Enriching Word Vectors with Subword Information 논문 리뷰

0. Abstract

- 문제 제기
  : 기존의 단어 벡터 표현(Word Embeddings) 모델들은 단어의
  형태론적 구조를 무시하고, 각 단어에 고유한 벡터를 할당하는 방식.

  ⇒ 어휘가 많고, 특히 형태론이 복잡한 언어에서 문제 발생

  > 💡 **형태론적 구조란?**
  >
  > - 단어의 내부 구조를 의미
  > - **어근, 접사**(접두사, 접미사 등), 단어가 변형되는 어형 변화(inflection)를 포함한 단어의 구성 요소와 규칙

- 해결방법

  : **Skip-Gram 모델**을 확장하여, 단어를 문자 n-그램(character n-grams)의 합으로 표현하는 방식을 제안
  ⇒ 각 문자 n-그램에 벡터를 할당하고, 이를 통해 단어를 표현

> 💡 **Skip-Gram 모델**
>
> - 중심 단어(center word)로부터 주변 단어(context word)를 예측하는 방법
> - ex) 예문에서 중심 단어인 'sat'을 통해 주변 단어인 ['The', 'fat', 'cat', 'on', 'the', 'table']을 예측
> - window: 중심 단어의 앞뒤 단어를 몇 개나 활용할지의 범위

- 장점

  : 매우 **빠른 학습** 가능 & 훈련 데이터에 포함되지 않은 **새로운 단어**에 대해서도 벡터 표현 계산
  가능

- 평가

  : 제안한 방법을 9개의 서로 다른 언어에서 **단어 유사도**와 **비유적 표현(유추) 과제**로 평가

  ⇒ 기존의 형태론적 단어 표현 방식과 비교해 SOTA 달성

1. Introuduction

- **연속적 단어 표현**: 주로 **공동 발생 빈도(co-occurrence statistics)**를 사용하여 대규모 데이터로부터 학습하는 방식

- **기존 방식의 한계**
  - 단어의 **내부 구조**를 고려하지 않고, **단어마다 고유한 벡터**를 할당
  - 형태론이 복잡한 언어(예: 터키어, 핀란드어)에서 문제 발생
  - 프랑스어나 스페인어의 동사에는 40개 이상의 다른 형태가 있고, 핀란드어 명사에는 15가지 이상의 격 변화 존재

⇒ 효과적인 단어 벡터를 학습하는 것이 어렵

- **형태론적 정보를 활용한 해결책**
  - 많은 언어에서 단어의 형태는 규칙에 따라 변화
  - **문자 n-그램(character n-grams)**을 사용하여 단어를 벡터로 표현하는 방식을 제안

- **제안 모델**
  - **Skip-Gram 모델**을 확장하여, 단어를 문자 n-그램의 합으로 표현하는 방식으로 학습
  - **단어의 내부 구조 정보**를 사용하므로, 단어 벡터를 더 정교하게 만들 수 있음

2. Related work

1. Morphological word representations

- **형태론적 정보를 이용한 단어 표현**에 대한 연구는 주로 rare words를 더 잘 표현하기 위해 발전해옴
- Alexandrescu와 Kirchhoff(2006): 단어를 여러 특성(feature)으로 표현하는 **factored neural language models**를 제안
  - 형태론적 정보를 단어 표현에 포함시킴
  - **터키어**와 같은 형태론적으로 복잡한 언어에 성공적으로 적용됨

- Lazaridou et al.(2013), Luong et al.(2013), Botha와 Blunsom(2014) 등의 연구: 어절을 형태소(morpheme)로 분해하여 단어 표현을 생성하는 다양한 조합 함수(composition functions)를 제안

⇒ **형태론적 분해**를 필요로 하지만, 이 논문에서 제안한 방법은 그렇지 X

- Chen et al.(2015): 중국어 **단어와 문자에 대한 임베딩을 함께 학습**하는 방법을 제안

- Cui et al.(2015): **형태론적으로 유사한 단어들이 유사한 벡터 표현**을 가지도록 제약을 가하는 방법을 제안

- Soricut과 Och(2015): **형태론적 변환의 벡터 표현을 학습**하는 방법을 제시

  ⇒ 훈련 데이터에 없는 단어(unseen words)의 표현도 형태 변환 규칙을 적용해 얻을 수 있음

- Cotterell과 Schütze(2015): 형태론적으로 주석된 데이터를 사용해 단어 벡터를 학습하는 방법

- Schütze(1993): **단어를 4-그램(four-grams)으로 나누고 각 4-그램의 벡터를 더해** 단어의 벡터 표현을 생성

- Wieting et al.(2016): **n-그램 카운트 벡터**를 사용해 단어를 표현하는 방법을 제안
  - 패러프레이즈(유의어) 쌍을 기반으로 학습
  - 이 논문에서는 일반적인 텍스트 데이터에 대해 학습이 가능

2. Character Level Features for NLP

- 문자 수준에서 학습하는 모델들은 **단어 단위로 분할하지 않고** 문자 자체로부터 언어 표현을 학습
- 재귀 신경망(recurrent neural networks)을 기반으로 한 문자 수준 모델들은 **언어 모델링**(Mikolov et al., 2012; Sutskever et al., 2011; Graves, 2013), **텍스트 정규화**(Chrupała, 2014), **품사 태깅**(Ling et al., 2015), **구문 분석**(Ballesteros et al., 2015) 등에 적용됨

