LogoType™

MUD

Meeting Using DeepLearning

201601364 박주영(T)　201500000 홍승환
201600599 김아연　201701138 김혜원

목차

1. 요약 주요 기능

문헌 내에서 **중요한 핵심 문장들을 뽑아서** 그 문장들로 요약문을 만드는 방법

핵심 키워드 추출	소재 뽑기	자동 요약문 추출 (Automatic Text Summarization)
• TF-IDF 이용 키워드 • TextRank 이용 키워드	텍스트분류Text Classification -딥러닝 이용 -RNN(LSTM)	• 추출적 요약 (Extractive) • 추상적 요약(Abstractive)

문헌의 내용을 잘 반영할 수 있는 **추상적인 문장을 직접 생성**함으로써 요약문을 만듬

2. 텍스트 마이닝

요약

인간의 언어(문법 + 형태소 + 의미)

→ 수치화(벡터화) / 수식화

→ 기계 Data

→ 중요도 / 순서 / 유사도 **요약**

비정형 텍스트 + 자연어 처리 기술

→ **유용 정보 추출/ 가공**

= 텍스트 마이닝

TEXT → 전처리 과정 → 의미정보 변환 → 의미정보 추출 → 패턴 및 경향 분석

3. 텍스트 마이닝 이용한 요약 과정

전처리 과정

- 텍스트 정규화 : ①토큰화(Tokenization) ②어간 추출(Stemming)
- 형태소 분석 KonLPY
- 불용어 제거
- 어간 추출(Stemming)

용도에 맞게 텍스트를 사전에 처리

용도에 따라 어떤 형태소 조합으로 분석 되어야 하는지가 다름!

요약

3. 텍스트 마이닝 이용한 요약 과정

텍스트를 분석하기 위해
텍스트 구조화

전처리 과정

- 텍스트
- 형태소
- 불용어
- 어간 추출(S ming)

(Stemming)

텍스트 분석

- 정보 추출과 탐색 : (키워드 분석 TF-IDF or TextRank 사용)
- 정보 추출과 탐색 : 연관어 분석
- 문서 유사도의 측정과 문서 단어 행렬 cosine similarity
- 문서 군집화
- 텍스트 분류

요약

3. 텍스트 마이닝 이용한 요약 과정

전처리 과정

- 텍스트 정규화 : ①토큰화(Tokenization) ②어간 추출(Stemming)
- 형태소 분석 KonLPY
- 불용어 제거
- 어간 추출(Stemming)

텍스트 분석

- 정보 추출과 탐색 : (키워드 분석 TF-IDF or TextRank 사용)
- 정보 추출과 탐색 : 연관어 분석
- 문서 유사도의 측정과 문서 단어 행렬 cosine similarity
- 문서 군집화
- 텍스트 분류

요약

- 추출적 요약
- 추상적 요약

3. 추출 요약 기술

추출 요약 기술 : 전처리 과정 + 텍스트 분석 → 수치화, 벡터화 활용한 통계적 텍스트 표현 모델
이런 모델들을 이용하여 요약 기술 알고리즘 : PageRank, TextRank, LexRank
------**그래프 알고리즘, 노드나 간선의 가중치 계산 기술**을 이용

