# 오타에 강건한 자모 조합 임베딩 기반 한국어 품사 태깅

서대룡○, 정유진, 강인호

네이버

daeryong.seo@navercorp.com, youjin.chung@navercorp.com, once.ihkang@navercorp.com

# A typing error-robust Korean POS tagging using Hangul Jamo combination-based embedding

Dae-Ryong Seo○, Youjin Chung, Inho Kang

Naver Corporation / NLP

**요 약**

본 논문은 한글 자모 조합 임베딩을 이용하여 오타에 강건한 한국어 품사 태깅 시스템을 구축하는 방법에 대해 기술한다. 최근 딥 러닝 연구가 활발히 진행되면서 자질을 직접 추출해야 하는 기존의 기계학습 방법이 아닌, 스스로 자질을 찾아서 학습하는 딥 러닝 모델을 이용한 연구가 늘어나고 있다. 본 논문에서는 다양한 딥 러닝 모델 중에서 sequence labeling에 강점을 갖고 있는 bidirectional LSTM CRFs 모델을 사용하였다. 한국어 품사 태깅 문제에서 일반적으로 사용되는 음절 임베딩은 약간의 오타에도 품사 태깅 성능이 크게 하락하는 한계가 있었다. 따라서 이를 개선하기 위해 본 논문에서는 한글 자모 임베딩 값을 조합시킨 음절 임베딩 방식을 제안하였다. 강제로 오타를 발생시킨 테스트 집합에서 실험한 결과, 자모 조합 임베딩 기법이 word2vec 음절 임베딩 방식에 비해 형태소 분할은 0.9%, 품사 태깅은 3.5% 우수한 성능을 기록하였다.

주제어: 한글 자모 조합 임베딩, 품사 태깅, Bidirectional LSTM CRFs

## 1. 서론

한국어 형태소 분석 및 품사 태깅은 주어진 어절을 형태소 단위로 분리하고 원형을 복원한 후, 각 형태소에 적절한 품사를 부여하는 과정이다. 한국어 형태소 분석은 기계번역, 음성인식, 개체명 인식을 비롯한 많은 자연어처리 응용 분야에서 필수적으로 사용되는 중요한 기술이다. 그러나 형태소 분석기를 구축하기 위해서는 다량의 언어 지식과 자원을 필요로 하기 때문에, 최근에는 기계 학습 방법론에 기반하여 형태소 분석 단계를 생략하고 바로 품사 태깅 시스템을 구축하려는 연구가 많이 시도되고 있다[1,2,4-7].

특히 의미 있는 자질을 직접 추출해야 하는 기존 기계학습 방법론과는 달리, 딥 러닝 모델은 자질 선정 과정 없이 스스로 의미 있는 자질을 찾아서 학습하고, 간단한 모델만으로도 기존 방법론을 뛰어넘는 우수한 성능을 낼 수 있기 때문에 최근 연구 방향에서 큰 흐름을 차지하고 있다.

딥 러닝 모델 기반의 품사 태깅 시스템들은 주로 end-to-end 방식으로 사전 정보나 자질 정보를 이용하지 않는 seq2seq 모델을 사용하거나[4,5], 또는 B(begin), I(inside) 태그를 각 음절에 맞춰서 부착하는 sequence labeling 방식을 사용한다[6,7]. 또한 품사 태깅 과정은 대부분 음절 단위로 처리를 진행하게 되는데, 이 때 음절 임베딩 값은 주로 대규모 말뭉치 상에서 pre-training 시켜 사용하게 된다. 이렇게 구축한 음절 임베딩 값에서는 서로 다른 두 음절은 완전히 다른 값을 갖고 있기 때문에 음절 간 구분이 명확해지는 장점이 있어서 일반적인 경우에는 좋은 성능을 보여줄 수 있다.

그러나 최근에는 폭발적으로 증가하는 신조어 수에 비례하여 동일한 개체명을 다양한 형태로 표기하는 사용자들이 많이 증가하게 되었으며 (예: 어벤져스, 어벤저스), 모바일 기기의 사용 빈도가 늘어남에 따라 오타의 발생 빈도 역시 높은 비율로 증가하게 되었다 (예: 햇어요). 음절 임베딩 값을 사용하는 경우에 '져'와 '저'와 같이 비록 음절의 자모 구성이 거의 유사한 음절들이더라도 두 음절의 임베딩 값이 서로 다르므로 완전히 다른 음절로 취급된다. 따라서 단어의 음절 상에 약간의 변화만 발생해도 (예: 어벤져스 -> 어벤저스) 기존에 학습한 적이 없는 생소한 단어가 되기 때문에 엉뚱한 품사 태깅 결과를 생성할 가능성이 높아지게 된다.

