What's the next factor for α? 시장수익률 이상을 넘어서는 초과수익은 불가능하다는 효율적 시장가설(Efficient Market Hypothesis)이 존재함에도, 금융시장의 플레이어들은 남들이 모르는 정보를 기반으로 초과수익을 달성하고자 합니다. 최근 주가예측과 관련해 새로이 개척되고 있는 자연어처리(NLP, Natural Language Process)를 Word Embedding과 Word2Vec 중심으로 살펴봅니다.

NLP for Finance

- 주가 예측 실증 분석에 활발히 사용
- 문자를 주식시장 예측 정보로 활용
- 주가이론과 관련해 새롭게 개척되고 있는 분야

> **"어떤 팩터를 조금 더 잘 찾아서 알파를 창출할까 하는 게 가장 핫한 이슈고 금융 연구의 주된 흐름 ... 최근 나오는 분야는 텍스트를 하나의 팩터로 보자는 것"**
>
> **"텍스트는 그만큼 분석이 어렵고 거기서 의미 있는 결과를 찾아낸다는 것은 솔직히 쉽지 않아 ... 지금 활발하게 연구되고 있는 분야는 텍스트 쪽이 많은 걸 확인"**
>
> 양승호 인사이저 대표

기호학

언어의 상징들은 여러 가지 양식으로 부호화된다.

Language?

- 의미를 전달하기 위하여 고안된 시스템
- 물리적 정보가 아닌 추상적인 의미 정보를 포함
- 기존 비전이나 다른 머신러닝 문제들과 다른 맥락

기표

- Significant, Signifier, Symbol
- 기호가 나타내는 물질적/물리적 형태
- 문자 또는 문자 등가물

기의

- Signifie, Signified, 아이디어나 어떤 물건
- 기호가 지시하는 개념 또는 의미
- "의미" > 구체적, 개별적 의미가 아닌 일반적인 의미를 뜻함
    - '사과' > 특정 사과가 아닌 포괄적, 일반적 사과를 의미

표시론적 의미론 (Denotational semantics)

- 프로그램이 무엇을 하는지 나타내는 domain이라는 수학적 객체를 찾는 것

> 컴퓨터 과학에서 **표시론적 의미론**(Denotational semantics)은 언어에서 수식의 뜻을 표현하는 수학적인 기호들로부터 만들어진 프로그래밍 언어의 의미를 형식화하는 접근법이다. 다른 접근법은 공리적인 의미론(Axiomatic semantics)과 연산 의미론(Operational semantics)을 포함하는 프로그래밍 언어의 형식적 의미를 제공한다.
>
> 간단히 말하자면, 표시론적 의미론이란 프로그램이 무엇을 하는지 나타내는 도메인이라는 수학적 객체를 찾는 것과 관련이있다. 예를들어, 프로그램(혹은 프로그램 구)은 부분 함수(Partial Function)로 표현되거나 어떤 환경과 시스템 간의 게임으로 표시될 수 있다.
>
> 위키피디아

자연어 처리

자연어 처리의 분류

- 쉬운 난이도
  - 철자 검사
  - 키워드 검색
  - 동의어 찾기
- 중간 난이도
  - 웹사이트 등에서 정보 추출(Web Crawling)
- 높은 난이도
  - 기계번역
  - 음성인식
  - 인공지능 개인화 서비스

자연어 처리의 어려움

언어의 유연함과 변화무쌍함이 근본적 원인

- 중의성
- 예외가 존재하는 규칙
- 유연성과 확장성

단어 임베딩

텍스트 전처리 (Text Preprocessing)

자연어처리 이전에 수행되는 과정

토큰화

- 토큰 단위로 텍스트를 분리
- 토큰 기준을 단어 혹은 다른 단위로 할 지 정하는 것
- 대부분 단어 토큰화(word tokenization)
    - 단어 자체 혹은 단어구, 문자열 등이 단어 토큰화에 포함

정제와 정규화

- 정제 (cleaning)
    - 말뭉치나 텍스트 데이터의 노이즈 데이터를 제거하는 작업
- 정규화 (normalization)
    - 다르게 표현되는 동일/유사 단어들을 통합시켜 같은 단어(토큰)로 만들어주는 과정

