급변하는 여론동향 파악을 위한 텍스트 분산 임베딩 방법

류성원[a], 김지영[a], 김누리[a], 이지훈[b] and 조성준[c]

[a]서울대학교 산업공학대학원 데이터마이닝 연구실
Tel: 010-8662-7948, E-mail: lyusungwon@dm.snu.ac.kr
Tel: 010-4149-9826, E-mail: jeeyung.kim@dm.snu.ac.kr
Tel: 010-9599-9075, E-mail: noori@dm.snu.ac.kr

[b]서울대학교 전기전자과 데이터사이언스 및 인공지능 연구실
Tel: 010-2238-3728, E-mail: t080205@gmail.com

[c]서울대학교 산업공학대학 및 산업시스템혁신연구소
Tel: +82-880-6275, E-mail: zoon@snu.ac.kr

개요

본 논문에서는 스트리밍 데이터의 임베딩을 위한 효율적인 분산처리 방법을 제안한다. 한 군집의 장비들의 자원을 최대한 활용하여 노드들에서는 여러 출처의 데이터 수집과 그래디언트를 계산을 하고 파라미터를 보관하는 서버는 이 노드들과 커뮤니케이션을 하는 동시에 임베딩에 대한 평가를 진행하여 임베딩에 대한 성능을 모니터링 할 수 있을 것이다. 또한 효율적인 학습을 위하여 기존의 강화학습에서 활용되었던 중요도 기반 추출 방식도 활용될 수 있을 것이다. 이렇게 가속화된 임베딩 학습 프레임워크를 통하여 여러 임베딩 기법들의 비교 뿐만 아니라 산업에서도 활용될 수 있을 것으로 기대된다.

키워드:

단어 임베딩, 분산 딥러닝, 스트리밍 딥러닝

서론

인터넷 상의 웹사이트 글, 뉴스 기사, 포털 댓글, SNS 등의 텍스트 데이터는 국민의 의견과 감정을 반영한다. 그렇기 때문에 여론을 파악하고 이를 국정에 반영하기 위해서는 인터넷 상의 텍스트 데이터를 분석하는 것이 필수적이다. 이 데이터들은 새로운 정책에 대한 국민의 반응이나 외국 특정 국가에 대한 국민 감정과 같이 국가를 운영하는데 필요한 국민에 대한 정보를 풍부하게 담고 있기 때문에 인터넷의 텍스트를 분석하는 일은 국가 안보에 매우 중요하다.

여론을 파악하기 위하여 인터넷의 텍스트를 분석하기 위해서는 1) 많은 양의 텍스트들에 담긴 의미를 충분히 담되 방대한 정보를 압축적으로 살펴볼 수 있어야 하고 2) 이에 대한 처리가 합리적으로 빨라야 하며 3) 다양한 웹사이트, SNS와 같이 다양한 출처로 부터 들어오는 데이터들을 실시간으로 빠르게 반영할 수 있어야 한다.

최근 자연어 처리기법의 급격한 발전은 단어를 분

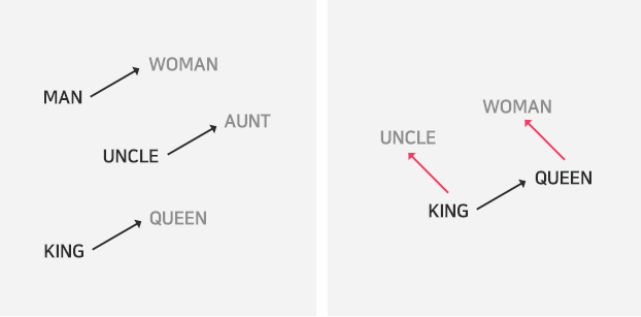

그림 1: 단어 임베딩 [5]

산으로 표현하는 방법에 기인한다. 기존의 텍스트 분석에서는 주로 사용되는 방법은 단어의 빈도수를 통하여 문서를 판단하던가 두 단어 간의 동시 빈도수를 통하여 단어를 표현하였다. 하지만 최근 등장한 신경망을 통하여 단어를 하나의 벡터로 표현하는 Word2Vec은 연속적 표현을 통하여 단어의 의미를 풍부하게 반영할 수 있게 되었다. 이 결과 나타나는 단어 임베딩은 단어가 쓰이는 맥락을 반영하여 단어의 의미론적 정보와 구문론적 정보를 함께 반영한다는 점이 밝혀졌다.