따라서 본 논문에서는 단어의 다양한 변이형 및 오타에 강건한 품사 태깅 시스템을 구축하기 위해 한글의 구성 원리에 착안하여 초성, 중성, 종성으로 구성된 자모 조합 기반의 임베딩 방식을 사용하였다. 자모의 구성이 비슷한 음절은 벡터 공간상에서 코사인 유사도가 높기 때문에, 단어에 일부 오타가 있더라도 매우 유사한 임베딩 값을 갖게 된다. 실험 결과, 강제로 오타를 발생시킨 테스트 집합에서 자모 조합 임베딩 방식은 word2vec을 이용해서 값을 생성 시킨 음절 임베딩 방식 대비 형태소 분할은 0.9%, 품사 태깅은 3.5% 높은 정확률을 기록하여 오타에 강건한 특성을 확인할 수 있었다.

## 2. 관련 연구

기존 품사 태깅 문제에서는 CRF 또는 SVM 기반 기계학습 방법이 많이 사용되었다. CRF와 SVM은 다양한 자질을 추출하고 이를 조합함으로써 우수한 성능을 얻을 수 있는 장점이 있으나, 자질을 추출하는 과정 자체가 어렵고 많은 비용을 요구하는 작업이기 때문에 어려움이 있었다 [1-2].

반면, 딥 러닝 모델은 별도의 자질을 추출하지 않아도 학습 과정에서 입력에 대한 추상화가 이루어지고 각각의 은닉층에서 특징에 대한 자질을 스스로 학습 한다. [3]은 미리 형태소 분할을 한 뒤 SENNA + CRF 모델을 이용하여 품사를 태깅하는 방법을 제시하였다. 형태소 단위의 단어 임베딩을 이용한 결과 98.23%의 정확률을 얻었으나, 미등록 형태소가 존재하면 성능이 낮아지는 문제가 있었다.

[4,5]는 seq2seq 모델을 이용한 방법을 제안하였다. RNN 기반이며 인코더와 디코더로 구성되어 있는 Seq2seq 모델은 특히 기계번역에서 많이 사용되는 모델이다. 또한 end-to-end 방식이기 때문에 형태소 후보 생성, 기분석 사전, 원형복원 사전 없이도 형태소 분석이 가능하다는 장점이 있다. [4]와 [5]는 음절 단위 임베딩과 attention 기법을 사용하여 seq2seq를 구현한 점에서 서로 유사하나, 각각 입력, 출력의 형식이 다르고, beam search의 이용 여부에서 서로 차이가 있다. Beam search는 각 단계에서 하나의 출력을 내는 게 아닌 top N개의 출력을 내보내는 방식이다. [5]의 실험 결과, beam search를 도입하고, 공백 정보까지 사용하면 태깅 성능 향상에 도움이 됨을 보였다.

[6]은 음절 기반의 bi-LSTM-CRFs 모델을 제안하고 있다. 음절의 임베딩 정보는 64차원으로써 word2vec으로 학습했으며, 추가로 음절의 품사 분포 벡터를 구축하여 태깅 모델에 반영하였다. 음절 임베딩 값을 bi-LSTM모델에 넣어 음절 단위로 품사 태깅을 진행한 후, 기분석 사전과 원형복원 사전을 이용하여 복합 형태소의 품사를 결정한다. 세종코퍼스 40만 어절 상에서 테스트 한 결과, 97.09%의 정확률을 기록하였다.

본 논문 역시 [6]과 마찬가지로 bi-LSTM-CRFs 모델을 사용하고 있으나, 자모 임베딩의 조합을 통해 음절 임베딩 값을 구성한다는 점에서 [6]과 차이가 있다.

## 3. 자모 조합 임베딩에 기반한 bi-LSTM-CRFs 품사 태깅

본 논문에서는 sequence labeling 문제 해결에 좋은 성능을 보이는 것으로 잘 알려진 bi-LSTM-CRFs 모델을 사용하였다. 학습은 문장 단위로 진행하였으며, 각 음절들은 초성, 중성, 종성의 자모들로 분할한 후, 이들 자모들의 임베딩 값을 조합하여 음절 임베딩 값을 구성하였다. 이후 음절 단위로 각 품사 태그와 조합된 B(beginning), I(inside), E(end) 태깅을 수행하게 된다.

