이름 : 이창윤
github : https://github.com/Changyoon-Lee

이메일: lckddbsl@gmail.com
전화번호: 010-4039-1940

개요

- 한양대학교 기계공학부 전공(2019.02 졸업)
- Python 개발, 머신러닝, 딥러닝, 자연어처리
- SK하이닉스 양산기술 정규직 근무(2019.01~2020.03)

사용 가능 기술

- 프로그래밍 언어 : Python3
- 데이터베이스 : MySQL
- 분석 툴 : sklearn, Tensorflow, Pytorch, matlab
- 협업 툴 : github, slack, notion

관련 교육/활동

• 혁신성장 청년인재 집중양성 인공지능 자연어처리(NLP) 기반 기업데이터 분석과정 수료	멀티캠퍼스, 강남 2020.05~2020.11
• 한국데이터진흥원 주관 2020 빅콘테스트 공모전 최우수상(sk텔레콤상) 수상	2020. 12. 15
• SQL 개발자(SQLD) 자격증	2020. 12. 29

관련 경험

1. 텍스트 마이닝을 활용한 금융통화위원회 의사록 분석	20.07.10~20.08.03
2. CNN 을 이용한 네이버 영화리뷰 데이터 감성분석	20.08.05~20.08.11
3. Attention+seq2seq을 이용한 한영 번역 구현	20.08.13~20.08.19
4. Big contest 공모전	20.08.23~20.09.28
5. 비지도 학습을 통한 영한 번역 구현	20.10.01~20.10.22

관련 경험

https://github.com/Changyoon-Lee/BOK_project

1. 텍스트 마이닝을 활용한 금융통화위원회 의사록 분석

20.07.10~20.08.03

Deciphering Monetary Policy Board Minutes through Text Mining Approach 논문 리뷰 및 구현

개요

한국은행에서 발간하는 '한국은행 금융통화위원회(이하 금통위) 의사록'에는 중앙은행이 시행하는 정책의 방향과 현 경제 상황에 대한 중앙은행의 판단이 포함되어 있다. 그러나 금통위 의사록의 문제는 매우 절제되어 있기 때문에 일반전인 독해로는 명확한 의미를 파악하는 것이 불가능하다. 따라서 텍스트 마이닝을 활용하여 금통위 의사록에 담겨있는 어조를 추출하여 수치화하고, 기준 금리의 변동과 얼마나 유사한지를 살펴보면서 지수의 설명력과 예측력을 검증한다.

진행과정

1. 금리관련 뉴스, 채권분석리포트, 금융통화위원회 의사록, 콜금리, 금리(2005~2017의 데이터를 크롤링을 이용하여 수집
2. 각각의 text데이터를 전처리하고, 콜금리 값을 기준으로 30일 전에 비해 상승했으면 hawkish 하락했으면 dovish로 문서들을 라벨링
3. 텍스트 문서들을 문장단위로 나누고 토큰화, ngram화 시킨다(ekonlpy 사용)
4. 각 토큰, ngram 들의 hawkish or dovish문서에 등장한 횟수를 모두 counting 하여 경향성 판단
5. 의사록 문서(토큰, ngram)의 경향성 수치화(-1~1의 값)
6. 실제 금리 변동과 상관분석

관련 경험

1. 텍스트 마이닝을 활용한 금융통화위원회 의사록 분석

역할

- 팀장(전체 flow 계획 및 역할 분담)
- Scrapy(frame work)라는 crawling tool을 이용해 20만개의 금리관련 뉴스기사 crawling
- 전처리 작업 : 정규표현식을 이용하여 수집된 자료 특수문자, 불필요한 내용(ex 뉴스기자 이름, email등) 제거
  ekonlpy 이용하여 tokenize, ngramize 진행
- 모든 수집된 데이터 통합/ 전월대비 콜금리 변동량을 통해 텍스트 문서들의 경향성(hawkish/dovish) 라벨링

	body	ngramized	up_down
time			
2005-05-01	[이른바 은행들의 전쟁 원년인 올해 4대 시중은행의 초반 전적표가 나왔다, 1일 금...	[[수익성/NNG;향상/NNG, 최대/NNG], [외형/NNG;성장/NNG], [자...	상승
2005-05-02	[조정을 거친 한진해운 이 반등에성공할 수 있을까 한진해운 주가는 연초 천원대에서 ...	[[미/NNG;경기/NNG;둔화/NNG;우려/NNG, 성공/NNG, 절상/NNG],...	상승
2005-05-03	[환율이 소폭 상승했다, 3일 서울 외환시장에서 원달러 환율은 전날 종가대비 달러...	[[저점/NNG;경계/NNG], [fed/NNG;ff/NNG;금리/NNG;인상/NN...	상승
2005-05-04	[미국 연방준비제도이사회(FRB)가3일(현지시간) 연방기금 금리를 5 포인트 인상하...	[[인상/NNG], [fed/NNG;금리/NNG;인상/NNG], [인플레이션/NNG...	하락
2005-05-05	[국내외 증시가 그동안의 약세에서 벗어나 오름세를 나타내면서 상승 추세로의 복귀 가...	[[기대/NNG;높/VA], [금리/NNG;인상/NNG;우려/NNG], [세계/NN...	하락

- 전체적인 알고리즘 pipeline 코딩
- 라벨링 된 문서에 나타난 토큰들의 빈도 수를 통해 token, ngram의 경향성 분류(hawkish/dovish)

관련 경험

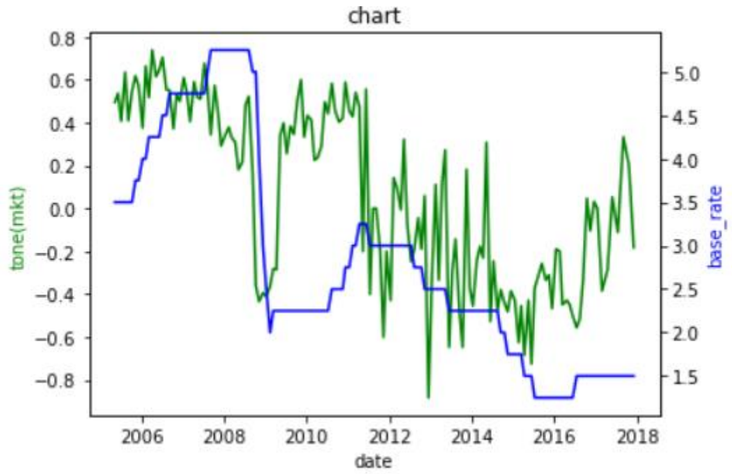
1. 텍스트 마이닝을 활용한 금융통화위원회 의사록 분석

역할

- 의사록의 문장&문서의 경향성 수치화

$$tone_s^{\square} = \frac{No.of\ hawkish\ features - No.of\ dovish\ features}{No.of\ hawkish\ features + No.of\ dovish\ features}$$

