task1

November 2, 2023

0.1 Task 1

[3]:
```
import numpy as np
import codecs, string

with codecs.open("common-english-words.txt", "r", "utf-8") as f:
    common_words = f.read().split(",")
```

1a)

[4]:
```
# Create and clean the text
text = "Intelligent behavior in people is a product of the mind. But the mind␣
 ↪itself is more like what the human brain does."
def clean(text):
    # To lower case
    text = text.lower()
    # Remove stop words
    text = " ".join(list(filter(lambda x: x not in common_words, text.split("␣
 ↪"))))
    # Remove punctutation
    text = "".join(list(filter(lambda x: x not in string.
 ↪punctuation+"\n\r\t\0", text)))
    return text
text = clean(text)
text
```

[4]:
```
'intelligent behavior people product mind mind itself more human brain does'
```

[5]:
```
# Create inverted index (for one document, simple)
vocab = dict([(key, {0: text.count(key)}) for key in set(text.split(" "))])
vocab
```

[5]:
```
{'product': {0: 1},
 'mind': {0: 2},
 'behavior': {0: 1},
 'human': {0: 1},
 'brain': {0: 1},
 'does': {0: 1},
```

```
 'more': {0: 1},
 'itself': {0: 1},
 'intelligent': {0: 1},
 'people': {0: 1}}
```

[39]:
```
# Construct a (normal) inverted index
# For one document this is just a frequency list
def gen_idx(corpus):
    # Initiate the index as a dict('term', dict('doc', num_occ))
    idx_list = dict([(key, {}) for key in set(" ".join(corpus).split(" "))])
    for doc_idx, doc in enumerate(corpus):
        # Increment number of occurrences for each occurrence
        for term in doc.split(" "):
            if doc_idx not in idx_list[term].keys():
                idx_list[term][doc_idx] = 0
            idx_list[term][doc_idx] += 1
    return idx_list
gen_idx([text])
```

[39]:
```
{'itself': {0: 1},
 'brain': {0: 1},
 'human': {0: 1},
 'intelligent': {0: 1},
 'people': {0: 1},
 'mind': {0: 2},
 'product': {0: 1},
 'more': {0: 1},
 'behavior': {0: 1},
 'does': {0: 1}}
```

1b)

[40]:
```
def blockify(corpus, block_size=3):
        corpus = [[doc.split(" ")[block_size*i:block_size*i+block_size] for i␣
 ↪in range(len(doc.split(" "))//block_size+1)] for doc in corpus] # Please␣
 ↪don't look at this monstrosity
        corpus = [list(filter(lambda x: len(x) > 0, doc)) for doc in corpus]
        return corpus

def gen_idx_block(corpus, block_size=3):
    # Initiate the index as a dict('term', dict('doc', [block_ids]))
    idx_list = dict([(key, {}) for key in set(" ".join(corpus).split(" "))])
    corpus_blocks = blockify(corpus, block_size)
    for doc_idx, doc in enumerate(corpus, 0):
        # Generate blocks
        blocks = corpus_blocks[doc_idx]
        corpus_blocks.append(blocks)
```

```
        # For each distinct term in the document
        for term in set(doc.split(" ")):
            if doc_idx not in idx_list[term].keys():
                idx_list[term][doc_idx] = []
            # Find occurrences and add block to block list:
            for block_idx, block in enumerate(blocks):
                if term in block:
                    idx_list[term][doc_idx].append(block_idx)
    return idx_list, corpus_blocks

# 'word': {doc_id: [block_indices]}
# Print the results:
print(text)
print("\nBlocks:")
idx, blocks = gen_idx_block([text], block_size=3)
for bid, block in enumerate(blocks[0]):
    print(bid,block)
idx
```

```
intelligent behavior people product mind mind itself more human brain does

Blocks:
0 ['intelligent', 'behavior', 'people']
1 ['product', 'mind', 'mind']
2 ['itself', 'more', 'human']
3 ['brain', 'does']
```

[40]:
```
{'itself': {0: [2]},
 'brain': {0: [3]},
 'human': {0: [2]},
 'intelligent': {0: [0]},
 'people': {0: [0]},
 'mind': {0: [1]},
 'product': {0: [1]},
 'more': {0: [2]},
 'behavior': {0: [0]},
 'does': {0: [3]}}
```

1c) Partial Vocabulary Suffix Tree Here we assume "vocabulary" means word-level instead of character-level suffixes, and "partial" means without stopwords. "$" marks end condition. Here, unary paths ending in a leaf node are removed to decrease the amount of nodes, as suggested in the lecture.

```
     1          2        3        4      5     6      7      8      9     10    11  12
intelligent behavior people product mind mind itself more human brain does $

(root)
    +-$-(12)
```

```
   +-does-(11)
   +-brain-(10)
   +-human-(9)
   +-more-(8)
   +-itself-(7)
   +-mind
   |   +-itself-(6)
   |   +-mind-(5)
   +-product-(4)
   +-people-(3)
   +-behavior-(2)
   +-intelligent-(1)
```

1d) Indexing a corpus

[41]:
```
# Create the corpus and clean it
corpus = [
    "Although we know much more about the human brain than we did even",
    "ten years ago, the thinking it engages in remains pretty much a total",
    "mystery. It is like a big jigsaw puzzle where we can see many of the",
    "pieces, but cannot yet put them together. There is so much about us",
    "that we do not understand at all.",
]
corpus = [clean(text) for text in corpus]
corpus
```

[41]:
```
['although know much more human brain even',
 'ten years ago thinking engages remains pretty much total',
 'mystery big jigsaw puzzle see many',
 'pieces put together much',
 'understand all']
```

[42]:
```
# Generate the inverted index for the corpus
# Note: Document ID is one lower than in the assignment text for simplicity
index = gen_idx(corpus)
index
```

[42]:
```
{'total': {1: 1},
 'big': {2: 1},
 'pretty': {1: 1},
 'pieces': {3: 1},
 'understand': {4: 1},
 'mystery': {2: 1},
 'even': {0: 1},
 'brain': {0: 1},
 'ten': {1: 1},
 'put': {3: 1},
 'puzzle': {2: 1},
```

```
 'human': {0: 1},
 'much': {0: 1, 1: 1, 3: 1},
 'ago': {1: 1},
 'more': {0: 1},
 'all': {4: 1},
 'although': {0: 1},
 'jigsaw': {2: 1},
 'remains': {1: 1},
 'years': {1: 1},
 'thinking': {1: 1},
 'see': {2: 1},
 'know': {0: 1},
 'many': {2: 1},
 'engages': {1: 1},
 'together': {3: 1}}
```

[51]:
```
# It could also be interesting to use block indexing on the corpus
index, _ = gen_idx_block(corpus, block_size=3)
index
```

[51]:
```
{'total': {1: [2]},
 'big': {2: [0]},
 'pretty': {1: [2]},
 'pieces': {3: [0]},
 'understand': {4: [0]},
 'mystery': {2: [0]},
 'even': {0: [2]},
 'brain': {0: [1]},
 'ten': {1: [0]},
 'put': {3: [0]},
 'puzzle': {2: [1]},
 'human': {0: [1]},
 'much': {0: [0], 1: [2], 3: [1]},
 'ago': {1: [0]},
 'more': {0: [1]},
 'all': {4: [0]},
 'although': {0: [0]},
 'jigsaw': {2: [0]},
 'remains': {1: [1]},
 'years': {1: [0]},
 'thinking': {1: [1]},
 'see': {2: [1]},
 'know': {0: [0]},
 'many': {2: [1]},
 'engages': {1: [1]},
 'together': {3: [0]}}
```