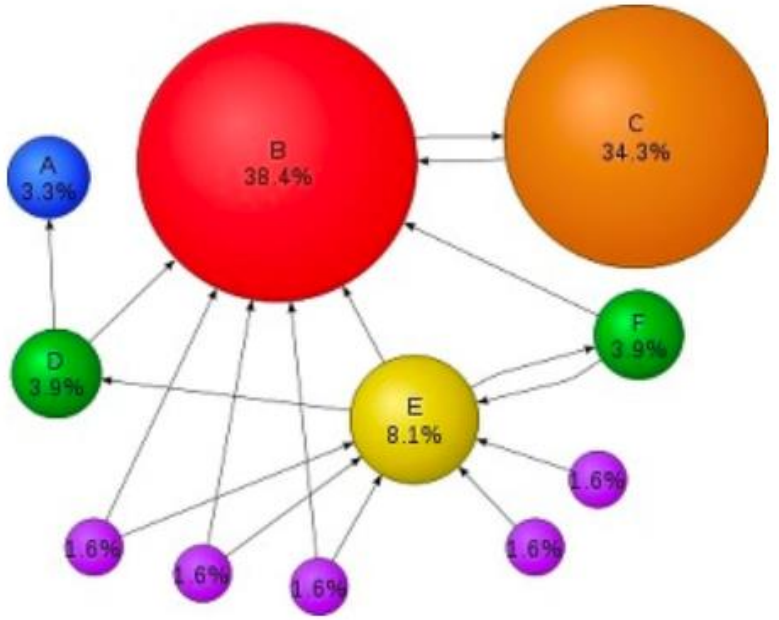
PageRank

$$PR(A) = (1-d) + d\left(\frac{PR(B)}{L(B)} + \frac{PR(C)}{L(C)} + \frac{PR(D)}{L(D)}\right)$$

그래프 랭킹 알고리즘. 웹 페이지들의 랭킹을 위해 개발
웹 문서에 상대적 중요도에 따라 가중치 부여 방법(간선 연결된 다른 노드들의 개수와 중요도로 평가)

TextRank

기본적으로 undirected 그래프를 가정하며, 유사도를 계산하기 위해 간선에 가중치가 부여된 그래프까지 고려하였다. 예를 들어 간선에 가중치가 부여된 그래프의 경우 노드의 가중치는 다음과 같은 수식으로 계산된다



LexRank

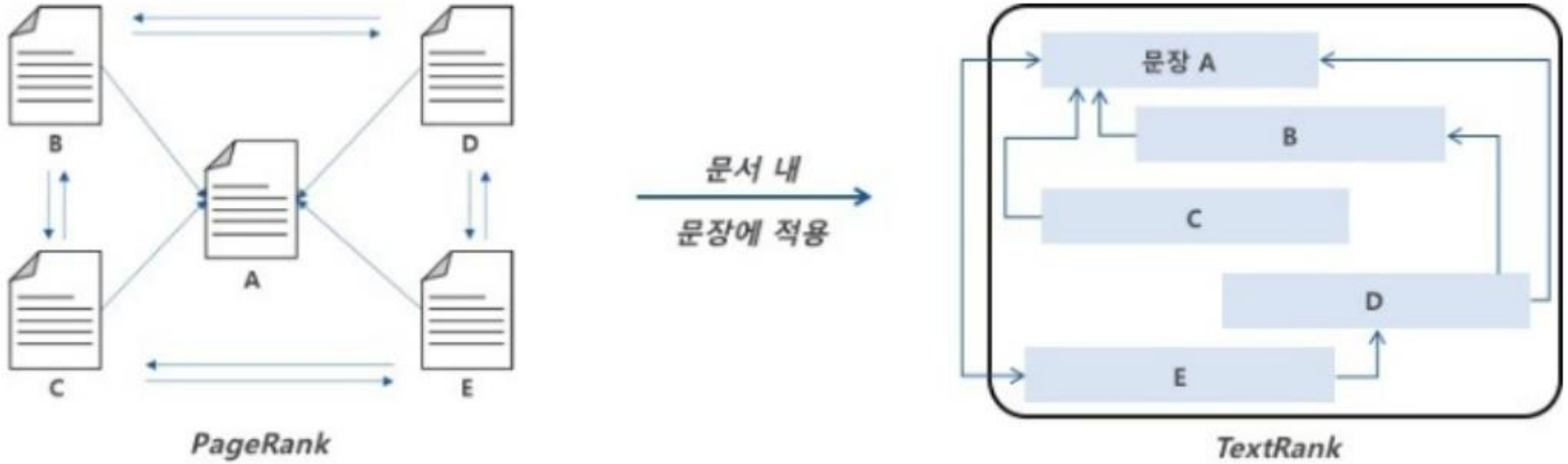
textRank와 유사. 대용량 문서 요약을 위한 일부 방법 중 하나로 이 기법 을 이용하였다는 것이며, TextRank와는 달리 다중 문서(multi-document) 요약에 적용