임베딩에 의미를 정교하게 담으려면 많은 데이터를 반영해야 하는데 이를 학습하기 위해서는 오랜 시간이 걸린다. 많은 양의 데이터를 처리하는데 걸리는 시간을 단축하기 위하여 사용되는 방법은 분산 처리 방법이다. 특정 양의 데이터를 한 컴퓨터에서 처리하지 않고 여러 컴퓨터의 유기적인 통신을 통하여

데이터를 처리하는 것이다. 이 덕분에 고성능 컴퓨터가 없다고 하더라도 여러 보통 성능의 컴퓨터를 병렬적으로 연결하여 데이터를 처리하는 것이 가능해졌다.

하지만 기존의 데이터 분산 처리 방법론들은 딥러닝의 모델을 학습하는데에는 적합하지 않다. 왜냐하면 딥러닝 모델을 학습하기 위해서는 목적 함수에 대한 각 파라미터들의 미분값을 구하는 역전파가 이루어져야 하는데 이와 같은 복잡한 연산의 교류는 기존의 분산 방법에서 불가능하기 때문이다. 그리하여 딥러닝만의 분산 학습을 위한 새로운 연구가 지속적으로 이루어지고 있고 가장 효율적인 자원의 활용을 위하여 모델과 데이터에 적합한 분산 방법을 선택하는 것이 중요하다.

반면 기존의 데이터 처리 방법들에서 지속적으로 축적되거나 새로 생기는 데이터를 처리하기 위한 스트리밍 처리에 대한 연구는 많이 이루어지고 있었다. 하지만 훈련과 추론 과정이 분리되어있는 딥러닝의 특성 상 지속적으로 들어오는 데이터를 빠르게 반영하는 연구는 드물게 이루어졌다. 시뮬레이터로부터 축적되는 데이터를 학습해야 하는 강화 학습 분야에서 그나마 이러한 연구가 이루어지고 있다.

기존에 제안된 여러 방법들을 바탕으로 본 논문에서는 스트리밍 데이터의 임베딩을 위한 효율적인 분산 처리 방법을 제안한다. 임베딩은 방대한 양의 텍스트 데이터를 효과적으로 표현할 수 있는 대신 많은 양의 데이터를 학습해야 하기 때문에 데이터가 많아야 하고 시간도 오래 걸린다. 특히 인터넷의 데이터를 임베딩하려면 데이터를 크롤링하여 저장하여야 하기 때문에 시간과 장비가 낭비될 수 있다. 이 두 가지 문제를 해결하기 위하여 분산 처리 기법을 사용할 것이다. 한 군집의 장비들은 각각 다양한 출처의 데이터들을 크롤링하는 즉시 임베딩을 위한 전처리를 실행하고 이를 모델을 위한 데이터로 추가한다. 그리고 이 군집 장비들은 딥러닝의 데이터 분산 처리 방법을 활용하여 모델을 학습한다. 이 결과 데이터를 크롤링과 모델의 분산 학습을 동시에 진행하여 시간을 압축할 수 있고 실시간으로 들어오는 데이터들을 임베딩에 반영할 수 있게 된다.

관련 연구

단어 임베딩

신경망을 통한 단어 임베딩 알고리즘은 현재 광범위하게 사용되고 있다. 중심단어로 부터 주위단어를 예측하는 Skip-gram[1]은 효율적인 연산을 위한 부정 표본 추출 기법과 결부되어[2] 을 사용하여 현재 가장 보편적으로 사용되고 있다(Skip-Gram Negative Sampling: SGNS). 이후 SGNS에서 단어 발생빈도에 대한 정보를 반영하기 어렵다는 점을 보완한 GloVe[3]가 제안되었으며 최근에는 텍스트 분류를 위한 목적으로 N-gram 단위의 작은 단위를 임베딩한 후에 이를 조합하여 사용하는 FastText[4]가 사용되고 있다.

