**Improvement of context-based meaning selection using Bi-Sent2Vec**

**경희대학교 컴퓨터공학과 캡스톤디자인2**

2015104203 이예준

**요 약**

최근 다양한 영상매체를 통해 전파되는 K-POP이나 K-DRAMA등 한류의 영향으로 자연스럽게 한국어 공부를 희망하는 외국인들의 수도 증가하고 있다. 이러한 수요에 맞추어 미리내는 다양한 한국어 교육 서비스들을 제공한다. 이 중에서 한국어 문장에 대한 영어 번역 문장의 문맥을 기반으로 각 문장 내 단어들의 영어 의미를 결정해주는 기능이 있는데, 가끔씩 문장 내 문맥이 고려되지 않은 채 결정되거나 한 문장 내에 동음이의어가 존재하는 경우 의미 결정에 오류가 생기는 문제가 있다. 이에 본 연구에서는 Bi-Sent2Vec을 통한 Cross-lingual representation과 특정 조사를 활용하여 기존 알고리즘을 재설계해 문제점 개선과 더불어 성능 향상을 시키고자 한다.

**1. 서론**

**1.1 연구배경**

최근 몇 년간 BTS로 인해 K-POP이 다시 외국에서 큰 인기를 누리고 있다. 강렬했던 2013년도 싸이의 강남스타일 이후로 K-POP이 주목받기 시작하며 꾸준하게 한류가 전파 되었고 BTS를 통해 그 인기가 더욱 커졌다. 이에 따라 수많은 해외 미디어에서 한류가 다루어지고 자연스럽게 한국어 교육을 희망하는 외국인들이 증가하고 있다는 내용도 보도되었다. 추가적으로 외국인들의 기준에서 한국어 공부가 어렵다는 것을 알 수 있었다. 외국인들의 기준에서 파열음으로 인해 발음이 어렵고 자음이 합쳐진 된소리나 특정 모음들 간의 발음 차이를 자국의 언어로 표현하지 못 하여 이를 인식하고 소리내기가 어렵다. 또한 동음이의어가 어느 나라에나 있다고 해도 표현법에 있어서도 그 가짓수가 너무나도 다양하다. 따라서 미리내는 한국어에 관심이 있지만 진입장벽이 다소 높아 어려움을 겪는 외국인들을 대상으로 보다 쉽게 한국어를 공부할 수 있도록 돕는 서비스를 제공한다. 미리내는 한국어 이해를 위해 여러 기능을 제공하는데, 본 연구에서는 문맥을 기반으로 하여 다의어의 의미를 결정해주는 Context-based meaning selection에 대해 다루고자 한다.

**1.2 연구목표**

현재 미리내에서 서비스 중인 문맥을 기반으로 한 다의어 의미 결정 기능은 다른 어느 기업에서도 제공하지 않는 서비스이다. 또한 첫 번째 버전이기 때문에 성능 면에서 개선의 여지가 존재한다. 따라서 본 연구에서는 해당 기능에 대해 조사하고 문제점을 찾아내어 기능 재설계를 통한 문제점 개선 및 성능 향상을 연구 목표로 설정하고자 한다.

**2. 관련 연구**

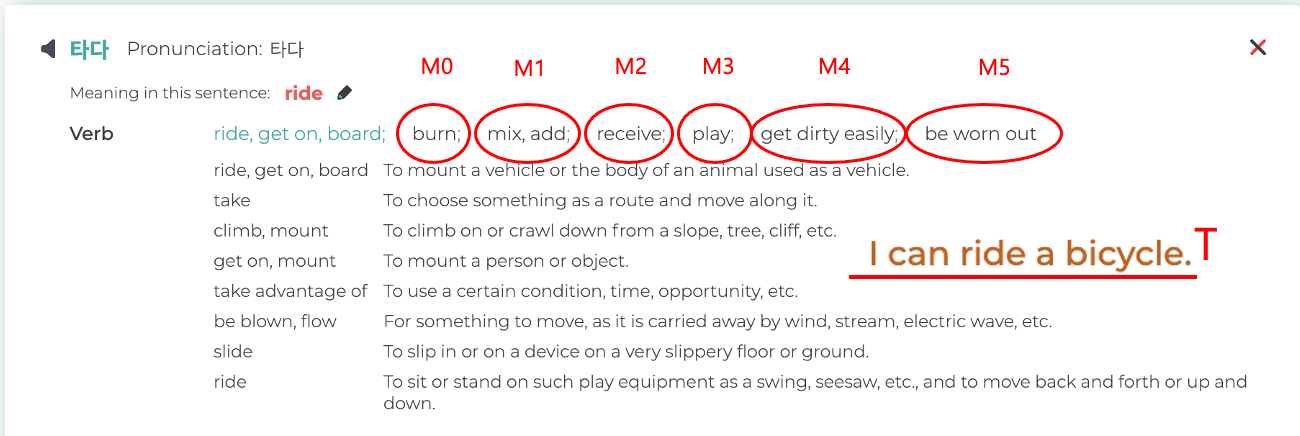
**2.1 기존 연구**

문맥 기반 다의어 의미 결정 기능(Context-based meaning selection)은 한국어 문장이 입력되었을 때 각각의 단어, 자세히 말해 의미를 가지는 가장 작은 단위인 형태소들에 대하여 해당 문장에서 어떤 의미로 사용되었는지 영어 의미를 결정해주는 것이다. 아래 [그림1]과 같이 ‘타다’를 기준으로 ride, burn, mix 등의 여러 단어가 있지만 해당 문장에서는 ride라는 의미로 쓰였음을 알려준다.



