XGBoost와 교차 검증을 이용한 구문분석 말뭉치에서의 오류 탐지

최민석[O1], 김창현[2], 천민아[1], 박혁로[3], 김재훈[1]
한국해양대학교[1], 한국전자통신연구원[2], 전남대학교[3]
ehdgus5136@naver.com, chkim@etri.re.kr, minah@kmou.ac.kr, hyukro@jnu.ac.kr, jhoon@kmou.ac.kr

Detecting Errors in Dependency Treebank through XGBoost and Cross Validation

Min-Seok Choi[O1], Chang-Hyun Kim[2], Min-Ah Cheon[1], Hyuk-Ro Park[3], Jae-Hoon Kim[1]
Korea Maritime and Ocean University[1], Electronics and Telecommunications Research Institute[2], Chonnam National University[3]

요 약

의존구조 말뭉치는 자연언어처리 분야에서 문장의 의존관계를 파악하는데 널리 사용된다. 이러한 말뭉치는 일반적으로 오류가 없다고 가정하지만, 현실적으로는 다양한 오류를 포함하고 있다. 이러한 오류들은 성능 저하의 요인이 된다. 이러한 문제를 완화하려고 본 논문에서는 XGBoost와 교차검증을 이용하여 이미 구축된 구문분석 말뭉치로부터 오류를 탐지하는 방법을 제안한다. 그러나 오류가 부착된 학습말뭉치가 존재하지 않으므로, 일반적인 분류기로서 오류를 검출할 수 없다. 본 논문에서는 분류기의 결과를 분석하여 오류를 검출하는 방법을 제안한다. 성능을 분석하려고 표본집단과 모집단의 오류 분포의 차이를 분석하였고 표본집단과 모집단의 오류 분포의 차이가 거의 없는 것으로 보아 제안된 방법이 타당함을 알 수 있었다. 앞으로 의미역 부착 말뭉치에 적용할 계획이다.

주제어: 오류 탐지, 의존구조 말뭉치, XGBoost, 교차검증

1. 서론

말뭉치란 자연언어 연구를 위해 특정 목적을 가지고 언어 표본을 추출한 집합을 의미한다. 본 논문에서는 구문분석 말뭉치를 다룬다. 한국어 구분분석에는 의존구조가 널리 사용되며, 한국어 의존구조 말뭉치를 일반적으로 세종 구구조 말뭉치[1]를 의존구조로 변환하여 사용하고 있다[2]. 또한 최근에는 국립국어원에서는 구문분석 말뭉치[3][1]를 공개하여 많은 연구자들이 활발하게 사용될 수 있을 것으로 기대된다. 이러한 말뭉치들은 다양한 사람들이 오랜 시간 제작하여, 다양한 오류들을 포함하고 있다[4]. 이런 말뭉치를 학습말뭉치로 사용하는 경우, 학습된 시스템의 성능 저하 요인이 되므로 가능한 한 많은 오류들은 수정되어야 한다. 일반적으로 오류 수정에는 말뭉치를 구축하는 것과 비슷한 시간과 노력이 필요할 뿐 아니라 일관성과 신뢰성 문제를 가지고 있다[5]. 이러한 문제를 해결하기 위해 오류를 검출하고 수정하는 많은 연구들이 진행되었다[6-9]. 하지만 여전히 오류 검출을 위한 학습 말뭉치가 부족하고 작업자들마다 판단하는 오류가 다르다는 문제가 존재한다.

이러한 문제를 다소 완화시키기 위해 본 논문에서는 품사부착 말뭉치에서 사용되었던 시스템[10]을 구문분석 말뭉치에 적용하는 방법을 제안한다. 제안된 시스템은 문맥표상의 표현(context embedding representation)과 구문태그 확률의 예측, 기준값의 설정(threshold setting), 오류 후보 선택(error candidate selection)으로 구성된다. 문맥 표상은 기준 어절 주변의 문맥을 벡터로 표현한 것이며, 여기서 문맥은 기준 어절의 부모(head), 조부모 어절과 구문분석 태그를 말한다. 기준값의 설정은 품사부착말뭉치에서 사용한 기준값을 동일하게 사용한다. 최종적으로 기준값과 교차검증을 이용하여 최종 오류 후보로 선택한다.