### 3.1 Bidirectional LSTM CRFs 모델

Bidirectional LSTM CRFs 의 모델은 그림 1과 같이 구성된다. RNN 기반 모델은 이전 입력에 대한 결과가 현재의 입력에도 영향을 준다는 점에서 특징이 있다. 특히 품사 태깅 문제와 같이, 주변의 단어와 음절이 현재 단어와 음절 태깅에 영향을 주는 경우, 다른 딥 러닝 모델들에 비해 RNN 모델이 보다 좋은 효과를 기대할 수 있다. 이전 음절 정보와 다음 음절 정보를 모두 참조하여 현재 음절의 품사 결정에 사용하기 위해 bidirectional RNN 구조를 이용하였다.

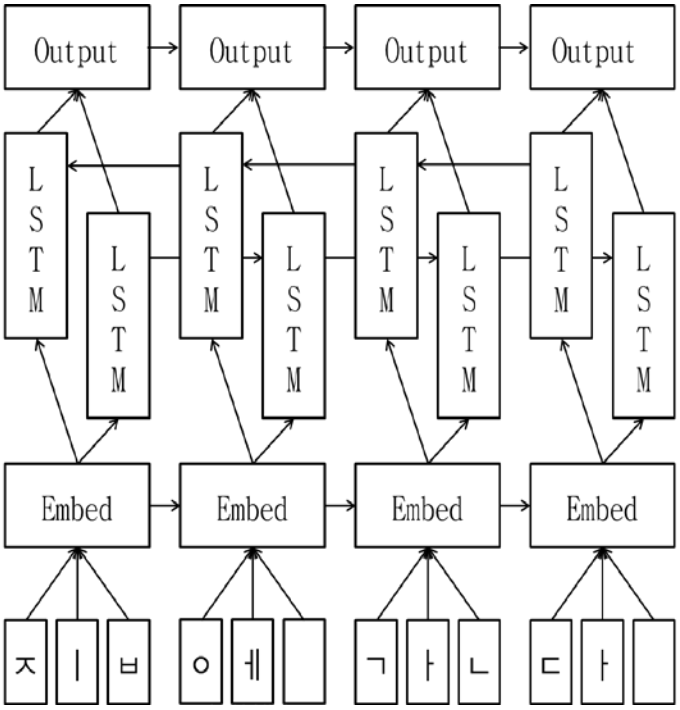


그림 1. 자모 임베딩을 이용한 Bidirectional LSTM CRFs 모델

문장 전체를 대상으로 각 음절마다 임베딩 값을 계산하여 bidirectional RNN 모델의 입력으로 넣으면 forward, backward 두 단계로 출력 값 계산이 이루어진다. 최종적으로 각 음절의 출력 값과 Viterbi 알고리즘을 이용하여 태그 사이의 전이 확률 값을 구하고 태그를 결정하게 된다. 4절의 실험 결과에서는 CRF를 사용한 모델과 사용하지 않은 모델의 성능도 함께 비교하였다.

RNN 모델은 동일한 구조의 cell이 연속적으로 이어져 있는 구조이기 때문에 gradient vanishing 문제가 발생할 수 있으나, LSTM은 cell 내부에 입력, 출력, 제거를 담당하는 게이트를 갖고 있기 때문에 긴 시퀀스에 대해서도 강건한 것으로 알려져 있다[8]. 본 논문에서도 LSTM을 사용하였으며, 시퀀스 길이 100인 경우에서 문제없이 학습이 되는 것을 확인하였다.

### 3.2 자모 조합 기반의 음절 임베딩

한글 음절은 기본적으로 초성, 중성, 종성으로 분리할 수 있다. 모음은 오직 중성 자리에만 사용될 수 있기 때문에 모두 고유한 값을 가질 수 있지만, 자음의 경우에는

“ㄶ”과 같이 받침으로만 사용될 수 있는 것을 일부 제외하면 초성과 종성 양쪽 모두에서 사용될 수 있는 것들이 대부분이다. 만약 초성과 종성을 구분하지 않고 동일한 자소에 대해서는 동일한 임베딩 값을 부여할 경우, “안 = ㅇ+ㅏ+ㄴ”, “낭 = ㄴ+ㅏ+ㅇ”으로써 “안”과 “낭”의 음절 임베딩 값이 동일해지는 문제가 발생하게 된다. 따라서 동일한 자소 “ㄱ”이더라도 초성인지, 종성인지 여부에 따라 구분하여 서로 다른 값을 갖도록 자모 임베딩 값을 생성하였다.

