Document Processing

Search Engines

Master Degree in Computer Engineering
Master Degree in Data Science
Academic Year 2023/2024

Nicola Ferro

Intelligent Interactive Information Access (IIIA) Hub
Department of Information Engineering
University of Padua

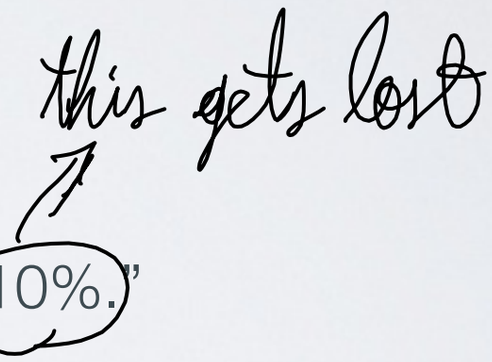
Outline

- Document Processing
- Hands-on Document Processing

Document Processing

Tokenization

- Surprisingly complex in English, can be harder in other languages
- Early IR systems:
  - any sequence of alphanumeric characters of length 3 or more
  - terminated by a space or other special character
  - upper-case changed to lower-case
- Example:
  - Bigcorp's 2007 bi-annual report showed profits rose 10%." becomes
    bigcorp 2007 annual report showed profits rose
- Too simple for search applications or even large-scale experiments
  - Too much information lost
  - Small decisions in tokenizing can have major impact on effectiveness of some queries



Tokenization Problems

- Small words can be important in some queries, usually in combinations
  - xp, pm, el paso, j lo, world war II
- Both hyphenated and non-hyphenated forms of many words are common
  - Sometimes hyphen is not needed
    e-bay, wal-mart, active-x, cd-rom, t-shirts
  - At other times, hyphens should be considered either as part of the word or a word separator
    mazda rx-7, e-cards, pre-diabetes, t-mobile, spanish-speaking
- Special characters are an important part of tags, URLs, code in documents
- Capitalized words can have different meaning from lower case words
  - Bush, Apple

Tokenization Problems

- Apostrophes can be a part of a word, a part of a possessive, or just a mistake
  - rosie o'donnell, can't, don't, 80's, 1890's, men's straw hats, master's degree, england's ten largest cities
- Numbers can be important, including decimals
  - nokia 3250, top 10 courses, united 93, quicktime 6.5 pro, 92.3 the beat, 288358
- Periods can occur in numbers, abbreviations, URLs, ends of sentences, and other situations
  - I.B.M., Ph.D., dei.unipd.it

Stop Lists

- Stopword list can be created from high-frequency words or based on a standard list
- Lists are customized for applications, domains, and even parts of documents
- Best policy is to index all words in documents, make decisions about which words to use at query time
- Some lists (`https://github.com/igorbrigadir/stopwords`):
  - Atire terms: 988
  - Terrier terms: 733
  - Smart terms: 571
  - Zettair terms: 469
  - Indri terms: 418
  - Glasgow terms: 319
  - Okapi terms: 222
  - Snowball terms: 174
  - Lucene terms: 33

Stemming

- Three basic types
  - **Dictionary-based**: uses lists of related words
  - **Algorithmic**: uses program to determine related words
  - **Statistical**: uses document corpora statistics and training data to learn how to stem
- The simplest algorithmic stemmers
  - suffix-s: remove ‘s’ endings assuming plural
    - e.g., cats → cat, lakes → lake
  - Many false negatives: supplies → supplie
  - Some false positives: ups → up

Harman, D. (1987). A Failure Analysis on the Limitations of Suffixing in an Online Environment. In Yu, C. T. and van Rijsbergen, C. J., editors, *Proc. 10th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 1987)*, pages 102–107. ACM Press, New York, USA.

Porter Stemmer

- Algorithmic stemmer which produces **stems** not words
  - possibly the most used stemmer in IR
- Consists of a series of rules designed to replace the longest possible suffix at each step
- Makes a number of errors and it is difficult to modify

Step 1a:

- Replace *sses* by *ss* (e.g., stresses → stress).
- Delete *s* if the preceding word part contains a vowel not immediately before the *s* (e.g., gaps → gap but gas → gas).
- Replace *ied* or *ies* by *i* if preceded by more than one letter, otherwise by *ie* (e.g., ties → tie, cries → cri).
- If suffix is *us* or *ss* do nothing (e.g., stress → stress).

Step 1b:

- Replace *eed*, *eedly* by *ee* if it is in the part of the word after the first non-vowel following a vowel (e.g., agreed → agree, feed → feed).
- Delete *ed*, *edly*, *ing*, *ingly* if the preceding word part contains a vowel, and then if the word ends in *at*, *bl*, or *iz* add *e* (e.g., fished → fish, pirating → pirate), or if the word ends with a double letter that is not *ll*, *ss*, or *zz*, remove the last letter (e.g., falling→ fall, dripping → drip), or if the word is short, add *e* (e.g., hoping → hope).

Martin Porter

Porter, M. F. (1980). An algorithm for suffix stripping. *Program*, 14(3):130– 137.

Porter Stemmer

- Porter2 stemmer addresses some of these issues
- Approach has been used with other languages
  - `https://snowballstem.org/`

False positives	*False negatives*
organization/organ	european/europe
generalization/generic	cylinder/cylindrical
numerical/numerous	matrices/matrix
policy/police	urgency/urgent
university/universe	create/creation
addition/additive	analysis/analyses
negligible/negligent	useful/usefully
execute/executive	noise/noisy
past/paste	decompose/decomposition
ignore/ignorant	sparse/sparsity
special/specialized	resolve/resolution
head/heading	triangle/triangular

Martin Porter

Porter, M. F. (1980). An algorithm for suffix stripping. *Program*, 14(3):130– 137.

