Goal

- Produce a set of indexing terms
  - make the best use of resources
  - accurately match user query terms
Text Processing

- Lexical Analysis
- Extract terms
- Removing stop words
- Stemming
- Term selection
Lexical Analysis

- Control information
  - Html tags : <.*>

- Numbers and digits
  - “The first class was held in the Cathedral of Learning in 1931”
  - \{1931\} => keep or remove?

- Hyphens
  - Keep or remove? When to keep? When to remove?
  - index as phrase?

- Punctuations
  - “B.C.”,”B.S.”
  - URLs

- Case-sensitive V.S case-insensitive
  - “YMCA” =? “yMca”
Lexical Analysis

- Converting byte stream to tokens
  - “The first class was held in the Cathedral of Learning in 1931.”
  - {The, first, class, was, held, in, the, Cathedral, of, Learning, in, 1931, .}
Extract Terms

- PHP

- JAVA
  [http://download.oracle.com/javase/1.4.2/docs/api/java/util/StringTokenizer.html](http://download.oracle.com/javase/1.4.2/docs/api/java/util/StringTokenizer.html)

- Python
  [http://docs.python.org/library/tokenize.html](http://docs.python.org/library/tokenize.html)

- Automatic services
Removing stop words

- Stop words:
  - 20-500 English words (an, and, by, for, of, the, ...)
  - Subject-dependent stop lists

- Pros
  - Improving storage efficiency
  - Improving search performance
  - ....

- Cons
  - “to be or not to be”
  - AT&T
  - “C” => programming index
CACM text collection:
a, about, above, accordingly, across, after, afterwards, again, against,
all, almost, alone, along, already, also, although, always, am, among,
amongst, an, and, another, any, anybody, anyhow, anyone, anything,
anywhere, apart, are, around, as, aside, at, away, awfully, b, be,
became, because, become, becomes, becoming, been, before,
beforehand, behind, being, below, beside, besides, best, better,
between, beyond, both, brief, but, by, c, can, cannot, cant, certain,
co, consequently, could, d, did, do, does,

.....
x, y, yet, you, your, yours, yourself, yourselves, z, zero, /*, manual,
unix, programmer's, file, files, used, name, specified, value, given,
return, use, following, current, using, normally, returns, returned,
causes, described, contains, example, possible, useful, available,
associated, would, cause, provides, taken, unless, sent, followed,
indicates, currently, necessary, specify, contain, indicate, appear,
different, indicated, containing, gives, placed, uses, appropriate,
automatically, ignored, changes, way, usually, allows,
indicating, specifying.
Removing stop words

- Table lookup
  - Make a table with stop words
  - Match each token against the table
  - Hashes

- Build into the lexical analyzer (embedded in the tokenizor)
Stemming

- Reduce variant word forms to a single form
  - retrieve, retrieving, retrieval, retrieved, ⇒ retriev

- Stemming approaches:
  - Table lookup – use a dictionary
  - Successor variety – fancy suffix removal
  - Affix removal - Strips prefixes of suffixes (-s, -ed, -ly, -ness)
Stemming Algorithms

- Lovins (1968)
- Porter (1980)
- Paice/Husk (1990)
- Krovetz (1993)

http://www.comp.lancs.ac.uk/computing/research/stemming/Links/algorithms.html
Term Selection

- Single word (uni-gram)
- Word pairs (bi-grams, n-grams)
- Noun phrases
  - Natural Language Processing (NLP)
  - Identify nouns along with adjectives and adverbs in the same phrase
  - “Adaptive Web” and “Interactive system design”
- Name entities
  - “Jim bought 300 shares of Acme Corp. in 2006.”
  