또한 받침이 없는 음절들을 위해 ‘종성 없음’의 의미로 <J+NONE> 태그를 만들어 임베딩에 이용하였다. 한 음절의 임베딩 값은 식 (1)에 따라 해당 음절을 구성하고 있는 자모 임베딩 값들을 weighted sum 하여 생성한다. 본 논문에서는 $\alpha = \beta = \gamma = 1$ 값을 사용하였다.

$$\mathrm{E}(\text{음절}) = \alpha * \mathrm{E}(\text{초성}) + \beta * \mathrm{E}(\text{중성}) + \gamma * \mathrm{E}(\text{종성}) \quad (1)$$

### 3.3 Layer Normalization

딥 러닝 모델 학습은 워낙 많은 계산량을 필요로 하기 때문에 학습시간이 길어지게 되는데, 이 시간을 단축시키기 위한 다양한 방법론들이 연구되고 있다. [9]는 학습 데이터를 많은 부분으로 분할시켜 여러 개의 서버에서 동시에 처리하는 방법을 사용하였으나, 데이터의 이동과 학습 모듈 자체가 복잡해지는 문제가 있었다. [10]에서는 deep neural network 내부에서 학습 데이터의 평균과 표준편차에 기반한 batch normalization 을 사용하여 학습 속도를 높였다. 하지만 batch normalization은 고정된 길이의 network에서는 잘 동작하나, RNN과 같이 입력마다 시퀀스의 길이가 달라지는 모델에서는 좋은 성능을 내기 힘들며, 학습 과정과 평가 과정에서 계산 방법이 달라지는 문제가 있었다. [11]에서 사용하는 layer normalization 은 batch normalization과 유사하게 평균과 표준편차를 이용하지만, normalize 하는 위치가 각 layer에서 이루어진다는 점에서 차이가 있다.

현재 입력 값을 $x^t$, 이전 hidden states $h^{t-1}$라고 할 때, 각 layer의 입력 $a^t$는 식 (2)와 같이 표현된다.

$$a^t = w_{hh}h^{t-1} + W_{xh}x^t \quad (2)$$

Layer에서 입력 값의 평균과 분산을 구하는 식은 (3), (4)으로 표현 가능하고, t의 hidden output은 식 (5)에 의해 구할 수 있다. g, b의 차원은 h와 동일하고 모든 time-step에서 공유되는 값이다.

$$\mu^t = \frac{1}{H}\sum_{i=1}^{H} a_i^t \quad (3)$$

$$\sigma^t = \sqrt{\frac{1}{H}\sum_{i=1}^{H}(a_i^t - \mu^t)^2} \quad (4)$$

$$h^t = f\left[\frac{g}{\sigma^t} \odot (a^t - \mu^t) + b\right] \quad (5)$$

이렇게 각 layer의 값을 정규화시킴으로써 covariate shift 문제를 해결할 수 있다. 따라서, loss 값이 수렴하는 시간이 짧아지고 학습 속도를 개선 할 수 있다. 또한 learning rate를 설정할 때 조금 큰 값을 사용하여도 loss 값이 튀지 않고 일정하게 학습이 되는 것을 확인할 수 있었다. RNN-LSTM을 사용하고 있는 본 논문에서의 실험 결과 역시, layer normalization 적용에 의해 동일 epoch 대비 빠르게 성능이 올라가는 것을 확인 할 수 있었다.

## 4. 실험 및 성능 평가

### 4.1 자모 및 음절 임베딩

자모 조합 임베딩 방식에 사용할 자모 임베딩 값은 랜덤으로 생성하여 실험을 진행하였다. 전체 자모의 개수는 기호류를 정규화한 경우에는 77개, 기호류를 정규화하지 않은 경우에는 234개 였다.

본 논문에서 제안하는 자모 조합 임베딩 방식과의 성능 비교를 위한 음절 임베딩 데이터는 word2vec을 이용한 pre-training 방식, 랜덤 초기화 방식 두 가지 방법을 이용하여 구축하였다. Word2vec을 이용한 학습에는 뉴스 50만 문서를 사용하였다. 전체 음절의 개수는 기호류를 정규화한 경우에는 5,602개, 기호류를 정규화하지 않은 경우에는 6,034개였다.

또한, 최종 품사 태그는 NNG, NNP, VV를 비롯한 43개의 세종 코퍼스 품사에 B, I, E 태그를 조합하여 43 x 3 = 129개를 생성하였다 (예: NNG-B, NNG-I, NNG-E). 여기서 B는 형태소에서 시작을(beginning), I는 중간을(inside), E는 끝을(end) 의미한다. 그리고 추가로 문장의 처음, 끝, PADDING을 나타내는 3개의 태그를 정의하여 129 + 3 = 총 132개의 태그를 사용하였다.

### 4.2 학습 및 평가 데이터 구축

학습 및 평가 데이터로는 세종코퍼스 16만 문장을 이용하였으며, RNN 학습에 15만 문장, 평가에 1만 문장을 이용하였다. 본 연구에서는 색인어 추출을 위한 품사 태거 개발을 염두에 두고 진행한 관계로, 세세하게 분리된 어미 정보는 필요하지 않기 때문에 각 어미들은 앞의 어간과 결합하여 하나의 용언으로 구성하였다. 예를 들어 “가/VV + 았/EP + 다/EF”는 어간과 어미를 결합시킨 표층형 단어에 품사 정보를 부여하여 “갔다/VV”로 변환시켜 사용했다.