Lovins Stemmer

- The **first** ever published stemming algorithm
- Algorithmic stemmer which produces **words**
- The design of the algorithm was much influenced by the technical vocabulary with which Lovins found herself working (subject term keywords attached to documents in the materials science and engineering field)
- The stemmer removes the longest suffix from a word then the word is recoded to convert this stem into valid words. This process can be split into two phases.
  - In the first phase, a word is compared with a pre-determined list of endings, and when a word is found to contain one of these endings, the ending is removed, leaving only the stem of the word.
  - The second phase standardizes spelling exceptions that come out of the first phase
- The Lovins algorithm is noticeably **bigger** than the Porter algorithm, because of its very extensive endings list but it is **faster**

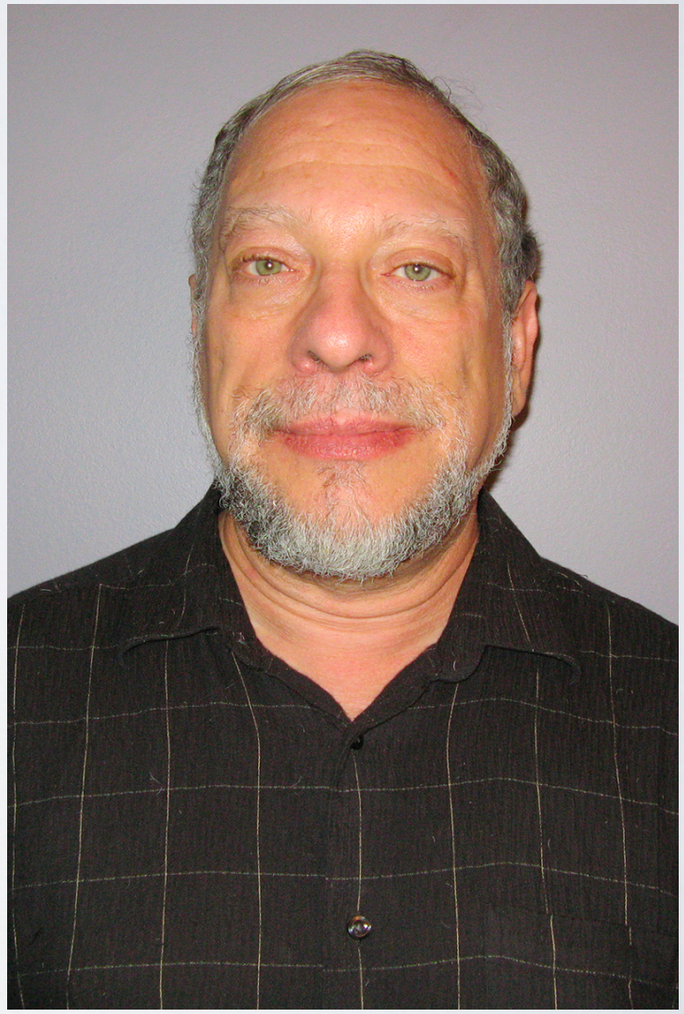
molto basato su regole

Julie Beth Lovins

Lovins, J. B. (1968). Development of a Stemming Algorithm. *Mechanical Translation and Computational Linguistics*, 11(1/2):22–31.

Krovetz Stemmer

- Hybrid algorithmic-dictionary
  - Word checked in dictionary
- If present, either left alone or replaced with "exception"
- If not present, word is checked for suffixes that could be removed
- After removal, dictionary is checked again
- Produces **words** not stems
- Lower false positive rate than Porter, somewhat higher false negative