💡 **재귀 신경망(recurrent neural networks)**

- **순차적 데이터**를 처리하는 데 특화된 신경망 구조
  ⇒ 시계열 데이터, 자연어 처리 등
  시간에 따라 순서가 있는 데이터에 주로 사용
- 이전의 출력이 다음 입력에 영향을 미치는 순환 구조

  ⇒ **이전 단계의 정보**를 **현재 단계**에서 사용

  ⇒ **시간의 흐름**에 따라 변하는 데이터를 학습 가능

- RNN은 시간이 길어질수록 과거의 정보가 현재로 전달되는 과정에서 **기억 상실**이 발생하는 경향이 있음

  ⇒ 이를 해결하기 위해 LSTM, GRU(Gated Recurrent Unit) 같은 RNN의 변형 모델들이 제안됨

- 합성곱 신경망(convolutional neural networks)을 기반으로 한 문자 수준 모델들은 **품사 태깅**(dos Santos and Zadrozny, 2014), **감정 분석**(dos Santos and Gatti, 2014), **텍스트 분류**(Zhang et al., 2015), **언어 모델링**(Kim et al., 2016) 등 다양한 자연어 처리 작업에 사용됨

💡 **합성곱 신경망(convolutional neural networks)**

- **이미지**나 **공간적 데이터**를 처리하는 데 특화된 신경망 구조
- 이미지와 같은 **공간적 데이터**에서 중요한 **패턴**을 학습하기 위해 필터(커널)
  를 사용하여 입력 데이터의 특정 local features 추출
- **합성곱**: 입력 데이터에 필터를 적용하여 특성 맵(feature map)을 생성
- CNN은 입력 이미지의 **공간적 구조**를 유지하면서 특징을 학습
- 합성곱 연산 후 **풀링(pooling)**이라는 과정을 통해, 특성 맵의 크기를 줄이고 **계산량을 줄이며**, **과적합**을 방지
- CNN은 이미지와 같은 **고정된 크기와 구조**를 가진 데이터에 적합하지만, RNN처럼 **순차적**이고 **시간에 따라 변화하는 데이터**를 처리하는 데는 적합하지 X

- Sperr et al.(2013): 단어를 **문자 n-그램**으로 인코딩한 **restricted Boltzmann machines** 기반의 언어 모델을 소개
  - **restricted Boltzmann machines:** 비지도 학습을 위한 확률적 신경망 모델

    ⇒ 입력 데이터를 **확률적으로 재구성**하는 데 초점

    ⇒ feature learning, 차원 축소, 생성 모델링 등에 사용됨

    ⇒ 각 층 내에서 노드들이 **서로 연결되지 않지만**, **입력층과 숨겨진 층 사이에는 완전 연결**

- **기계 번역**에서도 서브워드(subword) 단위를 사용해 드문 단어의 표현을 생성하는 방법이 제안됨
    - subword: 단어보다 작은 단위로, **단어를 세분화**해 처리하는 방식

3. Model

- 단어 표현(Word Representations)을 학습하는 방법

1. General model

- Mikolov et al.(2013b) 의해 제안된 방법

- 주어진 단어가 주어졌을 때 그 주변 문맥에 해당하는 단어들을 예측하는 방식

- 목표: 단어 벡터(word vector)를 학습하여 **해당 단어와 문맥 단어의 결합 확률**을 최대화하는 것

- 모델은 큰 말뭉치(corpus)에서 단어들이 **순차적으로 등장**하는 방식으로 학습됨
- 각 단어는 문맥 속에서 발생할 확률이 높은 단어들을 예측하는 방식으로 벡터를 학습
- 주어진 문맥에서 단어 w_c가 나타날 확률을 **소프트맥스(Softmax)** 함수로 정의

$$p(w_c \mid w_t) = \frac{e^{s(w_t,\, w_c)}}{\sum_{j=1}^{W} e^{s(w_t,\, j)}}.$$

⇒ s(w_t,w_c): 두 단어w_t와 w_c 간의 **스코어 함수**(두 단어 벡터의 내적(Scalar Product)으로 계산됨)

$$s(w_t, w_c) = u_{w_t}^{\top} v_{w_c}$$

⇒ u_wt와 v_wc는 각각 단어와 문맥 단어의 **입력 벡터**와 **출력 벡터**

- **부정 샘플링(Negative Sampling)**
  - **소프트맥스 함수**는 계산 비용이 크기 때문에, Skip-Gram 모델에서는 효율성을 높이기 위해 부정 샘플링(Negative Sampling)을 사용
  - **단어와 문맥 단어의 양성 예시뿐만 아니라 무작위로 선택된 부정 예시(negative samples) 도 학습에 사용**
  - **이진 분류 문제**로 변환하여 모델 학습 가능
  - 손실 함수:
    문맥 단어에 대해 부정 예시를 포함하여
    로지스틱 손실 함수(logistic loss)를 사용해 학습

$$\log\left(1+e^{-s(w_t,\,w_c)}\right)+\sum_{n\in\mathcal{N}_{t,c}}\log\left(1+e^{s(w_t,\,n)}\right)$$