[그림 1] 기존 문맥 기반 다의어 의미 결정 기능

위의 [그림1]의 예시 문장을 가지고 계속해서 기존 기능에 대한 원리를 살펴보고자 한다. 우선 아래 [그림2]와 같이 ‘타다’에 대한 영단어 후보들을 벡터 Mn이라고 지정한다. 그리고 papago에서 해당 한글 문장에 대한 영어 해석 문장을 불러와 해당 문장에 대한 T벡터를 생성한다. 현재 미리내에서는 일반적으로 형태소 분해에는 kakao에서 오픈 소스로 제공하는 Khaiii를 통해 분해한다. 그리고 각각의 형태소에 대한 Word embedding은 fasttext를 사용해 Dense-representation으로 나타낸다. Fasttext를 사용하는 이유는 각각의 단어를 Character n-gram의 합으로 보기 때문에 각각의 Character에 대한 hash-table을 따로 생성하여 OOV(Out-Of-Vocabulary)에 대해서도 vector로 표현할 수 있기 때문이다.



[그림 2] 단어 및 문장에 대한 벡터 초기화

결과적으로 이렇게 생성해낸 벡터 Mn과 T 를 n만큼의 반복문을 통해 서로간의 유사도를 구하고 가장 유사도가 높은 벡터 M을 가진 단어를 선별해내어 영어 의미를 결정해주는 원리이다.

**2.2 기존 연구의 문제점 및 해결 방향**

**2.2.1 기존 연구의 문제점**

이와 같은 기존 연구의 방식은 다음 [그림3]과 같은 문장에서 바로 문제점이 나타난다. 각각의 벡터 M이 전체 문장 벡터인 T와 함께 유사도가 구해지는 방식이기 때문에 동음이의어가 문장에 나타난다면 한 가지의 정답으로 통일이 되어버린다. 아래 예시에선 ‘배’라는 의미가 ‘pear’로 통일되어 나타나는 것을 확인할 수 있다. 또한 ‘위’에 대하여 ‘wi’라는 영단어가 나왔는데, 이 ‘wi’라는 단어는 단순하게 한국어 모음인 ‘ㅟ’를 의미하며 이를 통해 문맥이 제대로 고려되지 못 하는 문제점도 존재한다는 것을 알 수 있다.



[그림 3] 문제점이 나타나는 예시 문장

**2.2.2 해결 방안**

우선 바로 위의 절에서 2가지의 문제점을 파악할 수 있다. 그리고 각각의 원인들은 서로 상관관계를 가지고 있다. 우선 동음이의어에 대한 해석을 한국인의 입장에서 보았을 때 자연스럽게 첫 번째로 등장하는 ‘배’는 ‘boat’이고 두 번째 ‘배’는 ‘pear’임을 알 수 있다. 첫 번째 ‘배’를 해석하는 데에는 뒤에 따라오는 ‘위 에서’라는 단서를 통해서 짐작할 수 있고 두 번째 ‘배’는 서술어인 ‘먹다’를 통해서 파악할 수가 있다. 하지만 기존 방식으로 추적해보면 첫 번째 ‘배’의 의미를 결정 지을 때 벡터 T가 전체 문장을 지칭하기에 이와는 상관이 없는 ‘먹다’까지 포함이 되어서 결정이 되었고 두 번째 역시 필요하지 않은 부분인 ‘위 에서’가 같이 포함되었기 때문에 결과적으로 문장을 구성하는 데에 가장 큰 힘을 가지고 있는 서술부와 관련된 ‘pear’가 나오게 되었음을 알 수가 있다. 그리고 ‘wi’와 같은 경우, 유사도를 구하는 벡터 T가 전체 문장을 표현하기 때문에 ‘위’에 대한 후보 단어들이 연관성을 짓기에 모호해 나온 결과임을 유추해볼 수가 있었다.

따라서 문장 벡터인 T를 상황에 맞게 부분적으로 추출해보고자 한다. 가령 첫 번째 ‘배’에 대해 의미를 결정할 때에는 전체 문장인 ‘I eat a pear on a boat’를 벡터 T로 만드는 것이 아니라 ‘on a boat’를 벡터 T로 선정하고 두 번째 ‘배’에 대해서는 벡터 T를 ‘eat a pear’로 생성해내는 것이다. 또한 ‘wi’와 같이 문맥 고려가 약한 부분을 보완하고자 주변 단어들을 통해 중심 단어를 예측하는 CBOW(Continuous Bag of Words) 방식에 착안하여 의미 결정하는 부분의 형태소 주변 형태소들에 대한 정보들과의 연관성도 함께 따져 총 두 가지 score를 합산하는 방식을 도입해보고자 한다.

**3. 프로젝트 내용**

**3.1 시나리오**

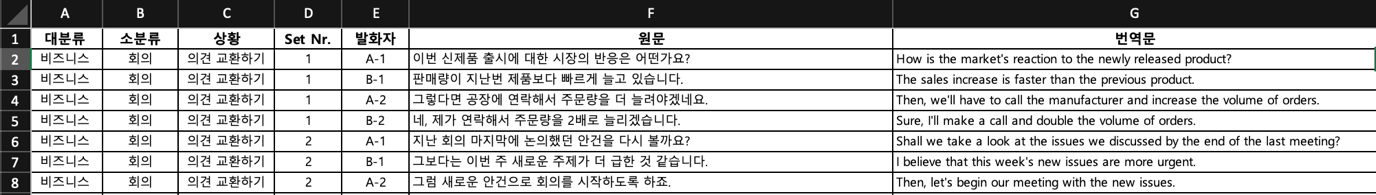


[그림4] Default GUI

본 연구를 진행하는 데에 있어서 중간 테스트 등은 미리내 서버를 local상태에서 확인할 것이며 시연에서의 비교대상 역시 local상태로 개선점 및 차이점을 확인하고자 한다.