- 실제금리와 corr계산



결과 및 알게 된 점

- Corr 계산결과 0.61 정도로 상당한 설명력을 가진다고 볼 수 있었다
- Framework(Scrapy) 를 사용 목적에 맞게 직접 수정하여 사용 해 보았다
- EkonIpy를 이용하여 경제 corpus에 특화된 token사전을 이용해 분석해보았고, 통계적 방식의 텍스트 마이닝을 이용하여 유용한 정보를 추출하는 구체적인 방식에 대해 깊게 알 수 있었다.

관련 경험

https://github.com/Changyoon-Lee/CNN_Project

2. CNN 을 이용한 네이버 영화리뷰 데이터 감성분석

20.08.05~20.08.11

Convolution Neural Networks for Sentence Classification 논문 리뷰 및 참고

개요

CNN 모델은 컴퓨터 비전을 위해 고안 되었지만, 자연어 처리에 대해서도 효과적임을 보인다
네이버영화리뷰 training set으로 Fasttext, word2vec, Contextualized Embedding 등의 Token embedding 방식 차이에 따라 성능을 비교해 보았다. 추가적으로 RNN모델과도 비교해 보았다.

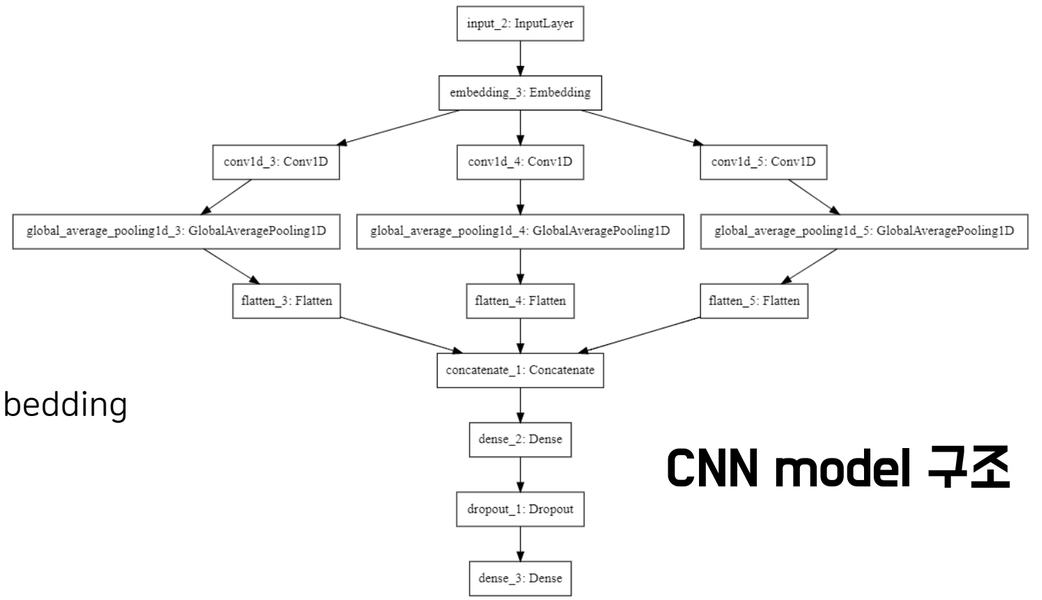
사용 데이터 set : 네이버 영화리뷰 데이터(https://github.com/e9t/nsmc/)

임베딩 방식에 차이를 두어 비교분석

Model1 : Fasttext
Model2 : 네이버영화리뷰의 training set으로 하여 기존의 word2vec을 이용한 word embedding
Model3 : Contextualized Embedding

- Static word Embeding : Fasttext, Word2Vec
  단어 단위, CBoW, Skip-gram의 방식으로 pre trained 되어야함

- Contextualized Word Embedding
  문장을 입력받아 각 단어에 대한 representation을 산출해준다.
  문맥에 따른 단어의 의미를 잡아 낼 수 있음



CNN model 구조

관련 경험

2. CNN 을 이용한 네이버 영화리뷰 데이터 감성분석

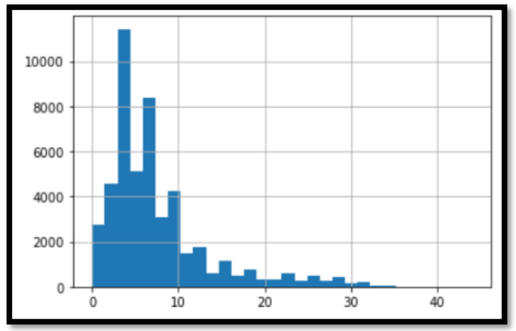
역할

Fastext를 이용하는 model1을 맡아 아래의 전체적 flow를 진행하였다

1. 중복데이터, null 값 제거
2. 영문,한글 이외 값 제거
3. Konlpy 패키지의 Okt이용하여 토큰화 (Noun, Adjective, Alpha값만 취급)

```
def preprocessing(data):
    data.drop_duplicates(subset=['document'], inplace=True)
    data = data.dropna(how = 'any')
    data['document'] = data['document'].str.replace("[^ㄱ-ㅎㅏ-ㅣ가-힣 ]","")
    data['document'].replace('', np.nan, inplace=True)
    data = data.dropna(how = 'any')
    sentences = data['document'].tolist()
    label = data['label']
    print('data len = {}'.format(len(sentences)))
    return sentences, label
```

