

Content Engineering Tutorial

Sujit Pal, Elsevier Labs
September 25-27, 2018

Outline

- Background (Vector Space Model / Lucene)
- Basic Text Processing
- Keyword Extraction
- Content Search and Discovery Enhancement
- Dimensionality Reduction
- Ontology Construction
- Content based Recommendations

Pre-requisites

- Notebooks and web tool available at <https://github.com/sujitpal/content-engineering-tutorial>
- Instructions in data/README.md and models/README.md about locations of data and models to download.
- Parts of the tutorial depends on Solr 7.x, please download and install.
- Data and models (along with all processing) available at
 - https://drive.google.com/file/d/1uGbrGu5v9yaUKB_26oz_asKe2-SRGtsw/view?usp=sharing
 - https://drive.google.com/file/d/1xTB2Qx6roKYxUCWR_uTu4pwGp4iW-rY7/view?usp=sharing
- Code mostly written using Anaconda Python 3, see requirements.txt for full list of additional libraries needed.

Vector Space Model

- Collection of documents can be thought of as term-document matrix.
- Vector space – each token represents a dimension, weight represents the value along that dimension for the document.
- Term Frequency, Inverse Document Frequency
- Inverted Index for Search
- Dimensionality Reduction as a useful technique for extracting semantics of data.
- See Notebook - <https://github.com/sujitpal/content-engineering-tutorial/blob/master/notebooks/00-vector-space-model.ipynb>

Processing Data

- Preprocessing NIPS data for Solr
 - Solr Field Types
 - Solr Analyzer Chain
- See notebook - <https://github.com/sujitpal/content-engineering-tutorial/blob/master/notebooks/01-preprocess.ipynb>
- Baseline search implemented
 - <http://localhost:5000/search0>
 - Entire search string must appear in title or body.

Search Improvement – tokenize query

- Baseline search may be too restrictive.
 - Example: “neural network with attention mechanism” will return 0 results.
- Solution – tokenize search string and do OR query with each token.
- Example in tool - <http://localhost:5000/search1>
- High recall – too high?

Exploring Data

- Word Counts
- Figuring out what to keep and what to discard
- See notebook - <https://github.com/sujitpal/content-engineering-tutorial/blob/master/notebooks/02-wordcounts.ipynb>

Custom Stopwords

- Removing stopwords maybe a way to make the tokenized OR search less noisy?
- Leveraging IDF to detect potential stopwords for the corpus.
- See notebook - <https://github.com/sujitpal/content-engineering-tutorial/blob/master/notebooks/03-stopwords.ipynb>
- See it applied in tool - <http://localhost:5000/search2>

How about keywords?

- Detect keywords in text using Log Likelihood Ratio (LLR)
 - Frequent bigrams (can be extended to longer grams)
 - Likely collocations – Log Likelihood Ratio method
 - General family of statistical methods – others are Chi-squared, t-test, etc.
- Notebooks
 - Generating Frequent Bigrams - <https://github.com/sujitpal/content-engineering-tutorial/blob/master/notebooks/04-bigrams.ipynb>
 - Finding likely bigrams - <https://github.com/sujitpal/content-engineering-tutorial/blob/master/notebooks/05-bigrams-llr.ipynb>

Other keyword detection algorithms

- RAKE – rule based, unsupervised.
 - See notebook - <https://github.com/sujitpal/content-engineering-tutorial/blob/master/notebooks/06-rake.ipynb>
- MAUI – machine learning based
 - Provide text and keywords as labels for training – used combination of keywords from bigrams LLR and RAKE.
 - Apply trained MAUI model on same text to predict more keywords
 - See notebook - <https://github.com/sujitpal/content-engineering-tutorial/blob/master/notebooks/07-maui.ipynb>
- Merged keywords from LLR, RAKE and MAUI manually curated.