본 논문의 구성은 다음과 같다. 2장에서 오류 검출을 소개하고, 3장에서는 오류 후보 탐지 시스템을 소개한다. 4장에서는 실험을 통한 오류 탐지 성능을 분석하고, 5장에서 결론 및 향후 연구 방향을 기술한다.

2. 관련 연구

본 장에서는 일반적인 오류 검출의 의미와 구문분석 말뭉치에서 오류 검출에 대해서 간단히 서술할 것이다.

2.1 오류 검출과 XGBoost

오류 검출이란 전체 데이터에서 다른 형태의 데이터를 검출하는 것을 말하며[11], NN 기반 방법[12], 스펙트럴 기반 방법[13], 군집화 기반 방법[14], 앙상블 방법[15] 등의 다양한 방법이 존재한다. 본 연구에서는 앙상블 방법의 일종인 XGBoost(eXtreme Gradient Boosting)[16]를 이용한다. 앙상블이란 다음 결과 예측 시 여러 개의 학

1) 국립국어원에서 공개한 구문분석 말뭉치는 의조구조 말뭉치이다.

습된 모델의 결과를 종합하여 사용하는 방법이다[17]. 이러한 방식은 크게 배깅(bootstrap aggregation)과 부스팅(boosting)으로 나눌 수 있다. 랜덤포레스트(random forest)는 배깅의 대표적인 방법이며, XGBoost는 부스팅의 대표적인 방법이다.

2.2 구문분석 오류 검출

구문분석 오류는 크게 부모 부착 오류와 구문 태그 부착 오류가 있다. 이러한 오류를 수동으로 검축하고 수정하는 것은 구문분석 말뭉치를 구축하는 것과 비슷한 시간과 노력이 소요된다. 이를 해결하기 위해 많은 연구들은 오류수정 도구를 개발하여 오류를 찾고 수정하였[18]. 또한 말뭉치가 구축된 이후가 아닌 구축 과정에서 오류를 최소화하려는 연구도 진행되었다. 이 방법으로 많이 사용되는 방법은 여러 부착 시스템의 결과가 불일치하는 경우 오류로 검출하는 방법이다[15]. 여러 부착 시스템이 일치하는 경우에도 오류가 포함되어 있을 수 있지만 오류가 없는 것으로 가정한다. 즉 완전한 학습말뭉치(gold standard corpus)보다는 준완전 학습말뭉치(silver standard)[19]를 목표로 한다.

3. 구문태그 오류 후보 검출 시스템

2.2절에서 언급했듯이 구문분석 오류는 크게 부모 부착 오류와 구문 태그 부착 오류가 있다. 본 논문에서는 부모 부착은 모두 정확하다는 가정을 바탕으로 한다. 위 가정을 바탕으로 선택된 의존관계 즉 구문 태그의 오류를 탐색한다. 본 논문에서 제안하는 오류 검출 시스템은 크게 세 단계(문맥표상 표현, 기준값 설정, 오류 후보 선택)로 구성된다. 이하의 절에서는 각 단계를 설명할 것이다.

3.1 문맥표상 표현

문장 내에서 의존 관계는 연관된 어절에 의해서 결정된다. 즉, 주어진 어절의 구문 태그가 오류인지를 판단하기 위해서는 주변의 문맥 정보를 활용해야 한다. 문맥 정보는 많을수록 정확한 예측이 가능하지만, 본 논문에서는 부모, 조부모 두 어절을 문맥으로 정의한다.


```
{
    "word_id": 2,
    "word_form": "성남시",
    "head": 3,
    "label": "NP",
    "dependent": [
        1
    ]
},
{
    "word_id": 3,
    "word_form": "판교신도시에서",
    "head": 5,
    "label": "NP_AJT",
    "dependent": [
        2
    ]
},
{
    "word_id": 4,
    "word_form": "이달",
    "head": 5,
    "label": "NP_AJT",
    "dependent": [
    ]
},
{
    "word_id": 5,
    "word_form": "분양하는",
    "head": 7,
    "label": "VP_MOD",
    "dependent": [
        3,
        4
    ]
},
```


그림 1. 국립국어원 구문분석 말뭉치의 일부

그림 1은 국립국어원 구문분석 말뭉치(버전 1.0)[2]에서 추출한 문장 “경기 성남시 판교신도시에서 이달 분양하는 중대형 아파트의 3.3㎡ 당 분양가가 2006년보다 200만 원 정도 싼 1500만 원 후반대로 결정될 것으로 보인다.”의 의존구조의 일부이다. 그림 1의 ‘성남시’를 기준으로 문맥 표상을 정의하면 그림 2와 같다.