4. 문장의 길이의 분포를 확인하여 max length 설정
5. 토큰화, OOV 처리, padding 진행



6. Fasttext의 wordvector 파일용량 크기가 4G이상으로 컸기 때문에 num_word의 크기의 맞추어 필요 word에 대해서만 따로 저장하여 사용하였다.
7. 모델 학습 및 model2, model3코드 통일화, 필요함수 module화 통합 된 결과 분석 진행

```
simple_ko_vec = {}
oov_list=[]
simple_ko_vec['<PAD/>'] = ko_model.wv.word_vec('<PAD/>')
for sent in token_train_data+token_test_data:
    for token in sent:
        try:
            simple_ko_vec[token] = ko_model.wv.word_vec(token)
        except:
            oov_list.append(token)
oov_list= list(set(oov_list))
```

\+ RNN 모델 기반 분석 추가 진행
https://github.com/Changyoon-Lee/CNN_Project/blob/master/rnnmodel_main.py

관련 경험

2. CNN 을 이용한 네이버 영화리뷰 데이터 감성분석

결과 분석 및 알게된 점

model1	model2	model3
0.8147	0.8209	0.8303

- Model3(contextualized word embedding) 방식이 test set에 대해 더 좋은 결과를 보였다.
- Model1,2 방식은 사전에 학습된 wordvector가 필요한 반면 model3의 방식은 그렇지 않다.
- 다른 영화리뷰 platform의 데이터에 대해서도 test를 하였을 시 model1 과 model3는 0.8에 근접하여 좋은 결과를 보였는데 fasttext 관련성이 떨어지는 Wikipedia 문서를 기반으로 학습되었지만 좋은 결과를 보여 더 인상이 깊었다.
- 도메인 별 사용되는 단어나 문맥의 형태가 다르므로 각각의 영역에서 좋은 성능을 보이기에는 3번 방식이 훨씬 더 효율적이라고 볼 수 있다.

관련 경험

https://github.com/Changyoon-Lee/seq2seq_project

3. Attention+seq2seq을 이용한 한영 번역 구현

20.08.13~20.08.19

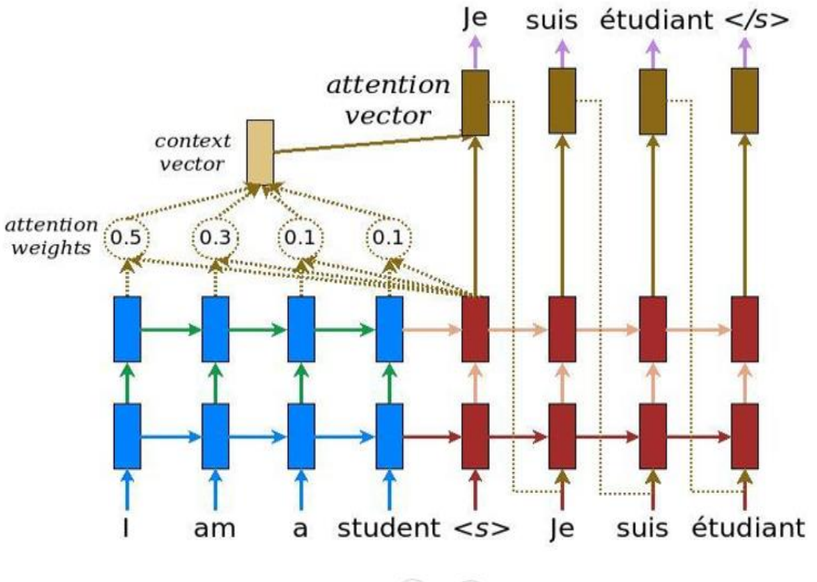
Sequence to Sequence Learning with Neural Networks 논문 리뷰 및 참고/ tensorflow 코드* 참고

개요

논문 및 사이트를 기반으로 attention 기법이 포함된 seq2seq 한영변환을 구현해보고 BLEU로 성능을 테스트 해보았다.
korpus는 aihub의 한국어-영어 번역 말뭉치*를 사용하였다.

논문리뷰

- 기존 RNN 모델에서 input 과 output 길이가 다를 때 적용이 어려움
- Input sequence를 RNN을 통해 고정된 길이의 context vector로 만든 후 다시 RNN에 넣음
- Input sequence와 output sequence에 대해 두개의 LSTM 사용
- 빈도 순으로 영어단어 16만개, 불어 8만개의 단어 사용(논문에선 EN-FR dataset 사용)
- 기존 SMT보다 보편적으로 우수한 성능을 보이나, 문장의 길이가 길어지거나 자주 언급되지 않는 단어 등장하는 경우 성능이 급격히 저하됨 -> 이후 Attention 기법 등장



* tensorflow 참고자료(https://www.tensorflow.org/tutorials/text/nmt_with_attention)
* 한국어-영어 번역 말뭉치(https://www.aihub.or.kr/aidata/87)

관련 경험

3. Attention+seq2seq을 이용한 한영 번역 구현

역할

알고리즘 코딩 및 결과분석 진행

1. data 불러와서 train, val, test set으로 분리
2. Preprocessing

   - 문장 앞뒤로 start, end tag 추가, 영문/한글 각각 토큰화 진행
3. Keras tokenize 이용하여 train set fitting
4. Parameter 설정 및 model 훈련
5. Blue 스코어 계산

```
def preprocess_sentences_eng(sentences):
    sentence = []
    for line in sentences:
        w = '<start> '+line+' <end>'
        sentence.append(w)
    return sentence
def preprocess_sentences_kor(sentences):
    okt = Okt()
    sentence = []
    for line in sentences:
        tokens = [i[0] for i in okt.pos(line)]
        tokens.insert(0,'<end>')
        tokens.append('<start>')
        sentence.append(' '.join(tokens[::-1]))
    return sentence
```

```
from nltk.translate.bleu_score import sentence_bleu, SmoothingFunction
score_list = []
smoothie = SmoothingFunction().method4
for i in range(len(test_kor.values)):

    reference = [test_en.values[i].split()]
    result,_,_= evaluate([test_kor.values[i]])
    candidate = result.capitalize().split()
    score = sentence_bleu(reference, candidate, smoothing_function=smoothie)
    score_list.append(score)
    if i<20:
        print('실제값 : {}\n예측값 : {}'.format(reference, candidate))

score = sum(score_list)/len(score_list)
print('score는 {}'.format(score))
```

