Wi·**cc**·i·**fh**·y

Christian Caspers, Felix Hamann

Information Retrieval & Link Detection

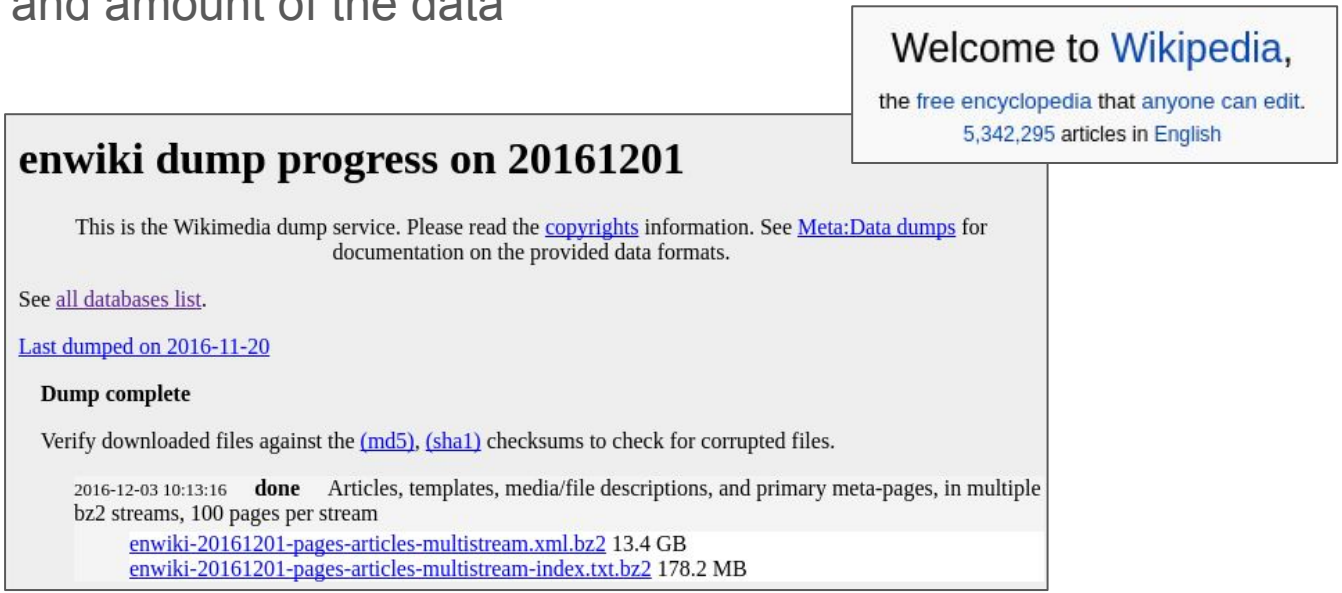
What do we need

A disambiguator and a link detector

- Training is based on a combination of the link structure and wikitext
- Samples are page titles of wikipedia articles
- For training we need a model of wikipedia’s link structure
- Ambiguities as defined by anchor text
- Counts for mentions of articles from other articles

Goals for information retrieval

- Necessary metrics for building training/test data should be stored in memory
- Random access to the text of all wikipedia articles
- Problem: Size and amount of the data



Welcome to Wikipedia,

the free encyclopedia that anyone can edit.
5,342,295 articles in English

enwiki dump progress on 20161201

This is the Wikimedia dump service. Please read the copyrights information. See Meta:Data dumps for documentation on the provided data formats.

See all databases list.

Last dumped on 2016-11-20

Dump complete

Verify downloaded files against the (md5), (sha1) checksums to check for corrupted files.

