Age Estimation System

Natural Language Processing & Information Retrieval

어연자(Reverse-NL team)

20153378 Lee, Rohee

20150804 Jung, Minjoon

20131699 Jung, Inseok

Index

001/ Subject

002/ Schedule

003/ Process

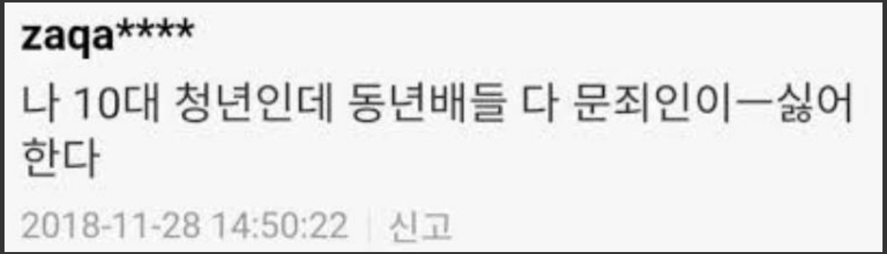
004/ Result

005/ Conclusion

01

Subject

1. Introduction & Subject

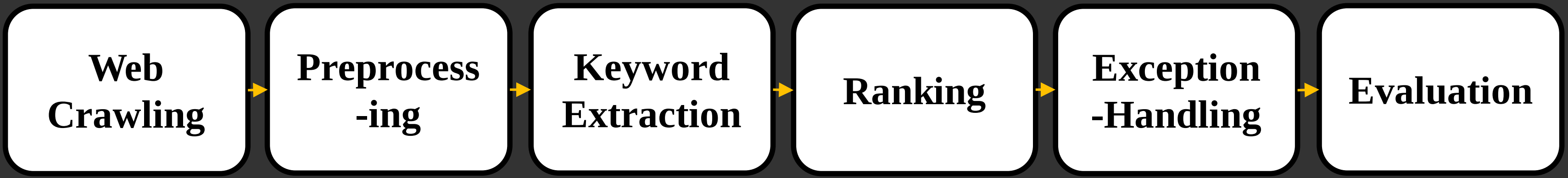


zaqa****

나 10대 청년인데 동년배들 다 문죄인이ㅡ싫어
한다

2018-11-28 14:50:22 | 신고

Age Estimation System



Web Crawling → Preprocess-ing → Keyword Extraction → Ranking → Exception-Handling → Evaluation

2. Summary

Process of building Age Estimation System

1.	We collect data by age in Nate Pann.	**Web Crawling**
2.	We use the KoNLPy library to perform preprocessing process that extracts only Korean nouns.	**Preprocessing**
3.	We extract keywords that represent each age group using TF and IDF.	**Keyword Extraction**
4.	We calculate the cosine similarity using VSM, and select the vector of the document which is close to the vector of query.	**Ranking**
5.	Also, we process words that are not in the Indexing list.	**Exception Handling**
6.	We calculate Precision of test-dataset.	**Evaluation**

02

Schedule

3. Schedule

Mon	Tue	Wed	Thu	Fri	Sat	Sun
5/13	14	15	16	17	18 First Presentation	19
20	21	22 Meeting New Idea	23	24 New Subject Web Crawling	25	26
27 Preprocessing	28	29 Meeting Keyword Extraction	30	31 Ranking	6/1	2
3 Merge	4	5 Meeting	6 Exception Handling	7	8	9
10 Evaluation	11	12	13	14	15 Final Presentation	

3. Role

Age Estimation System

Web Crawling	Preprocess-ing	Keyword Extraction	Ranking	Exception-Handling	Evaluation
Rohee	🟩				🟩
	Minjoon			🟪	🟪
		Inseok	🟦		🟦