⇒ N_t, c: 부정 샘플로 선택된 단어들

⇒ 문맥 단어와 부정 샘플들 간의 **점수 차이**를 학습

2. Subword model

- 기존 Skip-Gram 모델: 각 단어를 **하나의 벡터**로 학습 & 단어 내부 구조(예: 접두사, 접미사 등)를 무시
  - ex) "play"와 "playing"은 각각 독립적인 벡터로 처리 되며, 두 단어 사이의 관계가 학습되지 X

- 서브워드 단위(n-grams)로 단어를 표현
  - ex) 단어 "where", n = 3일 때 <wh, whe, her, ere, re>로 분해됨 + **특별한 시퀀스**로서 단어 전체인 "<where>"도 포함

    ⇒ **her**는 단어 "her"와 "where"에 각각 다른 의미로 등장하지만, 이 모델에서는 그 차이를 인식 가능

- 단어를 여러 개의 서브워드로 나누고, **각 서브워드에 대한 벡터를 학습한 후 이들의 합을 통해 단어의 최종 벡터**를 얻음

- **형태론적으로 복잡한 언어**에서 더 나은 성능을 발휘
- 훈련 데이터에 없는 새로운 단어도 서브워드 단위를 통해 벡터 표현을 생성 가능

- **스코어링 함수**
  - 주어진 단어 w는 그 단어를 구성하는 **n-그램들의 집합 G_w**로 표현됨
  - 각 n-그램 g에 대해 벡터 z_g가 학습되며, 단어와 문맥 단어 간의 스코어는 각 n-그램 벡터와 문맥 단어 벡터 v_c 간의 내적 합으로 계산됨

$$s(w, c) = \sum_{g \in \mathcal{G}_w} \mathbf{z}_g^\top \mathbf{v}_c.$$

  ⇒ 단어와 문맥 간의 **유사성**을 측정하는 데 사용되며, 단어의 내부 구조를 반영해 더 세밀한 표현을 제공

- **해싱을 통한 효율적인 메모리 사용**
  - n-그램의 수가 매우 많아질 수 있기 때문에, 메모리 사용량을 줄이기 위해 **해싱 함수(Hashing Function)를 사용하여 n-그램을 고정된 크기의 벡터 공간으로 매핑**
  - **FNV-1a 해싱 함수**를 사용해 n-그램을 정수 값으로 변환하고, 이를 통해 메모리 사용을 제한
    ⇒ 이 논문에서는
    K = 2.106이라는 크기의 해시 공간을 사용해 n-그램을 처리

💡 **FNV-1a 해싱 함수**

- **FNV-1** 해싱 함수의 변형
- **곱셈 (multiplication)**: 특정한 소수 값으로 현재 해시 값을 곱하는 연산
- **XOR 연산**: 해시 값을 입력 바이트와 XOR하여 해시 값에 변화 주기

<연산 과정>

- **초기화 (Initialize)**
  - **해시 초기값(Offset Basis)**: FNV 해싱은 특정한 **초기 해시 값**으로 시작. 일반적으로 32비트, 64비트, 128비트 등의 다양한 비트 길이에 따라 초기값이 다르게 설정됨.
- **해시 계산 (Hash Computation)**
  - 문자열이나 입력 데이터의 **각 바이트**를 순차적으로 처리하면서 다음의 연산을 수행:
    1. 현재 해시 값에 입력 바이트를 **XOR** 연산
    2. 결과 값을 고정된 소수(FNV 프라임)로 **곱셈**
- **결과 반환**
  - 모든 입력 데이터에 대해 연산이 끝나면, 그 결과 해시 값을 반환

4. Experimental setup

1. Baseline

- 실험에서 base model로 **Word2Vec** 패키지의 **skipgram** 및 **cbow** 모델을 사용
  ⇒ 널리 사용되는 단어 벡터 학습 모델
  - skipgram: 주어진 단어로 문맥을 예측
  - cbow: 주어진 문맥으로 단어를 예측

- 두 모델 모두 **Word2Vec** 패키지의 **C 구현** 버전을 사용하여 비교 실험을 수행

2. Optimization

- **확률적 경사 하강법(Stochastic Gradient Descent, SGD)** 사용
- 손실 함수: negative log likelihood를 최소화하는 방식으로 최적화
- 학습률(step size): 시간이 지남에 따라 **선형으로 감소**하는 방식을 사용

$$\gamma_0(1 - \frac{t}{TP}).$$

⇒ γ0: 초기 학습률

t: 현재 시간 단계

T: 전체 단어 수

P: 전체 학습 반복 횟수

- **병렬 처리**를 위해 **Hogwild** 알고리즘을 사용

⇒ **비동기 방식**으로 여러 스레드가 동시에 모델 파라미터를 업데이트할 수 있도록 해줌

💡 **Hogwild** 알고리즘

- **비동기적인** 방식으로 경사 하강법을 수행하는 **확률적 경사 하강법 알고리즘**
- 공유된 데이터에 대해 여러 **스레드가 비동기적으로 동시에 접근**하고, 각 스레드가 데이터를 읽고 **모델 파라미터를 업데이트**할 수 있는 구조
- **멀티코어 CPU** 또는 **병렬 처리 환경**에서 SGD의 성능을 크게 향상시킬 수 있음

