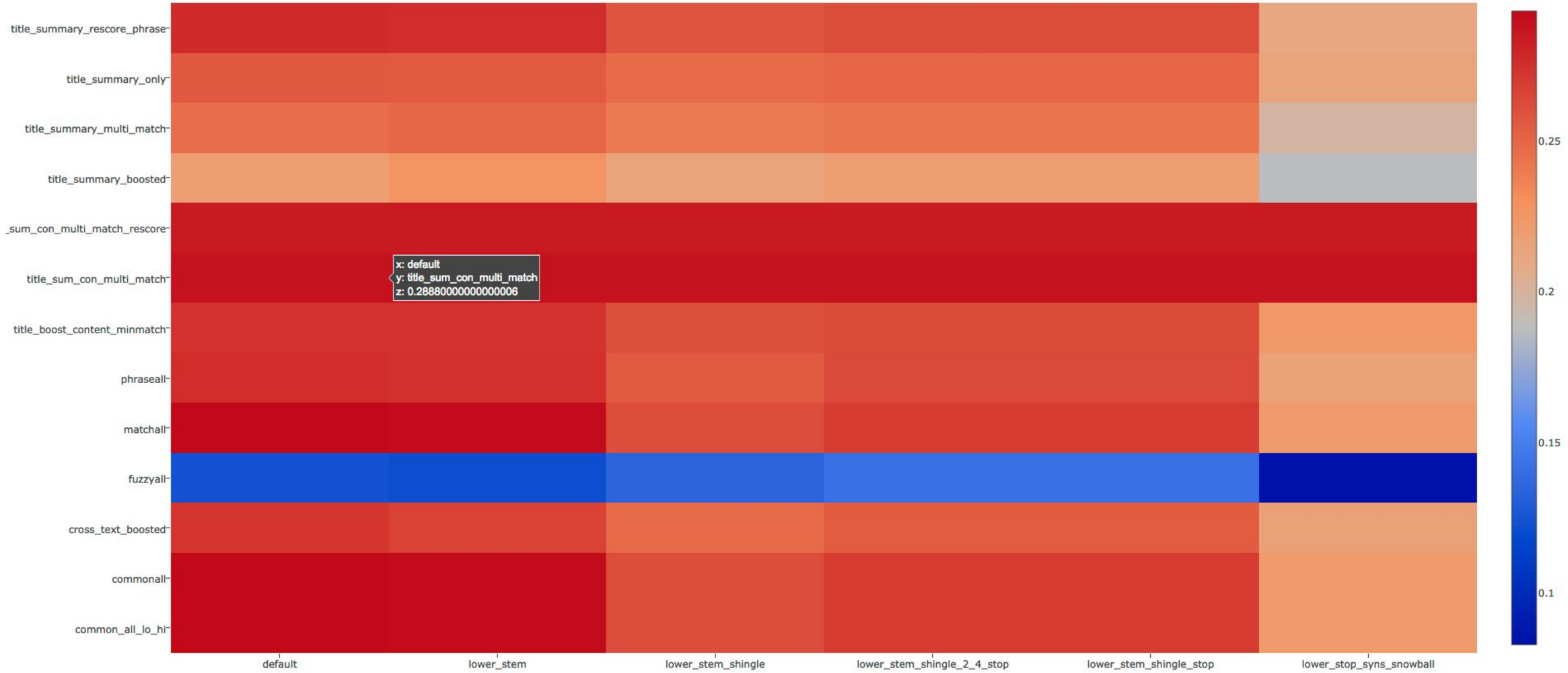
Use this as a training set and calculate precision/recall for every keyword phrase * analyzer * query plan

Relevancy Classification Matrix

	Cost of tooth implant	Latest hearing ad technology	Broken tooth	How to get rid of warts	...
The Use of Remote Controls With Hearing Aids		1			
What to do when your tooth cracks?	1		1		
Is it a skin tag or a wart?				1	
Tips for treating a dry cough					1
...			1		

SYSTEM1

Compare Effectiveness of Relevancy Metric



SYSTEM1

Quantifying and Applying Relevancy

Relevancy classifiers:

- Title lexical similarity (dice and jaccard)
- Content vectorization using the word2vec data model
- Cosine similarity clustering (e.g. dbscan)
- Mechanical Turk

Tools used:

- Pandas for processing the combinations as dataframes
- Elasticsearch-py to leverage elasticsearch's tokenizing ability
- SpaCy for evaluating cosine vector distance
- SciPy for clustering
- plotly for visualization

Did we solve the problem? Well... maybe.

- Depends on how close our metric corresponds to actual search relevancy
- Maybe we haven't thought of the best analyzer or query plan
- Flexible; we can swap out keyword phrase sets, query plans, index mappings, and relevancy classifications to generate new results very quickly
- Scalable; could adapt relatively easily to pyspark in the future

Future refinements include:

- softImpute (https://arxiv.org/pdf/1410.2596.pdf)
- Named Entity Recognition
- Sentiment Analysis
- Real-time feedback

Thanks!

system1.com/careers

SYSTEM1