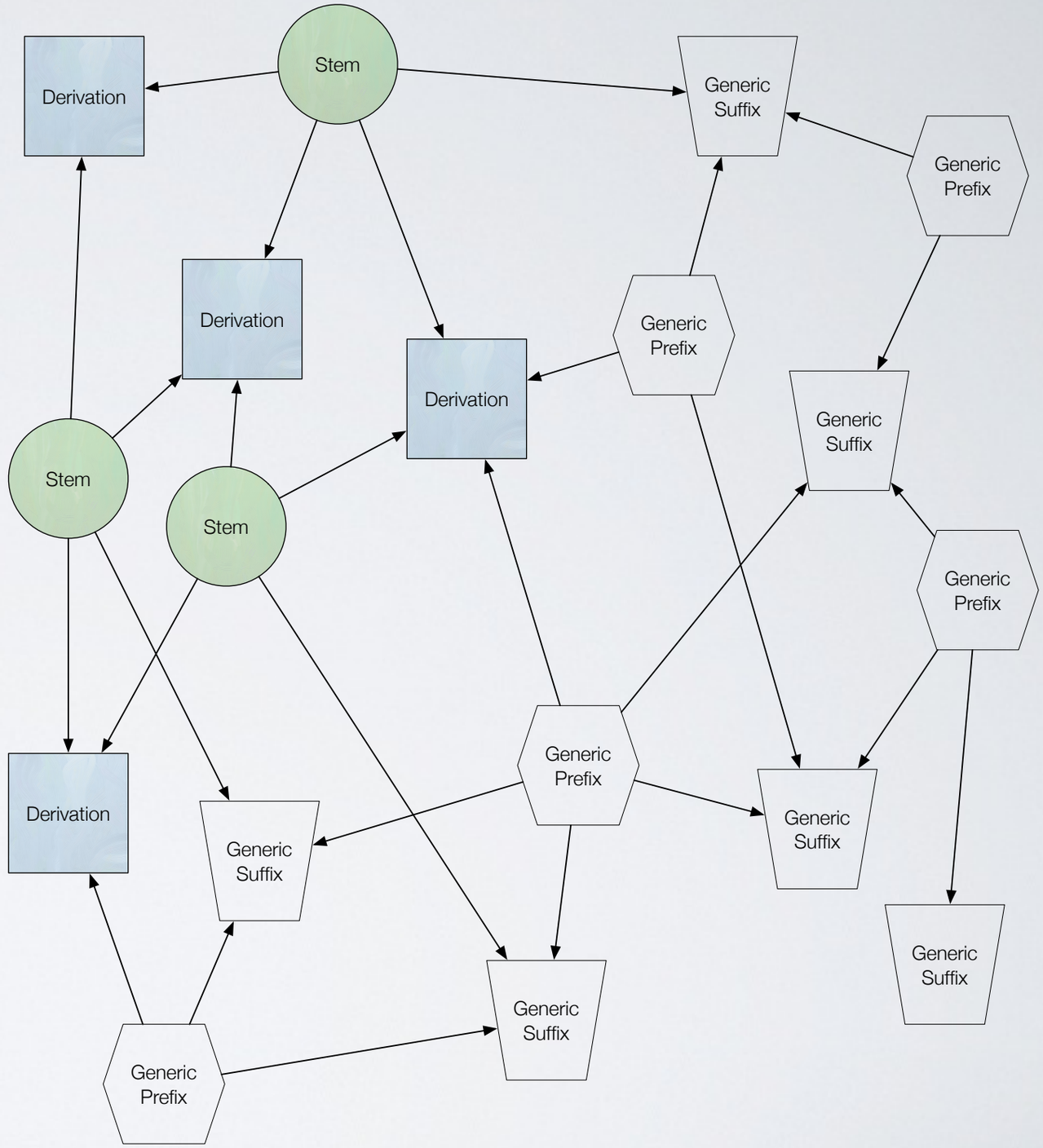


Robert Krovetz

Krovetz, R. (1993). Viewing Morphology as an Inference Process. In Korfhage, R., Rasmussen, E., and Willett, P., editors, *Proc. 16th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 1993)*, pages 191–202. ACM Press, New York, USA.

SPLIT Stemmer

- A language-independent unsupervised statistical stemmer
- We split each word into all the possible two parts: the prefix is the first part of the word, and the suffix is the second part
  - Each prefix or suffix is a node of a graph where each link corresponds to the word obtained by concatenating them
- Mutual reinforcement between stems and derivations => nodi di grafo bipartito
  - stems are prefixes which are completed at a high frequency rate by the derivations
  - derivations, in turn, are suffixes which complete, at a high frequency rate, the stems
  - in graphical terms, derivations tend to be linked to stems, and viceversa, thus forming communities in the graph
  - PageRank-like algorithm for computing the probability of being a stem and a derivation



Bacchin, M., Ferro, N., and Melucci, M. (2005). A Probabilistic Model for Stemmer Generation. *Information Processing & Management*, 41(1):121– 137.

Phrases

- Many queries are 2-3 word phrases
- Phrases are
  - More precise than single words
    e.g., documents containing “black sea” vs. two words “black” and “sea”
  - Less ambiguous
    e.g., “big apple” vs. “apple”
- Can be difficult for ranking
  - e.g., Given query “fishing supplies”, how do we score documents with
  - exact phrase many times, exact phrase just once, individual words in same sentence, same paragraph, whole document, variations on words?
- Text processing issue – how are phrases recognized?
- Three possible approaches:
  - Identify syntactic phrases using a **part-of-speech (POS)** tagger
  - Use **word n-grams**
  - Store word positions in indexes and use **proximity operators** in queries

Part-of-Speech (POS) Tagging

- POS taggers use statistical models of text to predict syntactic tags of words
- Example tags:
  - NN (singular noun), NNS (plural noun), VB (verb), VBD (verb, past tense), VBN (verb, past participle), IN (preposition), JJ (adjective), CC (conjunction, e.g., “and”, “or”), PRP (pronoun), and MD (modal auxiliary, e.g., “can”, “will”).
- Phrases can then be defined as simple noun groups, for example

Original text:
Document will describe marketing strategies carried out by U.S. companies for their agricultural chemicals, report predictions for market share of such chemicals, or report market statistics for agrochemicals, pesticide, herbicide, fungicide, insecticide, fertilizer, predicted sales, market share, stimulate demand, price cut, volume of sales.

Brill tagger:
Document/NN will/MD describe/VB marketing/NN strategies/NNS carried/VBD out/IN by/IN U.S./NNP companies/NNS for/IN their/PRP agricultural/JJ chemicals/NNS ,/, report/NN predictions/NNS for/IN market/NN share/NN of/IN such/JJ chemicals/NNS ,/, or/CC report/NN market/NN statistics/NNS for/IN agrochemicals/NNS ,/, pesticide/NN ,/, herbicide/NN ,/, fungicide/NN ,/, insecticide/NN ,/, fertilizer/NN ,/, predicted/VBN sales/NNS ,/, market/NN share/NN ,/, stimulate/VB demand/NN ,/, price/NN cut/NN ,/, volume/NN of/IN sales/NNS ./.

Word (Character) N-grams

- POS tagging too slow for large collections
- Simpler definition – phrase is any sequence of n **words** – known as **N-grams**
  - bigram: 2 word sequence, trigram: 3 word sequence, unigram: single words
  - **N-grams** also used at **character** level for applications such as OCR or as an alternative to stemming (poorer performance)
- N-grams typically formed from **overlapping sequences** of words (characters)
  - i.e. move n-word "window" one word at a time in document
- Frequent N-grams are more likely to be meaningful phrases
- Could index all N-grams up to specified length
  - Much faster than POS tagging
  - Uses a lot of storage

Google N-grams back in 2006

https://ai.googleblog.com/2006/08/all-our-n-gram-are-belong-to-you.html

- Web search engines index word N-grams
- Google sample from 2006 (24 Gbyte compressed)
  - Number of tokens: 1,024,908,267,229
  - Number of sentences: 95,119,665,584
  - Number of unigrams: 13,588,391 that appear at least than 200 times
  - Number of bigrams: 314,843,401
  - Number of trigrams: 977,069,902
  - Number of fourgrams: 1,313,818,354
  - Number of fivegrams: 1,176,470,663 that appear at least 40 times
- Most frequent trigram in 2007
  - English: “all rights reserved”
  - Chinese: “limited liability corporation”

Yang, S., Zhu, H., Apostoli, A., and Cao, P. (2007). N-gram Statistics in English and Chinese: Similarities and Differences. In Sheu, P. C. Y., Croll, P., Muhlhauser, M., Yamaguchi, Y., Naphade, M., Yu, H., and Koiti, H., editors, *Proc. International Conference on Semantic Computing (ICSC 2007)*, pages 454–460. IEEE Computer Society, Los Alamitos, CA, USA.