3. Implementation Details

- 모델 파라미터 설정
  - **단어 벡터 크기**: 300차원
  - **양성 샘플**마다 5개의 **부정 샘플**을 무작위로 선택. 부정 샘플의 선택 확률은 단어 빈도의 제곱근에 비례.
  - **context window size**: 1에서 5 사이의 값을 무작위로 선택.
  - **빈도 높은 단어의 다운 샘플링**: 가장 빈번한 단어는 10^-4의 임계값을 사용해 **서브샘플링됨**.
    - 빈번한 단어를 자주 학습하지 않도록 하기 위함
  - **단어 사전**: 학습 데이터에서 5번 이상 등장한 단어들만 사용하여 단어 사전을 구성
  - **학습률 설정**: Skip-Gram 기준 모델은 0.025, CBOW와 이 논문의 모델은 0.05로 설정됨.
    - 이는 Word2Vec 패키지에서 기본으로 사용되는 값
- 학습 속도

- 제안된 **서브워드 모델**은 **Skip-Gram** baseline 모델보다 약 1.5배 느리며, 105,000 단어/초/스레드 속도로 처리됨.
- 반면, baseline **Skip-Gram** 모델은 145,000 단어/초/스레드 속도를 기록
- 논문의 모델은 C++로 구현되었으며, **오픈소스로 공개됨**

4. Datasets

- **Wikipedia Data**를 사용하여 모델 학습
- 사용된 언어: 아랍어, 체코어, 독일어, 영어, 스페인어, 프랑스어, 이탈리아어, 루마니아어, 러시아어
- **Matt Mahoney**의 **perl 스크립트**를 사용해 데이터를 정규화 진행

> 💡 **perl 스크립트 주요 기능**
>
> - **텍스트 정규화**: 원시 텍스트 데이터에서 불필요한 요소(예: HTML 태그, 메타데이터, 특수 기호)를 제거하여 깨끗한 텍스트 만들기
> - **토큰화(Tokenization):** 문장을 단어 단위로 분리하는 작업
> - **소문자 변환 및 특수 문자 제거**: 대문자를 소문자로 변환하여 같은 단어를 다른 토큰으로 인식하지 않게 만들기
> - **불용어(Stop words) 처리:** 자주 등장하지만 실제 의미 전달에는 크게 중요하지 않은 단어들 제거

- 모든 데이터셋은 **무작위로 섞은 후, 5번의 학습 반복**을 통해 모델을 훈련

5. Results

1. Human similarity judgement

1. **평가 방법**

- 사람의 유사도 판단과 모델이 계산한 단어 벡터 간의 **코사인 유사도(cosine similarity)**를 비교하여 모델의 성능을 평가
- 사용 지표: **Spearman의 순위 상관계수(Spearman's rank correlation coefficient)**

  ⇒ 모델이 사람의 판단과 얼마나 일치하는지 측정

> 💡 **Spearman 상관계수**
>
> - **단조 관계**: Spearman 상관계수는 두 변수 사이의 **단조(monotonic)** 관계를 측정
>   - 변수 간의 값이 일정한 방향으로 증가하거나 감소하는 관계를 의미
>   - 두 변수 간의 순위가 일관되게 증가하거나 감소하면 높은 Spearman 상관계수
> - **순위 기반**: Spearman 상관계수는 각 변수의 **값 자체가 아니라 순위**에 기초하여 상관관계를 측정
>   - 값 자체보다는 각 값의 순위가 서로 얼마나 일관되게 변화하는지를 평가

2. **모델 성능 비교**

- 다양한 언어에 대해 제안된 모델(sisg, 서브워드 기반 스킵그램 모델)과 기존 **Skip-Gram (sg)** 및 **CBOW** 모델의 성능을 비교

		sg	cbow	sisg-	sisg
AR	WS353	51	52	54	**55**
DE	GUR350	61	62	64	**70**
	GUR65	78	78	**81**	**81**
	ZG222	35	38	41	**44**
EN	RW	43	43	46	**47**
	WS353	72	**73**	71	71
ES	WS353	57	58	58	**59**
FR	RG65	70	69	**75**	**75**
RO	WS353	48	52	51	**54**
RU	HJ	59	60	60	**66**

→ **sisg 모델**은 대부분의 데이터셋에서 baseline **모델**보다 **우수한 성능**을 보임
(특히
아랍어, 독일어, 러시아어와 같은 형태론적으로 복잡한 언어에서는 성능 향상이 두드러짐)

→ **영어 WS353 데이터셋**에서는 sisg가 다소 낮은 성능을 보였으나, **희귀 단어(Rare Words, RW) 데이터셋**에서는 기존 모델보다 우수한 성능을 보임
= 흔한 단어보다
희귀한 단어에서 서브워드 정보를 사용하는 것이 더 효과적

3. **형태론적 복잡성의 중요성**

- **독일어와 러시아어**는 문법적 격변화(declensions)가 많고, 복합어(compound words)가 자주 사용됨
- sisg 모델은 **문자 수준에서의 유사성**을 반영하여 두 단어를 더 비슷하게 처리 가능