**3.2 요구사항**

**3.2.1 Word embedding을 위한 KO-EN영어 corpus**

****

[그림5] AI-hub data 중 일부

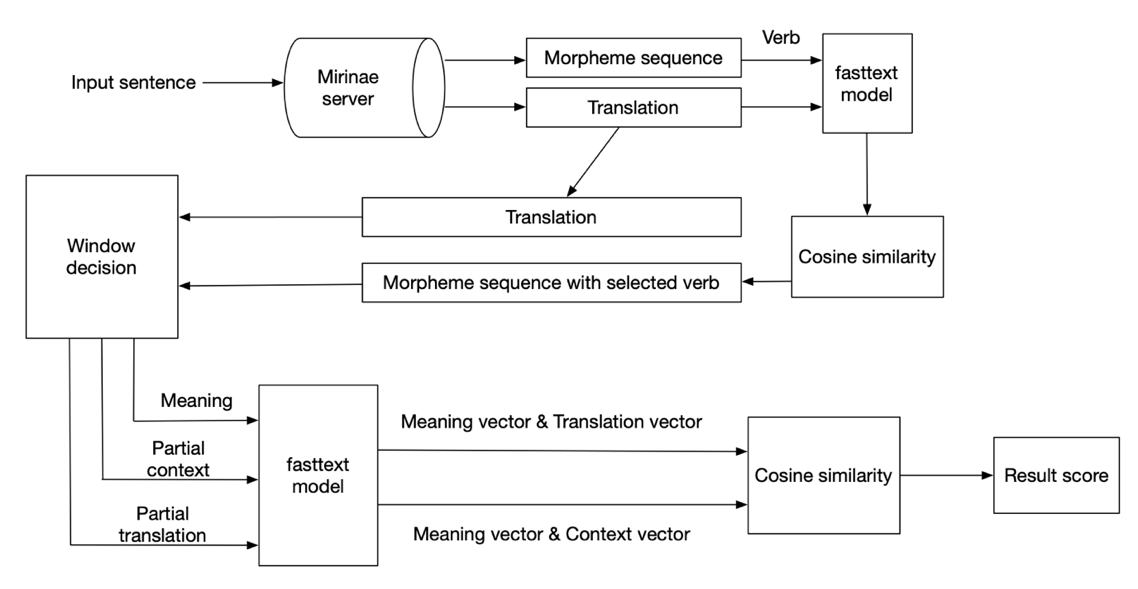
기존 연구에서는 유사도를 구하는 과정에서 오로지 영어 단어만이 쓰였기 때문에 fasttext를 통해 영단어들에 대해서만 Word embedding과정을 거쳤다. 하지만 이번에 재설계하고자 기능 구현을 위해선 의미를 결정짓고자 하는 한글 형태소에 따른 유의미한 Translation area를 부분적으로 찾아내야 하기 때문에 한글 형태소 역시 동시에 학습을 시켜야 한다. 그래서 현재 미리내에서 보유하고 있는 AI-hub의 한글-영어 문장 Corpus를 사용한다. 대략 140만개 가량의 Sentece-pair로 이루어져 있으며 구어체, 대화체, 문어체 등으로 구성되어 있다.

**3.2.2 Information of input sentence**

기능이 동작하기 위해 입력된 문장에 대한 정보를 알 수 있어야 한다. 이 정보는 미리내만의 자원이기 때문에 그 구성을 정확하게 서술할 수는 없지만 일반적으로 문장이 형태소 단위로 분해되어 있고 각각 indexing 정보가 있으며 각 형태소에 대해 국립국어원에서 가져온 매칭되는 영단어 후보들을 얻을 수 있다. 그리고 papago를 통해 English translation을 받아온다.

**3.3 시스템 설계**

전체적인 시스템 구성은 아래 [그림6]과 같다. 문장이 입력되고 미리내 서버에서는 입력된 문장을 형태소 단위로 분해하는 과정을 거친다. 이를 통해 Morpheme sequence가 나오고 추가적으로 papago를 통해 해당 입력된 문장에 대한 영어 Translation값도 반환해준다. 의미 결정을 하는 데에 정확도를 높이고자 문장 해석에 가장 큰 영향을 끼치는 서술부를 우선적으로 의미 결정하는 과정을 거친다. Translation과 Morpheme-sequence에서 Verb만을 가져와 fasttext를 통해 학습된 model에 통과시켜 각각의 형태소에 vector를 생성해낸 뒤에 서술부 의미를 결정해준다. 이로써 Morpheme sequence with selected verb가 탄생한다. 이 다음은 각각의 형태소에 대하여 의미를 결정짓는 데에 영향을 끼치는 범위를 부분적으로 지정하기 위한 Window decision algorithm으로 향한다. [그림3]의 문장을 예로 들자면, 첫 번째 ‘배’에 대해서 Meaning은 pear, boat, stomach 등이 될 것이고 Partial context는 ‘위’, ‘에서’가 될 것이다. 마지막으로 Partial translation은 ‘on’, ‘a’, ‘boat’가 된다. 이를 통해 선택된 형태소들은 다시 fasttext model을 통과하여 벡터로 치환된다. 마지막으로 Cosine similarity를 통해 Meaning vector와 Translation vector의 유사도를 Result score에 더하고, Meaning vector와 Context vector의 유사도를 추가적으로 score에 합산시켜 가장 높은 score를 보인 Meaning을 찾아낸다.