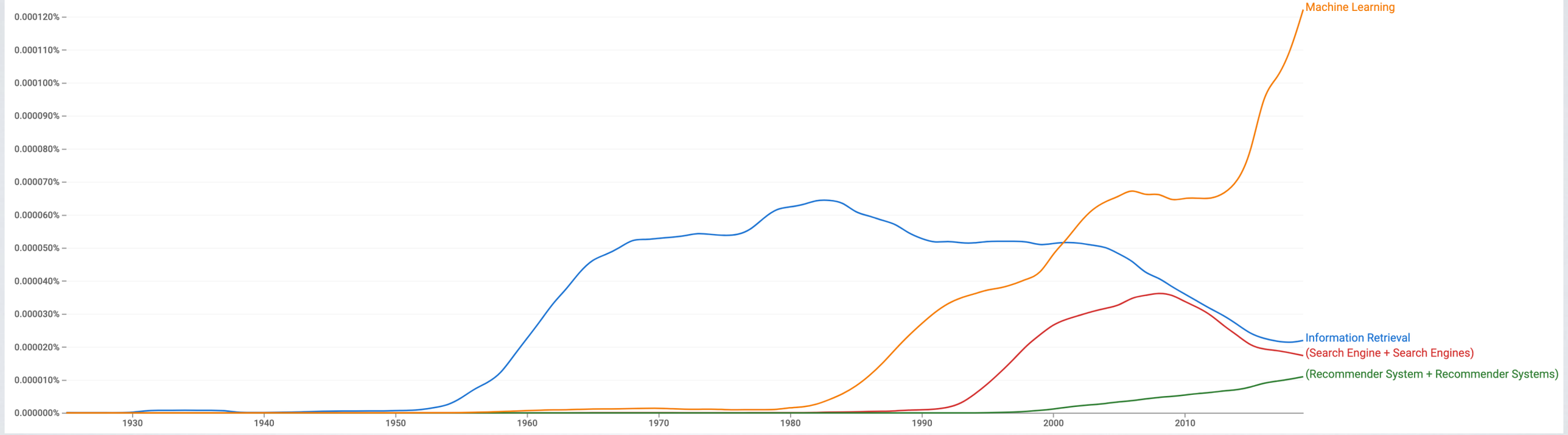
Google Books N-grams

https://books.google.com/ngrams

- When you enter phrases into the Google Books Ngram Viewer, it displays a graph showing how those phrases have occurred in a corpus of books (e.g., "British English", "English Fiction", "French") over the selected years.
- Corpora (roughly 1500-2019 CE)
  - All corpora were generated in July 2009, July 2012, and February 2020 from Book OCR (may suffer for poor OCR quality)
  - English, Chinese, French, German, Hebrew, Spanish, Russian, Italian
- Advanced use
  - **Wildcard search**: when you put a `*` in place of a word, the Ngram Viewer will display the top ten substitutions
  - **Inflection search**: an inflection is the modification of a word to represent various grammatical categories such as aspect, case, gender, mood, number, person, tense and voice
  - **Part-of-speech Tags**: you can use POS tags (either alone or appended to N-grams)
  - **Ngram Compositions**: The Ngram Viewer provides five operators that you can use to combine ngrams: `+`, `-`, `/`, `*`, and `:`

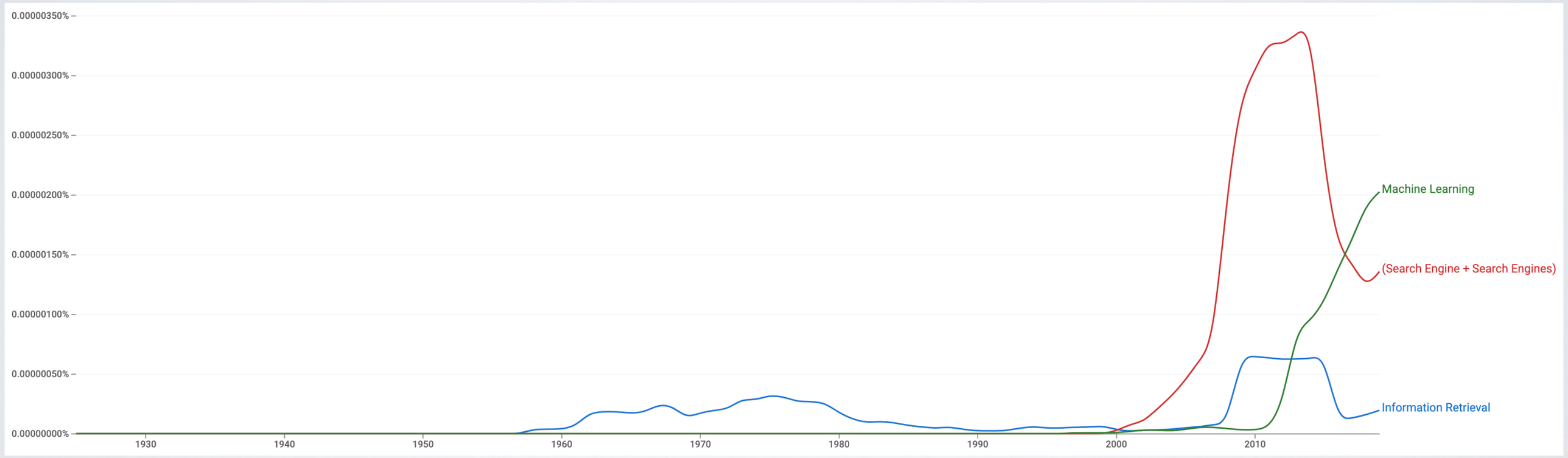
Google Books N-grams: Our Field

- Information Retrieval, (Search Engine+Search Engines),(Recommender System + Recommender Systems), Machine Learning
  - 1925-2019
  - English