이러한 알고리즘을 통한 연속적 단어 표현이 좋은 성능을 보이는 것에 대한 여러 연구가 진행되었다. 위와 같은 기법들을 통하여 임베딩된 단어들은 비슷하게 사용되는 단어들 끼리 모여있는 경향을 보일 뿐만 아니라 벡터화된 단어들 간의 차이에서 의미론적, 구문론적 규칙을 찾아내는 것을 보여주었다[5, 6]. [7]에서는 인공 신경망으로 학습된 단어의 임베딩이 궁극적으로는 전통적으로 사용되었던 단어의 발생 통계를 반영한다는 것을 밝혔고 [8]에서 인공 신경망 계열의 단어 임베딩 알고리즘들의 우수한 성과는 몇 가지의 구조 선택과 하이퍼파라미터 최적화 덕분이라는 것을 밝혀내었다.

분산 딥러닝 학습

기존의 데이터 분산 처리 프레임워크들은 데이터 마이닝이나 기계 학습에 초점을 맞추어서 이루어지고 있었다. 많은 양의 데이터와 연산에 대하여 하나의 좋은 장비 대신 군집의 장비를 활용하여 분산 처리를 하기 위한 프레임워크들이 등장하여 보편적으로 사용되고 있다[9, 10, 11]. 하지만 이러한 분산 프레임워크들은 딥러닝에서 필수적인 역전파의 구현이 제한되어 딥러닝에서는 전처리 역할에 그쳤다. 딥러닝이 발전하기 시작하면서 딥러닝을 위한 분산 방법들이 연구되기 시작하였다.

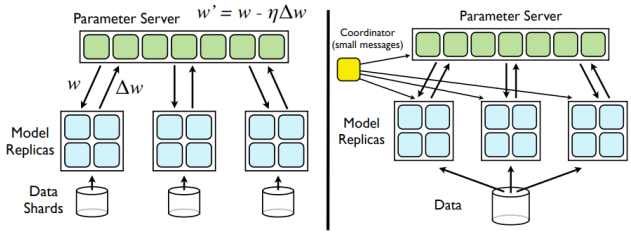

그림 2: 딥러닝 동기화 방식 [14]

[14]는 딥러닝을 위한 병렬로서 모델 분산과 데이터 분산을 제안하였다. 모델 분산은 모델의 크기가 너무 클 경우 여러 장비에 모델을 나누어 담고 이를 학습하는 것이다. 반대로 데이터 분산은 한 모델에 대하여 많은 데이터를 학습해야 할 경우 여러 장비에 데이터를 나누어 담아서 학습하는 방식이다. 이 두 가지 방법을 혼용하여 사용할 수 있다.

데이터 분산을 위해서 여러 장비에 한 모델을 복사해서 각각 학습하게 되는데 이 복사본들의 동기화를 하는 방식이 딥러닝 분산 학습의 중요한 주제이다. [14]에서는 두 가지 동기화 방식으로 동기적 방식인 Downpour SGD와 비동기적 방식인 L-BFGS을 제안한다. Downpour방식은 각 장비가 학습을 한 후 그래디언트를 파라미터 서버에 보내어 파라미터 서버를 비동기적으로 관리한다. 반면 L-BFGS방식은 Coordinator가 메시지를 통하여 파라미터 서버와 모

델 복사본들을 관리하여 동시에 학습한다.