RNN sequence length 는 100, LSTM의 hidden layer 개수는 256, learning rate는 0.03으로 설정하였고, 최대 140 epoch까지 학습을 진행하였다.

### 4.3 자모 조합 임베딩 기반 태깅 성능

성능 비교는 형태소 분할만 하는 경우와(segmentation) 형태소 분할에 이어 품사 태깅까지 하는 경우(POS tagging) 두 가지 문제로 구분하여 실험하였다.

먼저 형태소 단위 분할(segmentation) 문제는 입력문 “집에 간다”에 대해 품사 정보 없이 “집/에/간다”로

형태소 분할 경계를 구분하는 것까지만을 목표로 한다. 용언은 어간과 어미를 분리하지 않고 하나의 형태소로 취급하였으며, 태그셋은 B, I, E 3개를 사용하였다.

두번째로 품사 태깅(POS tagging) 문제는 형태소 분할 결과에 품사 정보까지 추가하여 "집/NNG", "에/JKB", "간다/VV"의 결과를 얻는 것까지를 목표로 한다. 모델 학습에는 앞서 4.1절에서 설명한 132개의 태그셋을 사용하였다. 자모 임베딩 및 음절 임베딩은 모두 동일하게 256차원으로 설정하였다.

표 1은 자모 조합 임베딩을 사용하여 실험한 결과를 보여주고 있다. 기존의 논문들에서도 확인된 바 있듯이, 일반적으로 CRF를 사용하는 경우 품사 태깅 성능이 더 향상되는 것을 확인할 수 있었으며, 특히 기호류를 정규화 하지 않고 그대로 사용하는 편이 정규화 시킨 경우에 비해 더 좋은 성능을 기록하였다. 이는 형태소 분할 뿐만 아니라 품사 태깅 과정에서 특별한 기호들의 정보가 주변 음절의 태깅 성능 향상에 도움을 주는 것으로 보여진다.

표 1. 자모 조합 임베딩 사용 시 정확률 (256차원)

| 학습 방식 | 형태소 분할 | 품사 태깅 |
|---|---|---|
| 기호류 정규화 | 98.7% | 96.9% |
| CRF + 기호류 정규화 | 98.7% | 97.2% |
| 기호류 비정규화 | 99.2% | 97.1% |
| CRF + 기호류 비정규화 | **99.2%** | **97.4%** |

### 4.4 음절 임베딩 기반 태깅 성능

표 2는 4.1절에서 기술한 바와 같이 뉴스 50만 문서상에서 word2vec으로 pre-training 시킨 음절 임베딩 값을 이용해서 실험한 결과이다. 4.3절의 자모 조합 임베딩 결과와 비슷하게 CRF를 이용하는 경우 형태소 분할에서는 큰 차이가 없었으나 품사 태깅 시에는 음절 임베딩 방식이 약간 더 높은 정확률을 기록하였다. 그림 2의 결과에서 볼 수 있듯이, 일반적인 문서 분석의 경우에는 word2vec 음절 임베딩 성능이 자모 조합 임베딩에 비해 전반적으로 비슷하거나 또는 0.1~0.2% 정도 높은 성능을 보여주는 것을 확인할 수 있다.

Word2vec 음절 임베딩 방식과의 성능 비교를 위해 별도의 학습 과정 없이 랜덤으로 초기화하여 생성한 음절 임베딩 데이터를 이용한 실험도 진행하였다. 표 3는 랜덤으로 값을 생성한 음절 임베딩 데이터를 사용했을 때의 성능이다. 표 2,3에서 보이듯이, 음절 임베딩 방식은 대규모 말뭉치 상에서 학습시켜 사용하거나 또는 랜덤으로 초기화시켜 사용하거나 어느 방식이건 관계없이 둘 간에는 성능 상의 차이가 거의 발생하지 않음을 확인할 수 있었다.