관련 경험

3. Attention+seq2seq을 이용한 한영 번역 구현

결과분석 내용

Score 를 높이기 위한 여러 시도 및 결과 비교

1. Hidden state initial 방식: 단순 영행렬로 한 것 보다 균일분포의 랜덤값으로 초기화 했을 때 성능이 좋았음
2. 한글 전처리방식(Okt/Mecab)에 차이에 따라 vocab size, 토큰화된 문장의 length에 차이가 컸으며, 성능에도 영향이 큼
3. 영문 전처리방식(nltk)를 이용함에 따라 성능이 크게 향상되었음
4. Input 문장의 입력 순서(정방향/역방향)를 다르게하여 다양한 상황에서 실험했을 시 모든경우에서 역방향의 경우가 score가 높았음

	Okt(전처리)	Okt	Mecab
score	0.1921	0.1856	0.2012
vocab size	28341	28361	17827
maxlen	26	26	32

input 순서	정방향	역방향
score	0.1954	0.2012

	정방향_nltk_	역방향_nltk_
score	0.2299	0.2326

```python
def initialize_hidden_state(self):
    return tf.zeros((self.batch_sz, self.enc_units))
```

```python
def initialize_hidden_state(self):
    return tf.random.uniform(shape=(self.batch_sz, self.enc_units), minval=-0.08, maxval=0.08)
```

```
input : 저는 오늘 그 이유에 대해 말하고자해요.
실제값 : and i 'm here to explain the reason for this .
예측값 : today , i want to talk about that professor today .
--------------------
input : 오늘 저는 성형수술의 위험성에 대해 말하고자해요.
실제값 : i want to say about the cosmetic sergery today .
예측값 : today , i want to talk about the importance of the truth of the future .
--------------------
input : 상기와 같은 조건으로 재계약을 하고자해요.
실제값 : we would like to renew the contract in the condition above .
예측값 : i want to sign the contract with the above above .
--------------------
input : 우리의 방학이 다음주에 시작 돼.
실제값 : our summer vacation starts next week .
예측값 : our vacation begins for next week .
--------------------
input : 그녀는 새로운 것을 창조해 내는 재밌는 작가야.
실제값 : she is an interesting writer who creates new things .
예측값 : she is a fun that i can enjoy the new zealand .
--------------------
input : 비오는 계절이 끝나자마자 시작돼요.
실제값 : it begins soon after the rainy season is over .
예측값 : taeyeon is completed as the weather season .
--------------------
```

실제 번역 된 결과

관련 경험

4. Big contest 공모전 20.08.23.~ 20.09.28

분야 : 감염병으로 인한 소비/경제/행동 변화, 사회적 영향 분석을 통해 뉴노멀시대의 서비스 아이디어 제시
주제 : 인구 밀집도 기반 카페/음식점 추천 서비스

역할

팀장 : 일정 계획 및 역할 분담, 중간발표 진행, 팀원 코드리뷰

- 신한카드 이용데이터를 이용해 코로나 전후 연령별 카테고리별 변화 분석
- 공공데이터 포털 서울시 상권데이터, 지하철 하차인원데이터를 이용하여 코로나 전후 동별 상권 수 변화량과 유동인구의 상관성 분석
- 지하철역 하차 인원수(유동인구) 예측 모델링
- Google map api이용하여 상권정보 및 이용자 데이터 수집 및 데이터 전처리
- 지도위에 특정구역 위 격자화를 하여 격자 별 통계계산(상권 수, 혼잡도)
- Folium 이용하여 지도 visualizing, 추천 장소 list up 구현

관련 경험

4. Big contest 공모전

데이터분석

이용 data : 노원구, 중구 카드매출데이터(신한)

전처리

- 행정동코드, 업종코드를 dict 형식으로 변환 후, 데이터 프레임에 mapping
- Pandas의 query, groupby method를 이용
- 각 동별 평균이용건수 50회 이상인 업종만 filtering

분석 내용

- 동별, 월별 전년대비 카드이용 변화율
- 업종별 변화율
- 나이대별 변화율

분석 결과

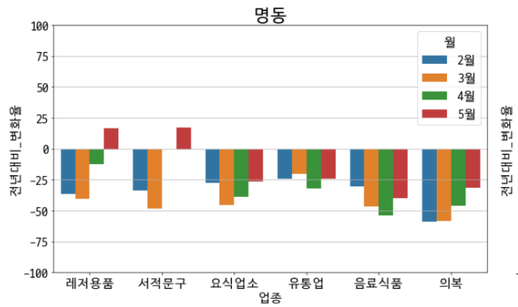
- 노원구, 중구의 데이터가 주어졌었는데 주거지역에 비해 주요상권지역(명동 필동 등)의 카드이용량 감소율이 컸다.
- 주요 이용업종은 요식업소, 유통업, 사무통신이었고 20대의 카드 이용량이 제일 많음

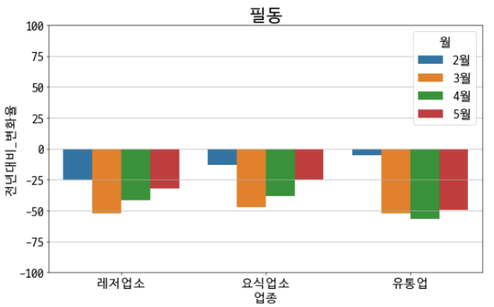
	기준일자	행정동코드	업종코드	성별코드	나이코드	이용건수	이용금액
구코드							
140	20190201	520	10	F	20	14	1038
140	20190201	520	10	F	25	59	4635
140	20190201	520	10	F	30	41	3046
140	20190201	520	10	F	35	59	5995
140	20190201	520	10	F	40	45	5172
...	...	...	...	...	...	...	...
350	20200531	720	81	M	45	19	127
350	20200531	720	81	M	50	33	211
350	20200531	720	81	M	55	9	61
350	20200531	720	81	M	60	19	194
350	20200531	720	81	M	65	5	29