03

Process

1. Web Crawling

Process of building Age Estimation System

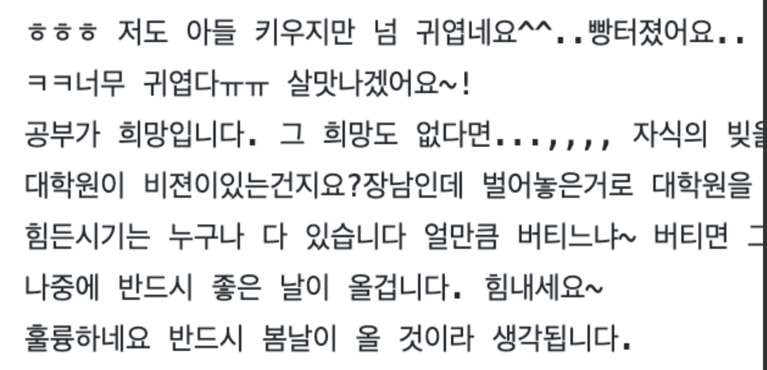
1.	**We collect data by age in Nate Pann.**	**Web Crawling**
2.	We use the KoNLPy library to perform preprocessing process that extracts only Korean nouns.	**Preprocessing**
3.	We extract keywords that represent each age group using TF and IDF.	**Keyword Extraction**
4.	We calculate the cosine similarity using VSM, and select the vector of the document which is close to the vector of query.	**Ranking**
5.	Also, we process words that are not in the Indexing list.	**Exception Handling**
6.	We calculate Precision of test-dataset.	**Evaluation**

1. Web Crawling

- We collect data from Nate Pann
- Use python package bs4, selenium
- Result : 10.txt, 20.txt, 30.txt, 40.txt, 50.txt

10대 이야기
20대 이야기
30대 이야기
40대 이야기
50대 이야기

Nate Pann Board



ㅎㅎㅎ 저도 아들 키우지만 넘 귀엽네요^^..빵터졌어요..
ㅋㅋ너무 귀엽다ㅠㅠ 살맛나겠어요~!
공부가 희망입니다. 그 희망도 없다면...,,,, 자식의 빚
대학원이 비젼이있는건지요?장남인데 벌어놓은거로 대학원을
힘든시기는 누구나 다 있습니다 얼만큼 버티느냐~ 버티면
나중에 반드시 좋은 날이 올겁니다. 힘내세요~
훌륭하네요 반드시 봄날이 올 것이라 생각됩니다.

Result Example – 40대

2. Preprocessing

Process of building Age Estimation System

1.	We collect data by age in Nate Pann.	Web Crawling
2.	**We use the KoNLPy library to perform preprocessing process that extracts only Korean nouns.**	**Preprocessing**
3.	We extract keywords that represent each age group using TF and IDF.	Keyword Extraction
4.	We calculate the cosine similarity using VSM, and select the vector of the document which is close to the vector of query.	Ranking
5.	Also, we process words that are not in the Indexing list.	Exception Handling
6.	We calculate Precision of test-dataset.	Evaluation

2. Preprocessing

Twitter – KoNLPy

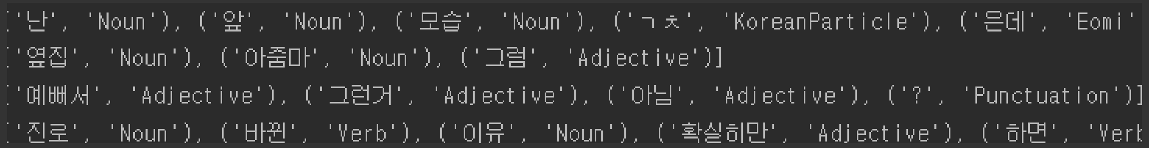
Python package

1. POS tagging

Ex) Noun, Josa, Verb, Number, KoreanParticle, Eomi ..

2. Extract necessary word

Noun + KoreanParticle



```
('난', 'Noun'), ('앞', 'Noun'), ('모습', 'Noun'), ('ㄱㅊ', 'KoreanParticle'), ('은데', 'Eomi'
('옆집', 'Noun'), ('아줌마', 'Noun'), ('그럼', 'Adjective')]
('예뻐서', 'Adjective'), ('그런거', 'Adjective'), ('아님', 'Adjective'), ('?', 'Punctuation')]
('진로', 'Noun'), ('바뀐', 'Verb'), ('이유', 'Noun'), ('확실히만', 'Adjective'), ('하면', 'Verk
```