  ```
  <ENAMEX TYPE="PERSON">Jim</ENAMEX> bought <NUMEX TYPE="QUANTITY">300</NUMEX> shares of <ENAMEX TYPE="ORGANIZATION">Acme Corp.</ENAMEX> in <TIMEX TYPE="DATE">2006</TIMEX>.
  ```
Term processing in Python

- [http://pypi.python.org/pypi/topia.termextract/#downloads](http://pypi.python.org/pypi/topia.termextract/#downloads)
Lemur (a toolkit in IR)

http://sourceforge.net/apps/trac/lemur/wiki/Quick%20Start
Download and compile


  - Run the commands:
    tar -zxf indri-5.0.tar.gz
    cd indri-5.0
    ./configure
    make
Index this document text.
Create parameter_file

<parameters>
  <memory>200m</memory>
  <index>/path/to/outputIndex</index>
  <stemmer> <name>krovetz</name> </stemmer> <corpus>
    <path>/path/to/collection1/</path>
    <class>trectext</class>
  </corpus>
  <corpus>
    <path>/path/to/collection2/</path>
    <class>trecweb</class>
  </corpus>
  <field><name>title</name></field>
  <field><name>date</name><numeric>true</numeric><parserName>DateFieldAnnotator</parserName></field>
</parameters>
Indexing

- Run the commands:
  
  \texttt{indri-5.0/buildindex/IndriBuildIndex parameter_file}

- In Python, you can extract indexed info from indri
  
  \texttt{f = os.popen('dumpindex /home/yiling/hci/indri/index dv \%d' %docid)}
Implementation steps

- Register [http://www.opencalais.com/user](http://www.opencalais.com/user)
- Request API Key
- Preparing input content
  - TEXT
  - HTML,
  - XML
- Start implementing your scripts
Example in Python

```python
license = "the requested key from OpenCalais"
params =
""

<c:params xmlns:c="http://s.opencalais.com/1/pred/" xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#">
<c:processingDirectives c:contentType="text/txt" c:enableMetadataType="GenericRelations"
c:outputFormat="text/simple">
</c:processingDirectives>
<c:userDirectives c:allowDistribution="true" c:allowSearch="true" c:externalID="17cabs901" c:submitter="yiling">
</c:userDirectives>
<c:externalMetadata>
</c:externalMetadata>
</c:params>

content = "NEW YORK (CNNMoney.com) -- The U.S. Postal Service may be forced to eliminate"

def get_oc_result(license, content, params):
    url = "http://api.opencalais.com/enlighten/rest/"
    headers = {"Content-type": "application/x-www-form-urlencoded"}
    conn = httplib.HTTPConnection('api.opencalais.com')
    conn.request("POST", "/enlighten/rest/", urlencode({'content':content, 'licenseID':license, 'paramsXML':params}), headers)
    r1 = conn.getresponse()
    data = r1.read()
    return data

oc_out = get_oc_result(license, content, params)
print oc_out
```
Sample of OpenCalais Output

<CalaisSimpleOutputFormat>
  <Person count="2" relevance="0.121">Kenneth E. Kendall</Person>
  <Position count="2" relevance="0.338">designer</Position>
  <Position count="2" relevance="0.368">airport Ramp Planner</Position>
  <Company count="1" relevance="0.082" normalized="PRENTICE HALL">Prentice-Hall</Company>
  <IndustryTerm count="1" relevance="0.215">verbal protocol</IndustryTerm>
  <IndustryTerm count="1" relevance="0.133">computer applications</IndustryTerm>
  <IndustryTerm count="1" relevance="0.326">verbal protocol</IndustryTerm>
  <IndustryTerm count="1" relevance="0.353">userInteractive systems</IndustryTerm>
  <IndustryTerm count="1" relevance="0.171">software vendors</IndustryTerm>
  <Position count="1" relevance="0.326">control-room Ramp Planner</Position>
  <Position count="1" relevance="0.326">text editor</Position>
  <Technology count="1" relevance="0.326">verbal protocol</Technology>
  <Topics>
    <Topic Taxonomy="Calais" Score="0.926">Technology_Internet</Topic>
  </Topics>
</CalaisSimpleOutputFormat>
Questions, Comments or Discussions

ONLINE TEXT PROCESSING SOURCE
HTTP://GNOSIS.CX/TPIP/
HTTP://PROQUEST.SAFARIBOOKSONLINE.COM/BOOK/SOFTWARE-ENGINEERING-AND-DEVELOPMENT/PATTERNS/9781933988665