비동기적 방식의 장점은 시간 당 처리하는 데이터의 양이 많다는 점이다. 각 장비가 학습하는 시간의 편차가 있을 수 있는데 각 장비가 빨리 학습하는 대로 파라미터를 갱신하여 다음 학습을 할 수 있기 때문이다. 다만 이 때문에 파라미터 서버와 장비의 통신이 많아질 수 있다. 기존에는 이러한 통신을 할 때 마다 파라미터 서버 내용을 안전하게 보존하기 위하여 다른 통신들의 접근을 제한하는 잠금이 이루어 졌는데 [15]에서는 한 장비가 통신을 할때 잠금을 하지 않더라도 문제가 생길 확률이 낮고 오히려 갱신이 효율적으로 이루어 질 수 있다는 점을 보여주었다. 이러한 이유로 과거의 딥러닝 분산에는 비동기적 방식이 주를 이루었다.

모든 장비의 그래디언트 계산이 끝나야만 파라미터를 업데이트 할 수 있어서 가장 느린 장비의 학습 속도에 맞춰진다는 동기적 방식의 특성 때문에 딥러닝의 분산 학습에서 동기적 방식보다 비동기적 방식이 선호되었다. 하지만 점차 목적 함수의 수렴 측면에서는 비동기적 방식보다 동기적인 방식이 더 효율적이기 때문에 동기적 방식이 데이터 효율이 더 좋다는 관찰이 보고되면서 동기적 방식이 선호되기 시작하였다. 이와 더불어 위에서와 같이 하나의 장비가 전체의 학습 속도를 늦추는 현상을 방지하기 위하여 [16] 에서는 여러 장비 중 빠르게 학습되는 k개 장비의 결과만 활용하여 학습하는 방법을 제안하였다.

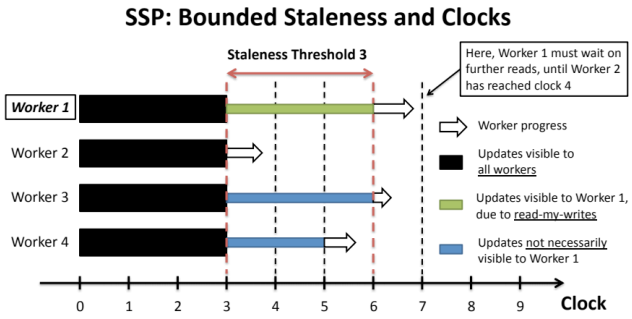

그림 3: Bounded Staleness [17]

동기 방식이 비동기 방식에 비하여 효율적인 이유는 각 장비가 그래디언트를 구하는 동안 다른 장비들이 파라미터를 갱신하기 때문에 그래디언트를 구한 파라미터와 업데이트 하려 파라미터간의 괴리가 생기기 때문이다. [17]에서는 이러한 차이를 최소화하여 동기 방식과 비동기 방식의 중간 형태인 제한된 동기 방식을 제안하였다.

동기화 방식과는 별개로 분산 학습 구조는 분산 학습의 결과에 중요한 영향을 미친다. 분산 학습의 구조는 크게 All-reduce/All-gather방식과 파라미터 서버 방식이 있다. All-reduce/All-gather방식은 서버를 따로 두지 않고 각 워커들이 다른 데이터들에 대하여 각각 학습한 후 워커들 간의 집단 통신(All-reduce/All-gather)을 통하여 동기화 하는 방식이다. 실제로 All-reduce/All-gather와 같은 오퍼레이션은 사용하지 않고 Unidirectional Ring과 같은 방식을 통하여 동기화한다. 파라미터 서버 방식은 말 그대로 워커의 외부에 파라미터를 담당하는 서버를 두고 워커들이 이 서버들과 통신하는 방식을 의미한다. [18]에서는 이러한 방식으로 학습할 때 업데이트 하는 그래디언트의 밀도가 희박하다는 점에서 착안하여 그래디언트를 계층화(Quantization)하고 인덱스만을 저장하여 서로에게 전파하는 방식을 제안하였다.

[19] 에서는 이러한 All-reduce/All-gather구조와 파라미터 서버의 구조의 성능을 측정해 보았는데 특이한 발견을 제보하였다. 파라미터 업데이트의 분포가 밀집한 모델의 경우에는 All-reduce/All-gather구조가 효과적이었고 분포가 희박한 경우에는 파라미터 서버 구조가 효과적이었다. 그리하여 [19]에서는 파라미터의 분포에 따라서 구조를 선택하는 혼합 구조를 제안하였다.