[그림 6] New context-based meaning selection system architecture

**3.4 구현**

**3.4.1 Preprocessing training data**



[그림 7] English, Korean tokenize function

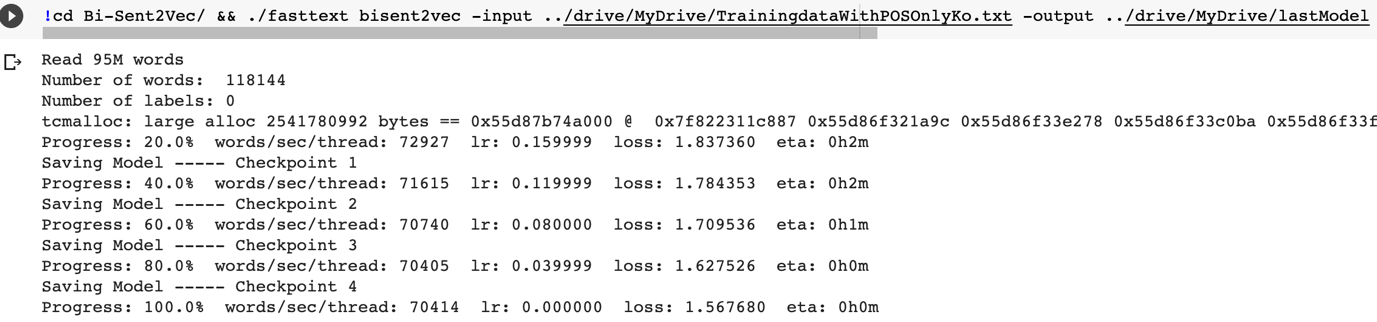
형태소가 주어지고 이에 대한 Vector를 생성해내기 위해선 이를 위한 Trained model이 필요하다. 물론 그 이전에 Model을 훈련시키는 데에 필요한 training data에 대한 전처리 과정이 필요하고 자연어처리에서 정말 중요한 부분이다. 이에 따라 결과 차이가 정말 커지기 때문이다. 우선 [그림5]와 같이 엑셀확장자로 정리가 된 한국어-영어 Senctence-pairs를 읽어들인 후 한국어는Khaiii를 통해 형태소 분해과정을 거치고 영어 문장은 nltk를 사용해 분해한다. 한국어 형태소 분석기는 Khaiii 이외에도 여러 개가 있지만 Mirinae server를 통해 받는 Morpheme sequence들은 전부 Khaiii를 활용한 것이기 때문에 차이를 없애고자 같은 형태소 분석기를 선택했다.



[그림 8] 한국어 문장 <<split>> 영어 문장

[그림 7]의 함수들을 통해 전처리 과정을 거치고 나면 위의 [그림8]과 같은 형태의 Sentence-pair가 생성된다. Fasttext기반의 Embedding을 차용하는 Bi-Sent2Vec의 training form을 맞춘 결과이다. 한국어 문장엔 형태소별로 \_ko가 붙어있고 영어 문장엔 \_en이 붙어있다. 특이하게 한국어 형태소들에는 :JKO 등의 Tag가 추가적으로 붙어있는 것을 확인할 수 있는데, 이는 위의 시스템 구성도에서 볼 수 있는 Window-decision에서 형태소의 종류에 따라 가변적인 window-size를 설정하기 위해서 꼭 필요한 정보이다.

**3.4.2 Training model with Bi-Sent2Vec**

****

[그림9] Training 과정

준비된 Training data를 가지고 Bi-Sent2Vec을 통해 학습을 진행한다. 총 4번의 Checkpoint를 거쳐 학습이 완료되는데 결과물로 .bin확장자와 .vec확장자 파일이 생성된다. 벡터 확장자로 된 모델은 단순히 학습에 사용됐던 형태소들에 대해서만 벡터를 반환할 수 있고 바이너리 파일로 된 모델은 학습에 사용되지 않았던 OOV에 대해서도 character별로 hash-map을 조회해 벡터를 생성해낼 수 있다.

**3.4.3 Window decision**

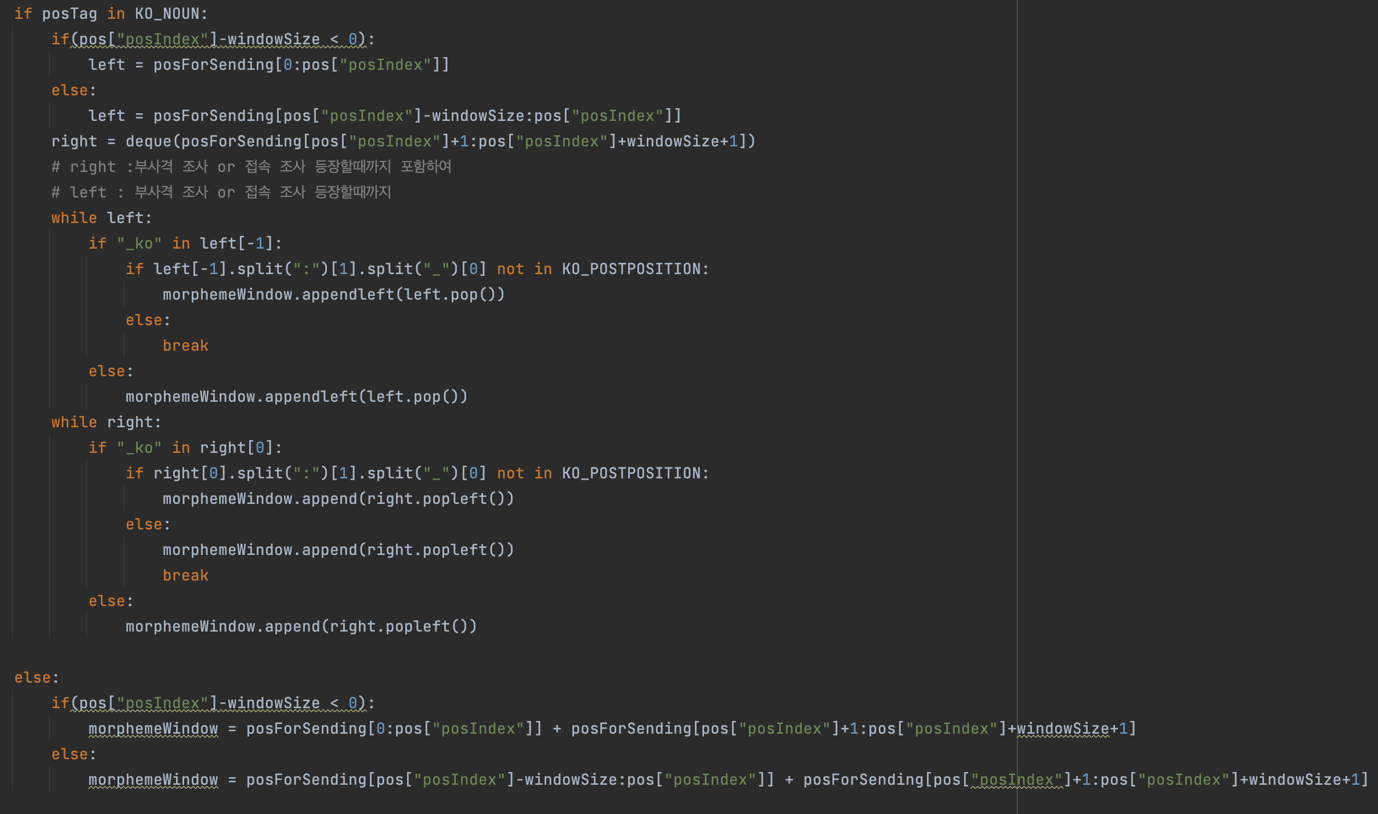
이제 본 연구에서 가장 핵심적인 부분에 대해 서술하고자 한다. 바로 Partial context와 Partial translation이 생성되는 부분이다. 이후 3.4.4절에서 시간 순서로 동작하는 코드 설명에 앞서 Partial context와 Partial translation의 의미를 명확히 하고자 한다. 본 연구에서 언급하는 Window는 총 2가지를 지칭한다.