Python Tools for NLP Preprocessing

- Web Crawling > BeautifulSoup4
- Preprocessing > NLTK (tokenizer)
- Sentiment Analysis > NLTK (VADER)
- Keyword Analysis > Wordcloud

감성 분석 (sentiment analysis)

텍스트에 담긴 사항을 이해하고 극성을 분석해서 정보를 추출하는 기법

- 규칙 및 어휘 기반 (Lexicon based)
- 머신러닝 기반 (Machine learning)
- 딥러닝 기반 (Deep learning)

단어 임베딩 (Word Embedding)

문자를 숫자들의 배열인 벡터로 변환하는 방법

- 의미적으로 유사한 단어는 유사한 단어 벡터로 생성
- 단어 벡터를 통해 유사성이나 의미의 차이를 표시
- 단어 임베딩은 텍스트 분석의 초석

단어 벡터 (Word Vector)

Vocabulary에 있는 단어들을 벡터로 표시한 것

- 단어 벡터의 Distance Measure를 통해 유사성이나 차이에 대한 개념 추출
- 예시 - 영어
  - 약 1,300만 개의 토큰이 있다고 가정
  - Word Space 상의 한 점으로 토큰을 표현
  - 시제, 수치, 성별 등에 따라 dimension 증가

One-hot vector

- 해당 단어(토큰)가 사전의 인덱스에 해당되면 1, 아니면 0 할당
- Column vector dimension = vocabulary 단어 수
  - 의미 파악이 되지 않아 유사성이나 차이점 설정이 어렵다

One-hot vector vs. 단어 임베딩

	One-hot vector	Embedding Vector
차원	고차원 (vocabularya size)	저차원
다른 표현	희소벡터 (sparse vector)	밀집벡터 (dense vector)
표현 방법	수동	훈련 데이터로 학습
값의 타입	1, 0	실수

분포 의미론 (Distributional semantics)

분포 가설 = 비슷한 위치에 등장하는 단어들은 비슷한 의미를 갖는다

> **"You shall know a word by the company it keeps"**
>
> J. R. Firth (1957)

...for example, *while* **deep learning** *based* machine translation that...

- deep learning = center word
- while, based = context with fixed-size window 1
  - fixed-size window 1 = 중심 단어 전후 하나씩 있는 단어를 본다
  - 중심 단어 바로 앞에 나오면 -1, 바로 뒤에 나오면 +1

단어 임베딩의 발전

"엄청난 양의 말뭉치와 비지도학습을 이용, 자연어 처리 시스템에서 범용 과를 발휘하는 단어 벡터를 만드는 것"

- 비지도학습 방법을 이용한 사전학습 (pre-training)
- 단어 표현 방법에 따라 자연어처리 성능이 크게 달라짐

- 핵심기술 Word2Vec 이후 ELMo, BERT, ALBERT 등이 많이 활용
- 벡터 차원 줄이기 위해 전처리 과정에서 **특이값분해**(SVD, Singular Value Decomposition) 실시

Word2Vec

Overview

- Mikolov et al.(2013)
- 단어 벡터를 학습하는 Framework

Word2Vec

1. 많은 양의 말뭉치 확보
2. 고정어휘의 모든 단어를 벡터로 표현
3. 말뭉치의 각 위치 t에 대해서 중심단어 c와 문맥단어 o 확인
4. 단어 벡터의 유사성을 사용하여 c가 주어졌을 때 o가 등장할 확률 계산
5. 목적함수 최대화하기 위해 단어 벡터를 계속 조정 (iteration)

Word2Vec 확률 계산 과정

...many *problems turning* **into** *banking crises* as ...