딥러닝의 분산 학습을 위해서는 위와 같은 고려 사항들이 있기 때문에 모델의 구조와 데이터를 보고 적절하게 선택하여야 한다.

스트림 딥러닝 학습

기존의 데이터 처리 프레임워크에서는 지속적으로 쌓이는 데이터들을 처리하기 위한 스트림 처리 프레임워크[12, 13]들이 등장하여 산업에서 활용되고 있다. 이러한 스트림 처리 프레임 워크들은 데이터의 생성 시간과 도달 시간 사이의 간극을 메우는 것이 중요한 과제였다. 하지만 딥러닝에서는 일반적으로 데이터가 모델로 학습하는 과정에서 데이터의 생성 시간이 중요하지 않다. 또한 데이터의 학습과 추론이 따로 이루어 진다는 점도 딥러닝의 스트림 처리에 대한 연구가 희박한 이유이다.

딥러닝에서 스트림 데이터의 처리가 필요하다면 변화하는 데이터를 빠르게 반영해야하기 때문이다. 이와 같은 환경은 환경과 지속적으로 상호작용하는 강화 학습 환경이 있다. 강화학습은 환경에서 얻은 경험을 통하여 모델을 학습하고 학습한 모델이 환경에 영향을 주어 다른 경험을 얻기 때문에 환경에서 나오는 데이터들을 즉각적으로 반영해야 한다. 이러한 측면에서는 강화학습 환경이 스트림 처리가 필요한 환경이라고 볼 수 있다. 또한 이러한 경험들을 최대한 빠르게 많이 학습해야 하기 때문에 강화학습에서 분산을 활용하려는 접근이 많이 있었다.

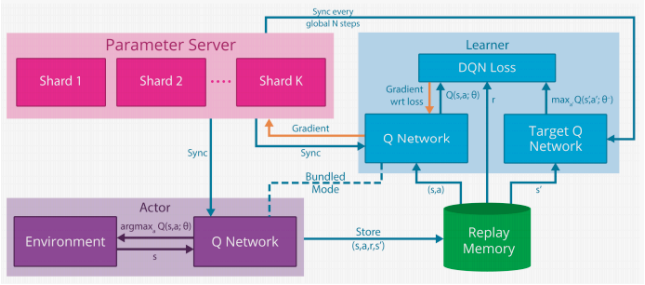

그림 4: Gorila 프레임워크 [20]

[20] 은 강화 학습에 딥러닝 분산 학습을 적용하였다. 파라미터 서버와 행동자, 학습자를 나누어 행동자들이 수집한 데이터를 재생 기억이라는 저장소에 저장한 후 학습자들은 이를 학습하기만 하도록 분업화하였다. 이 군집 환경에서 비동기적 분산 학습을 통하여 강화 학습을 빠르게 학습시키는데 성공하였다. [21]에서는 이러한 분산 학습을 통하여 단순히 학습을 속도를 향상 시킬 뿐만 아니라 환경으로부터 수집하는 데이터의 편향을 극복하고 다양화하기 위하여 활용하였다.

강화 학습 환경에서는 데이터의 경험을 여러 번 재사용하기 위하여 데이터들을 재생 기억이라는 저장소에 넣어두고 이를 무작위 추출하여 학습하는 방식을 취한다. 하지만 점차 효율적인 학습을 위하여 단순 표본 추출이 아닌 중요도 기반 표본 추출이 제안되었다. 데이터의 중요도를 판단하기 위하여 [22]에서는 해당 데이터의 그래디언트의 L2 노름을 활용하기도 하고 [23]에서는 해당 데이터의 손실 값을 활용하기도 한다.