- **서브워드 정보**를 활용하면 복합어가 많은 언어에서 단어 간의 관계를 더 잘 파악 가능

4. **Out-of-Vocabulary 단어 처리**

- baseline 모델(sg, cbow):

  훈련 데이터에 포함되지 않은 단어에 대한 벡터를 생성할 수 없기 때문에, **OOV(Out-of-Vocabulary)** 단어를 null vectors로 처리

- **sisg 모델: 서브워드 정보**를 사용하여 훈련 데이터에 없는 단어에 대해서도 **유효한 벡터**를 생성 가능
  ⇒ OOV 단어 처리에 유리

2. Word analogy tasks

1. **평가방법**

- 단어 유추 과제: **단어 간의 관계**를 학습하는 문제
  - ex) "king : queen = man : woman"과 같은 비율 관계를 학습하는 것

- 평가 지표: **문법적(syntactic)** 및 **의미적(semantic)** 관계

2. **모델 성능 비교**

- 체코어(CS), 독일어(DE), 영어(EN), 이탈리아어(IT)에 대해 **단어 유추 과제**의 성능 비교

		sg	cbow	sisg
CS	Semantic	25.7	27.6	27.5
	Syntactic	52.8	55.0	77.8
DE	Semantic	66.5	66.8	62.3
	Syntactic	44.5	45.0	56.4
EN	Semantic	78.5	78.2	77.8
	Syntactic	70.1	69.9	74.9
IT	Semantic	52.3	54.7	52.3
	Syntactic	51.5	51.8	62.7

→ **sisg 모델**은 특히 syntactic analogies에서 baseline 모델보다 더 나은 성능을 보임
:
형태론적 정보가 중요한 문법적 관계에서 sisg 모델이 더 효과적임을 보여줌

→
의미적 관계(semantic analogies)는 sisg 모델이 baseline 모델과 비슷하거나 약간 낮은 성능을 보임

3. **문법적 관계에서의 우수성**

- **체코어**와 **이탈리아어**에서 sisg 모델은 **문법적 관계**에서 매우 우수한 성능을 보임

⇒ 형태소가 중요한 역할을 하는 언어에서 sisg 모델의 강점을 보여줌

3. Comparison with Morphological Representations

1. **형태소 정보를 활용한 모델들과의 비교**

	DE		EN		ES	FR
	GUR350	ZG222	WS353	RW	WS353	RG65
Luong et al. (2013)	-	-	64	34	-	-
Qiu et al. (2014)	-	-	65	33	-	-
Soricut and Och (2015)	64	22	71	42	47	67
sisg	73	43	73	48	54	69
Botha and Blunsom (2014)	56	25	39	30	28	45
sisg	66	34	54	41	49	52

- 제안된 모델(sisg)은 기존의 **형태소 정보 기반 모델**들과 성능을 비교.
  - **Luong et al. (2013)**: 재귀 신경망(recursive neural network)을 사용한 모델
  - **Qiu et al. (2014)**: morpheme CBOW 모델을 사용
  - **Soricut and Och (2015)**: 접두사와 접미사 분석에 기반한 모델
  - **Botha and Blunsom (2014)**: 로그-바이리니어(log-bilinear) 언어 모델을 사용한 방법

- 제안된 모델(sisg)은 대부분의 평가 지표에서 **기존 형태소 기반 모델**보다 더 나은 성능을 보임
- 특히 **독일어**에서 성능 향상이 두드러졌는데, 이는 명사 합성(compounding)을 효과적으로 처리했기 때문
  ⇒ Soricut과 Och의 모델은 합성어를 모델링하지 못한 반면, sisg는 이를 효과적으로 처리
- **OOV(Out-of-Vocabulary) 단어**에 대해 제안된 모델은 **문자 n-그램**을 합산하여 유효한 벡터를 생성할 수 있었으며, 이를 통해 성능이 크게 향상

4. Effect of the size of the training data

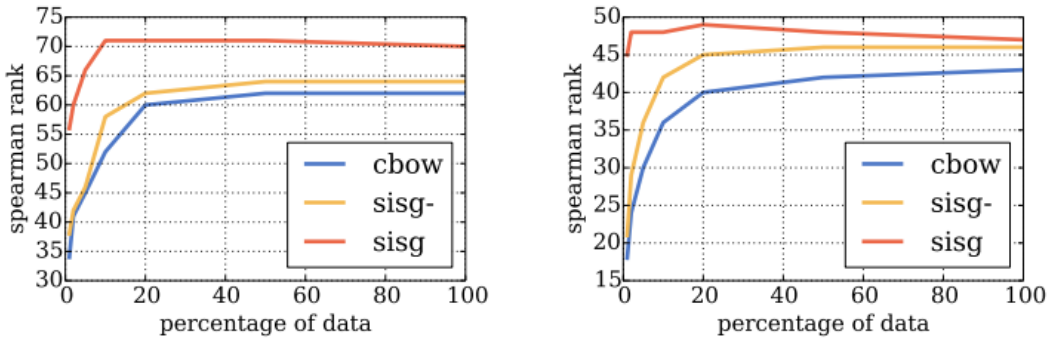
1. **실험 목적**

- 훈련 데이터가 클수록 더 다양한 단어를 학습 가능
- 제안된 모델(sisg)은 **문자 수준의 유사성(character-level similarities)**을 이용해 **희귀한 단어**를 더 잘 처리할 수 있기 때문에, 훈련 데이터가 적을 때도 더 **견고한 성능**을 유지할 수 있다는 가정을 평가하고자 함