2016-12-03 10:13:16 **done** Articles, templates, media/file descriptions, and primary meta-pages, in multiple bz2 streams, 100 pages per stream

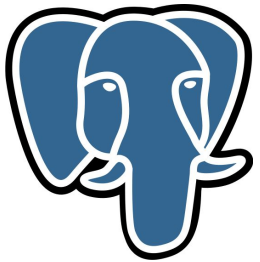
enwiki-20161201-pages-articles-multistream.xml.bz2 13.4 GB
enwiki-20161201-pages-articles-multistream-index.txt.bz2 178.2 MB

How to handle the data

- It is not possible to extract all required data at once
- Incremental improvement of our data set

Used databases

- Redis as in-memory data structure server for metrics
  - Really, really fast (access via loopback interface)
  - Proven python client: redis-py
  - Offers all data structures we need: sets, counter and key-value mapping
  - Transactional operations - important for testing and bug-fixing
  - Can become the bottleneck when accessing data with many clients
  - Concurrent access when spawning multiple processes
- Postgresql as database for wikitext
  - Random access via page title if armed with an index
  - Proven python client: psycopg2
  - In our case really easy to access (no complicated schema)



Stage I - Parsing Wikipedia

- Flat XML of 17092650 page elements
- 56G in size - cannot reside in memory
- No random access of elements
- Splitting the data is not easy
  - Pages are not uniformly distributed
  - No structural information easily obtainable

Our approach:

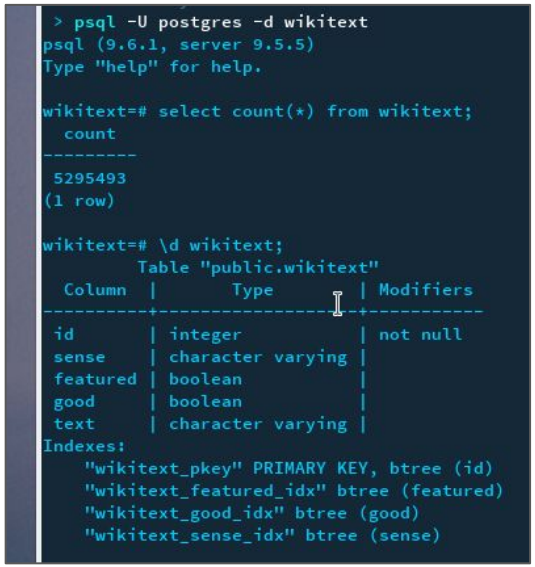
- Several processes crawl concurrently
- Each process parses its range

Stage I - Results

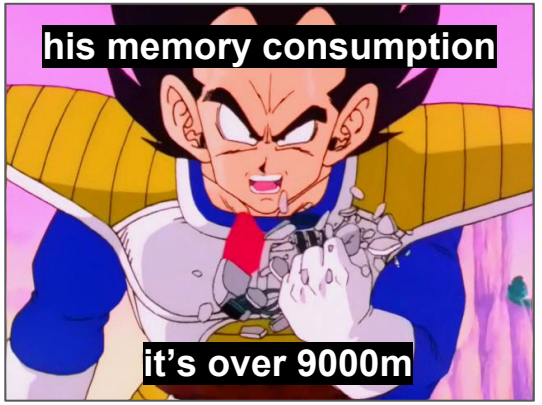
Postgres is populated and there’s no need to touch it anymore

Redis built the initial index structure

- Sets for senses, redirects and categories
- A queue of senses for worker pools
- Lookup table for redirects
- In total we imported
  - 5.289.723 senses
  - 7.273.736 redirects
  - 1.492.879 categories



Stage II - Extracting metrics



- We need backlink sets and mentioning counts
- All ambiguous terms must be extracted
- Should not take ages to do so: worker pool
- For keeping a somewhat sane memory footprint: 3-grams

Redis is suited well to incrementally improve data quality - dump.rdb

1. Extract links and ambiguous terms [[title|target]]
2. Merge redirects and ambiguities
3. Count mentions

Stage II - The extraction pipeline

Links and Ambiguities

150.397.424 Links!
4.725.190 Ambiguities

It's getting hot in here (memory wise)

Convert ambiguities and redirects

4.153.317 ambs removed
705.770 redirects to ambs
3.442.783 already senses

Prune to 3-grams

Could've done this earlier but now is the time.