[24]에서는 분산 학습과 데이터의 중요도 기반 추

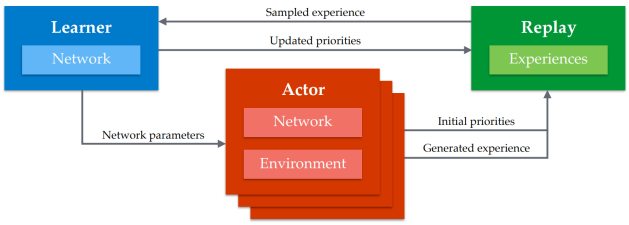

그림 5: Ape-X [24]

출 방식을 모두 활용하여 시간 대비 최고의 성과를 보여주었다. 여러 행동자들은 지속적으로 활동하며 비동기적으로 데이터를 경험 재생에 넣는다. 반면 하나의 학습자는 경험 재생에 쌓인 데이터를 주기적으로 추출하여 학습한다. 이 때 단순 무작위 추출이 아닌 우선순위를 기반하여 추출하여 학습하기 때문에 효율적인 학습이 가능하다. 이러한 구조가 효율적인 이유는 학습자는 하나의 GPU가 필요하지만 행동자들은 CPU만 필요로 하기 때문에 여러 행동자가 활동하는 것이 가능하기 때문이다. 본 논문의 방법론은 이 논문으로부터 많은 영감을 받았다.

제안 방법

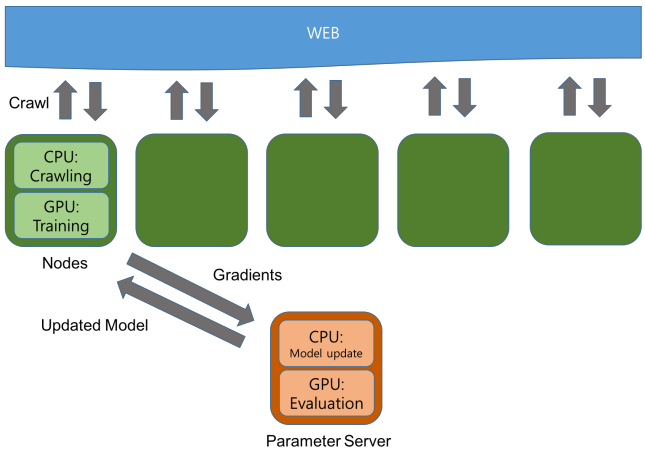

그림 6: 제안 구조

다양한 출처의 인터넷 문서를 효율적으로 임베딩하기 위하여 데이터 분산 딥러닝 학습을 제안한다. 임베딩은 모델 중에서도 파라미터의 갱신의 분포가 매우 희박하기 때문에 All-reduce/All-gather구조보다는 외부에 파라미터의 정보를 가지고 있는 서버를 둔 파라미터 서버 구조가 적합하다. 이에 따라 본 구조는 두 부분으로 나뉜다.

첫번째 부분은 노드들로 두 가지 프로세스를 병행한다. CPU는 각 노드에 할당된 인터넷의 출처로 부터 크롤링을 한다. 크롤링한 데이터는 디스크에 저장하지 않고 즉시 전처리하여 또 다른 프로세스를 위한 큐에 넣는다. 다른 프로세스는 크롤링을 하는 프로세스로부터 받은 데이터를 가지고 주어진 모델에 대하여 그래디언트를 계산하고 주기적으로 파라미터 서버로 보낸다.

파라미터 서버는 모델의 파라미터를 보관하고 있다가 노드들로 부터 받은 그래디언트를 통하여 모델을 업데이트하고 노드들에게 모델을 보내준다. 파라미터 서버의 GPU를 통하여 모델을 주기적으로 평가하여 모델의 학습의 정도를 추적할 수 있다.

동기화 방식은 각 노드들이 할당받은 크롤링 출처에 따라 달라질 수 있다. 만약 각 출처들의 한번에 긁어오는 데이터의 양들이 비슷하다면 데이터의 효율을 최대화 하기 위하여 동기적 방식을 사용할 수 있다. 반면 각 출처 별로 학습 속도의 편차가 커서 전체 학습 속도를 둔화할 것 같다면 큐를 이용하여 비동기적 방식을 사용할 수 있을 것이다.