구코드	행정동코드	구명	행정동명
140	520	서울 중구	소공동
140	540	서울 중구	회현동
140	550	서울 중구	명동
140	570	서울 중구	필동
140	580	서울 중구	장충동
140	590	서울 중구	광희동
140	605	서울 중구	을지로동
140	615	서울 중구	신당동
140	625	서울 중구	다산동
140	635	서울 중구	약수동
140	645	서울 중구	청구동
140	650	서울 중구	신당5동
140	665	서울 중구	동화동
140	670	서울 중구	황학동
140	680	서울 중구	중림동

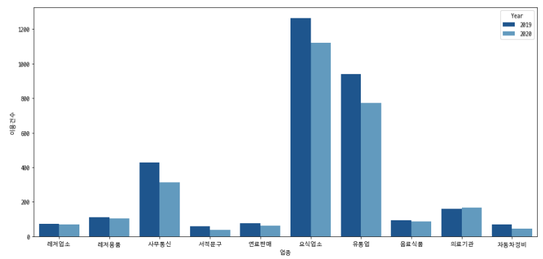
업종코드 (12개 분류)	세부 업종코드
숙박(10)	특급 호텔 1급 호텔 2급 호텔 콘도미니엄 모텔,여관,기타숙박
레저용품(20)	종합 테마파크 악기 피아노 대리점 운동경기,레저용품 레저스포츠 자전거
레저업소(21)	실내골프장 실외골프장

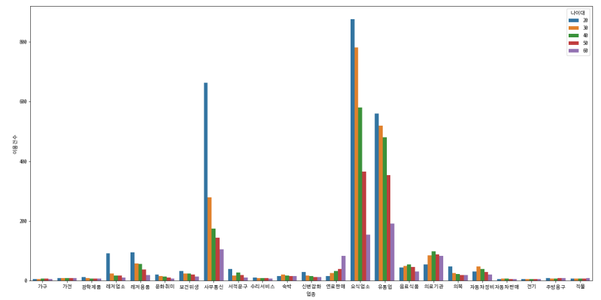
1428518 rows × 7 columns





업종별 이용건수 변화





관련 경험

4. Big contest 공모전

데이터분석

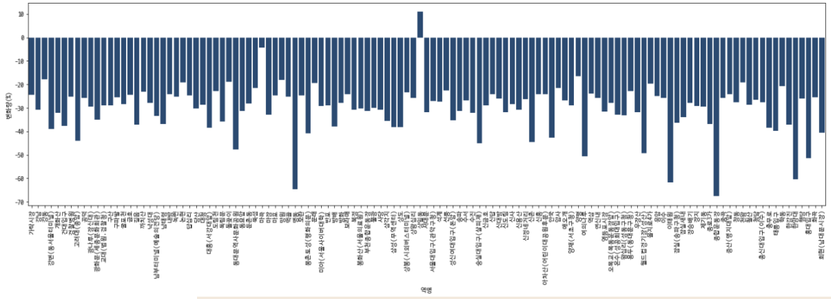
이용 data
서울시 상가업소정보 1,2분기(공공데이터 포털)
서울교통공사 연도별 일별 시간대별 역별 승하차인원(서울 열린데이터 광장)

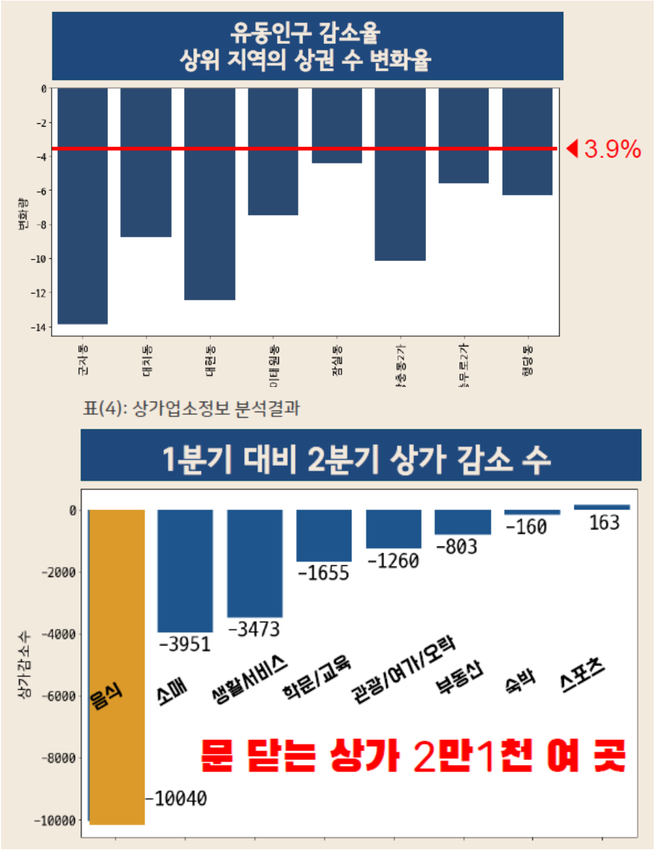
전처리
- stack unstack, 날짜 형식 변환, 역명이 변경된 부분 mapping 처리

분석내용
- 역별 전년대비 하차인원 변화량
- 감소율 상위지역의 상권 수 변화율
- 업종별 1분기 대비 2분기 상가수 변화

분석 결과
- 전반적으로 하차 인원 20%이상 감소 -> 유동량 감소
- 동별 상권수 변화율 평균 3.9%, 하차 인원 감소 상위지역의 상가피해 높음
- 1분기 대비 2분기 상가수 변화로 피해정도 확인

	date	호선	역번호	역명	구분	5-6	6-7	7-8	8-9	9-10	...	18-19	19-20	20-21	21-22	22-23	23-24	24-01	합 계	요일	공휴일
0	2019-01-01	1호선	150	서울	승차	348	321	348	741	940	...	3277	2898	2590	3004	2234	970	38	39559	화요일	0
1	2019-01-01	6호선	2630	녹사평(용산구청)	승차	171	117	58	81	104	...	272	212	238	239	188	72	0	3506	화요일	0
2	2019-01-01	6호선	2629	삼각지	하차	46	89	64	80	105	...	128	152	157	116	152	80	14	2411	화요일	0
3	2019-01-01	6호선	2629	삼각지	승차	48	68	51	80	110	...	269	150	125	100	74	50	0	2981	화요일	0
4	2019-01-01	6호선	2628	효창공원앞	하차	24	76	64	103	129	...	213	179	149	163	145	80	12	2533	화요일	0