**3.4.3.1 Variable length window of context for each morphemes with breaking rule**

첫 번째 Window의 의미를 설명하기 위해 예를 들어 ‘배 위에서 배를 먹었다.’라는 문장이 있다고 가정해보자. 일반적으로 한국인이 해당 문장에서 2가지 ‘배’에 대해서 해석을 시도할 때, 첫 번째 ‘배’에 대해서는 뒤에 따라오는 ‘위에서’를 기반으로 해석하고 두 번째 ‘배’는 뒤에 따라오는 ‘를 먹었다’에서 어떤 의미로 사용되었는지 추측할 수 있다. 이때 해석을 하는 데에 수집한 유의미한 정보들의 범위를 Window라고 한다. 본 연구에서는 의미를 결정짓고자 하는 형태소에 대하여 앞과 뒤 2개씩 총 4개의 형태소들을 묶어 하나의 Window라고 정의한다.

이렇게만 보면 고정 길이의 Window라고 생각할 수 있다. 하지만 여기서 또 하나의 규칙이 추가된다. 방금 제시한 Window의 규칙에 따르면 두 번째 ‘배’의 해석에 필요한 형태소는 ‘위’, ‘에서’, ‘를’, ‘먹’이 될 것이다. 그렇지만 사실상 앞의 2개의 형태소 ‘위’, ‘에서’는 필요가 없다. 왜냐하면 ‘위에서’ 자체는 뒤의 phrase를 꾸며주는 Adverbial phrase의 정보이기 때문이다. 따라서 유의미하지가 않다. 이렇듯 앞과 뒤로 2칸씩 Window가 뻗어나가는 와중에Phrase구간이 걸치게 되는 시점에서 멈출 수 있어야하기 때문에 특별한 Breaking rule을 적용했다.

일반적으로 위와 같이 Phrase가 구분되는 곳에는 특별한 조사가 존재한다. 바로 부사격조사와 접속조사이다. 이 성질을 활용하여 Breaking point의 명확한 기준을 설립했다. 이를 토대로 설계한 Variable length window algorithm이 바로 아래 [그림10]이다.

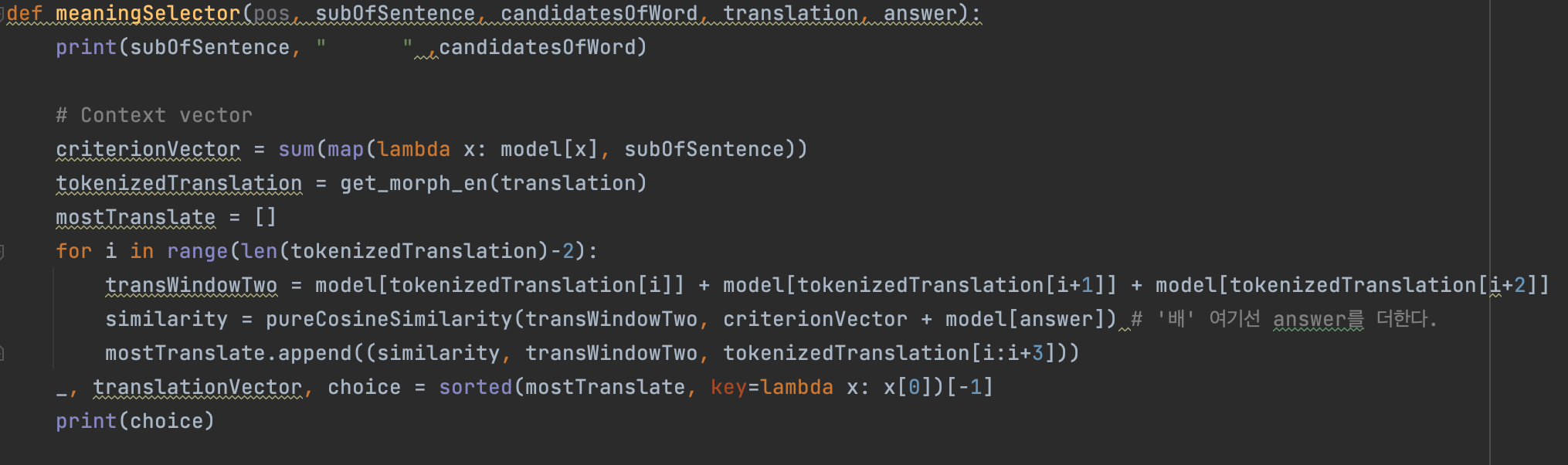


[그림10] Variable length window for each morpheme

위 [그림10]를 통해 완성되는 morphemeWindow변수를 본 연구에서는 해당 형태소에 대한 Partial context라고 정의한다. 이후 의미를 결정짓기 위한 Score 점수에 기여하는 항목 중 하나이다.