Removing near-duplicate keywords

- Using simhash algorithm to detect keywords that are very close to each other, e.g., data model and data models.
 - Uses hashing techniques (min-hash, sim-hash)
 - See notebook - <https://github.com/sujitpal/content-engineering-tutorial/blob/master/notebooks/08-keyword-neardup.ipynb>
- Using dedupe algorithm to detect keywords that are semantically equivalent, e.g., data set and dataset.
 - Uses Machine Learning
 - Training code (active learning) - https://github.com/sujitpal/content-engineering-tutorial/blob/master/scripts/dedupe_keyword_train.py
 - See notebook - <https://github.com/sujitpal/content-engineering-tutorial/blob/master/notebooks/09-keyword-dedupe.ipynb>

Incorporating keywords into search

- Detect keywords in search query using Aho-Corasick algorithm.
- Solr supports Aho-Corasick via SolrTextTagger.
- Expand query to incorporate keywords
 - See notebook - <https://github.com/sujitpal/content-engineering-tutorial/blob/master/notebooks/11-query-parsing-expansion.ipynb>
 - See it applied in tool - <http://localhost:5000/search4>

Extracting organizations

- NLTK + Stanford
 - NLTK with Stanford NER backend – very slow
 - Stanford NER command line
 - See notebook - <https://github.com/sujitpal/content-engineering-tutorial/blob/master/notebooks/12-org-ner-nltk-stanford.ipynb>
- Spacy NER
 - Quite fast and good quality
 - See notebook - <https://github.com/sujitpal/content-engineering-tutorial/blob/master/notebooks/13-org-ner-spacy.ipynb>
- Pruning predictions against dictionaries
 - See notebook - <https://github.com/sujitpal/content-engineering-tutorial/blob/master/notebooks/14-org-ner-ahocorasick.ipynb>

Incorporating Facets into Solr

- We extracted keywords and ORGs for each document
- We already have authors for each document
- We can enhance search by offering facet functionality
 - See in tool - <http://localhost:5000/search4>

Make Content more discoverable

- Similar item functionality
 - More Like This (MLT) – built into Solr
 - Similar keywords, authors and ORGs
 - Custom MLT using (keywords, authors and ORGs).
- Setting up new fields and MLT handler
 - New fields - https://github.com/sujitpal/content-engineering-tutorial/blob/master/scripts/create_schema_2.sh
 - Handler configuration - https://github.com/sujitpal/content-engineering-tutorial/blob/master/scripts/create_config_2.sh
- See notebook - <https://github.com/sujitpal/content-engineering-tutorial/blob/master/notebooks/15-load-solr.ipynb>
- See in tool - <http://localhost:5000/content1?id=6295>

Topic Modeling

- Uses gensim
- Preprocess text for cleaner topic models
- Dimensionality Reduction – projects original document into smaller and denser “taste” space.
- Find similar documents
 - See notebook - <https://github.com/sujitpal/content-engineering-tutorial/blob/master/notebooks/16-topic-modeling.ipynb>
 - See in tool - <http://localhost:5000/content1?id=6295>

Word and Document Embeddings

- Project document vectors into smaller embedding space
 - Lookup third party embeddings
 - Create your own embeddings (we created our own).
- Another dimensionality reduction technique, projects document into smaller, denser “meaning” space.
- Document vectors
 - Average of word vectors for word2vec (BoW model)
 - Model based for doc2Vec.
- See notebook - <https://github.com/sujitpal/content-engineering-tutorial/blob/master/notebooks/17-word-embeddings.ipynb>
- See in tool - <http://localhost:5000/content1?id=6295>

Building a keyword ontology

- Use keyword collocations across documents to build an ontology of keywords
- See notebook - <https://github.com/sujitpal/content-engineering-tutorial/blob/master/notebooks/18-build-ontology.ipynb>
- Applications
 - Query expansion
 - Ranking keyword similarity
 - Reverse approach yields navigable network of documents.
 - Explore options for Content Recommendations.

Content Recommendations

- Push Mechanism – recommend documents that a user might like to read, given the documents he has already read.
- Extension of Similar Documents, except this has multiple documents as input.
- Most techniques for computing similarity covered earlier (e.g., Topic Modeling, Word Vectors, etc) can be reused. Here we use NMF (Non-negative Matrix Factorization).
- See notebook - <https://github.com/sujitpal/content-engineering-tutorial/blob/master/notebooks/19-content-recommender.ipynb>

Thank you!

- Contact: sujit.pal@elsevier.com