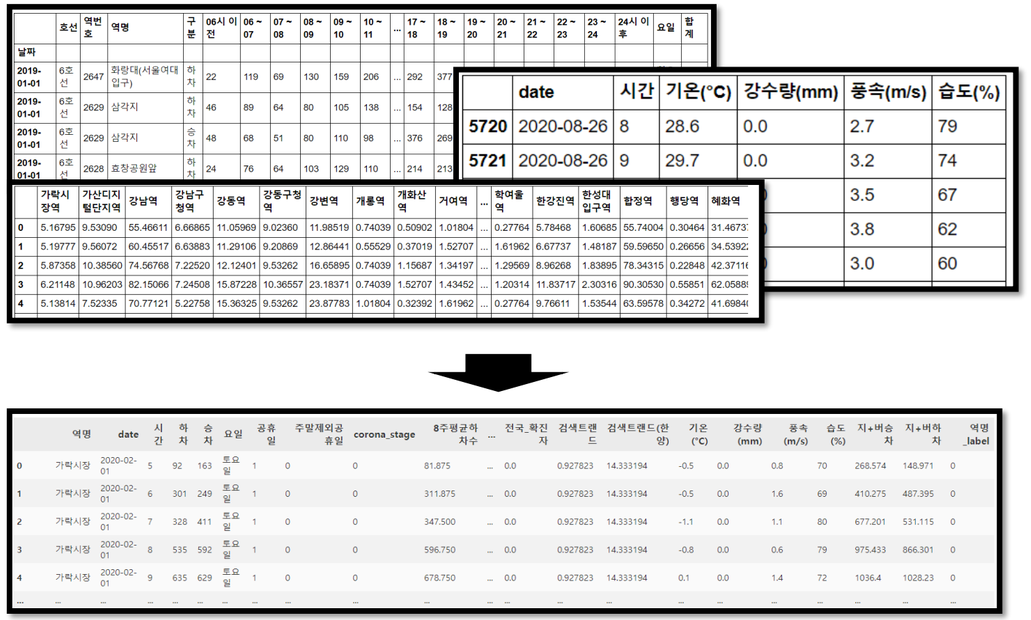
표(4): 상가업소정보 분석결과

관련 경험

4. Big contest 공모전

Data 취합

- 지하철 하차인원, 평균하차수 , 공휴일, 거리두기 단계, 역관련 검색어트랜드, 날씨정보 등
- 수집된 다양한 형태의 data는 column, index의 값, data의 형태, 형식들이 모두 달라 각각의 전처리 작업을 거쳐서 하나의 통합된 dataframe으로 역별 날짜별로 취합하였다



	호선	역번호	역명	구분	06시 이전	06 ~ 07	07 ~ 08	08 ~ 09	09 ~ 10	10 ~ 11	...	17 ~ 18	18 ~ 19	19 ~ 20	20 ~ 21	21 ~ 22	22 ~ 23	23 ~ 24	24시 이후	요일	합계
날짜																					
2019-01-01	6호선	2647	화랑대(서울여대입구)	하차	22	119	69	130	159	206	...	292	377								
2019-01-01	6호선	2629	삼각지	하차	46	89	64	80	105	138	...	154	128								
2019-01-01	6호선	2629	삼각지	승차	48	68	51	80	110	98	...	376	269								
2019-01-01	6호선	2628	효창공원앞	하차	24	76	64	103	129	110	...	214	213								

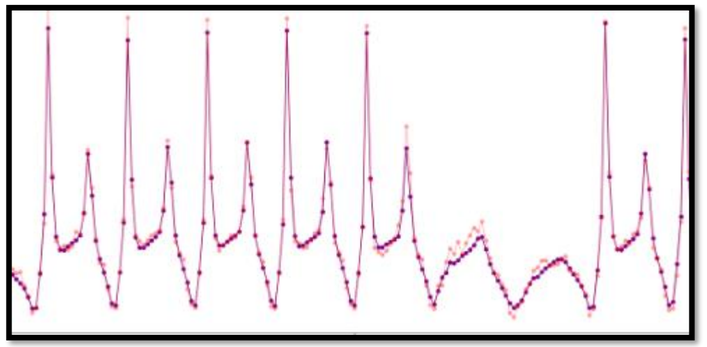
	date	시간	기온(°C)	강수량(mm)	풍속(m/s)	습도(%)
5720	2020-08-26	8	28.6	0.0	2.7	79
5721	2020-08-26	9	29.7	0.0	3.2	74
					3.5	67
					3.8	62
					3.0	60

	가락시장	가산디지털단지역	강남역	강남구청역	강동역	강동구청역	강변역	개롱역	개화산역	거여역	...	학여울역	한강진역	한성대입구역	합정역	행당역	혜화역
0	5.16795	9.53090	55.46611	6.66865	11.05969	9.02360	11.98519	0.74039	0.50902	1.01804	...	0.27764	5.78468	1.60685	55.74004	0.30464	31.4673
1	5.19777	9.56072	60.45517	6.63883	11.29106	9.20869	12.86441	0.55529	0.37019	1.52707	...	1.61962	6.67737	1.48187	59.59650	0.26656	34.5392
2	5.87358	10.38560	74.56768	7.22520	12.12401	9.53262	16.65895	0.74039	1.15687	1.34197	...	1.29569	8.96268	1.83895	78.34315	0.22848	42.3711
3	6.21148	10.96203	82.15066	7.24508	15.87228	10.36557	23.18371	0.74039	1.52707	1.43452	...	1.20314	11.83717	2.30316	90.30530	0.55851	62.0588
4	5.13814	7.52335	70.77121	5.22758	15.36325	9.53262	23.87783	1.01804	0.32392	1.61962	...	0.27764	9.76611	1.53544	63.59578	0.34272	41.6984