각 노드들은 크롤링한 데이터들을 재생 기억과 같은 데이터 저장소에 보관해 놓고 학습하기 때문에 강화학습에서 사용한 방법들과 같은 유사한 기법을 사용할 수 있다. 본 논문의 실험에서는 이 데이터들의 우선 순위를 따로 정하지 않지만 후속 연구에서 모델을 효율적으로 학습하기 위한 우선 순위를 설정할 수 있을 것이다.

예상 결과

데이터 공급이 안정화 되있다면 여러 소스의 텍스트이 생성되는 즉시 임베딩에 반영하여 임베딩에 여러

소스의 정보를 담을 수 있다. 여러 소스로 부터 들어오는 데이터는 더욱 다양하고 풍부한 정보가 들어있을 뿐 만 아니라 한 가지의 데이터 출처를 사용할 때에 나타날 수 있는 편향의 위험을 분산하는데에도 도움이 된다. 이를 통하여 만들어진 임베딩은 단순히 특정 데이터셋 내에서의 정보를 포착하는 것이 아닌 현실의 일반적인 의미를 포찰할 수 있게 된다.

이 방법은 기존의 임베딩에 새로운 데이터를 반영할 수 있기 때문에 만약 데이터들이 지속적으로 생성된다면 새로운 데이터의 정보를 임베딩에 반영할 수 있다. 이는 기존 임베딩의 큰 문제점 중 하나였던 단어 의미 고정의 문제를 해결 할 수 있다. 특히 단어의 의미가 달라지거나 새로운 단어가 나타날 경우 이를 반영할 수 있게 된다.

데이터 분석을 통한 현황 분석이 어려운 이유는 과거의 데이터를 기준으로 분석하기 때문이다. 하지만 시시각각 변화하는 여론을 지속적으로 파악하기 위해서는 변화하는 데이터들의 정보를 반영할 수 있는 구조가 필수적이다. 예를 들어, 국가에서 새로운 정책을 제안 했다면 이와 관련된 텍스트들이 인터넷에서 생성될 것이도 이러한 데이터들을 실시간으로 반영함으로써 여론의 동향이나 변화를 파악할 수 있을 것이다.

미래 연구 방향

제안된 방법이 효과적으로 작용하기 위해서는 장기간 지속적으로 크롤링할 수 있을 정도로 많은 양의 데이터가 있는 출처나 지속적으로 데이터가 창출되는 출처가 필요하다. 이를 위해서는 실시간으로 데이터를 받아올 수 있는 SNS나 뉴스기사와 같은 출처가 적당하다. 이러한 데이터들을 안정적으로 크롤링 할 수 있는 구조가 갖추어 있어야 하며 이를 효율적으로 탐색하여 배분할 수 있는 알고리즘이 있어야 한다. 그렇기 때문에 안정적인 크롤링을 위한 서칭 알고리즘은 이 방법론에 큰 도움이 될 것이다.

또한 데이터의 효율을 극대화 하기 위해서는 동기적 동기화방식이 필요하다. 하지만 이는 속도 측면에서 문제가 생길 수 있기 때문에 학습 속도를 배분하고 통신속도를 효율화 할 수 있는 방안이 필요하다. 이를 위해서 [18]에서와 같은 방법을 통하여 통신의 효율을 높이는 방안을 시도해 볼 만 하다.

텍스트 임베딩의 가장 어려운 점 중 하나는 신뢰할 만한 정량적 평가 지표가 없다는 것이다. Word-similarity task나 Word-analogy task는 단어 전체의 임베딩 품질을 판단하기에 턱없이 데이터가 부족하다. 뿐만 아니라 변화하는 텍스트의 의미를 올바르게 평가할 수 있는 지표는 거의 존재하지 않는다. 텍스트 임베딩의 발전을 위해서는 텍스트 임베딩을 객관적이고 정량화 하여 평가할 수 있는 시험이나 지표가 필요하다.