Removed a total of 5G memory consumption

Pickle some indexes

Redis becomes a bottleneck for member queries

Distribute frozensets

Extract Mentions

1.260.364.692

Now we have consistent data for training etc.

pages	17092650
senses	4062010
redirects	5665593
categories	1492879
subcategories	584189
unregarded	2684286
links	77118361
ambiguations	265062
mentions	1218631673

Stage III - Required features

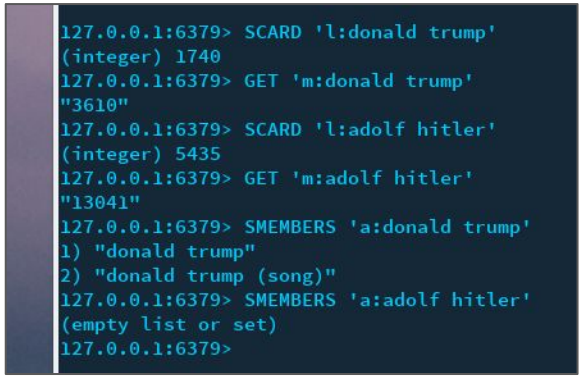
- Disambiguator
  - Commonness (cardinality of backlink sets per ambiguity)
  - Relatedness (normalized ratio of backlink sets)
  - Context Quality (sum of all averages between relatedness and link probability)
- Link Detector
  - Two link probabilities (average and maximum)
  - Two relatedness measures (per term and the average)
  - Confidence of the disambiguator (average and maximum)
  - Generality
  - Location and Spread

Stage III - Training the classifiers

- Unfortunately not much data to show - we stick close to the paper
- Pages are extracted randomly with a link count threshold (100)
- Training and test data is available for download (50, 100, 500) pages
  - ~14k samples per 100 pages
  - all necessary metrics are computed
- The disambiguator performs good: ~93% accuracy
  - sklearn’s c4.5 implementation
  - maximum tree height set to 5 performed best
  - changing minimum counts for leaves did not change much
  - confidence lies around 90%
- The link detector is implemented, but with a bug in the algorithm
  - so all data presented would be a lie

What went well

- Using redis as volatile data storage
  - incremental dumps
  - live inspection
- Extracting data with worker pools
- Calculating the metrics for the training data
- We have a consistent model of the whole wikipedia



What did not

- Redis becomes a bottleneck when doing many requests (even with pipelining)
- Implementing the data extraction took too long
- Lots of unnecessary data from mediawiki text (parsing is so slow)

General

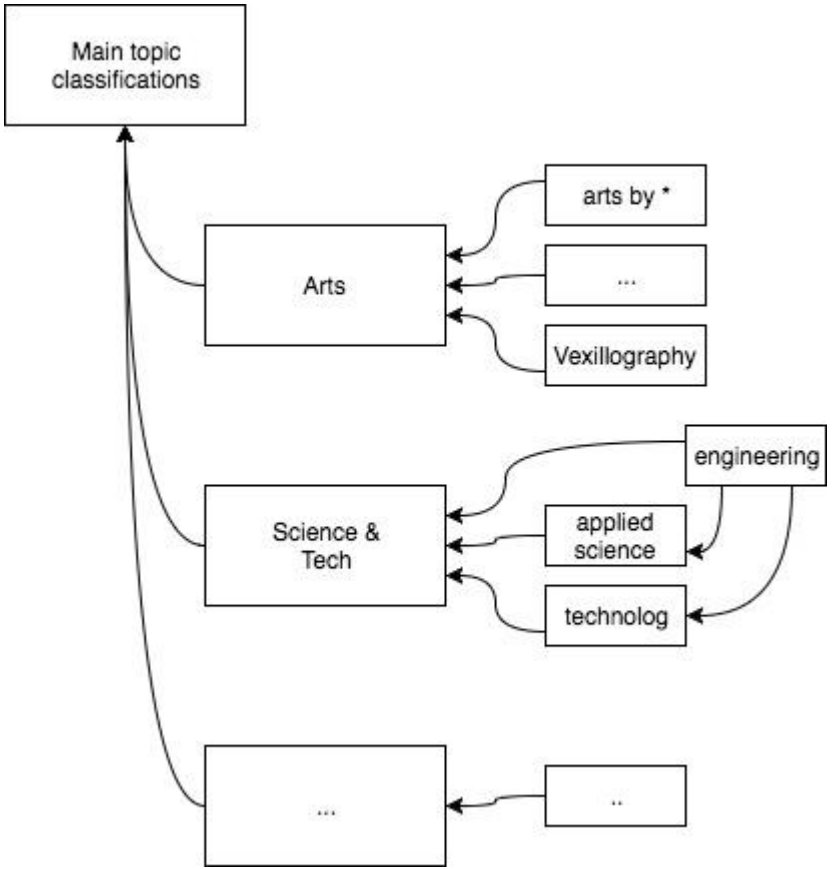
- We are case-insensitive - wikipedia is not
- Memory leaks kill, copy-on-write only really works for immutable types
- Premature optimization is the root of all evil