	역명	date	시간	하차	승차	요일	공휴일	주말제외공휴일	corona_stage	8주평균하차수	...	전국_확진자	검색트랜드	검색트랜드(한양)	기온(°C)	강수량(mm)	풍속(m/s)	습도(%)	지+버승차	지+버하차	역명_label
0	가락시장	2020-02-01	5	92	163	토요일	1	0	0	81.875	...	0.0	0.927823	14.333194	-0.5	0.0	0.8	70	268.574	148.971	0
1	가락시장	2020-02-01	6	301	249	토요일	1	0	0	311.875	...	0.0	0.927823	14.333194	-0.5	0.0	1.6	69	410.275	487.395	0
2	가락시장	2020-02-01	7	328	411	토요일	1	0	0	347.500	...	0.0	0.927823	14.333194	-1.1	0.0	1.1	80	677.201	531.115	0
3	가락시장	2020-02-01	8	535	592	토요일	1	0	0	596.750	...	0.0	0.927823	14.333194	-0.8	0.0	0.6	79	975.433	866.301	0
4	가락시장	2020-02-01	9	635	629	토요일	1	0	0	678.750	...	0.0	0.927823	14.333194	0.1	0.0	1.4	72	1036.4	1028.23	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...

예측 모델링

Linear regression, Random forest, LSTM 등 머신러닝,딥러닝을 이용하여 여러 feature 들을 통해 특정 요일 특정시간대의 하차인구수를 예측하는 모델링을 하였다. 코딩은 팀원에게 분배하여 진행 하였고, 추후 modeling, 구현 상의 오류들을 찾아내고 검수하면서 잘못된 점을 바로잡고 팀원들에게 피드백 하며 내용을 견고하게 할 수 있도록 하였다.



- 결과값으로 R2 score 이용하였는데 하차수 평균값을 이용한 방법의 미해 여러 feature를 이용한 방법의 score가 낮은 것을 보고 modeling 상의 오류가 있을 것으로 판단
- Scaling 과정에서 training set과 test set을 각각 전처리한 부분을 발견하여 수정
- 범주형 feature labeling 방식을 어떻게 할 것인가, feature importance에 따른 주요요소 필터링 방법에 대해 피드백을 하였고 최종적으로 결과가 개선됨

관련 경험

4. Big contest 공모전

추천 시스템 sequence & 알고리즘

주제가 혼잡도에 따른 약속장소를 추천해주는 것이기 때문에 그에 따른 추천 알고리즘을 구성하였다.

1. Google place api 이용, 서울 전지역에 대한 데이터를 반복문을 통해 수집
2. 수집된 시간별 데이터 전처리 및 시간, 장소, category 조건에 따른 listup 함수화
3. Folium을 이용하여 특정 위치로부터 반경내 지역을 격자화, 격자내 평균 혼잡도, 상가수 를 통해 scoring 후 지도 위에 시각화

구역을 격자화 하여 격자 당

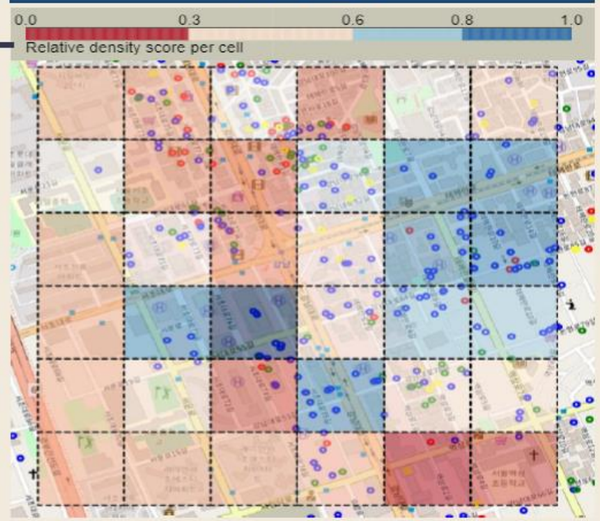
평균 혼잡도, 상가 분포수로 score를 구함

평균 혼잡도가 한산하고, 상가 수가 많은 지역이 1순위인 이유는,
한산하고 상가 수가 적은 지역의 상권이 발달하지 않았을 것이라 생각

평균 혼잡도	상가 수	score
한산	많음	1순위
한산	적음	2순위
붐빔	적음	3순위
붐빔	많음	4순위

$$score = \frac{a * 상가\ 수 + b}{c * 평균혼잡도 + d}$$ (a,b,c,d : 상수)

격자화 예시



관련 경험

5. 비지도 학습을 통한 영한번역구현 20.10.01~20.10.22

Mikel Artetxe, Gorka Labaka, Eneko Agirre, and Kyunghyun Cho. 2018. **Unsupervised Neural Machine Translation**.

논문 내용 요약

Abstract

최근 Neural Machine Translation(NMT) 분야의 성공에도 불구하고

많은 양의 **병렬 코퍼스 부족**은 대부분의 언어에서

실질적 문제점으로 대두됨

연구 목표

병렬 데이터를 전혀 사용하지 않고 완전한 비지도 학습의 방식으로서

단일 언어 코퍼스(monolingual corpora)을 사용하는 NMT 시스템을 사용할 것임

Model Building

- Unsupervised Embedding mappings
- Slightly modified attentional encoder-decoder
- Training on monolingual corpara alone
- Denosing
- Back-Translation

Unsupervised cross-lingual embeddings

- 각 언어에 대한 단일 언어 코퍼스(monolingual corpora)를 Word2vec을 통해 임베딩을 한 뒤, 같은 공간을 공유하게 한다. 이후 2개 국어 사전(bilingual dictionary)에 기반하여 서로를 매핑시키는 선형 변환을 학습한다.
- Mikolov et al(2013b)의 논문에서, 영어와 스페인어를 독립적으로 임베딩했음에도, 같은 의미를 지닌 단어의 임베딩 벡터는 유사한 값을 보인다는 점에서 착안
- GAN 모델과 같이, source word embedding을 target word embedding에 가깝게 변화시켜주는 Generator와, 실제 target word embedding과 Generator가 생성한 embedding을 분류하는 Discriminator로 구성된다

