Sent2Vec 문장 임베딩을 통한 한국어 유사 문장 판별 구현

박상길, 신명철

요약

본 논문에서는 Sent2Vec을 이용한 문장 임베딩으로 구현한 유사 문장 판별 시스템을 제안한다. 또한 한국어 특성에 맞게 모델을 개선하여 성능을 향상시키는 방법을 소개한다.

고성능 라이브러리 구현과 제품화 가능한 수준의 완성도 높은 구현을 보였으며, 자체 구축한 평가셋으로 한국어 특성을 반영한 모델에 대한 P@1 평가 결과 **Word2Vec CBOW에 비해 9.25%, Sent2Vec에 비해 1.93%** 더 높은 성능을 보였다.

모델

본 논문에서는 유사 문장을 판별하기 위한 모델로 Sent2Vec을 이용한다. Sent2Vec은 범용적인 문장 임베딩을 목표로 하는 비지도 학습 모델로, Word2Vec의 CBOW 모델을 문장 단위로 확장한 모델이다. 문장 임베딩은 다음의 수식으로 정의한다.

$$\boldsymbol{v}_S := \frac{1}{|R(S)|}\boldsymbol{V}\iota_{R(S)} = \frac{1}{|R(S)|}\sum_{w\in R(S)}\boldsymbol{v}_w$$

CBOW

CBOW는 컨텍스트(주변)가 타겟(중앙) 벡터를 갖도록 학습한다. 단어 위치를 보지 않는 Bag-of-Words 형태로, 학습 속도가 빠른 반면 일반적으로 Skip-gram에 비해 성능은 떨어지는 것으로 알려져 있다. Sent2Vec의 기본적인 학습 방식은 Word2Vec과 동일하며 문장의 네거티브 샘플링을 통해 컨텍스트 전체의 loss를 최소화 하는 형태로 학습한다.

$$\min_{U,V}\sum_{S\in\mathcal{C}}\sum_{w_t\in S}\left(\ell\left(\boldsymbol{u}_{w_t}^{\top}\boldsymbol{v}_{S\setminus\{w_t\}}\right)+\sum_{w'\in N_{w_t}}\ell\left(-\boldsymbol{u}_{w'}^{\top}\boldsymbol{v}_{S\setminus\{w_t\}}\right)\right) \quad (1)$$

네거티브 샘플의 수는 데이터셋이 작을 경우 5-20, 대형 코퍼스에서는 2-5 정도의 값을 사용한다. 코퍼스가 클 수록 이 값은 더 낮출 수 있다.

서브 샘플링
빈도 수가 높은 단어는 확률적으로 제외하여 불용어 제거와 유사한 효과를 갖도록 하며 또한 학습 속도를 높여준다. t = 10^{-5}를 초기값으로 사용한다.

$$P(w_i) = \sqrt{\frac{t}{f(w_i)}} + \frac{t}{f(w_i)}$$

다이나믹 컨텍스트 윈도우
CBOW는 사용자가 설정한 최대 윈도우 값을 기준으로 균등하게 샘플링된 윈도우 사이즈로 랜덤하게 컨텍스트 윈도우를 정한다.

저빈도 단어 삭제
출현 빈도가 낮은 단어는 학습에서 배제한다. 학습 배제는 성능에 영향을 끼치지 않으면서도 모델 압축에 탁월한 효과를 보인다.

문자 n-그램
fastText에는 문자 단위로 서브워드를 추출하여 학습에 참여해 동사의 변형을 잘 유추할 수 있다.

Table 2: 각 모델에 따른 성능 평가 결과

모델	P@1	P@3	P@5
Word2Vec CBOW	0.7574	0.8618	0.8927
Sent2Vec uni. (baseline)	0.7854	0.8997	0.9271
Sent2Vec bi.	0.8306	0.9296	0.9539
Sent2Vec bi. + 주어부 가중치 1.2	0.8369	0.9312	0.9544
Sent2Vec bi. + 주어부 가중치 1.2 + 가중치 감소*	0.8499	0.9374	0.9597

Sent2Vec

Sent2Vec은 문장 임베딩을 위한 특성을 지니기 위해 CBOW 모델에 비해 다음과 같은 차이점이 있다.

서브샘플링 비활성화
Sent2Vec은 서브샘플링을 사용하지 않는다. 서브샘플링은 n-그램 생성을 가로막고 문장에서 중요한 부분을 앗아갈 수 있다. 또한 서브샘플링된 단어만 남게되면 단어간 거리를 단축시켜 컨텍스트 윈도우의 크기를 암묵적으로 증가시키는 부작용을 낳는다.

다이나믹 컨텍스트 윈도우 비활성화
Sent2Vec은 문장 전체의 의미를 살리기 위해 문장의 모든 n-그램을 조합하여 학습 하기 때문에 다이나믹 컨텍스트 윈도우는 사용하지 않는다. Sent2Vec의 컨텍스트 윈도우 크기는 문장의 전체 길이로 고정한다.

단어 n-그램
Sent2Vec에서는 단어 단위의 n-그램을 적용하며, 바이그램의 최대 거리를 설정한다. 즉, 문장이 (A,B,C,D,E)로 구성되어 있을때,
- 단어 n-그램=3: (A), (A,B), (A,C)
- 단어 n-그램=4: (A), (A,B), (A,C), (A,D)

조합을 컨텍스트로 하여 학습한다.