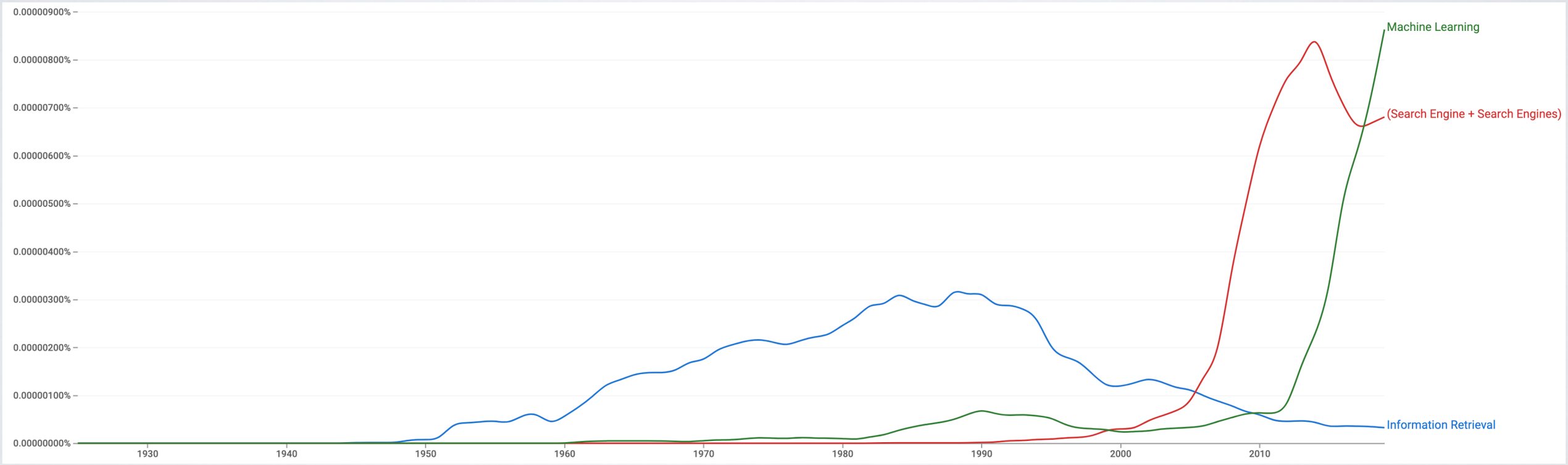
Google Books N-grams: Our Field

- Information Retrieval, (Search Engine+Search Engines),(Recommender System + Recommender Systems), Machine Learning
  - 1925-2019
  - Russian



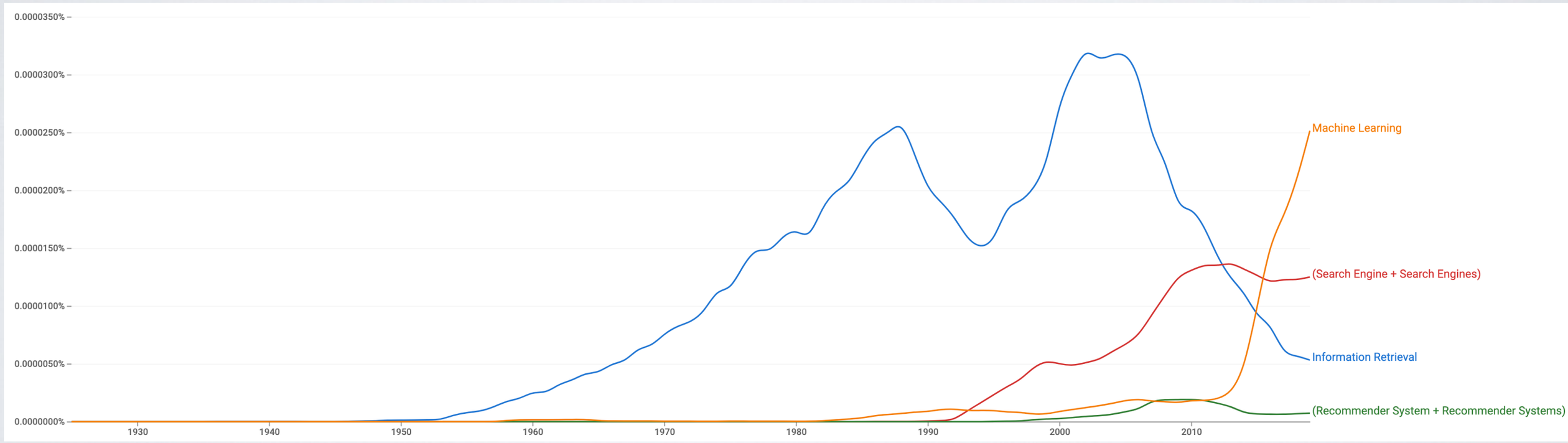
Google Books N-grams: Our Field

- Information Retrieval, (Search Engine+Search Engines),(Recommender System + Recommender Systems), Machine Learning
  - 1925-2019
  - French



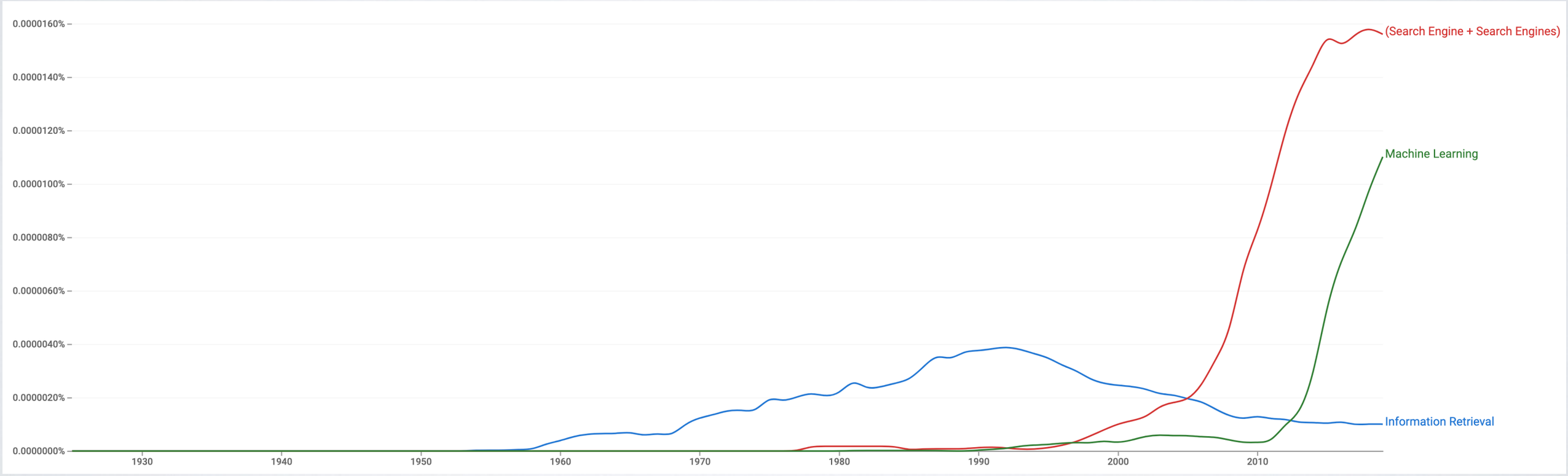
Google Books N-grams: Our Field

- Information Retrieval, (Search Engine+Search Engines),(Recommender System + Recommender Systems), Machine Learning
  - 1925-2019
  - German