관련 경험

5. 비지도 학습을 통한 영한번역구현

구현

이용한 framework
Cross-lingual-word embedding (github.com/artetxem/vecmap)
Unsupervised Translation (github.com/artetxem/undreamt)

DATA
한글 : Aihub 한국어-영어 번역말뭉치 중 한글 100만 문장 + 매일경제 뉴스기사 400만 문장 크롤링
영어 : WMT14 (news-crawl data) 500만 문장 추출

역할

- 영어 문서, 한글 문서 각각 약 500만문장 **전처리 및 토큰화** 진행(한글-kkma/영어-nltk)
- Word2vec 을 이용하여 영어, 한국어 데이터 embedding
- 이용한 모델의 torch 기반 **코드**를 상황에 맞게 **수정**
- **AWS이용**하여 학습진행 및 결과 정리

관련 경험

5. 비지도 학습을 통한 영한번역구현

주요 이슈 & 해결방법

문장 갯수가 많아 전처리과정에 많은 시간이 소요 됨

=> python multiprocess 패키지의 process를 이용하여 작업을 병렬 처리 진행(github)

참고한 코드가 pytorch 구버전으로 작성이 되어 현재의 CUDA 버전과 호환이 어려웠음

=> torch 1.+ 버전의 코드로 수정, 추가학습 진행 할 수 있도록 코드 추가

Error 처리 내용

undreamt/train.py

- train.py line 337

```
self.loss += loss.data[0]
->self.loss += loss.item()
```

Plain Text

> C:\Users\YOON\anaconda3\envs\torch151\lib\site-packages\torch\nn_reduction.py:43: UserWarning: size_average and reduce args will be deprecated, please use reduction='sum' instead.

translator.py line 38

```
self.criterion = nn.NLLLoss(weight, size_average=False)
--> self.criterion = nn.NLLLoss(weight, reduction='sum')
```

> c:\Users\YOON\torch151\undreamt\undreamt\translator.py:59: UserWarning: volatile was removed and now has no effect. Use with torch.no_grad(): instead

translator line 59

```
varids = self.device(Variable(torch.LongTensor(ids), requires_grad=False, volatile=not train))
-> with torch.no_grad():
varids = torch.LongTensor(ids)
```

> Exception has occurred: RuntimeError Expected object of device type cuda but got device type cpu for argument #3 'index' in call to _th_index_select

train.py line 105

```
device = device("cuda") if args.cuda else devices.cpu
```

> gpu할당 오래된 방식 으로 되어있음 ex) tensor~.cuda()/ tensor.cpu()

translator.py, train.py, encoder.py, decoder.py 전반적으로 코드 수정(gpu에 초점 맞추어 수정하여 cpu는 동작 안할지도..)

devices.py 사용 안함 필요없음

> tensor unit8 dtype warning 발생 bool type으로 변경

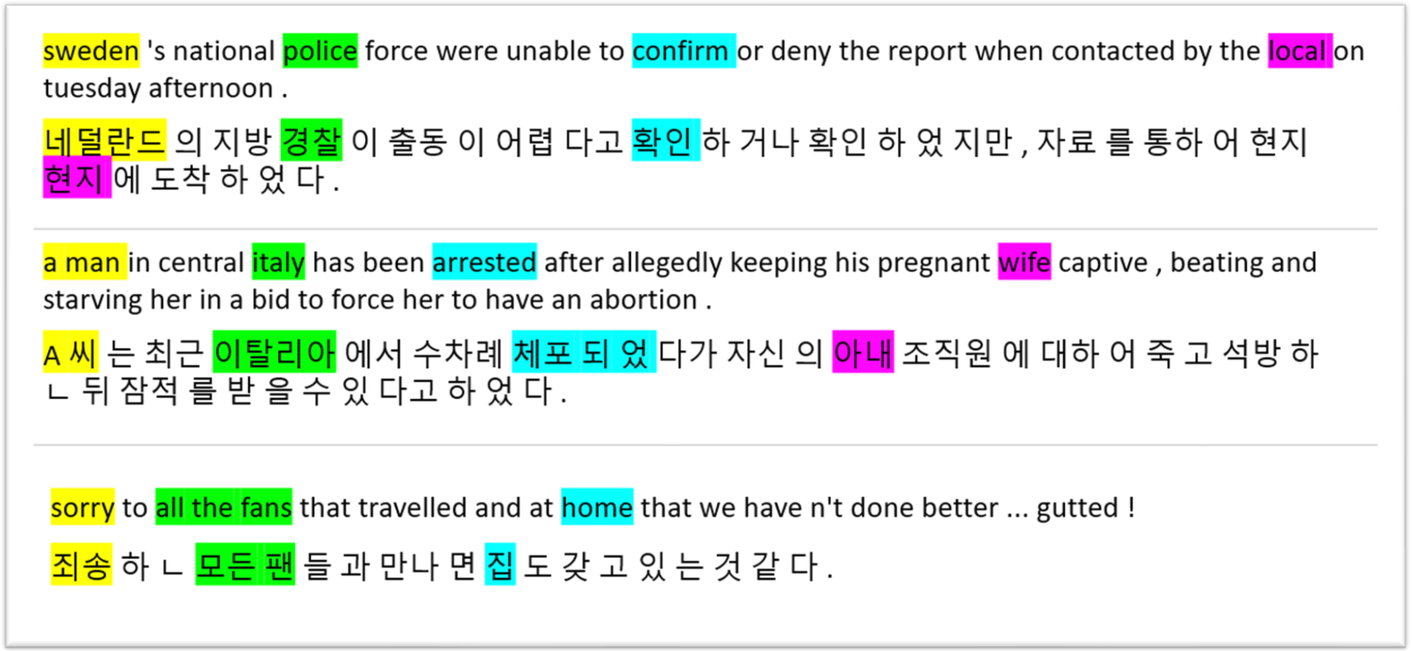
translator.py line 70

```
mask = torch.cuda.BoolTensor(batch_size, max_length).fill_(0)
```

관련 경험

5. 비지도 학습을 통한 영한번역구현

번역 결과



⇒비슷한 공간에 mapping 되어 있는 단어들을 확인 할 수 있었음
⇒문장 구성에 있어서 좋은 결과를 보임