3. Word processing by age group

3. Keyword Extraction

Process of building Age Estimation System

1.	We collect data by age in Nate Pann.	Web Crawling
2.	We use the KoNLPy library to perform preprocessing process that extracts only Korean nouns.	Preprocessing
3.	**We extract keywords that represent each age group using TF and IDF.**	**Keyword Extraction**
4.	We calculate the cosine similarity using VSM, and select the vector of the document which is close to the vector of query.	Ranking
5.	Also, we process words that are not in the Indexing list.	Exception Handling
6.	We calculate Precision of test-dataset.	Evaluation

3. Keyword Extraction

Indexing

TF

IDF

Weight

Keyword

3. Keyword Extraction

Indexing

TF

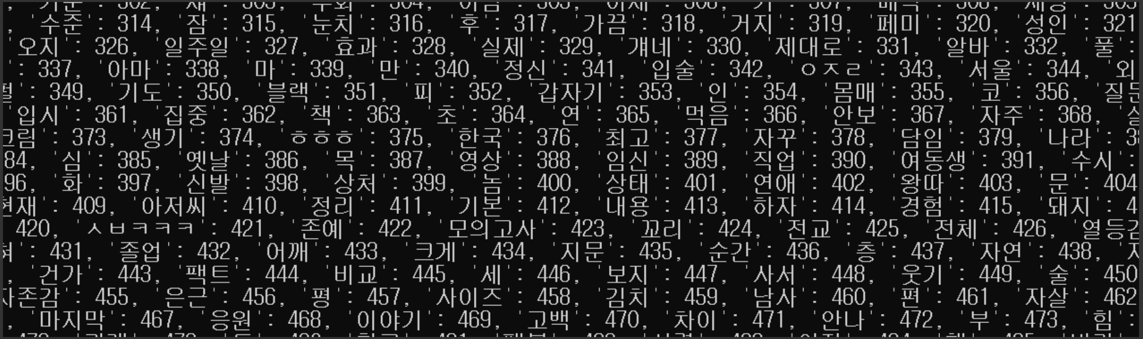
IDF

Weight

Keyword

Indexing_list [Dictionary]
= { word0: index0, word1: index1, word2: index2, … }

It contains all words of dataset and assign unique number to each word.



```
, '수준': 314, '잠': 315, '눈치': 316, '후': 317, '가끔': 318, '거지': 319, '페미': 320, '성인': 321
'오지': 326, '일주일': 327, '효과': 328, '실제': 329, '개네': 330, '제대로': 331, '알바': 332, '풀':
': 337, '아마': 338, '마': 339, '만': 340, '정신': 341, '입술': 342, 'ㅇㅈㄹ': 343, '서울': 344, '외
털': 349, '기도': 350, '블랙': 351, '피': 352, '갑자기': 353, '인': 354, '몸매': 355, '코': 356, '질문
'입시': 361, '집중': 362, '책': 363, '초': 364, '연': 365, '먹음': 366, '안보': 367, '자주': 368, '실
크림': 373, '생기': 374, 'ㅎㅎㅎ': 375, '한국': 376, '최고': 377, '자꾸': 378, '담임': 379, '나라': 3
84, '심': 385, '옛날': 386, '목': 387, '영상': 388, '임신': 389, '직업': 390, '여동생': 391, '수시':
96, '화': 397, '신발': 398, '상처': 399, '놈': 400, '상태': 401, '연애': 402, '왕따': 403, '문': 404
현재': 409, '아저씨': 410, '정리': 411, '기본': 412, '내용': 413, '하자': 414, '경험': 415, '돼지': 4
 420, 'ㅅㅂㅋㅋㅋ': 421, '존예': 422, '모의고사': 423, '꼬리': 424, '전교': 425, '전체': 426, '열등감
혀': 431, '졸업': 432, '어깨': 433, '크게': 434, '지문': 435, '순간': 436, '층': 437, '자연': 438, '자
, '건가': 443, '팩트': 444, '비교': 445, '세': 446, '보지': 447, '사서': 448, '웃기': 449, '술': 450
자존감': 455, '은근': 456, '평': 457, '사이즈': 458, '김치': 459, '남사': 460, '편': 461, '자살': 462
, '마지막': 467, '응원': 468, '이야기': 469, '고백': 470, '차이': 471, '안나': 472, '부': 473, '힘':
```

