Text Summarization

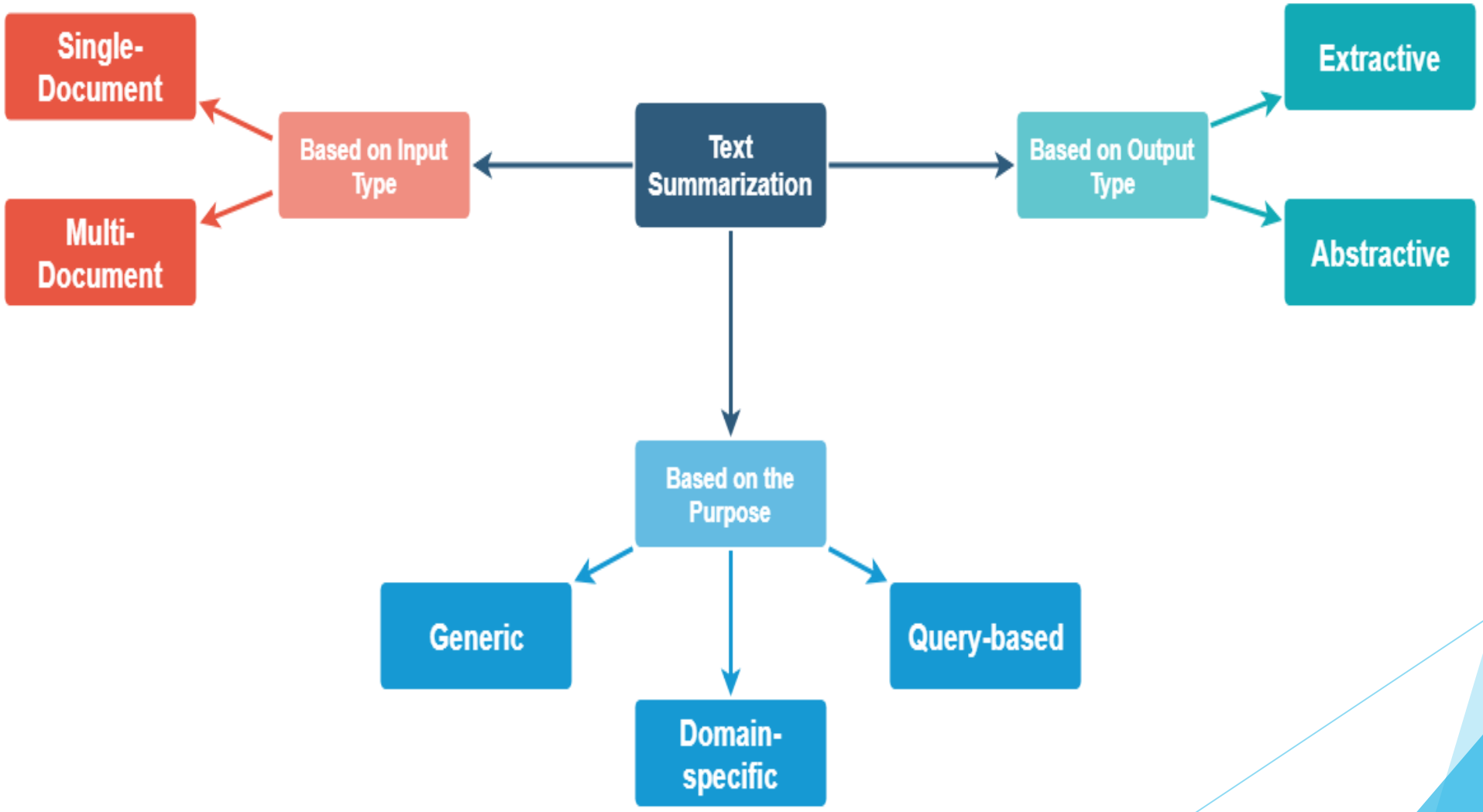
Dr. Liao

2/4/2020

Overview

- Text Summarization
  - What to Summarize?
  - How to Summarize?
    - Extractive Summarization
    - Abstractive Summarization
  - Steps of Simple Summarize Methods
  - TF-IDF
  - Web Scraping in BeautifulSoup
  - In-Class Hands-On Programming with **Python, NLTK, Gemsim, and BeatuifulSoup**

What to Summarize?



How to Summarize?

- **Extractive** Method
  - Extract key sentences from a text without modifying any word.
    - **Pros**: robust, straightforward
    - **Cons**: lack in flexibility
- **Abstractive** Method – Like what humans do
  - Generate new sentences to summarize the entire set.
    - **Pros**: more fluent and natural – intelligent
    - **Cons**: much harder
- Hybrid method – use both

Extractive Summarization

- Feature based
  - Weighted scores
    - Term frequency
    - Length of the sentence
    - Position of the sentence
    - Presence of the verb in the sentence
  - Luhn's Algorithm (1958) – Significance of the word (*frequency*)
  - And more …
- Graph based - TextRank (for single doc) & LexRank (for multi-docs)
- Topic based - Latent Semantic Analysis (LSA)
- Gramma based