**3.4.3.2 Fixed window for partial translation**

두 번째 Window의 의미는 바로 3이라는 고정된 길이를 갖는 Translation의 일부분이다. 앞서 해결방안 절에서 서술한 바와 같이 하나의 형태소에 대한 알맞은 영어 단어 의미를 결정짓는 데에 반드시 전체 영어 문장이 쓰일 필요가 없다. 바로 위 절의 예시를 그대로 가져와보자. 다시 말해 첫 번째 ‘배’에 대한 의미를 결정짓기 위해서 ‘I ate a pear on a boat’ 전체 문장이 쓰일 필요가 없다는 것이다. 첫 번째 ‘배’에 대해선 해석 문장의 일부분인 ‘on a boat’만 있다면 충분하다. 문제는 ‘on a boat’를 어떻게 추출해낼 것인가 이다. 우리는 이미 해당 부분을 탐색하기 위한 재료를 가지고 있다. 바로 위의 절에서 구한 Partial context이다. 위에서 구현한 Variable length window algorithm을 이용하면 첫 번째 ‘배’에 대한 Partial context는 ‘위’, ‘에서’이다.



[그림11] Extract most similar partial translation

[그림 11]의 함수에서 인자로 받아오는 subOfSentence가 Partial context 이다. 우선 찾고자 하는 형태소인 ‘배’까지 더해 ‘배’ + ‘위’ +. ‘에서’에 대한 Criterion vector를 만든다. 그리고 ‘on a boat’와 같이 가운데에 관사가 포함되는 것을 고려해 탐색할 Window를 3으로 고정시킨다. 이후 Sliding window기법으로 길이 3인 Window를 한 칸씩 뒤로 밀며 Criterion vector인 ‘배 위에서’와 가장 높은 유사도를 보이는 부분을 찾아낸다. 이를 통해 ‘on a boat’를 부분적으로 추출해낼 수가 있다. 해당 마지막 라인인 choice를 출력해보면 ‘on a boat’가 나온다. choice 이외에 두 번째 변수로 translation vector가 나오는 것을 알 수 있다. 이를 Partial translation이라 정의하며 이 역시 이후에 의미를 결정짓는 Score 합산에 영향을 주는 항목 중 하나이다.

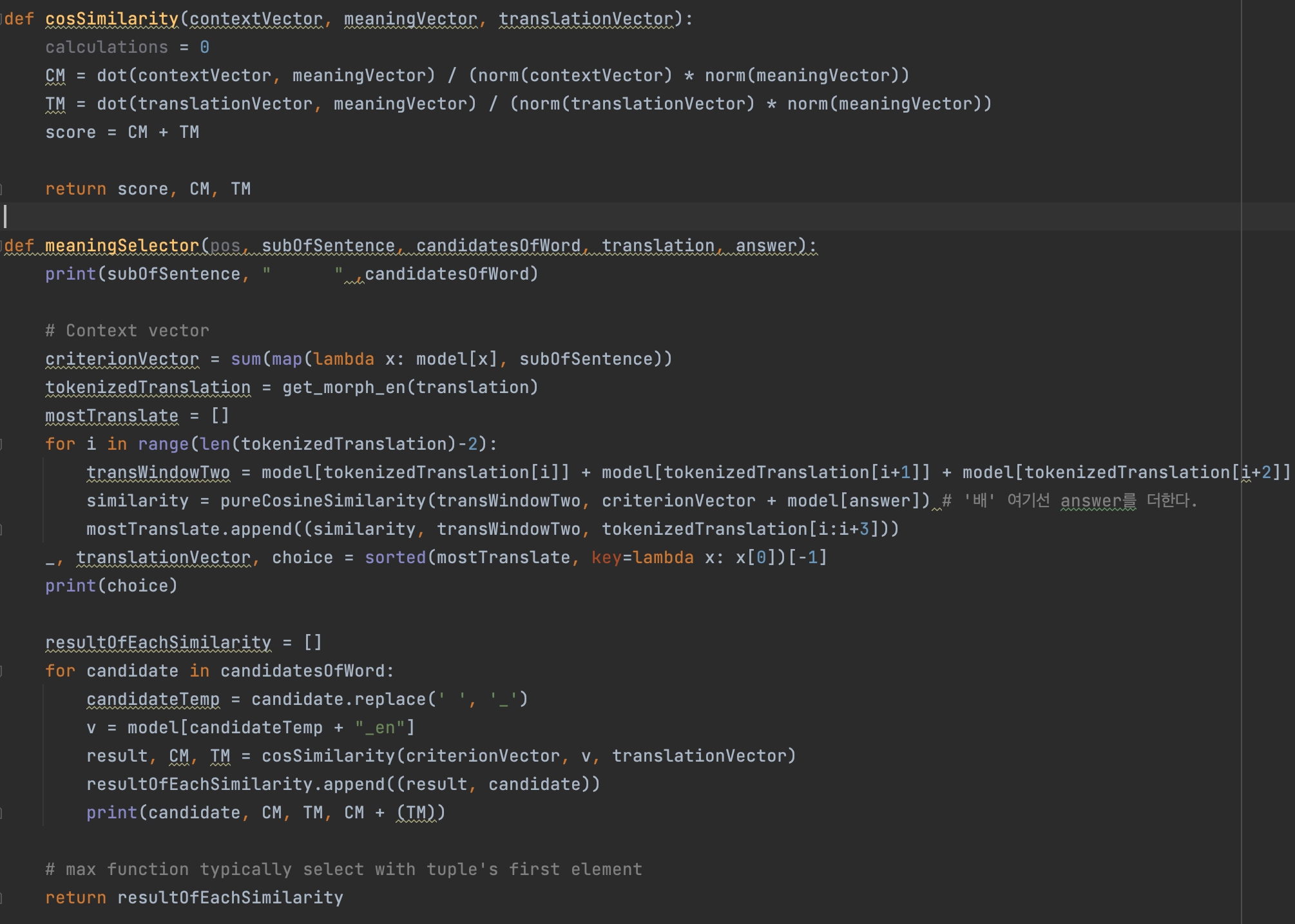
**3.4.4 How to run Context-based meaning selector**