2. **실험 방법**

- **Wikipedia 덤프**에서 1%, 2%, 5%, 10%, 20%, 50% 비율로 데이터를 나누어 제안된 모델(sisg)과
  baseline 모델(cbow) 훈련
- **OOV 단어**(훈련 데이터에 없던 단어) 처리
  - sisg-: null vector 사용
  - sisg: **n-그램 벡터**의 합을 이용해 새로운 벡터를 생성

3. **결과 분석**



(a) DE-GUR350 (b) EN-RW

Figure 1: Influence of size of the training data on performance. We compute word vectors following the proposed model using datasets of increasing size. In this experiment, we train models on a fraction of the full Wikipedia dump.

- 훈련 데이터 크기와 상관없이 sisg는 cbow보다 항상 더 좋은 성능을 보임
- 특히, **훈련 데이터가 적을 때**도 sisg 모델은 **우수한 성능**을 유지
  ⇒ 독일어 GUR350 데이터셋에서 전체 데이터의 5%만 사용한 sisg 모델은 전체 데이터를 사용한 cbow 모델보다 더 높은 성능(66 vs. 62)을 보임

  ⇒ **영어 RW 데이터셋**에서도 1%의 데이터만 사용한 sisg 모델이 전체 데이터를 사용한 cbow 모델보다 성능이 좋음

4. **주요 관찰**

- **sisg 모델**은 적은 양의 데이터로도 **OOV 단어**에 대해 유효한 벡터를 생성할 수 있으므로, 작은 데이터셋에서도 **효과적인 성능**을 보임

- 반면, **cbow 모델**은 훈련 데이터가 증가할수록 성능이 개선되지만, 데이터를 충분히 많이 사용하지 않으면 성능이 떨어짐

- **특정 응용 분야**에서 사용되는 **도메인 특화 텍스트**가 많지 않은 경우, sisg 모델을 사용하면 **제한된 데이터**로도 좋은 단어 벡터를 학습할 수 있다는 장점

5. Effect of the size of n-grams

1. **실험 목적**

- **n-그램의 크기**(예: 2, 3, 4, 5, 6 문자)가 단어 벡터의 성능에 미치는 영향을 분석
- **n-그램**을 통해 단어를 더 세밀하게 표현하기 + 형태소적 정보(예: 어미 변화, 접두사, 접미사) 반영 가능

2. **실험 방법**

- 제안된 모델은 기본적으로 3~6 글자의 **n-그램**을 사용하여 단어를 벡터로 표현
- 실험에서 **n의 범위**(예: n=2부터 n=6까지)를 변화시키며 **단어 유사도 및 단어 유추 성능을 평가**
- **독일어와 영어**에 대해 **단어 유사도** 및 **단어 유추** 성능을 측정하여 n-그램 크기가 성능에 어떻게 영향을 미치는지 분석

3. **결과 분석**

	2	3	4	5	6
2	57	64	67	69	69
3		65	68	70	70
4			70	70	**71**
5				69	**71**
6					70

(a) DE-GUR350

	2	3	4	5	6
2	59	55	56	59	60
3		60	58	60	62
4			62	62	63
5				64	64
6					**65**

(b) DE Semantic

	2	3	4	5	6
2	45	50	53	54	55
3		51	55	55	**56**
4			54	**56**	**56**
5				**56**	**56**
6					54

(c) DE Syntactic

	2	3	4	5	6
2	41	42	46	47	**48**
3		44	46	**48**	**48**
4			47	**48**	**48**
5				**48**	**48**
6					**48**

(d) EN-RW

	2	3	4	5	6
2	78	76	75	76	76
3		78	77	78	77
4			79	79	79
5				**80**	79
6					**80**

(e) EN Semantic

	2	3	4	5	6
2	70	71	73	74	73
3		72	74	**75**	74
4			74	**75**	**75**
5				74	74
6					72

(f) EN Syntactic

- **n-그램 크기**에 따라 성능이 달라지며, 일반적으로 **3~6 글자**의 n-그램이 적절한 성능을 제공
- 독일어(DE)에서는 특히 **긴 n-그램**이 더 좋은 성능을 보임

  ⇒ 독일어 명사가 합성어(compound nouns)로 이루어진 경우가 많기 때문에, 긴 문자 시퀀스가 더 유용하다는 것을 의미
- 영어(EN)에서는 **n=4~5** 범위가 더 좋은 성능을 보임 & **n=2**는 정보가 부족하여 성능이 떨어짐

4. **주요 관찰**

- **짧은 n-그램**(ex. n=2): 충분한 형태소 정보를 제공하지 못하여 성능이 떨어짐
  - 특히 어미 변화나 접미사 등을 포착하는 데에는 n=2가 부족하기 때문
- **n=3 이상**의 n-그램: 더 많은 형태소 정보를 포함하므로, **문법적** 및 **의미적 유추 과제**에서 성능을 크게 향상시킴
- **독일어**와 같은 **합성어**가 많은 언어에서는 긴 n-그램(n=5, 6)이 유리함