- problems = context word (-2) = $P(W_{t-2}|W_t)$
- turning = context word (-1) = $P(W_{t-1}|W_t)$
- banking = context word (+1) = $P(W_{t+1}|W_t)$
- crises = context word (+2) = $P(W_{t+2}|W_t)$

Word2Vec Objective Function

$$L(\theta) = \prod_{t=1}^{T} \prod_{\substack{-m \le j \le m \\ j \neq 0}} P(w_{t+j}|w_t;\theta)$$

- $-m \le j \le m \ (j \neq 0)$; window size
- window size = 3
  - 중심단어(center word) 앞, 뒤로 주변단어(context word) 3개씩

$$J(\theta) = -\frac{1}{T} logL(\theta) = -\frac{1}{T} \sum_{t=1}^{T} \sum_{\substack{-m \le j \le m \\ j \neq 0}} logP(w_{t+j}|w_t;\theta)$$

- 모든 j와 t에 대하여 확률을 구한 뒤 합한다
- 목적함수 계산 편의성을 위하여 로그를 취한다
- 이후 T로 나누어 평균을 취한다
- -1을 곱함으로써 최대화 함수를 최소화 함수로 변형한다
  - 이후 Loss Function을 거쳐 θ에 대하여 편미분
  - Gradient Descent 방법으로 최적화 값을 계산

Loss Function이 목적함수로 주어질 때 조건부 확률의 계산

중심 단어가 주어질 때 주변 단어가 나타나는 확률 계산에 대하여 2개의 단어 벡터 활용

- 말뭉치 단어에는 center와 context가 있으며, 두 가지 경우의 수를 모두 사용
  - 중심 단어일 때 v 표시
  - 문맥 단어일 때 u 표시

중심 단어와 문맥 단어에 대해서 Softmax Function을 통해 확률값으로 표시, 기본적으로 분자에는 한 가지 경우에 대한 값, 분모에는 모든 단어에 대한 합산값

$$P(o|c) = \frac{exp(u_o^T v_c)}{\sum_{W \in V} exp(u_t^T v_c)}$$

- 분모 - 전체 vocabulary에 대하여 summation
- 분자 - 분모에서 하나를 택한 경우
- exponential function - 0과 1 사이의 값을 가짐 (확률값)
- $u_o^T v_c$
  - Dot product
  - 문맥 단어와 중심 단어의 similarity 비교
  - 각각에 대하여 곱한 뒤 합산
  - 값이 클수록 확률이 커짐을 나타냄

Word2Vec Softmax Function

$$softmax(X_i) = \frac{exp(X_i)}{\sum_{j=1}^{n} exp(X_j)} = p_i$$

Softmax Function

- 어떤 함수 값들을 확률값으로 변환하는 함수
  - Soft > 아주 작은 값도 어느 정도의 확률로 할당
  - Max > 가장 큰 값을 가진 경우 가장 높은 확률 할당

Gradient 계산

$$\theta = \begin{bmatrix} V_{aarduark} \\ V_a \\ . \\ . \\ . \\ V_{zebra} \\ u_{aarduark} \\ u_a \\ . \\ . \\ . \\ u_{zebra} \end{bmatrix} \subset \mathbb{R}^{2dV}$$

- θ : 하나의 길다란 벡터 내의 모든 모형들의 파라미터
- d차원 벡터와 V개 단어를 사용하는 경우 : $\theta \in \mathbb{R}^{2dV}$
  - 엄청나게 많은 차원 = 엄청나게 큰 계산비용

Loss Function

$$J(\theta) = \frac{1}{T}\sum_{t=1}^{T}\sum_{\substack{-m \le j \le m \\ j \neq 0}} log\, p(w_{t+j}|w_t)$$

Softmax Function

$$log\, p(o|c) = log\frac{exp(u_o^T v_c)}{\sum_{w=1}^{V} exp(u_t^T v_c)}$$

- 윈도우 내 중심단어에 대해서 gradient 계산
- 문맥단어에 대해서도 gradient 계산
  - 각 윈도우에서 사용되는 모든 매개 변수에 대한 업데이트 계산
  - 한 단어에 대해서 2개의 벡터를 생성

Word2Vec: More Details

벡터가 두 개인 이유?

- 중심 단어인 경우 1개, 주변 단어인 경우 1개
- 2개의 벡터가 더 최적화에 용이함
- 최종적으로 2개 벡터를 평균하여 사용하는 등 편리

Word2Vec 모형 변형

- Skip-grams (SG)
  - 중심단어 주어진 상태에서 외부단어 예측
- Continuous Bag of Words (CBOW)
  - 문맥단어를 활용하여 중심단어 예측
- Negative Sampling
  - 학습의 효율성 위해 사용

Word2Vec parameters and computations

우리가 원하는 모형?