앞서 Window 정의와 역할 등에 대하여 서술하며 시스템 구성도에서 Window decision의 output중 Partial context와 Partial translation에 대하여도 정의하였다. 나머지 하나인 Meaning은 별다른 알고리즘 과정을 거치지 않은 의미를 결정짓기 위한 해당 형태소의 영어 의미 후보들을 말한다. 이제 부분적인 구현 설명이 끝났으니 전반적인 시스템 구성도 흐름에 대하여 서술하고자 한다. 먼저 기능에 대한 성능을 정확하고 강하게 만들기 위해 문장에서 가장 큰 영향력을 가지고 있어 그 목적이 뚜렷한 서술부를 우선적으로 의미를 결정지어 준다. 아래 [그림12]처럼 posForContext를 순차적으로 조회하며 해당 형태소가 서술부인 경우 전체 문장 벡터와 해당 서술부의 후보 단어들을 비교하여 가장 유사도가 높은 것을 찾아 서술부를 가장 먼저 의미를 찾아준다. 그 다음 한글로 되어 있는 서술부를 결정된 영어 의미로 치환한다.



[그림12] Select verb priorily

바로 위 그림과 같이 우선적으로 서술부를 결정해준 뒤에 verbIdx에 posIndex에 이미 결정이 끝난 index를 추가해준다. 이후엔 posForContext를 똑같이 순회하며 서술부 이외의 의미 결정이 필요한 형태소들에 대하여 meaningSelector를 호출한다.

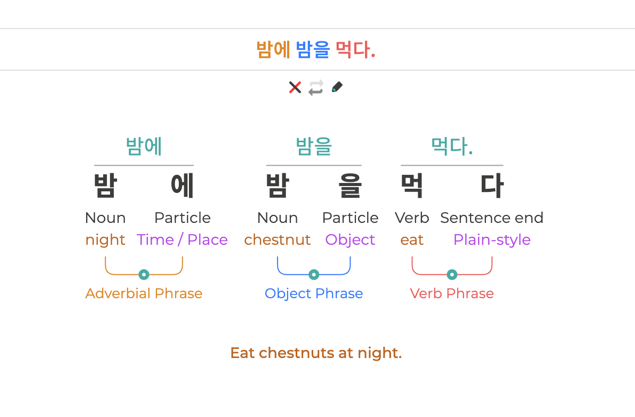
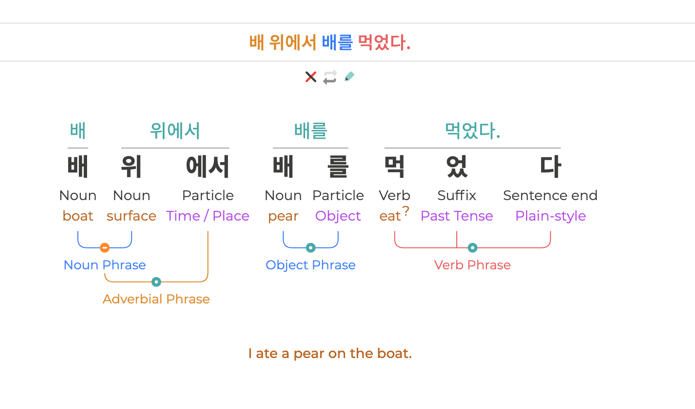


[그림 13] Meaning selector with Parital context, Meaning, Partial translation

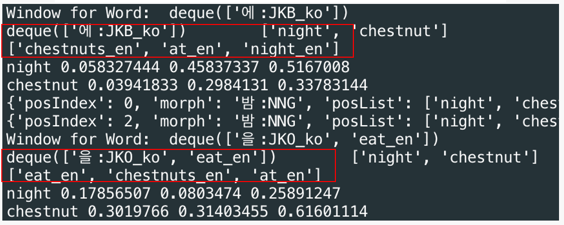
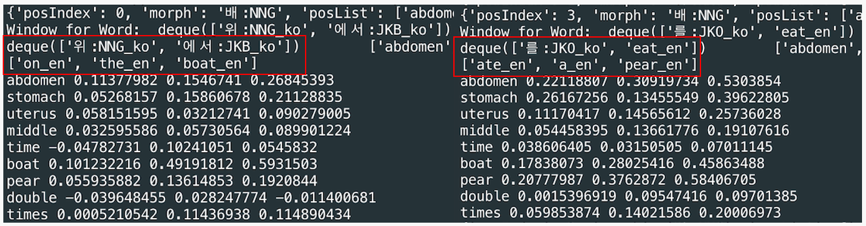
위 [그림13]와 같이 구성된 meaningSelector가 호출되며 해당 함수에서는 위의 절에서 서술한 바와 같이 subOfSentece 즉, Partial context를 통해 Partial translation을 추출하게 되고 현재 이미지 상에는 나오지 않았지만 전역변수로load시킨 훈련된 model로 각각 Context vector, Translation vector를 구해주며 각각의 후보 영어 단어들에 대한 Meaning vector도 생성해준다. 이후에 Cosine similarity함수를 호출하게 되며 합산된 Score중 가장 높은 유사도를 보이는 영어 단어로 의미를 결정지어 준다. 코드 상 CM은 Context vector와 Meaning vector의 유사도, TM은 Translation vector와 Meaning vector의 유사도를 의미한다.