3. Keyword Extraction

Indexing

TF

IDF

Weight

Keyword

tf_list [List]
= [[(size of indexing_list)], [ ], [ ], [ ], [ ]]

TF = The number of times the word corresponding to the index appeared in the document / the highest number of each vector

$$\boldsymbol{tf_{ij}} = f_{ij} \,/\, max_i\{f_{ij}\}$$

3. Keyword Extraction

Indexing

TF

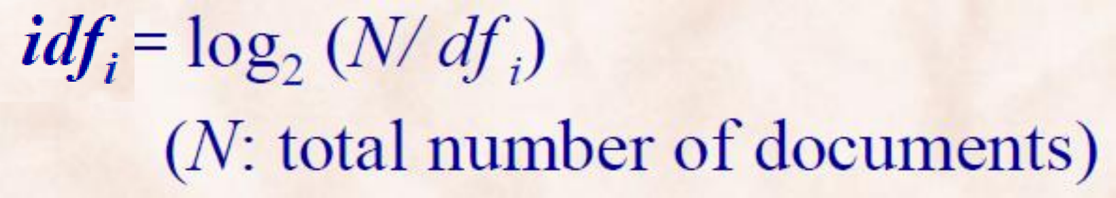
IDF

Weight

Keyword

idf_list [List]
= [(size of indexing_list)]

IDF = whole number of documents / The number of documents in which the word corresponding to the index appeared



$$idf_i = \log_2 (N/ df_i)$$

(N: total number of documents)

3. Keyword Extraction

Indexing

TF

IDF

Weight

Keyword

weighting_list [List]
= [[(size of indexing_list)], [ ], [ ], [ ], [ ]]

Weight = TF * IDF

3. Keyword Extraction

Indexing

TF

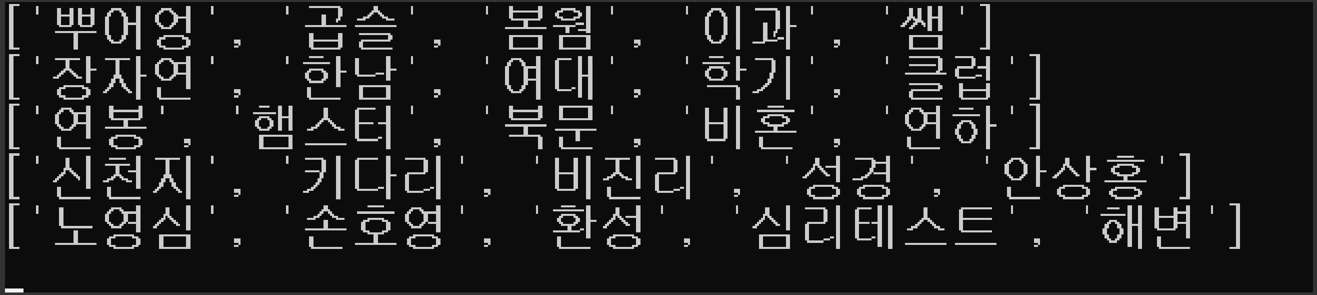
IDF

Weight

Keyword

keyword_list [List]
= [[(top K keyword of each age)], [ ], [ ], [ ], [ ]]

Keyword = top K(number) in order of Weight by age



```
['뿌어엉', '곱슬', '봄웜', '이과', '쌤']
['장자연', '한남', '여대', '학기', '클럽']
['연봉', '햄스터', '북문', '비혼', '연하']
['신천지', '키다리', '비진리', '성경', '안상홍']
['노영심', '손호영', '환성', '심리테스트', '해변']
```