3. 추출 요약 기술

TextRank

PageRank의 중요도가 높은 웹 사이트는 다른 많은 사이트로부터 링크를 받는다는 점에 착안하여
문서 내의 문장 or 단어를 이용하여 문장의 Ranking을 계산하는 알고리즘

* 핵심 단어를 선택 ← **단어 간의 co-occurrence graph** 생성
* 핵심 문장을 선택 ← **유사도를 기반으로 sentence similarity** 생성 (Cosine Similarity이용)
* 랭킹 : 각각 그래프에 PageRank 를 학습하여 각 마디 (단어 혹은 문장) 의 랭킹을



[그림 2] PageRank > TextRank 적용

3. 추출 요약 기술

TextRank

PageRank의 중요도가 높은 웹 사이트는 다른 많은 사이트로부터 링크를 받는다는 점에 착안하여
문서 내의 문장 or 단어를 이용하여 문장의 Ranking을 계산하는 알고리즘

* 핵심 단어를 선택 ← **단어 간의 co-occurrence graph** 생성
* 핵심 문장을 선택 ← **유사도를 기반으로 sentence similarity** 생성 (Cosine Similarity이용)
* 랭킹 : 각각 그래프에 PageRank 를 학습하여 각 마디 (단어 혹은 문장) 의 랭킹을

$$TR(V_i) = (1-d) + d * \sum_{V_j \in In(V_i)} \frac{w_{ji}}{\sum_{V_k \in Out(V_j)} w_{jk}} TR(V_j)$$

- $TR(V_i)$: 문장 또는 단어$(V)i$에 대한 TextRank값
- w_ij: 문장 또는 단어 i 와 j 사이의 가중치
- d : dampingfactor , PageRank에서 웹 서핑을 하는 사람이 해당 페이지를 만족하지 못하고 다른페이지로 이동하는 확률로써, TextRank에서도 그 값을 그대로 사용(0.85로 설정)
- TextRank $TR(V_i)$를 계산 한 뒤 높은 순으로 정렬

3. 추출 요약 기술

TextRank

* 핵심 단어를 선택 ← **단어 간의 co-occurrence graph** 생성
* 핵심 문장을 선택 ← **유사도를 기반으로 sentence similarity** 생성 (Cosine Similarity이용)
* 랭킹 : 각각 그래프에 PageRank 를 학습하여 각 마디 (단어 혹은 문장) 의 랭킹을

```
def word_graph(sents, tokenize=None, min_count=2, window=2, min_cooccurrence=2):
    idx_to_vocab, vocab_to_idx = scan_vocabulary(sents, tokenize, min_count)
    tokens = [tokenize(sent) for sent in sents]
    g = cooccurrence(tokens, vocab_to_idx, window, min_cooccurrence, verbose)
    return g, idx_to_vocab
```

<단어 그래프>

두 단어 간의 유사도를 정의←두 단어의 co-occurrence 계산
Co-occurrence = 문장 내 두 단어의 간격이 window 인 횟수
dict of dict 형식의 그래프를 scipy 의 sparse matrix 로 변환
명사, 동사, 형용사와 같은 단어만 단어 그래프를 만드는데 이용
Tokenize 함수 = 전처리 과정

3. 추출 요약 기술

TextRank

* 핵심 단어를 선택 ← **단어 간의 co-occurrence graph** 생성
* 핵심 문장을 선택 ← **유사도를 기반으로 sentence similarity** 생성 (Cosine Similarity이용)
* 랭킹 : 각각 그래프에 **PageRank 를 학습** 하여 각 마디 (단어 혹은 문장) 의 랭킹을

```python
import numpy as np
from sklearn.preprocessing import normalize

def pagerank(x, df=0.85, max_iter=30):
    assert 0 < df < 1

    # initialize
    A = normalize(x, axis=0, norm='l1')
    R = np.ones(A.shape[0]).reshape(-1,1)
    bias = (1 - df) * np.ones(A.shape[0]).reshape(-1,1)

    # iteration
    for _ in range(max_iter):
        R = df * (A * R) + bias

    return R
```