Abstractive Summarization

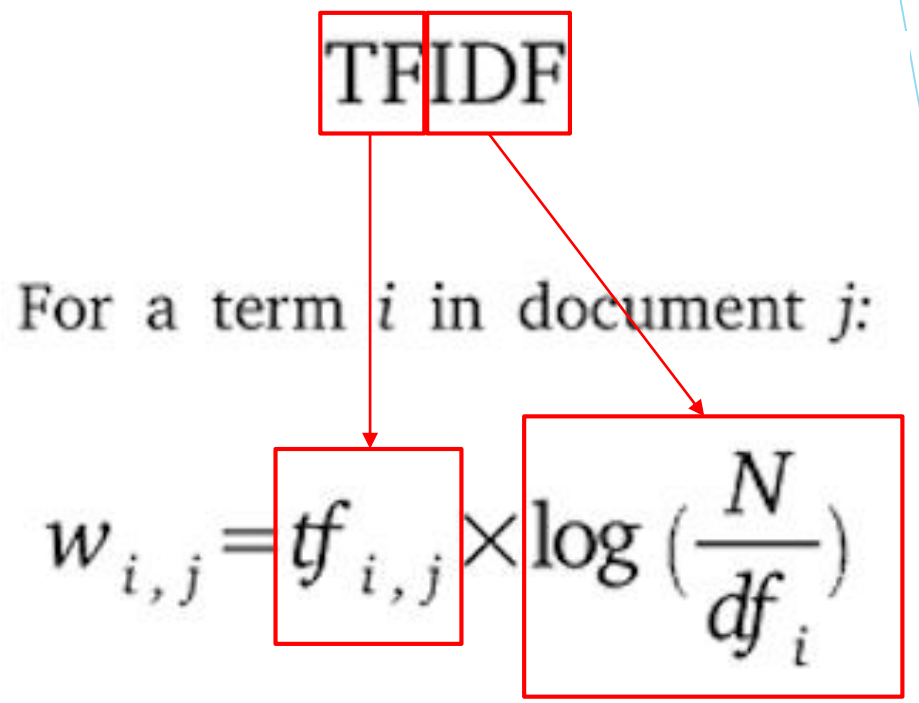
- Recurrent Neural Network (RNN)
- Sequence to Sequence
- Pointer-generator Network
- Deep Reinforced Model
- And more…

Steps of the Simple Summarization Method

- Word Frequency Based Statistical Methods
  - Text preprocessing
  - Word frequency calculation
    - Weighted frequency *(many ways)*
    - **TF-IDF**
    - Other mathematical methods
  - Score each sentence with each word weighted frequency
  - Build a summary with the sentences with highest scores

TF-IDF - Vectorization

- TF(Term Frequency)
  - defines the probability of finding a word in the document
- IDF (Inverse Document Frequency)
- TF-IDF
  - The multiplication of TF and IDF values
  - Intend to reflect how important a word is to a document in a collection or corpus
  - Weight rare words higher than common words
  - One of the most popular term-weighting schemes today



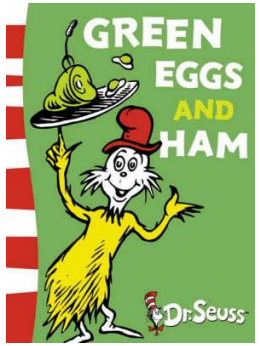
Vector Space Model

- Documents and queries are represented as a vector of features representing terms (words) that occur with in the collection

Features

cat	hat	green	eggs	ham	sam	grinch	stole	….	tree

All of the content words with in the collection



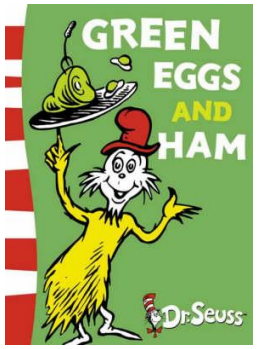
Feature Vector for Document

0	0	1	1	1	1	0	0	….	0

whether the feature exists with in the document

Adopted from Dr. Uzuner's NLP590 Course Slide

Term weighting



Binary Vector

0	0	1	1	1	1	0	0	….	0

Frequency Vector

0	0	5	10	6	50	0	0	….	0

IDF Vector

0	0	2.9	2.3	2.8	.69	0	0	….	0

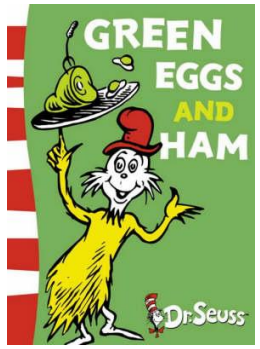
$$\mathrm{idf}(t, D) = \log \frac{N}{|\{d \in D : t \in d\}|}$$

with

- N: total number of documents in the corpus
- $|\{d \in D : t \in d\}|$: number of documents where the term t appears (i.e., $\mathrm{tf}(t, d) \neq 0$). If the term is not in the corpus, this will lead to a division-by-zero. It is therefore common to adjust the denominator to $1 + |\{d \in D : t \in d\}|$.

Adopted from Dr. Uzuner's NLP590 Course Slide

Term weighting



Binary Vector

0	0	1	1	1	1	0	0	….	0

Frequency Vector

0	0	5	10	6	50	0	0	….	0

IDF Vector

0	0	2.9	2.3	2.8	.69	0	0	….	0

TF-IDF Vector

0	0	14.5	23	16.8	16.8	0	0	….	0

$$\mathrm{tfidf}(t, d, D) = \mathrm{tf}(t, d) \times \mathrm{idf}(t, D)$$

Adopted from Dr. Uzuner's NLP590 Course Slide

NLP Hands-On Programming in Class

- Code Examples & Tutorials for **Text Summarization** with NLTK, Gensim, BeautifulSoup, Python
  - Dr. Liao wrote them **particularly** for
    - this course learning
    - **Assignments, labs, and final project** examples
- All programming tutorials & code example demos
  - Using online Jupyter Lab in class
- **More** code examples & tutorials in coming classes…

Homework

- Video Lecture Report 1
  - Watch Robot Sophia video
  - Due on 2/11
- Optional Individual Lab 1
  - Text Summarization
  - Due on 2/18
  - Extra credit
- Programming Assignment 1

  - Chatbot Eliza
  - 3 weeks
  - Due on 2/25