4. Ranking

Process of building Age Estimation System

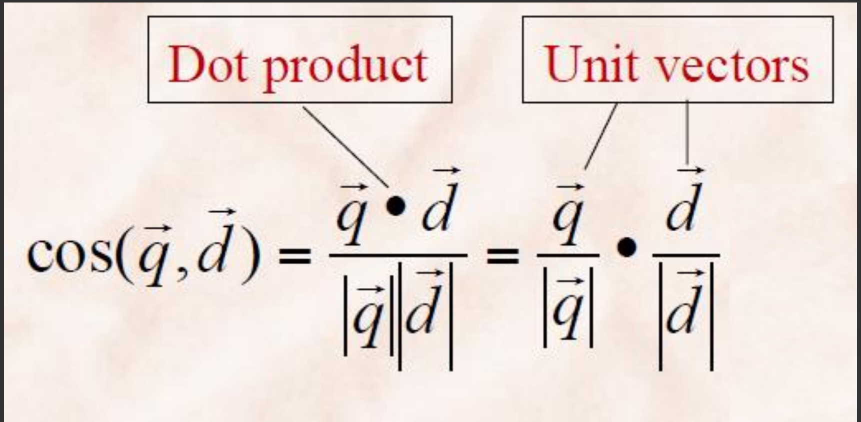
1.	We collect data by age in Nate Pann.	Web Crawling
2.	We use the KoNLPy library to perform preprocessing process that extracts only Korean nouns.	Preprocessing
3.	We extract keywords that represent each age group using TF and IDF.	Keyword Extraction
4.	**We calculate the cosine similarity using VSM, and select the vector of the document which is close to the vector of query.**	**Ranking**
5.	Also, we process words that are not in the Indexing list.	Exception Handling
6.	We calculate Precision of test-dataset.	Evaluation

4. Ranking

Normalize

Similarity

Ranking



Dot product

Unit vectors

$$\cos(\vec{q},\vec{d}) = \frac{\vec{q} \bullet \vec{d}}{|\vec{q}||\vec{d}|} = \frac{\vec{q}}{|\vec{q}|} \bullet \frac{\vec{d}}{|\vec{d}|}$$

5. Exception Handling

Process of building Age Estimation System

1.	We collect data by age in Nate Pann.	Web Crawling
2.	We use the KoNLPy library to perform preprocessing process that extracts only Korean nouns.	Preprocessing
3.	We extract keywords that represent each age group using TF and IDF.	Keyword Extraction
4.	We calculate the cosine similarity using VSM, and select the vector of the document which is close to the vector of query.	Ranking
5.	**Also, we process words that are not in the Indexing list.**	**Exception Handling**
6.	We calculate Precision of test-dataset.	Evaluation

5. Exception Handling

If the comment include new word?

Word decomposition

감 → ‘ㄱ’‘ㅏ’‘ㅁ’