그림 2. ‘성남시’ 기준의 문맥표상

각 어절과 구문태그의 표상은 FastText[20]를 이용하여 학습하였다. FastText는 주변 단어와 단어의 부분 단어(subword)를 이용하여 미등록어 문제에 좀 더 좋은 결과를 보여주는 장점이 있다. 이와 같은 방법으로 구해진 문맥표상은 XGBoost의 입력으로 제공되어 기준 어절에 대한 구문태그의 확률을 구한다.

3.2 기준값 설정

본 논문에서 완전한 하나의 분류기(XGBoost)가 있다고 가정한다. 이 분류기는 주어진 문맥에 대해 각 어절의 확률 $P(t|C)$를 출력하고, 이 확률을 내림차순으로 정렬하면 $p_1 = P(t_1|C)$, $p_2 = P(t_2|C)$, ..., $p_n = P(t_n|C)$과 같다. 여기서 t는 구문태그이고, C는 문맥이고, n은 구문태그의 개수이다. 또한 부착된 구문태그 t_a의 확률이 $p_a = P(t_a|C)$일 때, 본 논문에서는 다음과 같은 두 가지 가정을 전제로 구문태그 t_a를 오류로 판단한다.

가정 1: $p_1 - p_a > \theta_1$

그림 3은 실제로 부착된 구문태그의 확률 p_a과 분류기의 가장 높은 확률 p_1의 차이가 지정된 기준값 θ_1보다 클 경우, 오류로 간주한다. 이 가정은 일반적으로 분류기의 출력과 부착된 구문태그가 다를 경우로서 일반적으로 흔히 사용하는 가정이다.

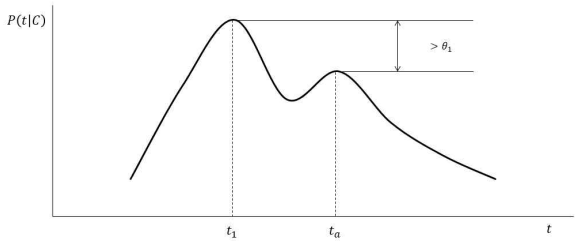

그림 3. 가정 1의 개념 그래프

가정 2: $p_1 - p_a < \theta_1$ and $p_1 - p_2 < \theta_2$:

그림 4는 가정 1의 조건을 만족하지 않더라고 분류기의 가장 높은 확률 p_1과 두 번째로 높은 확률 p_2의 차이가 클 작은 경우에도 오류로 간주한다. 일반적으로 확률 분포가 균등할 경우, 그 시스템의 엔트로피(entropy)가 가장 높다. 즉, 엔트로피가 높을 경우에 시스템에 불안하여 오류가 발생된다. 따라서 p_1과 p_2가 거의 차이가 없다면 오류로 간주할 수 있다.

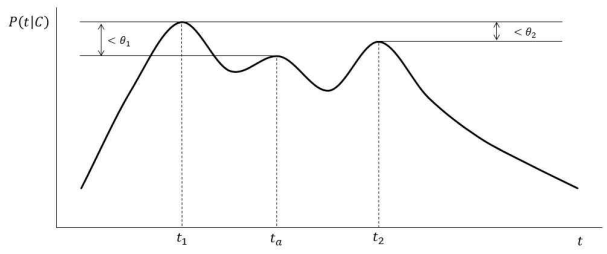

그림 4. 가정 2의 개념 그래프

여기서 기준값에 해당하는 θ_1과 θ_2을 매개변수(hyper-parameters)로서 실험을 통해 결정되며, θ_2는 θ_1에 비해 매우 작은 값이 될 것이다($\theta_2 \ll \theta_1$). 이 값들을 설정하기 위해서는 오류가 부착된 말뭉치(error-tagged corpus)가 필요하나, 본 논문에서는 같은 시스템을 사용한 품사부착말뭉치에서의 오류 탐지에서의 값들을 사용하였다[10][2].