Text Categorization

Wikipedia Categories

- categories have
  - multiple parents
  - multiple children
- (probably) acyclic graph
- categories and articles reference parent categories

- **Main topic classifications**
  - kind-of root-category
  - 17 subcategories
  - only few directly associated articles



Building a Corpus for Classification I

- assign articles to categories
  - need to reverse relations
  - happens during data-import
- get enough articles
- no article exists in only one category
- depth creates overlap
  - the deeper the article within Wikipedia's structure, the more main topics are associated with it
- overlap creates noise
- multiclass, multilabel

corpus wanted

- 17 main categories
- 10,000 samples/category
- max 3 categories per article
- theoretically 170.000 samples
  - ~3% of all articles

Building a Corpus for Classification II

- iterate persisted graph
- for each main category
  - search breadth first for articles
  - memorize visited categories
  - memorize articles
  - loop over subcategories
- stop criteria
  - depth
  - number of articles per category
  - no subcategories left
- remove articles with > 3 categories
- create split 75:25
- remove overlap from training
- remove wiki markup

- **Results**
  - wanted 17 * 10.000 articles
  - received only 20.000 articles
  - Train: 14041
  - Test: 6442
- numbers vary slightly for each run due to redis' set operations

Corpus Issues I - Overlap and Balance

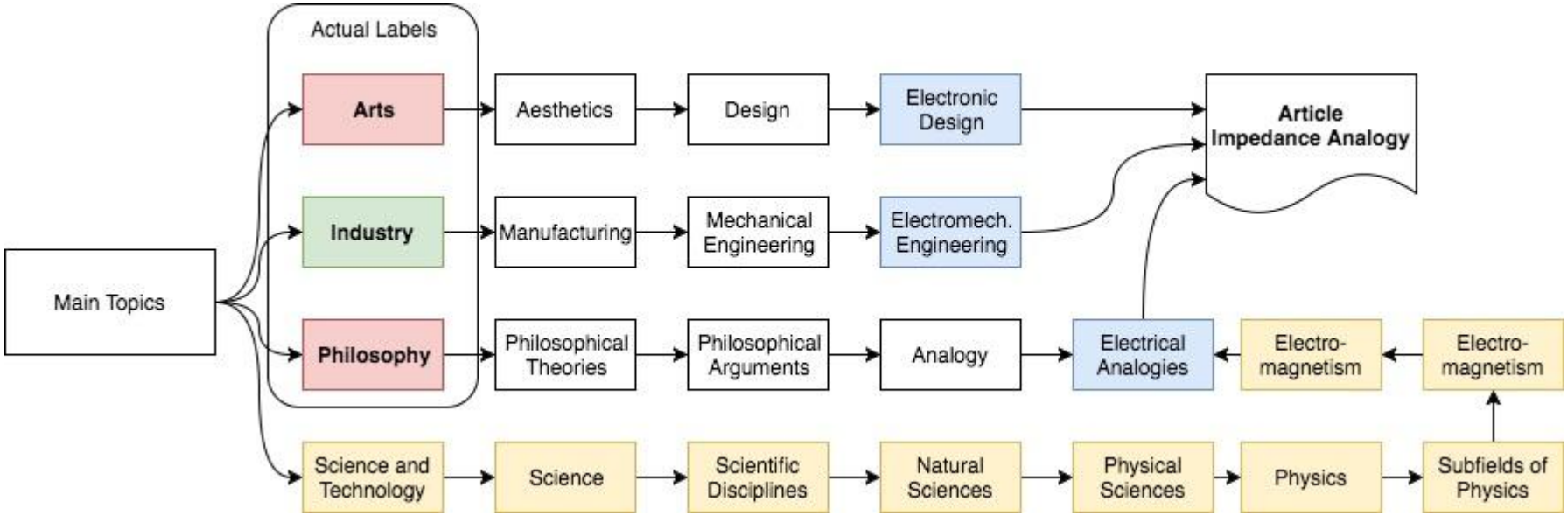
- search 10K articles, depth 10

```
Corpus
============================================================
TOTAL 170000
UNIQUE 27004
Name                    Count
--------------------  -------
arts                    10000
games                   10000
geography               10000
health                  10000
history                 10000
industry                10000
law                     10000
life                    10000
mathematics             10000
matter                  10000
nature                  10000
people                  10000
philosophy              10000
reference works         10000
religion                10000
science and technology  10000
society                 10000
```

- pruned corpus
- categories/article ≤ 3

```
Corpus
============================================================
TOTAL 17731
UNIQUE 8796
Name                    Count
--------------------  -------