5. Exception Handling

심쿵
심쿵

대마초
대마도

움짤
움찔

글렷다
글렀어

살랭
살랑

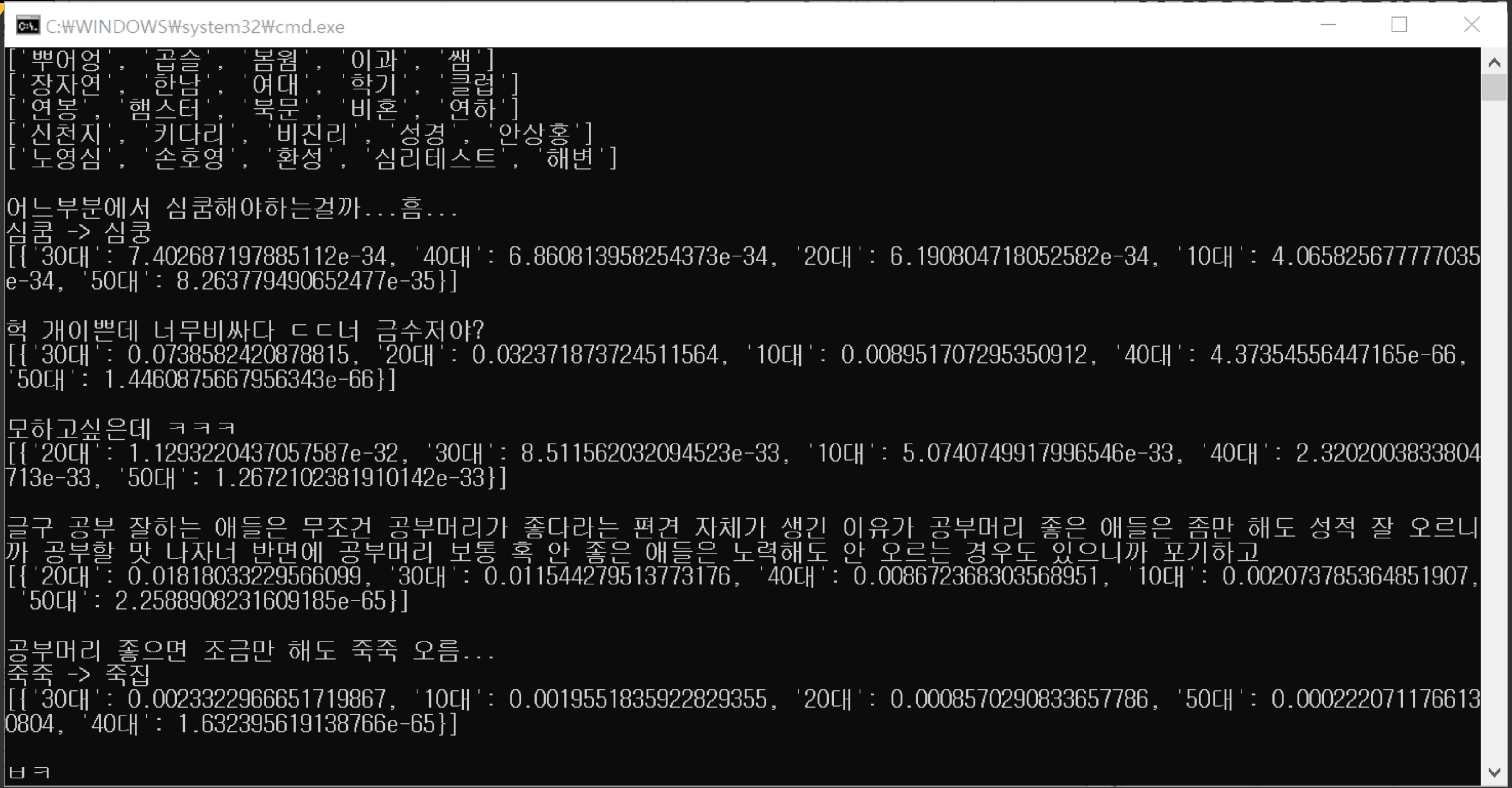
죽죽
죽집

Replacing with words which is similar to the words in the Indexing list.

04

Result & Evaluation

4-1. Result



```
['뿌어엉', '곱슬', '봄웜', '이과', '쌤']
['장자연', '한남', '여대', '학기', '클럽']
['연봉', '햄스터', '북문', '비혼', '연하']
['신천지', '키다리', '비진리', '성경', '안상홍']
['노영심', '손호영', '환성', '심리테스트', '해변']

어느부분에서 심쿵해야하는걸까...흠...
심쿵 -> 심쿵
[{'30대': 7.402687197885112e-34, '40대': 6.860813958254373e-34, '20대': 6.190804718052582e-34, '10대': 4.065825677777035
e-34, '50대': 8.263779490652477e-35}]

헉 개이쁜데 너무비싸다 ㄷㄷ너 금수저야?
[{'30대': 0.0738582420878815, '20대': 0.032371873724511564, '10대': 0.008951707295350912, '40대': 4.37354556447165e-66,
'50대': 1.4460875667956343e-66}]

모하고싶은데 ㅋㅋㅋ
[{'20대': 1.1293220437057587e-32, '30대': 8.511562032094523e-33, '10대': 5.0740749917996546e-33, '40대': 2.3202003833804
713e-33, '50대': 1.2672102381910142e-33}]

글구 공부 잘하는 애들은 무조건 공부머리가 좋다라는 편견 자체가 생긴 이유가 공부머리 좋은 애들은 좀만 해도 성적 잘 오르니
까 공부할 맛 나자너 반면에 공부머리 보통 혹 안 좋은 애들은 노력해도 안 오르는 경우도 있으니까 포기하고
[{'20대': 0.01818033229566099, '30대': 0.011544279513773176, '40대': 0.008672368303568951, '10대': 0.002073785364851907,
 '50대': 2.2588908231609185e-65}]

공부머리 좋으면 조금만 해도 죽죽 오름...
죽죽 -> 죽집
[{'30대': 0.0023322966651719867, '10대': 0.0019551835922829355, '20대': 0.0008570290833657786, '50대': 0.000222071176613
0804, '40대': 1.6323956191387 66e-65}]


ㅂㅋ
```