3.3 오류 후보 선택

이미 구축된 말뭉치의 오류를 선정해야 하므로 모든 말뭉치를 대상으로 오류를 검출해야 한다. 본 논문에서는 기준값을 바탕으로 말뭉치 전체를 교차검증(cross validation) 방법을 통한 오류 후보를 선택한다. 그림 5와 같이 교차 검증을 통한 오류 검출 방법은 전체 데이터를 N 등분하여 하나의 조각을 제외한 나머지 조각으로 학습하고 제외한 조각에서 오류를 검출하는 방법으로 이와 같은 과정을 N번 반복하면 전체 말뭉치에 대한 오류를 검출할 수 있다.

A	B	C
Train	Train	Error detection
Train	Error detection	Train
Error detection	Train	Train

그림 5. 오류 탐지를 위한 3-겹 교차검증

4. 실험 및 평가

4.1 오류 검출을 위한 대상 말뭉치

본 실험에서는 최근에 공개된 국립국어원 구문 분석 말뭉치(버전 1.0)[3]을 사용하였다. 표 1은 성능을 평가하기 위해 사용된 구문분석 말뭉치에서 임의로 추출한 말뭉치 일부의 통계치이다.

표 1. 전체 말뭉치와 성능 평가 말뭉치의 통계

	전체 말뭉치	성능 평가 말뭉치
문장	150,082	200
어절	2,000,213	2,796

2) 엄밀히는 구문분석 환경에 적합한 매개변수를 설정하는 것이 바람직하다. 그러나 분류기의 값에 의존하여 오류를 판단하는 문제이므로 다른 문제에서도 큰 차이는 없을 것으로 판단한다.

전체 말뭉치와 성능 평가 말뭉치에서의 구문 태그 및 오류 분포는 4.3절의 표 2에서 자세히 나타낼 것이다.

4.2 기준값 설정

3.2절에서 언급한 θ_1과 θ_2를 설정하기 위해서는 오류가 포함되어 있고 수동으로 검증된 말뭉치가 필요하다. 따라서 본 실험에서는 따로 θ_1과 θ_2를 설정하지 않고 품사부착말뭉치에서의 오류 탐지에서 사용된 θ_1과 θ_2를 사용한다[10]. 사용된 θ_1는 0.015, θ_2는 0.005이다(3.2절 참조).

4.3 오류 검출

4.2절에서 정해진 θ_1과 θ_2를 바탕으로 구문분석 말뭉치 전체에 대해서 오류 후보를 선택하였다. 오류 검출은 3.3절에서 설명한 교차검증 방법으로 수행되었으며 그 결과는 표 2와 같다. 구문분석 말뭉치에서 100,794개의 구문 태그를 오류로 판단하여 약 5%를 오류로 검출하였다. 성능을 확인하기 위해서는 검출된 오류를 수동으로 확인하여야 하지만 이를 확인하는 일은 너무 많은 시간과 노력이 소요된다. 따라서 성능 평가 말뭉치에서 검출된 오류를 직접 분석하였다.

4.4 모집단과 표본집단의 오류 분석

본 논문에서 제안된 방법의 성능을 평가하기 위하여 표본 집단(평가 말뭉치)의 오류의 비율과 모집단(전체 말뭉치)의 오류 비율을 비교해 보았다. 그림 6은 표 2를 바탕으로 일정 빈도수보다 적은 구문 태그를 제외한 태그별 모집단과 표본 집단에서의 오류 비율을 비교한 그래프이다. 이를 정량적으로 분석하려고 식 (6)과 같은 상대 엔트로피(Kullback-Leibler divergence, relative entropy)[21]를 사용한다.

$$KL(p\|q) = \sum_{x \in T} p(x) \log\left(\frac{p(x)}{q(x)}\right) \quad (6)$$

여기서 T는 구문태그 집합이고, $p(x)$와 $q(x)$는 각각 표본집단(평가 말뭉치)과 모집단(전체 말뭉치)에서 구문태그 x의 오류율이다. $p(x)$가 0인 경우, 상대 엔트로피를 0으로 하였으며, 그 결과, $KL(p\|q)$는 0.038과 $KL(q\|p)$는 0.024로 표본집단과 모집단이 유사함을 알 수 있었다. 따라서 두 집단의 오류율의 거의 유사하여 제안된 방법이 타당한 방법임을 알 수 있다.