표 2. Word2vec 음절 임베딩 사용 시 정확률 (256차원)

| 학습 방식 | 형태소 분할 | 품사 태깅 |
|---|---|---|
| 기호류 정규화 | 98.7% | 97.1% |
| CRF + 기호류 정규화 | 98.8% | 97.3% |
| 기호류 비정규화 | 99.3% | 97.3% |
| CRF + 기호류 비정규화 | **99.3%** | **97.5%** |

표 3. 랜덤 음절 임베딩 사용 시 정확률 (256차원)

| 학습 방식 | 형태소 분할 | 품사 태깅 |
|---|---|---|
| 기호류 정규화 | 98.7% | 97.1% |
| CRF + 기호류 정규화 | 98.6% | 97.2% |
| 기호류 비정규화 | 99.3% | 97.3% |
| CRF + 기호류 비정규화 | **99.3%** | **97.5%** |

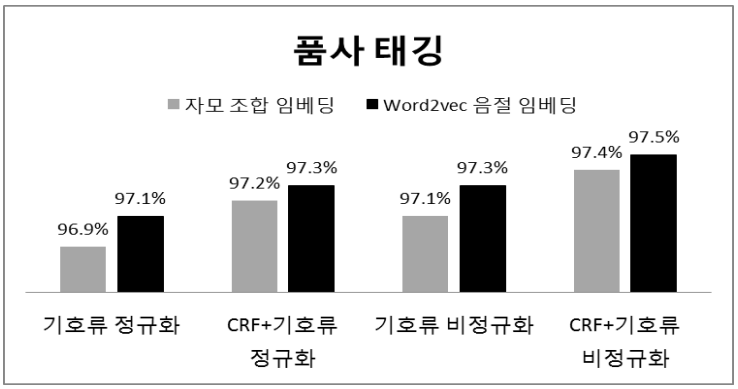


그림 2. 학습 방식별 품사 태깅 결과 비교

### 4.5 자모, 음절 임베딩 차원에 따른 성능 비교

입력으로 사용되는 임베딩 벡터의 차원을 줄이더라도 동일한 결과를 얻을 수 있다면 메모리와 속도 측면에서 이득을 얻을 수 있다. 따라서 최적의 임베딩 크기를 찾기 위해 자모 및 음절 임베딩 벡터를 32, 64, 128, 256 차원으로 변화시키면서 성능을 비교해 보았다.

표 4는 각 임베딩 차원에 따른 품사 태깅 정확률 추이를 보여주고 있다. 실험 결과를 보면 어떤 임베딩 방식을 사용하더라도 전체적으로 임베딩 크기가 증가할수록 태깅 성능이 비례하여 향상되는 경향을 확인할 수 있다.

또한 McNemar's test를 이용하여 word2vec 128차원 결과와 256차원 결과를 비교한 결과 p-value 값으로 0.015가 나왔고, 통계적으로 유의미하다는 결과를 얻을 수 있었다. 본 논문에서는 임베딩 벡터의 크기를 256차원으로 결정하여 사용하였다.

표 4. 임베딩 차원에 따른 품사 태깅 정확률 변화 (CRF+기호류 비정규화)

| 임베딩 차원 | 자모 조합 임베딩 | Word2vec 음절 임베딩 | 랜덤 음절 임베딩 |
|---|---|---|---|
| 32 | 97.0% | 96.0% | 96.6% |
| 64 | 97.3% | 97.1% | 97.0% |
| 128 | 97.3% | 97.4% | 97.3% |
| 256 | **97.4%** | **97.5%** | **97.5%** |

### 4.6 오타가 빈번한 환경에서의 성능 비교

다수의 오타가 발생하는 경우에서의 성능 평가를 위해 평가 데이터에서 각 어절 당 1개씩의 자모를 랜덤으로 변경시켰다. 그 후 평가 데이터 상에서 4.3, 4.4절과 동일한 조건으로 성능 평가를 진행하였다. 예를 들어 "중요한 일을 앞두고"이라는 문장이 있다면, 랜덤으로 한 어절 당 1 개씩 자모 변경이 적용되어 "중**쵸**한 일**일** 앞두**교**"과 같은 문장이 생성된다. 단, 자모가 변경되었더라도 품사는 변경 없이 원래 값을 그대로 유지하였다.

표 5,6은 강제로 오타를 생성한 문장에서의 형태소 분할 및 품사 태깅 성능을 보여주고 있다. 일반 문장 분석 시에는 자모 조합 임베딩, word2vec 음절 임베딩, 랜덤 음절 임베딩의 성능이 거의 비슷했던 것에 비해 (표 1,2,3), 강제 오타를 발생시킨 경우에는 자모 조합 임베딩의 성능이 word2vec 음절 임베딩 방식보다 모든 조건에서 우월하게 나타나는 것을 확인할 수 있다 (표 5,6). 자모 조합 임베딩이 형태소 분할만 하는 경우에는 0.9%, 품사 태깅 시에는 약 3.5% 이상 높은 정확률을 기록하였다.