4-2. Evaluation

Process of building Age Estimation System

1.	We collect data by age in Nate Pann.	**Web Crawling**
2.	We use the KoNLPy library to perform preprocessing process that extracts only Korean nouns.	**Preprocessing**
3.	We extract keywords that represent each age group using TF and IDF.	**Keyword Extraction**
4.	We calculate the cosine similarity using VSM, and select the vector of the document which is close to the vector of query.	**Ranking**
5.	Also, we process words that are not in the Indexing list.	**Exception Handling**
6.	**We calculate Precision of test-dataset.**	**Evaluation**

4-2. Evaluation

Precision – input_data

input/10.txt
전체 댓글 : 8713
결과 : 4834
정확도 : 0.554803

input/20.txt
전체 댓글 : 4974
결과 : 3318
정확도 : 0.667069

input/30.txt
전체 댓글 : 19829
결과 : 13429
정확도 : 0.677240

input/40.txt
전체 댓글 : 2612
결과 : 1589
정확도 : 0.608346

input/50.txt
전체 댓글 : 1727
결과 : 522
정확도 : 0.302258

input_data
총 댓글 : 37855
결과 : 23692
정확도 : 0.625861

4-2. Evaluation

Precision – test_data

test_data/test_10.txt
전체 댓글 : 479
결과 : 155
정확도 : 0.323591

test_data/test_20.txt
전체 댓글 : 58
결과 : 25
정확도 : 0.431034

test_data/test_30.txt
전체 댓글 : 280
결과 : 127
정확도 : 0.453571

test_data/test_40.txt
전체 댓글 : 29
결과 : 10
정확도 : 0.344828

test_data/test_50.txt
전체 댓글 : 80
결과 : 26
정확도 : 0.325000

test_data
총 댓글 : 926
결과 : 343
정확도 : 0.37041036

05

Conclusion

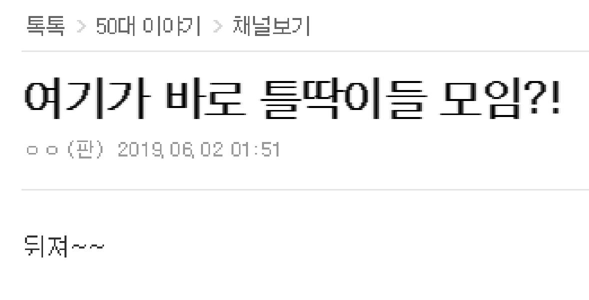
5. Conclusion

Unfortunately..<Dataset>

1. Dataset don’t reflect common attributes because they are made up of data from specific group.

2. The data for 10, 40, and 50 are too low compared to the 20s and 30s.

3. Some people use a board that does not fit their age.



톡톡 > 50대 이야기 > 채널보기

여기가 바로 틀딱이들 모임?!

ㅇㅇ(판) 2019.06.02 01:51

뒤져~~

5. Conclusion

Unfortunately..<Tech>

1. Problems with finding similar words

The composition of a vowel or consonant is similar to a word,
but it is small in terms of finding similar words.

2. Preprocessing Problem

It often happens that words are separated and preprocessed
in a form that does not preserve the meaning of words.

5. Conclusion

We could find little data on the Internet about our subject.
One of the papers used deep-learning, which made it difficult to access.

Therefore, it is very valuable that we select unique subject,
and apply Vector Space Model (which we learned from lecture) to real project.