1.키워드 추출

+문장 그래프

= **2.문장 요약 (Ranking)**

3. 추출 요약 기술

TextRank

* 핵심 단어를 선택 ← **단어 간의 co-occurrence graph** 생성
* 핵심 문장을 선택 ← **유사도를 기반으로 sentence similarity** 생성 (Cosine Similarity이용)
* 랭킹 : 각각 그래프에 PageRank 를 학습하여 **각 마디 (단어 혹은 문장) 의 랭킹을**

```
from textrank import KeysentenceSummarizer

summarizer = KeysentenceSummarizer(tokenize = komoran_tokenize, min_sim = 0.5)
keysents = summarizer.summarize(sents, topk=10)
```

```
summarizer = KeysentenceSummarizer(tokenize = komoran_tokenizer, min_sim = 0.3)
keysents = summarizer.summarize(sents, topk=3)
```

```
print(summarize(news.text, ratio=0.1))
```

```
In [29]: print(summarize(news.text, ratio=0.1))
```

```
요즘은 개발은 안하고 맨날 제안서만 쓴다.
하지만 자야 될 시간이므로 일단 인사만 하고 모르쇠로 일관하며 취침 모드로 들어간다.(9시가 넘어서
리) 아이들이 잔다 이제 쓰레기 정리하고 분리수거 하러 가야 된다.
```

3. 추출 요약 기술

LexRank

:문장 수반 관계를 고려한 문서 요약

TextRank + **유사도 선택 가능**
(주로 TF-IDF + Cosine similarity)
→긴 문서에 적합

비슷한 주제를 가진 문장들은 서로 가깝게 있을 것 가정

+중복 문장 제거

+ **여러 문서 요약가능**

```
In [2]: from lexrankr import LexRank

In [7]: lexrank = LexRank()
        lexrank.summarize('''
        사건(?)이 발생하면, 원인을 "과학적"으로 분석한다.

        짐작과 추측으로 결론내지 않고, "데이터"와 "수치"로 가설들을 증명 또는 반증하면서 결론을 찾아간다.
        물증없이 짐작과 추측만으로 섣불리 내려진 결론은 전진을 방해하고 낭비를 초래한다.
        사건의 증거를 수집하고 수사기록을 남긴다. 사건의 증거와 기록은 영구히 보존한다.

        당시의 지식과 증거로는 완벽한 결론이 나지 않는 미제사건도 발생한다.
        이후 유사한 사건이 발생했을 때, 동일범의 사건인지/아닌지 확인할 수 있는 증거를 남겨두어야 한다.
        증거가 수집/보관되지 않으면, 교묘한 연쇄살인마는 잡을 수 없다.
        최대한 증거를 많이 남기는 범인이 되라.

        사건의 범인은 "다행히" 많은 경우, 우리자신이다. 이왕 범죄자가 될 거라면, 흔적과 증거를 많이 남겨서
        빨리 잡히는 범인이 되어야 한다. 한번 발생한 사건은 반드시 다시 발생한다.

        사건이 다시 발생하는 것이 꼭 문제는 아니다.
        다시 발생한 사건을 좀 더 쉽고 명확하게 해결할 수 있도록 하지 못한다면 문제다.''')
        summaries = lexrank.probe(None)
        for i, summary in enumerate(summaries):
            print(str(i), summary)

0 당시의 지식과 증거로는 완벽한 결론이 나지 않는 미제사건도 발생한다
1 이후 유사한 사건이 발생했을 때, 동일범의 사건인지/아닌지 확인할 수 있는 증거를 남겨두어야 한다
```

Thank You

감사합니다.