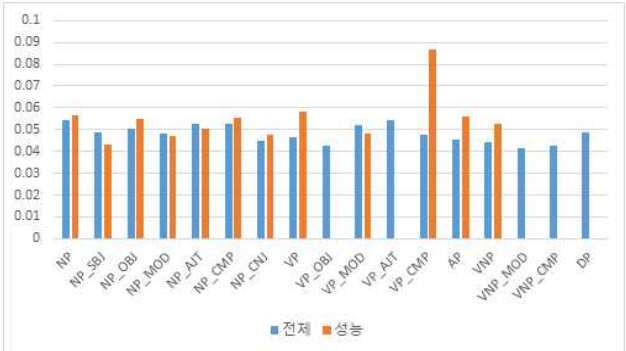

그림 6. 각 태그별 오류 비율 그래프

표 2. 말뭉치에서의 오류 탐지 결과

	전체 말뭉치		성능 평가 말뭉치	
	전체어절	오류어절	전체어절	오류어절
NP	495,441	26,812	862	49
NP_SBJ	233,042	11,348	324	14
NP_OBJ	175,530	8,819	218	12
NP_MOD	75,006	3,619	106	5
NP_AJT	263,922	13,971	338	17
NP_CMP	9,757	513	18	1
NP_CNJ	53,601	2,418	84	4
VP	294,605	13,718	359	21
VP_SBJ	2,332	132	0	0
VP_OBJ	5,153	219	3	0
VP_MOD	195,542	10,213	228	11
VP_AJT	12,254	667	12	0
VP_CMP	18,590	881	23	2
VP_CNJ	380	15	1	0
AP	69,647	3,181	107	6
AP_SBJ	5	0	0	0
AP_OBJ	3	0	0	0
AP_MOD	37	2	0	0
AP_AJT	59	1	0	0
AP_CMP	1	0	0	0
AP_CNJ	10	0	0	0
VNP	33,611	1,489	38	2
VNP_SBJ	65	1	0	0
VNP_OBJ	442	31	1	0
VNP_MOD	19,690	817	30	2
VNP_AJT	130	3	0	0
VNP_CMP	5,160	219	8	0
VNP_CNJ	79	2	0	0
DP	34,326	1,668	36	0
DP_SBJ	2	0	0	0
DP_MOD	5	0	0	0
DP_AJT	1	0	0	0
IP	299	13	0	0
IP_SBJ	2	0	0	0
IP_OBJ	1	0	0	0
IP_MOD	1	1	0	0
IP_AJT	5	0	0	0
IP_CMP	10	0	0	0
IP_CNJ	2	0	0	0
X	1,385	22	0	0
X_SBJ	8	0	0	0
X_OBJ	2	0	0	0
X_MOD	7	0	0	0
X_AJT	9	0	0	0
X_CMP	2	0	0	0
X_CNJ	10	0	0	0
L	15	0	0	0
L_MOD	3	0	0	0
R	24	0	0	0
총 합	2,000,213	100,794	2,796	146

5. 결론

본 논문에서는 XGBoost와 교차 검증을 이용하여 구축된 구문분석 말뭉치로부터 오류를 탐지하는 방법을 제안한다. 제안된 방법은 먼저 오류가 포함된 구문분석 말뭉치와 XGBoost를 사용해서 구문태그 부착기를 학습하고, 교차검증을 이용해서 구문분석 오류를 검출한다. 그러나 오류가 부착된 학습 말뭉치가 존재하지 않으므로 일반적인 분류문제로 해결할 수 없다. 따라서 매개변수를 조절하며 학습된 구문태그 부착기의 출력을 비교함으로써 오류를 검출한다. 본 논문에서는 품사부착말뭉치에서의 오류 탐지에서 조절된 매개변수를 이용한다. 또한 모집단의 모든 오류 후보를 수작업으로 확인할 수 없으므로 표본집단과 모집단의 오류 분포를 비교하여 표본집단을 바탕으로 모집단을 예측하는 본 논문의 타당성을 보였다.

앞으로 구문분석 말뭉치 환경에 적합한 매개변수를 찾아 준지도학습 기반 오류 검출 시스템을 개발할 계획이며, 또한 의미역 부착말뭉치에도 같은 방법으로 적용할 계획이다.