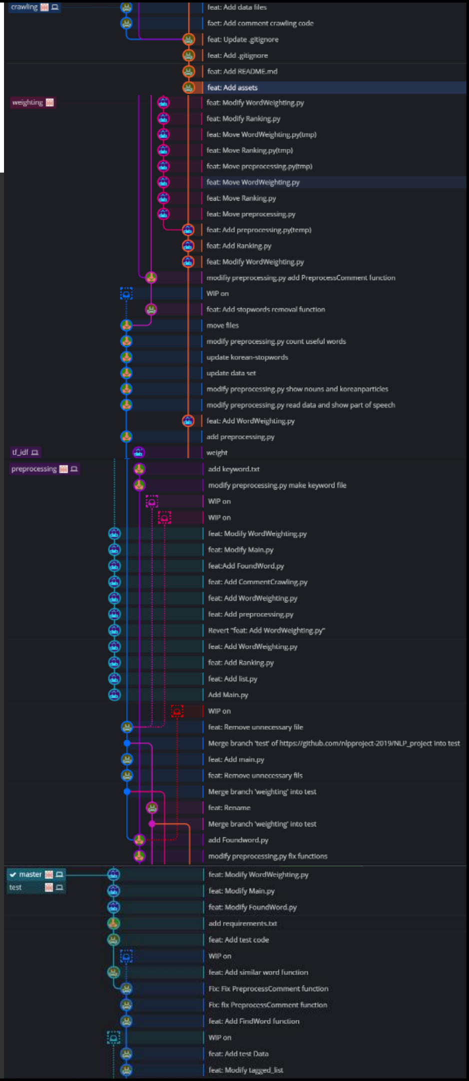
5. Conclusion

We conducted the project based on Korean language that was familiar to us.
Because the Korean dataset is significantly less than that of English,
I think it is one of the unique aspects of our project
that we collect the necessary parts directly through crawling.

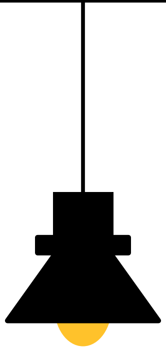
Also, I was worried about how to handle words
that were not in the indexing list when they came into the query.
Finally we apply solutions in terms of morphology.

5. Github

https://github.com/nlpproject-2019/NLP_project.git



```
feat: Add data files
feat: Add comment crawling code
feat: Update .gitignore
feat: Add .gitignore
feat: Add README.md
feat: Add assets
feat: Modify WordWeighting.py
feat: Modify Ranking.py
feat: Move WordWeighting.py(tmp)
feat: Move Ranking.py(tmp)
feat: Move preprocessing.py(tmp)
feat: Move WordWeighting.py
feat: Move Ranking.py
feat: Move preprocessing.py
feat: Add preprocessing.py(temp)
feat: Add Ranking.py
feat: Modify WordWeighting.py
modifiy preprocessing.py add PreprocessComment function
WIP on
feat: Add stopwords removal function
move files
modify preprocessing.py count useful words
update korean-stopwords
update data set
modify preprocessing.py show nouns and koreanparticles
modify preprocessing.py read data and show part of speech
feat: Add WordWeighting.py
add preprocessing.py
weight
add keyword.txt
modify preprocessing.py make keyword file
WIP on
WIP on
feat: Modify WordWeighting.py
feat: Modify Main.py
feat:Add FoundWord.py
feat: Add CommentCrawling.py
feat: Add WordWeighting.py
feat: Add preprocessing.py
Revert "feat: Add WordWeighting.py"
feat: Add WordWeighting.py
feat: Add Ranking.py
feat: Add list.py
Add Main.py
WIP on
feat: Remove unnecessary file
Merge branch 'test' of https://github.com/nlpproject-2019/NLP_project into test
feat: Add main.py
feat: Remove unnecessary fils
Merge branch 'weighting' into test
feat: Rename
Merge branch 'weighting' into test
add Foundword.py
modify preprocessing.py fix functions
feat: Modify WordWeighting.py
feat: Modify Main.py
feat: Modify FoundWord.py
add requirements.txt
feat: Add test code
WIP on
feat: Add similar word function
Fix: Fix PreprocessComment function
Fix: fix PreprocessComment function
feat: Add FindWord function
WIP on
feat: Add test Data
feat: Modify tagged_list
```



Thank you