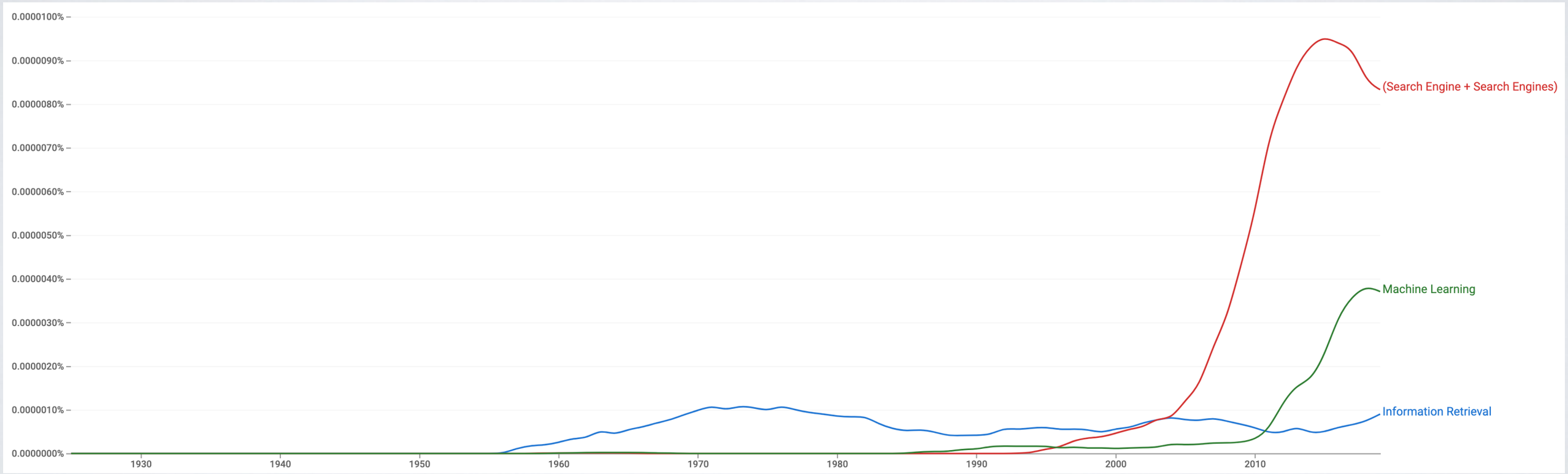
Google Books N-grams: Our Field

- Information Retrieval, (Search Engine+Search Engines),(Recommender System + Recommender Systems), Machine Learning
  - 1925-2019
  - Italian



Google Books N-grams: Our Field

- Information Retrieval, (Search Engine+Search Engines),(Recommender System + Recommender Systems), Machine Learning
  - 1925-2019
  - Spanish



Named Entity Recognition (NER)

- Named Entity Recognition
  - identify words that refer to something of interest in a particular application

    e.g., people, companies, locations, dates, product names, prices, etc.
  - It is a branch of Information Extraction, which aims at automatically extracting structure from text, e.g. entities, relationships, events, …

    Fred Smith, who lives at 10 Water Street, Springfield, MA, is a long-time collector of **tropical fish.**

    identificare categoria

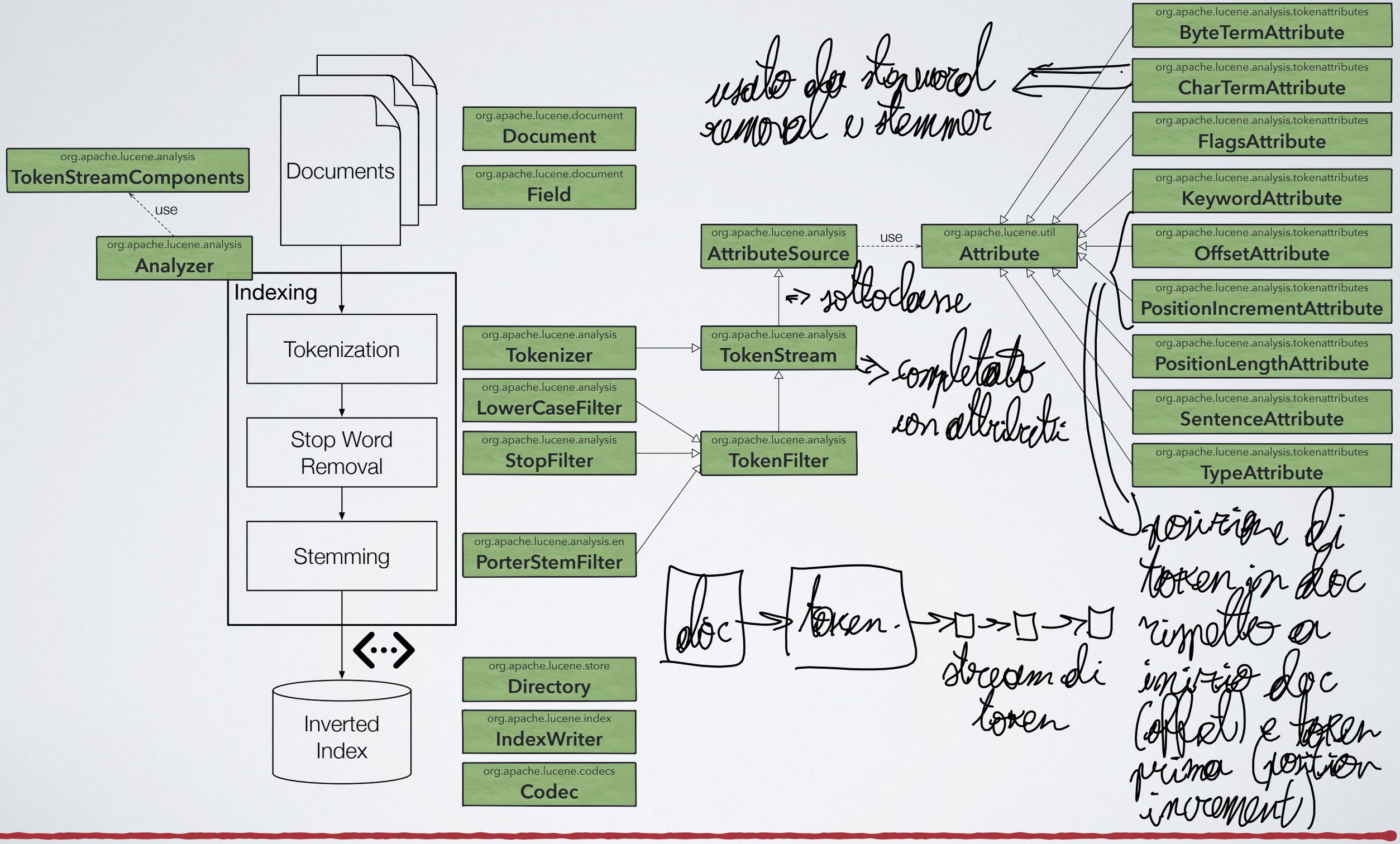
    <PersonName><GivenName>Fred</GivenName> <Sn>Smith</Sn>
    </PersonName>, who lives at <address><Street >10 Water Street</Street>,
    <City>Springfield</City>, <State>MA</State></address>, is a long-time
    collector of tropical fish.