**4. 프로젝트 결과**

**4.1 연구 결과**



[그림14] 결과



[그림 15] 결과에 대한 분석

위 [그림14]과 같이 ‘배 위에서 배를 먹었다.’라는 문장이 입력되었을 때 2.2.1절에서 문제제기 한 부분들이 해결된 것을 볼 수 있다. 그리고 [그림 15]에서 결과에 대한 분석이 이어진다. 먼저 빨간 박스의 Deque 자료구조에 들어가 있는 형태소들은 앞서 구현한 Variable length window algorithm을 통해 생성된 Partial context들이고 그 바로 밑 배열에 존재하는 형태소들은 Fixed 3 length sliding window algorithm을 통해 생성된 Partial translation이다. 이후 밑으로 나오는 단어들은 의미 결정을 위한 후보 단어들이며 CM, TM, CM + TM 순으로 출력되어 나타난다. 결과적으로 CM + TM이 시스템 구성도의 가장 마지막에 나타나는 Score를 의미하며 Score가 가장 높은 단어가 최종적인 의미로 결정되게 된다.

**4.2 성능 평가**

|  |  |  |
| --- | --- | --- |
| 예시 문장 | 기존 기능 오답률 | 개선 기능 오답률 |
| 눈이 오는 날 눈을 감고 기도했다. | 16.6% | 16.6% |
| 사과의 의미로 사과를 주다. | 12.5% | 0% |
| 차 안에서 차를 마신다. | 28.5% | 0% |
| 큰 다리를 건너자니 다리가 너무 아팠다. | 9% | 0% |
| 아침엔 샌드위치를 아침으로 먹는다. | 12.5% | 0% |
| 우리 아이가 첫 돌에 돌을 집었다. | 9% | 9% |
| 해변가 굴에 들어가 굴을 먹었다. | 11.1% | 0% |
| 벌을 괴롭힌 아이가 벌을 받았다. | 10% | 0% |
| 배에서 음식을 먹었더니 배가 울렁거린다. | 27.2% | 0% |
| 머리를 묶은 저 여학생은 머리가 작아서 더 예쁜 것 같다. | 6.6% | 6.6% |
| 2는 소수이다. | 20% | 0% |
| 평균 | 14.8% | 2.9% |

위의 2.2.1절에서 기존 연구의 문제점으로 의미 결정에 있어서 문맥 고려의 부족함과 무엇보다도 동음이의어에 대한 오류를 지적했고 그들에 대한 해결을 연구의 방향으로 잡고 진행하였다. 따라서 성능 평가를 위해 기존 연구 결과와의 비교를 실시했으며 예시 문장 선정은 최대한 동음이의어가 존재하는 문장들로 구성했다. 결과적으로 동음이의어 측면에서 문맥 기반 다의어 의미 결정 기능의 오답률이 14.8%에서 2.9%로 크게 감소해 성능이 상당히 좋다는 것을 확인할 수 있다.

**5. 결론**

**5.1 기대효과**

머신러닝 분야의 여러 인공 신경망들의 각 단위가 사람의 신경세포를 모방했다는 점에 착안하여 본 연구의 다의어 의미 결정 기능 재설계에 있어서 최대한 실제 사람이 문장에서 단어의 의미를 추측하는 과정을 코드로 녹여내고자 했다. 간단히 말해 단어 주변의 유의미한 정보들을 바탕으로 최대한 가능성이 높은 의미를 찾아내는 것이다. 물론 일반적으로 동음이의어를 한 문장 내에서 여러 번 사용하는 경우가 드물고 명사에 한해선 그 수가 그렇게 많진 않다. 하지만 동음이의어 문제를 해결하는 과정에서 주변 정보들의 문맥을 효율적으로 사용함으로써 기존 연구의 문맥 고려가 부족했던 점을 한층 더 강화할 수 있었다고 볼 수 있다. 실제로 본 연구에서 새롭게 개발한 Parital context와 Partial translation을 생성해 체계적으로 정답을 도출해내는 알고리즘이 기존 연구보다 효율적이라고 판단되어 현재 미리내 서비스의 기능이 본 연구의 새로운 알고리즘으로 대체되었다.

**5.2 추후 연구 방향**

본 연구에서 새롭게 재설계한 문맥 기반 의미 결정 기능은 CM과 TM에 의해 결과가 도출되기 때문에 이 두 가지에 대한 의존도가 굉장히 높다. 또한 이 둘은 서로 상호 보완적이다. 우선 CM만으로 결과를 도출하기가 어렵다. 왜냐하면 CM은 실제로 문맥상 말이 되기만 한다면 모두 높은 유사도를 나타내며 그 차이들이 미세하기 때문이다. 따라서 명확한 답을 얻을 수 없다. 또한 TM만으로도 결과를 도출하기 어렵다. TM에서의 Translation 즉, 본 연구에서 표현하는 Partial translation은 완성된 전체 영어 해석 문장에 기반하는데 처음부터 이 영어 해석 문장에 오역이 생겨버리면 잘못된 TM을 생성하게 될 수 있기 때문이다. 따라서 추후에 다양한 예시 문장들을 넣어보며CM과 TM을 적정 비율로 합산하는 방식을 도입해보거나 TM의 경우 오역에 취약한 부분을 개선할 수 있는 추가적인 로직을 도입해볼 수 있겠다.

**6. 참고 문헌**

[1] Morpheme analyzer: https://github.com/kakao/khaiii

[2] Part of speech tags: https://github.com/kakao/khaiii/wiki/%EC%BD%94%ED%8D%BC%EC%8A%A4

[3] fasttext Python module: https://github.com/facebookresearch/fastText/tree/master/python

[4] Bi-Sent2Vec with fasttext for cross-lingual representation: https://github.com/epfml/Bi-Sent2Vec

[5] AI-hub corpus: https://www.aihub.or.kr/aidata/87

[6] Mirinae: https://mirinae.io/

[7] 국립국어원: https://krdict.korean.go.kr/eng/mainAction?nation=eng

[8] NLTK tokenizer: https://www.nltk.org/api/nltk.tokenize.html