음절 임베딩을 사용하는 경우에는 자모 한 개만 달라져도 완전히 다른 단어로 간주되기 때문에, 오타가 존재하는 경우 오타를 낸 형태소뿐만 아니라 주변 형태소까지 제대로 분석하지 못하는 경우가 발생한다. 하지만 자모 조합 임베딩 방식에서는 자모 한 개의 변형 정도로는 기존 단어와 거의 유사한 임베딩 값을 유지하게 된다. 따라서 이와 같은 특성 덕분에 표 7에서 확인 할 수 있듯이 자모 조합 임베딩은 word2vec 음절 임베딩보다 오타에 강건한 분석 결과를 생성해 낼 수 있다.

표 5. 강제 오타 생성한 경우 형태소 분할 정확률 비교

| 학습 방식 | 자모 조합 임베딩 | Word2vec 음절 임베딩 |
|---|---|---|
| 기호류 정규화 | 96.3% | 95.9% |
| CRF + 기호류 정규화 | 96.3% | **96.0%** |
| 기호류 비정규화 | 96.9% | 95.8% |
| CRF + 기호류 비정규화 | **96.9%** | 95.9% |

표 6. 강제 오타 생성한 경우 품사 태깅 정확률 비교

| 학습 방식 | 자모 조합 임베딩 | Word2vec 음절 임베딩 |
|---|---|---|
| 기호류 정규화 | 88.2% | **84.9%** |
| CRF + 기호류 정규화 | 88.2% | 84.7% |
| 기호류 비정규화 | 88.4% | 84.7% |
| CRF + 기호류 비정규화 | **88.4%** | 84.8% |

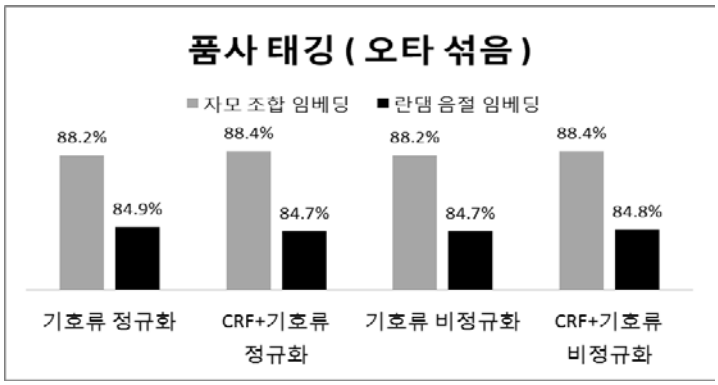


그림 3. 학습 방식별 오타 데이터 품사 태깅 결과 비교

[정답]
중요/NNG+한/XSV 일/NNG+을/JKO 앞두고/VV

-> 강제 오타 생성 (어절당 한 개)
중**쵸**/NNG+한/XSV 일/NNG+**일**/JKO 앞두**교**/VV

[자모 조합 임베딩 태깅 결과]
**중쵸/NNG**+한/XSV (O) 일/NNG+**일/VCP** (X) **앞두교/VV** (O)

[랜덤 음절 임베딩 태깅 결과]
**중쵸한/VA** (X) 일/NNG+**일/VCP** (X) **앞두교/NNG** (X)

그림 4. 임베딩 방식 별 품사 태깅 결과 비교

### 4.7 Layer Normalization 적용 결과

그림 5, 6는 학습 시 LSTM에 layer normalization을 적용한 경우와 그렇지 않은 경우의 loss 및 정확도 추이를 보여주고 있다. Layer normalization을 사용하면 훨씬 더 빠르게 loss값이 수렴하는 것을 확인할 수 있으며, 약 절반 정도의 epoch만으로도 동일한 정확률에 도달함으로써 학습 시간을 단축 할 수 있었다.