감사의 글

이 논문은 2019년도 정부(과학기술정보통신부)의 재원으로 정보통신기획평가원의 지원(R7119-16-1001, 지식증강형 실시간 동시통역 원천기술 개발)과 2017년도 정부(과학기술정보통신부)의 재원으로 한국연구재단의 지원을 받아 수행된 연구임(NRF-2017M3C4A7068187, 한국어 정보처리 원천 기술 연구 개발).

참고문헌

[1] 국립국어원, “21세기세종계획”, 2012.

[2] 최용석, 이공주, “한국어 구절 구문 코퍼스의 의존 구문 구조 트리로의 변환에서 중심어 전파 규칙”, *한글 및 한국어 정보처리 학술대회 논문집*, pp.514-519, 2018.

[3] 국립국어원, 국립국어원 구문 분석말뭉치(버전1.0), URL:https://corpus.korean.go.kr, 2020.

[4] 이미경, 정한민, 성원경, 박동인, “품사 표지 부착 말뭉치 검증”, *한글 및 한국어 정보처리 학술대회 논문집*, pp. 145-150, 2005.

[5] 최명길, 서형원, 권홍석, 김재훈 “한국어 품사 부착 말뭉치의 오류 검출 및 수정”, *한국마린엔지니어링 학회지*, 제37권, 제2호, pp. 227-235, 2013

[6] E. Eskin, “Detecting errors within a corpus using anomaly detection”, in *Proceedings of the 1st North American chapter of the Association for Computational Linguistics Conference*, pp. 148-153, 2000.

[7] Q. Ma, B. Lu, M. Murata, M. Ichikawa, and H. Isahara, “On-line error detection of annotated corpus using modular neural networks”, *Lecture Notes in Computer Science*, Vol.2130, pp. 1185-1195, 2001.

[8] T. Nakagawa and Y. Matsumoto, “Detecting errors

in corpora using support vector machines", in *Proceedings of the 19th international conference on Computational linguistics*, pp.1-7, 2002.
[9] M. Dickinson, "Detection of annotation errors in corpora", *Language and Linguistics Compass*, Vol.9, No.3, pp.119-138, 2015.
[10] 최민석, 김참현, 박호민, 천민아, 윤호, 남궁영, 김재균, 김재훈, "XGBoost와 교차검증을 이용한 품사부착말뭉치에서의 오류 탐지", *정보처리학회논문지: 소프트웨어 및 데이터공학*, 제9권, 제7호, pp.221-228, 2020.
[11] V. Chandola, A. Banerjee, and V. Kumar, "Anomaly detection: Survey", in *Proceedings of ACM Computing Surveys*, Vol.41, No.3, pp.15, 2009.
[12] S. Bybers, and A. E. Raftery, "Nearest-neighbor clutter removal for estimating features in spatial point", in *Proceedings Journal of the American Statistical Association*, Vol.93, No.442, pp.572-584, 1998.
[13] A. Agovic, A. Banerjee, A. R. Ganguly, and V. Protopescu, "Anomaly detection in transportation corridors using manifold embedding", in *Proceedings of the 1st International Workshop on Knowledge Discovery from Sensor Data*, pp.435-455, 2007.
[14] D. Yu, G. Sheikholeslami, and A. Zhang, "Findout: finding outliers in very large datasets", in *Proceedings of Knowledge and Information Systems*, Vol.4, No.4, pp.387-412, 2002.
[15] I. Rehbein, "POS error detection in automatically annotated corpora", in *Proceedings of the 8th Linguistic Annotation Workshop*, pp.20-28, 2014.
[16] C. Tianqi, and G. Carlos, "XGBoost: a scalable tree boosting system", in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, Vol.16, pp.785-794, 2016.
[17] T. G. Thomas, "Ensemble methods in machine learning", in *Proceedings of Multiple Classifier Systems. MCS 2000. Lecture Notes in Computer Science*, Vol.1857, 2000.
[18] 김재훈, 박은진, "PPEditor: 한국어 의존구조 부착을 위한 반자동 말뭉치 구축 도구", *정보처리학회논문지*, Vol.13-B, No. 1, pp.63-70, 2006.
[19] N. Kang, E. M. van Mulligen, and J. A. Kors, "Training text chunkers on a silver standard corpus: can silver replace gold?", *BMC Bioinformatics*, Vol.13, No.1, 2012.
[20] 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.
[21] S. Kullback, Information Theory and statistics, Dover Publications, 1968.