Named Entity Recognition (NER)

- Rule-based
  - Uses lexicons (lists of words and phrases) that categorize names
    e.g., locations, peoples' names, organizations, etc.
  - Rules also used to verify or find new entity names
    e.g., "<number> <word> street" for addresses
    "<street address>, <city>" or "in <city>" to verify city names
    "<street address>, <city>, <state>" to find new cities
    "<title> <name>" to find new names
  - Rules either developed manually by trial and error or using machine learning techniques
- Statistical
  - uses a probabilistic model of the words in and around an entity
  - probabilities estimated using training data (manually annotated text)

Document Processing Hands-on

Apache Lucene Analysis



Developing Your Own Analyzer

- Subclass the Analyzer class
- Implement the `TokenStreamComponents createComponents(String fieldName)` method unico metodo astratto

  TokenStreamComponents:
  - source di tokens
  - sink: dove far uscire token

  - It defines which `Tokenizer` and `TokenFilter` to apply
  - It wraps everything into a TokenStreamComponents, an utility class encapsulating the outer components of a token stream, i.e. the source (a `Reader`) and the outer end (sink), an instance of `TokenFilter` which also serves as the `TokenStream` returned by `Analyzer.tokenStream(String, Reader)`
  - You may leverage the `fieldName` parameter if you wish your analyzer to behave differently for different document fields
- (Optional) Override `initReader(String fieldName, Reader reader)` if you need to pre-process the character stream, e.g. to strip HTML tags, before lexical analysis happens

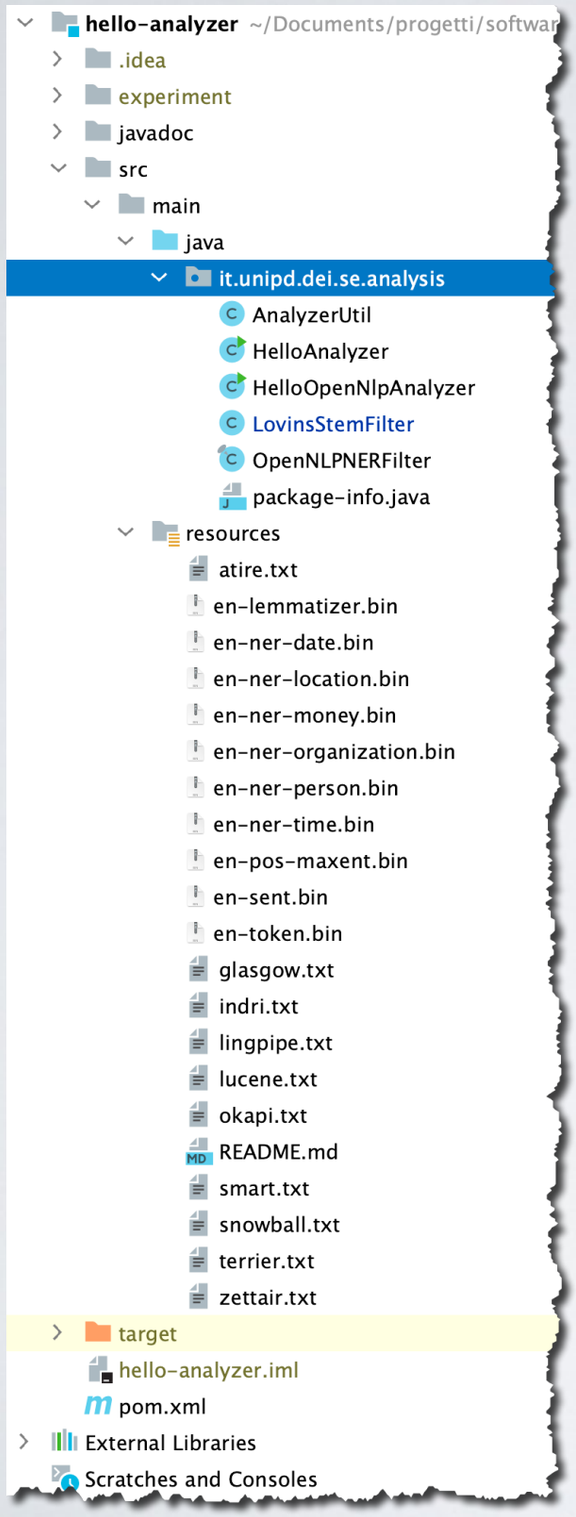
Developing Your Own TokenFilter

- Implement the `boolean incrementToken()` method
  - Consumers (i.e., `IndexWriter`) use this method to advance the stream to the next token
  - Implementing classes must implement this method and update the appropriate `AttributeImpl`s with the attributes of the next token
  - This method is called for every token of a document, so an efficient implementation is crucial for good performance

evitare concatenazione di stringhe -> usarle solo alla fine -> piuttosto StringBuilder (non thread safe) -> per thread safe, keyword "synchronized" -> evitare instance variables