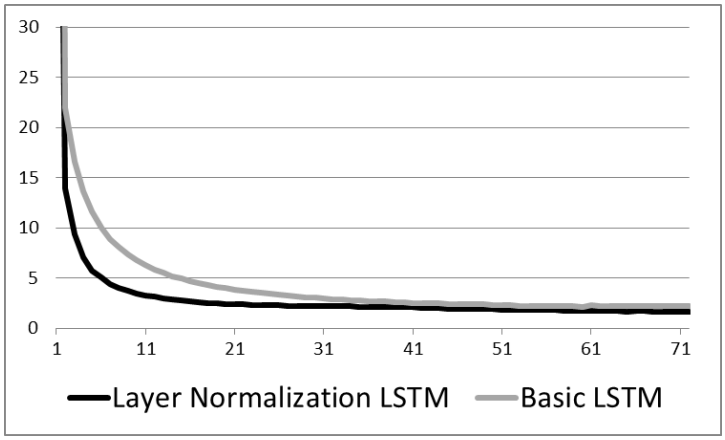


그림 5. LN_LSTM과 LSTM의 Loss 값 변화

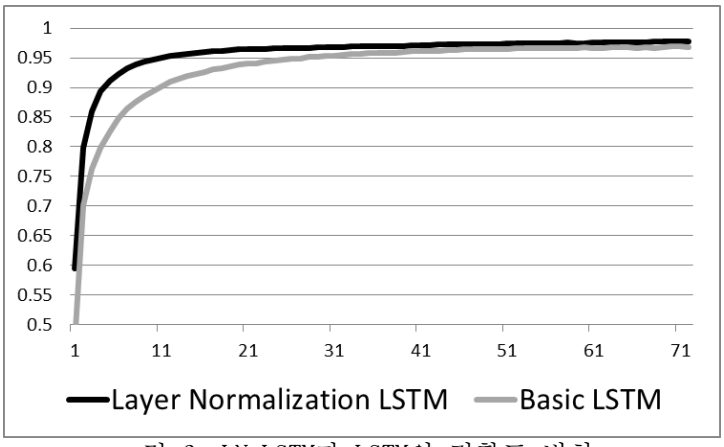


그림 6. LN_LSTM과 LSTM의 정확도 변화

## 5. 결론

본 논문은 기존의 딥 러닝 기반 품사 태깅 방법론들에서 사용하던 pre-training 시킨 음절 임베딩 값이나 랜덤 초기화된 음절 임베딩 값을 이용하지 않고 한글 자모 임베딩 값을 조합시킨 음절 임베딩 값의 사용을 제안하였다. 음절 임베딩 방식은 오타가 있거나 또는 기존에 학습되지 못한 미등록어가 들어오는 경우 이로 인한 영향을 많이 받기 때문에 품사 태깅 성능이 크게 하락하는 문제가 있었다. 그러나 자모 조합 임베딩을 사용하면 오타 및 변이형에 강건한 품사 태거 구축이 가능함을 실험으로 보였다. 자모 조합 임베딩의 이러한 특성은 신조어가 끊임없이 증가하고 오타가 빈번하게 발생하는 모바일 환경에서의 품사 태깅 시 안정적인 태깅 성능을 유지하는데 큰 도움이 되리라 생각된다. 또한 layer normalization을 적용하면 훨씬 빠른 학습 모델이 수렴이 가능하여 학습 시간 단축에 매우 효과적임을 확인할 수 있었다.

향후에는 보다 실제 조건에 근접한 오타 발생 실험을 위해, 랜덤 자모 변경이 아닌 실제 물리적인 키보드 또는 모바일 자판 상의 인접 키 위주로 오타 생성 실험을 진행해보면 더 의미 있는 결과를 얻을 수 있을 것으로 예상된다.

## 참고문헌

[1] 심광섭, “형태소 분석기 사용을 배제한 음절 단위의 한국어 품사 태깅”, 인지과학 2011.

[2] 나승훈, 양성일, 김창현, 권오욱 “CRF에 기반한 한국어 형태소 분할 및 품사 태깅”, 한글 및 한국어 정보처리 2012.

[3] 나승훈, 정상근. “딥 러닝에 기반한 한국어 품사 태깅”, 동계학술발표회 2014.

[4] 정의석, 박전규. “seq2seq 주의집중 모델을 이용한 형태소 분석 및 품사 태깅”, 한글 및 한국어 정보처리 2016.

[5] 이건일, 이의현, 이종혁, “Sequence-to-sequence 기반 한국어 형태소 분석 및 품사 태깅”, 한국정보과학회 2016.

[6] 김혜민, 윤정민, 안재현, 배경만, 고영중. “품사 분포와 Bidirectional LSTM CRFs를 이용한 음절 단위 형태소 분석기”, 한글 및 한국어 정보처리 2016

[7] 이충희, 임준호, 임수종, 김현기, “기분석사전과 기계학습 방법을 결합한 음절 단위 한국어 품사 태깅”, 한국정보과학회 2016

[8] 이창기, “Long Short-Term Memory 기반의 Recurrent Neural Network를 이용한 개체명 인식”, 한국컴퓨터종합학술대회 2015

[9] Jeffrey Dean, Greg Corrado, Rajat Monga, Kai Chen, Matthieu Devin, Mark Mao, Andrew Senior, Paul Tucker, Ke Yang, Quoc V Le, et al. “Large scale distributed deep networks.” NIPS 2012.

[10] Sergey Ioffe, Christian Szegedy. “Batch normalization: Accelerating deep network training by reducing internal covariate shift.” ICML 2015.

[11] Jimmy Lei Ba, Jamie Ryan Kiros, Geoffrey E. Hinton. “Layer Normalization.” NIPS 2016.