Table 1: 단어 n-그램=3, 단어 ID 조합과 새로운 단어 ID

타입	단어	단어 ID
단어	카드결제	14
	알림	17
	서비스	26
	계좌변경은	1538
	어떻게	2
	하나요	6
	단어 ID 조합	**새로운 단어 ID**
단어 n-그램=3	14 17	692956
	14 26	1780052
	17 26	841078
	17 1538	711288
	26 1538	1285391
	26 2	888413
	1538 2	1747
	1538 6	100493
	2 6	758302

한국어 특성

주어부 가중치
우리 말은 술어부 보다 주어부에서 중요한 단어가 등장한다는 가정하에 일정 길이 이상의 단어를 지닌 문장에서 절반 지점 상위를 주어부, 하위를 술어부로 정하고 주어부 단어 벡터에 α 배율로 가중치를 적용한다. 여기서 α는 1.2-1.5 사이일때 가장 좋은 결과를 보였다.

가중치 감소
코퍼스 출현 빈도에 따라 빈도가 높은 단어에 대해서는 가중치를 감소시킨다. 이는 불용어 제거와 유사한 효과를 갖는다.

$$w = \frac{(t+1)\cdot k}{t+k}$$

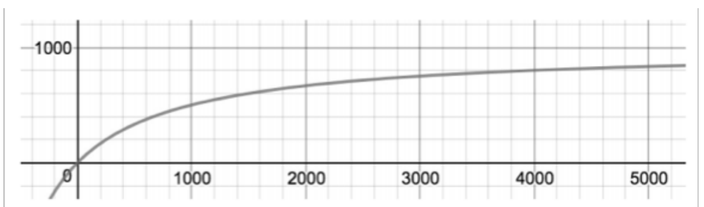

Figure 1: $t = 1000$, 단어 출현 빈도 순위 k에 대한 가중치 w

실험 및 평가

학습 데이터는 2013년 부터 2015년 까지의 모든 언론사의 뉴스를 형태소 분석한 결과로 4.5억개의 문장, 100억개의 단어, 85G 크기의 파일로 구성되어 있다. Sent2Vec은 비지도 학습 모델이므로 문장에 대한 별도의 레이블링은 진행하지 않는다.

문장의 유사도를 판별하는 방식은 유입 문장에 대한 임베딩 계산 후 비교 대상이 되는 모든 문장의 임베딩 벡터에 대해 코사인 유사도를 계산하여 가장 유사한 벡터에 해당하는 문장을 정답으로 간주하고 거리순으로 결과를 제시한다.

4명의 참가자가 약 2주간 5,000여개의 문장으로 구성된 테스트셋을 구축하였고, 이를 통해 모델의 성능을 기계적으로 측정할 수 있게 했다. 테스트셋에는 직접 선택한 정답 문장이 포함되어 있으며 각각 1위, 3위, 5위 내에 존재하는지 여부로 Precision at k(P@k)를 정하고 이 점수로 모델의 성능을 평가했다.

Contact

박상길, 신명철
{kaon.park, index.shin}@kakaocorp.com

References

1. J. Mueller and A. Thyagarajan, "Siamese recurrent architectures for learning sentence similarity," in Proceedings of the thirtieth aaai conference on artificial intelligence, 2016, pp. 2786–2792.
2. T. Mikolov, I. Sutskever, K. Chen, G. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," CoRR, vol. abs/1310.4546, 2013.
3. M. Pagliardini, P. Gupta, and M. Jaggi, "Unsupervised learning of sentence embeddings using compositional n-gram features," CoRR, vol. abs/1703.02507, 2017.
4. Q. V. Le and T. Mikolov, "Distributed representations of sentences and documents," CoRR, vol. abs/1405.4053, 2014.
5. M. J. Kusner, Y. Sun, N. I. Kolkin, and K. Q. Weinberger, "From word embeddings to document distances," in Proceedings of the 32Nd international conference on international conference on machine learning - volume 37, 2015, pp. 957–966.
6. S. Deerwester, S. T. Dumais, G. W. Furnas, T. K. Landauer, and R. Harshman, "Indexing by latent semantic analysis," JOURNAL OF THE AMERICAN SOCIETY FOR INFORMATION SCIENCE, vol. 41, no. 6, pp. 391–407, 1990.
7. Michael Hardy et al., "Latent semantic indexing — Wikipedia, the free encyclopedia." 2018.
8. C. Dyer, "Notes on noise contrastive estimation and negative sampling," CoRR, vol. abs/1410.8251, 2014.
9. P. Bojanowski, E. Grave, A. Joulin, and T. Mikolov, "Enriching word vectors with subword information," Transactions of the Association for Computational Linguistics, vol. 5, pp. 135–146, 2017.
10. O. Levy, Y. Goldberg, and I. Dagan, "Improving distributional similarity with lessons learned from word embeddings," Transactions of the Association for Computational Linguistics, vol. 3, pp. 211–225, 2015.
11. A. Joulin, E. Grave, P. Bojanowski, and T. Mikolov, "Bag of tricks for efficient text classification," in Proceedings of the 15th conference of the european chapter of the association for computational linguistics: Volume 2, short papers, 2017, pp. 427–431.
12. Danko Georgiev et al., "Cosine similarity — Wikipedia, the free encyclopedia." 2018.