- (Optional) Override `void reset()`
  - This method is called by a consumer before it begins consumption using `incrementToken()`
  - Stateful implementations must implement this method so that they can be reused, just as if they had been created fresh
  - If you override this method, always call `super.reset()`, otherwise some internal state will not be correctly reset

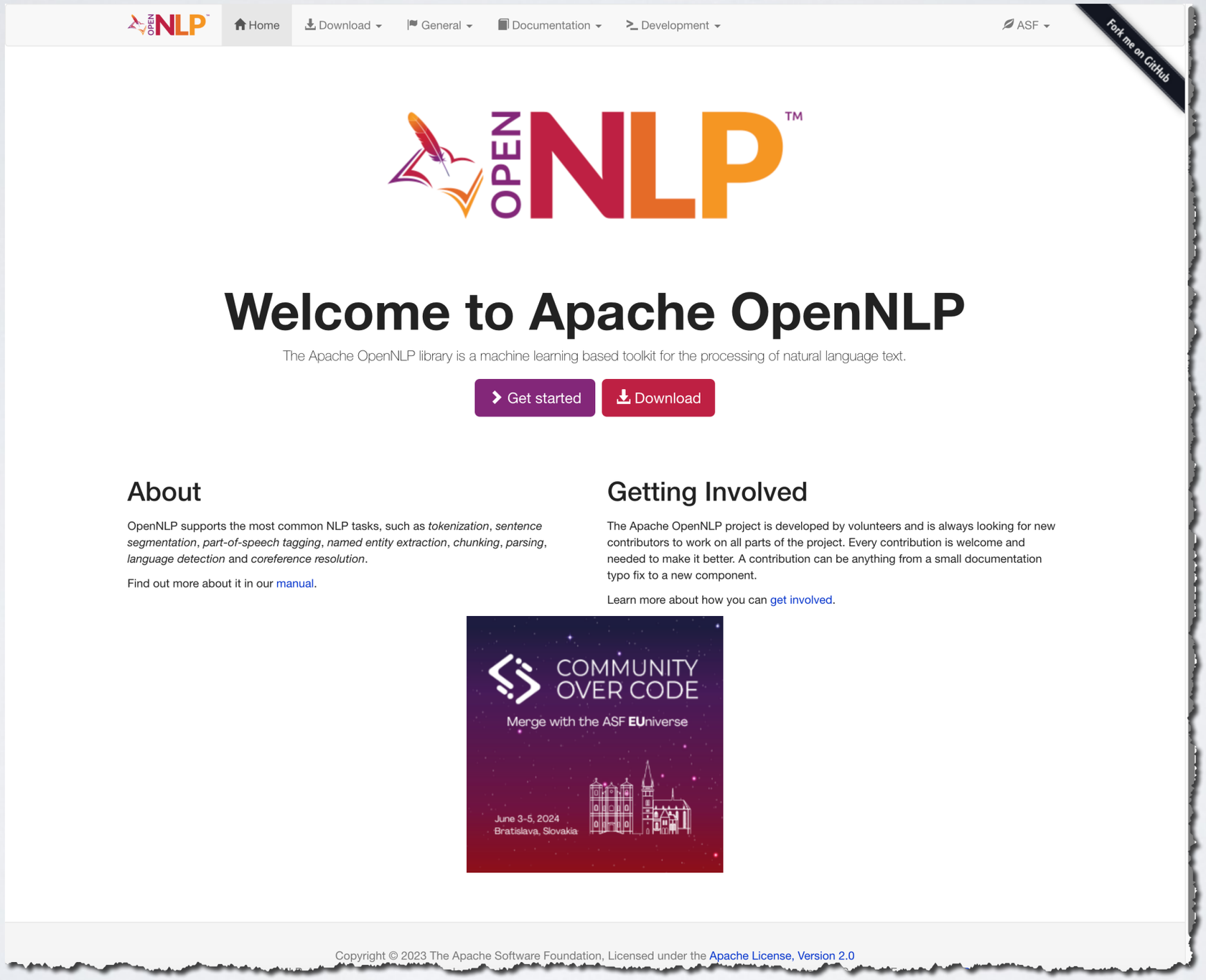
Hello, Analyzer!



- `HelloAnalyzer` shows how to create your own analyzer by using `Tokenizers` and `TokenFilters` available in Lucene
- `LovinsStemFilter` shows how to create a simple `TokenFilter` for the Lovins stemmer, which is not available in Lucene
- `HelloOpenNlpAnalyzer` shows how to create your own analyzer using advanced NLP techniques (sentence detection, tokenizations, POS tagging, NER tagging, lemmatization)
  - It relies on the integration between Apache Lucene and Apache OpenNLP
- `OpenNLPNERFiler` shows how to create an advanced `TokenFilter` for NER tagger of Apache OpenNLP, which is not available in Lucene
- `AnalyzerUtil` is an helper class containing utility methods for loading stop list and Apache OpenNLP models in the resource folder as well as for consuming a `TokenStream` and printing diagnostic information about it
- `Resource` is a folder containing several stop lists and Apache OpenNLP models. See `README.md` for more information

Apache OpenNLP

http://opennlp.apache.org/



Questions?


HOW MUCH DO YOU WANT TO CUT THE GRASS ?
NOT MUCH
Dik Browne
I MEAN HOW MUCH MONEY?
I MEAN I DON'T WANT TO CUT IT MUCH!
IT'S A HARD, HARD LANGUAGE
8-8