참고 문헌

[1] Mikolov, Tomas, et al. "Efficient estimation of word representations in vector space." *arXiv preprint arXiv:1301.3781*(2013).

[2] Mikolov, Tomas, et al. "Distributed representations of words and phrases and their compositionality." Advances in neural information processing systems. 2013.

[3] Pennington, Jeffrey, Richard Socher, and Christopher Manning. "Glove: Global vectors for word representation." Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP). 2014.

[4] Joulin, Armand, et al. "Bag of tricks for efficient text classification." arXiv preprint arXiv:1607.01759 (2016).

[5] Mikolov, Tomas, Wen-tau Yih, and Geoffrey Zweig. "Linguistic regularities in continuous space word representations." Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies. 2013.

[6] Levy, Omer, and Yoav Goldberg. "Linguistic regularities in sparse and explicit word representations." Proceedings of the eighteenth conference on computational natural language learning. 2014.

[7] Levy, Omer, and Yoav Goldberg. "Neural word embedding as implicit matrix factorization." Advances in neural information processing systems. 2014.

[8] Levy, Omer, Yoav Goldberg, and Ido Dagan. "Improving distributional similarity with lessons learned from word embeddings." Transactions of the Association for Computational Linguistics 3 (2015): 211-225.

[9] Dean, Jeffrey, and Sanjay Ghemawat. "MapReduce: simplified data processing on large clusters." Communications of the ACM 51.1 (2008): 107-113.

[10] Isard, Michael, et al. "Dryad: distributed data-parallel programs from sequential building blocks." ACM SIGOPS operating systems review. Vol. 41. No. 3. ACM, 2007.

[11] Zaharia, Matei, et al. "Resilient distributed datasets: A fault-tolerant abstraction for in-memory cluster computing." Proceedings of the 9th USENIX conference on Networked Systems Design and Implementation. USENIX Association, 2012.

[12] Toshniwal, Ankit, et al. "Storm@ twitter." Proceedings of the 2014 ACM SIGMOD international conference on Management of data. ACM, 2014.

[13] Zaharia, Matei, et al. "Discretized Streams: An Efficient and Fault-Tolerant Model for Stream Processing on Large Clusters." HotCloud 12 (2012): 10-10.

[14] Dean, Jeffrey, et al. "Large scale distributed deep networks." Advances in neural information processing systems. 2012.

[15] Recht, Benjamin, et al. "Hogwild: A lock-free approach to parallelizing stochastic gradient descent." Advances in neural information processing systems.

2011.
[16] Chen, Jianmin, et al. "Revisiting distributed synchronous SGD." arXiv preprint arXiv:1604.00981 (2016).
[17] Ho, Qirong, et al. "More effective distributed ml via a stale synchronous parallel parameter server." Advances in neural information processing systems. 2013.
[18] Strom, Nikko. "Scalable distributed DNN training using commodity GPU cloud computing." Sixteenth Annual Conference of the International Speech Communication Association. 2015.
[19] Parallax: Automatic Data-Parallel Training of Deep Neural Networks. Soojeong Kim, Gyeong-In Yu, Hojin Park, Sungwoo Cho, Eunji Jeong, Hyeonmin Ha, Sanha Lee, Joo Seong Jeong, Byung-Gon Chun. arXiv:1808.02621, August 2018.
[20] Nair, Arun, et al. "Massively parallel methods for deep reinforcement learning." arXiv preprint arXiv:1507.04296 (2015).
[21] Mnih, Volodymyr, et al. "Asynchronous methods for deep reinforcement learning." International conference on machine learning. 2016.
[22] Alain, Guillaume, et al. "Variance reduction in SGD by distributed importance sampling." arXiv preprint arXiv:1511.06481 (2015).
[23] Schaul, Tom, et al. "Prioritized experience replay." arXiv preprint arXiv:1511.05952 (2015).
[24] Horgan, Dan, et al. "Distributed prioritized experience replay." arXiv preprint arXiv:1803.00933 (2018).