- 각각의 위치에서 동일한 예측을 해 주는 모형 (예측의 일관성)
- 문맥에서 자주 등장하는 단어는 높은 확률을 부여하는 모형

Stochastic Gradient Descent (SGD)

Graident Descent 알고리즘의 문제점 = Gradient 계산에 많은 시간 소요

- Stochastic!
  - 반복적으로 window를 샘플링하고 하나씩 업데이트한다
  - SGD의 각 표본 window에서 반복적으로 gradient를 계산한다
- 장점
  - 계산 비용 효율화
- 단점
  - sample window의 단어 숫자가 적으므로 희소 벡터가 될 수 있음

Skip-gram model with Negative sampling

정규화 요인은 계산 비용이 너무 많이 든다!

- Negative sampling & Skip-gram
- 2013년 Mikolov가 논문으로 제안한 접근법
  - Distributed Representations of Words and Phrases and their Compositionality (Mikolov et al. 2013)

How to?

- 노이즈 쌍과 실제 쌍에 대한 이진 로지스틱 회귀(Binary logistic regerssions) 훈련
  - noise pair = 무작위 단어와 pairing한 중심 단어

최적화 목적함수

$$MaximizeJ(\theta) = \frac{1}{T}\sum_{t=1}^{T} J_t(\theta)$$

목적함수 앞부분에 Sigmoid Function 적용 (log part)

- 첫 번째 로그로부터, 두 단어가 동시 발생할 확률을 최대화한다

$$J_t(\theta) = log\,\sigma\left(u_o^T v_c\right) + \sum_{i=1}^{k} \mathbb{E}_j \sim P(w)\left[log\,\sigma(-u_j^T v_c)\right]$$

Negative Sampling

$$J_{neg} - sample(o, v_c, U) = -log(\sigma(u_o^T v_c)) - \sum_{k=1}^{K} log(\sigma(-u_k^t v_c))$$

- 단어 확률을 사용해서 k개의 음수 샘플을 취함
  - 실제 외부단어(outside word)가 나타날 확률 최대화
  - 무작위 단어가 중심단어 주위에 나타날 확률 최소화

동시발생 문제

동시발생을 capture하지 않는 이유?

- 동시발생 행렬을 X라고 두었을 때, 2가지 경우가 가능함
  1. 전체 문서 Approach
  2. Window Approach
     1. Word2Vec과 유사하게 각 단어 주변에 window 적용

*동시발생 행렬은 주로 잠재의미 분석에서 다루어짐

사례: 윈도우 기반 동시발생 행렬

- 말뭉치 3개
  - "I love Mira"
  - "I love Zion"
  - "I like Toby"
- Window size = 1 (usually 5~10)

counts	I	love	like	Mira	Zion	Toby	.
I	0	2	1	1	0	0	0
love	2	0	0	0	1	0	0
like	1	0	0	0	0	1	0
Mira	0	1	0	0	0	0	1
Zion	0	1	0	0	0	0	1
Toby	0	0	1	1	0	0	1
.	0	0	0	0	1	1	0

동시발생 행렬의 문제점

1. 어휘 증가에 따라 행렬 차원 증가
2. 저장 공간 및 계산비용 증가
3. 희소 벡터 문제 존재
4. 모형이 견고하지 않음

동시발생 행렬 문제의 해결책

중요한 정보 대부분을 고정된 저차원 벡터에 밀집 벡터(dense vector)로 만들어주기

- How to reduce number of dimensions?
  - 특이값분해(SVD, Singluar Value Decomposition)
  - 행렬 수의 크기 조정
  - 기능어 문제
    - The, He, Has <- min func or ignore
    - 카운트 대신 Correlation을 설정한 다음 음수에 대해 0을 부여함

K-mooc, cau, 금융 AI, 유시용 교수님

*자료 출처: 강의자료 (12-1강)