  ⇒ 이는 합성어에서 길이가 긴 문자가 더 많이 포함되기 때문

5. **n-그램 크기의 선택**

- **3~6 글자**의 n-그램이 가장 적합한 선택으로 보여짐

⇒ **짧은 접두사나 접미사**와 같은 형태소 정보를 포함하면서도, 긴 n-그램은 **복합어**와 같은 더 복잡한 단어 구조를 포착할 수 있기 때문

6. Language modeling

1. **실험 목적**

- **언어 모델링:** 텍스트의 확률 분포를 학습하는 작업으로, 주어진 문맥에서 다음 단어를 예측하는 모델
- **제안된 sisg 모델**이 기존의 **CLBL** 및 **CANLM** 모델보다 얼마나 더 좋은 성능을 보이는지 평가
- **형태론적으로 복잡한 언어**(예: 체코어, 러시아어)에서 **서브워드 정보**가 어떻게 도움이 되는지 분석

2. **실험 방법**

- **5개 언어**(체코어(CS), 독일어(DE), 스페인어(ES), 프랑스어(FR), 러시아어(RU))에 대해 실험을 수행
  - Botha와 Blunsom (2014)에서 사용한 데이터셋을 이용
  - 각 데이터셋은 약 **백만 개의 학습 토큰**으로 구성되어 있으며, 동일한 전처리와 데이터 분할을 사용
- **LSTM 기반 순환 신경망**(RNN)을 사용해 언어 모델을 학습
- 모델의 **lookup 테이블**을 초기화할 때 **pre-trained 단어 벡터**를 사용

- 2개의 Baseline 모델과 비교
  - **CLBL**: Botha와 Blunsom(2014)의 로그-바이리니어 언어 모델
  - **CANLM**: Kim et al. (2016)의 문자 기반 언어 모델

3. **Perplexity**

- 언어 모델의 성능을 측정하는 지표로, 모델이 다음 단어를 얼마나 잘 예측하는지 나타냄

- **Perplexity가 낮을수록** 모델이 더 좋은 성능을 보인다는 것을 의미

4. **결과 분석**

	CS	DE	ES	FR	RU
Vocab. size	46k	37k	27k	25k	63k
CLBL	465	296	200	225	304
CANLM	371	239	165	184	261
LSTM	366	222	157	173	262
sg	339	216	150	162	237
sisg	**312**	**206**	**145**	**159**	**206**

- **LSTM**, **기본 skip-gram 모델(sg)**, 그리고 **sisg 모델**의 성능을 **Perplexity**로 비교
- **sisg 모델**은 모든 언어에서 **가장 낮은 Perplexity**를 기록하며, 가장 좋은 성능을 보임
  - 특히, 체코어(CS)와 **러시아어(RU)** 같은 **형태소가 복잡한 언어**에서 성능 향상이 두드러짐
    - **체코어**에서는 skip-gram 대비 **8% 감소**
    - **러시아어**에서는 skip-gram 대비 **13% 감소**

- **로망스어 계열 언어**(스페인어, 프랑스어)에서는 성능 향상이 다소 적었지만 여전히 유의미한 개선
  - **스페인어**에서는 **3% 감소**, **프랑스어**에서는 **2% 감소**

5. **주요 관찰**

- **서브워드 정보**를 사용한 **sisg 모델**은 형태론적으로 복잡한 언어에서 **큰 성능 향상**을 보임
  ⇒ 복잡한 어미 변화나 합성어를 더 잘 처리할 수 있는 능력

- **로망스어 계열 언어**에서는 형태소적 복잡성이 낮아 상대적으로 성능 향상이 적음

6. Qualitative analysis

1. Nearest neighbors

- 목적: 제안된 모델(sisg)과 기존 skip-gram 모델을 비교하여, 두 모델이 **유사한 단어들**을 어떻게 학습하는지 분석

- **유사성 측정**은 **코사인 유사도**를 사용

- 결과: 복잡하고 **기술적**이거나 **희귀한 단어**에 대해, 제안된 모델(sisg)이 기존 모델보다 더 나은 Nearest Neighbors를 찾아냄.
  ⇒ sisg 모델이
  형태소 정보를 더 잘 포착할 수 있기 때문

💡 **Nearest Neighbors**

- **벡터 공간에서의 단어 표현:** 모든 단어는 벡터로 표현되는데, 벡터는 일반적으로 단어의 의미나 문법적 속성을 포함하는 다차원 공간에서 위치를 가짐
- **유사도 측정**
  - **코사인 유사도**는 벡터 간의 각도를 측정하여, 각도가 작을수록 두 벡터가 더 유사하다는 것을 의미
- **Nearest Neighbors 탐색:** 주어진 단어 벡터에 대해, 벡터 공간 내에서 **가장 유사한 단어 벡터들**을 찾아 반환
  - "dog"라는 단어 벡터를 기준으로 할 때, 최근접 이웃으로 "puppy", "cat"과 같은 단어들
- **Nearest Neighbors 분석의 목표:** 단어의 **의미적** 또는 **문법적 유사성**을 찾는 것