arts                     1776
games                    2971
geography                1302
health                    697
history                  1051
industry                 1419
law                       697
life                      462
mathematics               899
matter                    703
nature                    814
people                   1872
philosophy                226
reference works          1240
religion                  856
science and technology    420
society                   326
```

Corpus Issues II - Unintuitive Labels



Creating a Classifier

- which one? problem is multiclass and multilabel
  - Logistic Regression
    - multiclass, supports probabilities
  - Stochastic Gradient Descent
    - customizable loss functions
    - “hinge”: SVM, multiclass, good results, no probabilities
    - “log”: like logistic regression, multiclass, probas
  - **OneVsRest with Logistic Regression**
    - multiclass and multilabel
    - fits one classifier per class
    - chosen because most appropriate, works good enough
      (probabilities from multiclass-only classifieres seemed plausible, though)

Optimization

- remove footer sections
- remove wiki markup
- no stemming or lemmatization
  - really slow (+60 minutes)
  - didn’t improve scores
- TfidfVectorizer instead
  - term-frequencies
  - GridSearch for tuning parameters
  - best result after 2 hours of fitting (54 fits)
    - max_df: 0.75
    - min_df: 100
    - ngram_range (1,2)
  - vocabulary of vectorizer ~ 45.000 terms

- pseudo-measure for quality of results

$$mean\left(\frac{|prediction \bigcap labels|}{|labels|}\right)$$

- example: if 2 of 3 labels are correctly predicted, classifier achieved 66% accuracy
- classifier achieves ~ 60% accuracy
- why?
  - default score function for multi-label is “subset accuracy which is a harsh metric”

Room for Improvement

- additional classifiers for subcategories
- rebalance corpus, analyze causes for imbalance
- add sample weights based on distance to main topics
- tuning of parameters for
  - arbitrary depth, maybe dynamic per category
  - article limits
  - overlap
- define metrics for quality of articles (ex. only good and featured)
  - this was an issue due to overlap, without limiting depth every article belonged to every category
- more thorough gridsearch for tuning classifier-parameters
- computing-resources are the limit

Demo

Q&A

Thank You