Exercise 1: Boolean Retrieval

- Any questions?
Lecture last week: Term vocabulary
Index construction

Collect documents
- Tokenizing
- Linguistic preprocessing
- Build the index (postings list)
Lecture last week: Term vocabulary
Index construction

- Collect documents
  - What type of encoding?
  - What language?
  - What is a document?
- Tokenizing and linguistic preprocessing e.g:  
  - Throw away punctuation
  - Remove stop words
  - Normalization
  - Stemming
  - Lemmatization
Lecture last week: Term vocabulary
Skip lists

In practice \( \sqrt{\text{Number of postings}} \)
Lecture last week: Term vocabulary
Phrase search

- Phrase query
- Two types:
  - N-word index
  - Positional index
Exercise 2: Advanced Indices

- Part A: Implementing a n-word index
- Part B: Implementing a positional index
- Part C: Theoretical questions about stemming
Exercise 2: Advanced indices
A & B

- Phrase query: should only return documents that contain this exact phrase
- Updated parser to handle phrase queries
  - Phrase needs to be enclosed in double quotes
  - Output format: Ordered list of individual terms that make up the phrase
- Updated tokenizer to drop stop words
  - Provided python snippet to acquire list of used stop words
  - Use function `remove_stop_words()` to remove stop words from flattened queries
Exercise 2: Advanced indices
A: n-word index

- Mostly complete reference implementation for standard inverted index provided
- Modify `add_document()` to build a n-word index for configurable n
- Modify `execute_query()` to be able to process phrase queries on n-word index
Exercise 2: Advanced indices
A: n-word index

Modify `add_document()` to build a n-word index for configurable n

```python
def add_document(path):
    '''
    Add a document to the inverted index. Return the document's document ID.
    Remember the mapping from document ID to document in the 'documents' data structure.
    '''
    # make sure that we access the global variables we have defined
    global the_index, documents, document_id_counter, n
    # do not re-add the same document.
    if path in documents.values():
        # find and return document id for document which is already part of index.
        for docid, doc in documents.items():
            if doc == path:
                return docid
        docid = document_id_counter
        documents[docid] = path
        document_id_counter += 1
        print("Adding "'s to index" % path)
        for word in tokenize_document(path):
            # TODO for assignment: change this inner loop to create a n-word index instead
            # skip document, if it is already in the index for 'word'
            if word in the_index.keys() and the_index[word][-1] == docid:
                continue
            the_index.setdefault(word, []).append(docid)
    return docid
```
Exercise 2: Advanced indices
A: n-word index

- Example for n=2

Help ETH Zurich to flexibly react to new challenges and to set new accents in the future.
Exercise 2: Advanced indices

A: n-word index

- Modify `execute_query()` to be able to process phrase queries on n-word index.
  - Can be done by only adding code in the box

```python
# Feel free to remove this print() if you don't find it helpful.
print("Flat query repr:", flat)

args = []
# go through arguments and fall back on recursive evaluation if we
# could not completely flatten the query
for arg in flat.args:
    if isinstance(arg, Operation):
        # as soon as we find a argument to the top-level operation
        # which is not just a term, we fall back on the tree query
        # execution strategy.
        return execute_query_tree(flat)
    elif isinstance(arg, list):
        # Assume it is a phrase
        for w in arg:
            assert(isinstance(w, str))
            # TODO: Implement preprocessing for n-word queries
            # How exactly you do this depends on how you've constructed the
            # n-word index.
            print()
            print("\>>> Found phrase \'\$s\' in query \'\$s\', NYI!" % (' '.join(arg), query))
            print()
            return None
    else:
        args.append(arg)

if flat.op == 'OR':
    # For demonstration purposes, utilise python's set() datatype
```
Exercise 2: Advanced indices
A: n-word index

- Check with examples below
  - Use different n
- Answer question to `grep`
- Optional exercises:
  - Implement post processing step which actually correctly filters the results
  - Improve query execution engine to handle phrases in conjunction with Boolean operators
Exercise 2: Advanced indices
B: positional index

- Mostly complete reference implementation for Boolean queries provided
- Complete `add_document()` to build a positional index
- Complete `positional_intersect_two()` to be able to process phrase queries on the positional index
Exercise 2: Advanced indices
B: positional index

- **Complete** `add_document()`
  - Adds one document to the_index

```python
def add_document(doc):
    
    # Add a document to the inverted index. Returns the document's ID
    
    global documents, docid_counter, the_index

    # do not re-add the same document.
    if doc in documents.values():
        return documents[doc]

    docid = docid_counter
    documents[docid] = doc
    docid_counter += 1

    print("Adding document \%s to inverted index with document ID \%d" % (doc, docid))
    for pos, word in enumerate(tokenize_document(doc)):
        
        # TODO for the assignment, implement this (remove the next line)
        raise "unimplemented"

    return docid
```

- the_index should be a map
  - words → lists of (document_id, [list of positions])
Exercise 2: Advanced indices
B: positional index

- **the_index** should be a map
  - words → lists of (document_id, [list of positions])
  - No term frequency
Exercise 2: Advanced indices
B: positional index

- **Implement** `positional_intersect_two()`
  - Intersects two posting lists of a positional index
  - Should return a list of tuples
    - `(document_id, position_of_p1, position_of_p2)`

"ETH Zurich"
Exercise 2: Advanced indices
B: positional index

- This method is used by already implemented `positional_intersect()` which positionally intersects postings lists for a list of words.
- Check your implementation with provided examples

"ETH Zurich"
Exercise 2: Advanced indices
B: positional index

- Everything after that code segment does not need to be modified
  - `execute_query()` already implemented
  - `print_result()` already implemented
- Example queries in last code segment to test your code with Boolean queries
Exercise 2: Advanced indices
C: stemming

- Already implemented Porter stemmer in jupyter.
  - Play around with it to get a feel for how it transforms words
- Answer true/false questions on moodle