2. Character n-grams and morphemes

1. **목적**

- **n-그램**이 실제로 **형태소**에 해당하는지 평가
- 단어 벡터를 n-그램들의 합으로 표현하고, 특정 n-그램을 제거했을 때 단어 벡터가 얼마나 달라지는지를 측정

2. **방법**

- 각 단어는 n-그램 벡터들의 합으로 표현됨
- 각 n-그램을 제거한 후 단어 벡터가 얼마나 변화하는지를 코사인 유사도로 평가하여, 가장 중요한 n-그램을 순위별로 나열

- **독일어**, **영어**, **프랑스어** 세 언어에서 n-그램이 단어 표현에 미치는 영향을 분석

3. **결과 분석**

	word	n-grams		
DE	autofahrer	fahr	fahrer	auto
	freundeskreis	kreis	kreis>	<freun
	grundwort	wort	wort>	grund
	sprachschule	schul	hschul	sprach
	tageslicht	licht	gesl	tages
EN	anarchy	chy	<anar	narchy
	monarchy	monarc	chy	<monar
	kindness	ness>	ness	kind
	politeness	polite	ness>	eness>
	unlucky	<un	cky>	nlucky
	lifetime	life	<life	time
	starfish	fish	fish>	star
	submarine	marine	sub	marin
	transform	trans	<trans	form
FR	finirais	ais>	nir	fini
	finissent	ent>	finiss	<finis
	finissions	ions>	finiss	sions>

Table 6: Illustration of most important character n-grams for selected words in three languages. For each word, we show the n-grams that, when removed, result in the most different representation.

- **독일어(German)**: 독일어의 복합 명사(compound nouns)에서 n-그램들이 **유효한 형태소**와 잘 대응된다는 것을 확인
  - ex) **Autofahrer**(자동차 운전사)에서 가장 중요한 n-그램은 **Auto**(자동차)와 **Fahrer**(운전사)로, 실제 형태소와 정확히 대응됨

- **영어(English)**: 영어에서도 복합 명사에서 n-그램들이 형태소와 잘 대응됨
  - ex) **starfish**에서 **star**(별)와 **fish**(물고기)가 중요한 형태소로 나누어짐
  - 그러나, 영어에서는 n-그램이 **접두사**나 **접미사**와도 대응됨

        - ex) **kindness**에서는 **ness**가 중요한 형태소로, **unlucky**에서는 **un**이 중요한 접두사로 작용

- **프랑스어(French)**: 프랑스어에서는 **동사 활용**에서 중요한 형태소가 나타남
    - ex) **finirais**(끝낼 것이다)에서 **ais**, **ent**와 같은 동사 어미들이 중요한 역할

3. Word similarity for OOV words

1. **목적**

- 제안된 모델은 훈련 중에 나타나지 않은 OOV 단어(Out-of-Vocabulary Words)에 대해 어떻게 단어 벡터를 생성하고 유사성을 계산하는지 평가

2. **방법**

- 훈련 데이터에 없는 OOV 단어는 해당 단어의 **n-그램 벡터들의 평균**으로 표현됨
- **English RW similarity dataset**에서 일부 단어 쌍을 선택하여 그 중 하나가 OOV 단어인 경우에 대해 분석을 진행
- Wikipedia 데이터의 1%만을 사용해 모델을 훈련하여 더 많은 OOV 단어가 포함된 환경 구축

3. **결과**

- OOV 단어의 n-그램들과 훈련된 단어의 n-그램들 간의 코사인 유사도를 계산하여 **단어 쌍 간 유사성**을 평가
    - **ex1) chip**과 **microcircuit**: 두 단어의 **micro**와 **circuit** 부분이 **chip**과 잘 일치하는 그룹을 형성
    - **ex2) rarity**와 **scarceness**: **scarce**와 **rarity**가 유사하게 매칭되었고, **ness**와 **ity** 같은 접미사도 잘 대응됨
    - **ex3) preadolescent**와 **young**: **preadolescent**의 **adolesc** 부분이 **young**과 잘 대응됨

4. **결론**

- 제안된 모델은 훈련 데이터에 없는 OOV 단어에 대해서도 **n-그램 정보**를 사용해 단어 벡터 생성 가능

  ⇒ 단어 간 **형태소적 유사성**을 잘 포착

- 접두사와 접미사가 있는 단어에서 문법적 형태가 사전에 없어도 **의미적 유사성을 잘 계산** 가능

7. Conclusion

- 이 논문에서는 **서브워드 정보**를 사용하여 단어 벡터를 학습하는 단순하면서도 효과적인 방법을 제안

- 제안된 모델은 **n-그램을 포함하는 skip-gram 모델**로, 기존 방법들보다 **더 빠르고**, **사전 처리가 필요 없으며** 형태소적 정보를 잘 반영

- 제안된 모델은 **형태소 분석에 의존하지 않고**도 서브워드 정보를 반영함으로써 기존 모델보다 **우수한 성능**을 보임

- **형태소가 복잡한 언어**(예: 체코어, 러시아어)에서 더 나은 성능을 발휘하며, 단순한 형태소 기반 분석보다 **빠르고 정확하게** 학습할 수 있음을 보여줌