AI in Kakao

카카오미니의 명령어 분류 방법

사용자가 카카오미니에게 명령을 내렸을 때 카카오미니로 입력된 음성 신호는 먼저 텍스트 형태로 변환된다(음성인식). 변환된 텍스트는 어떤 분야와 관련된 기능을 수행하기 위한 명령인지 식별된 뒤(도메인(domain)/봇(bot) 분류), 해당 분야의 기능 중에서 어떤 특정 기능을 수행하기 위한 것인지 판별된다(인텐트(intent) 분류).

글 | 최동현 heuristic.085@kakaocorp.com

어느새 30대 중반으로 접어든, 그러나 마음만은 젊게 지내는 인공지능 개발자입니다. 작년에 카카오로 이직하며 카카오미니 개발에 참여하게 되었습니다. 카카오미니에 기여를 할 수 있어서 매우 기쁘고 미니를 좀 더 똑똑하게 만들기 위하여 열심히 공부 중에 있습니다.

문장 분류와 카카오미니의 관계

카카오미니의 음성 신호의 변환 단계(ex. "SBS 라디오 틀어 줘")

(1) 음성인식
- 텍스트 형태인 "SBS 라디오 틀어 줘"로 변환

(2) 봇 분류
- 라디오와 관련된 기능 요청임을 인지

(3) 인텐트 분류
- 라디오를 틀어 달라는 요청임을 파악(라디오를 멈추거나, 가능한 라디오 채널 목록을 알려 주거나, 지금 듣고 있는 라디오 채널의 이름을 들려 주는 게 아니다)

(4) 슬롯 추출
- 정확히 어떤 라디오 채널을 요청하는 것인지 파악(이 경우에는 'SBS')

(5) 요청 동작 수행
- 실제로 해당 라디오 채널의 주파수 및 재생과 관련된 정보를 찾아와 카카오미니 클라이언트에 전달

이 글에서는 (2)~(3) 단계에 대해 중점적으로 다룰 예정이다.

(2)와 (3) 단계는 주어진 문장을 여러 개의 분류 가능한 '클래스(class)' 중 하나로 분류하는 것을 공통적인 문제로 정의할 수 있다. 예를 들어, (2) 봇 분류의 경우 어떤 특정 문장이 주어졌을 때 여러 경우의 봇 중에서 하나의 봇을 선택하는 문제로 볼 수 있다. 카카오미니의 경우 음악/라디오/뉴스/주식/인물/실시간 검색어 등의 봇이 있다. (3) 인텐트 분류의 경우 어떤 특정 문장이 주어졌을 때, 라디오 재생/라디오 정지/라디오 채널 목록/현재 듣는 라디오 채널명 반환 등 라디오 봇과 관련된 여러 기능 중 하나를 선택하는 문제로 볼 수 있다.

두 단계가 동일한 문제로 정의될 수 있다면, 왜 두 단계를 하나로 합치지 않는지 궁금해 할 수 있다. 그 이유는 크게 두 가지다. 첫째, 서로 다른 봇에서는 동일한 명령어도 서로 다른 동작이 이루어지도록 해야 하기 때문이다. '재시작'이라는 명령어는 음악 봇에서는 현재 듣고 있는 음악을 반복 재생 해주어야 하지만, 알람 봇에서는 이전 타이머를 재시작 해주어야 한다. 둘째, 특정 명령은 단독으로 사용될 경우 해당 봇에서 실행되면 오류이지만, 해당 봇의 맥락(context)에서는 정상적으로 동작해야 하기 때문이다. 예를 들어, "정답은 코끼리야"라고 사용자가 단독으로 말했을 경우 카카오미니는 어떠한 기능도 수행하면 안된다. 하지만 사용자가 카카오미니와 스무고개 중이었다면 미니는 이 말을 정상적인 명령으로 받아들여 동작한다.

이러한 종류의 문장 분류 문제는 언뜻 보기에는 굉장히 쉽게 해결될 수 있는 것처럼 보인다. 대부분의 사람들은 특정 문장을 보았을 때 그 의미를 직관적으로 이해할 수 있기 때문이다. 그래서 보통 키워드나 패턴 기반의 문장 분류 문제를 풀고자 한다. 그러나 이러한 방법들은 아래와 같은 두 가지 이유로 대부분 실패한다.

첫 번째, 어떤 문장에 결정적인 키워드가 포함되어 있더라도 주변의 맥락에 따라 문장 전체의 의미는 얼마든지 달라질 수 있다. 이는 아래 발화의 예제에서 잘 드러난다.

키워드 기반의 문장 분류 예시

"뉴스 라디오 틀어줘" → 라디오에서 뉴스 틀어주기(라디오 봇)

"라디오 뉴스 틀어줘" → 라디오에 관련된 뉴스를 틀어주기(뉴스 봇)

"라디오로 뉴스 틀어줘" → 라디오에서 뉴스 틀어주기(라디오 봇)

두 번째, 패턴 기반일 경우 서로 다른 봇과 관련된 패턴간 충돌이 다수 발생한다(ex. "윤종신 나이 들려줘" → 인물 VS 음악). 또한, 패턴 기반일 경우 발화의 변동에 매우 취약하다. 본 시스템의 입력은 사람이 직접 타이핑하는 것이 아닌, 음성인식 모듈을 거쳐서 들어오는 입력 값이기에 (어느 정도의 오류를 포함할 수 있음) 발화의 변동에 취약하다. 카카오미니에서는 문장 분류를 위하여, CNN(convolutional neural network)[*1]을 이용하는 기계학습 기반 접근 방법을 취했다. CNN은 수많은 이미지 관련 문제를 해결하는 데 성공적으로 적용된 인공신경망 구조로, 네트워크에 들어온 입력 벡터를 미리 선언된 수용 영역(receptive field) 범위만큼의 단위로 묶어서 처리한다. 이를 통해 입력에서 너무 세부적이어서 분류에 있어 중요하지 않은 부분은 제거하고 실제로 분류에 중요한 부분들만 도출할 수 있도록 설계되어 있다.

문장 분류를 CNN에 적용하기 위해서는, 먼저 사용자의 발화를 벡터 형태로 변환시켜 CNN 입력으로 사용 가능한 형태로 만들어야 한다.

발화를 벡터로 표현하기

발화를 벡터로 표현하기 위해서 본 작업에서는 먼저 발화를 이루는 각 단어들을 해당 단어를 나타내는 벡터와 맵핑(mapping)한다. 그 후, 주어진 발화를 구성 단어들을 나타내는 벡터로 변환한다.

단어를 벡터로 바꾸는 방법 중 잘 알려진 것들로는 원-핫 인코딩(one-hot encoding), Word2Vec, GloVe, FastText 등이 있다. 원-핫 인코딩 및 Word2Vec에 대해서는 카카오 AI리포트의 이전 호[*2]에 간략히 설명이 되어 있으므로 GloVe와 FastText에 대해서만 간략히 설명한다.

GloVe[*3]는 워드 임베딩 벡터(word embedding vector)로 단어 i와 j의 차이를 표현하기 위하여, '프로브 워드(probe word)' k를 사용한다. [표 1]은 i = 'ice', j = 'steam' 일 때, 프로브 워드 k를 사용하여 두 단어의 의미를 구분하는 예시다.

[표 1] 대상 단어 'ice' 와 'steam'에 대한 프로브 워드 k의 동시 발생 확률 및 그 비율[*3]

Probability and Ratio	k=solid	k=gas	k=water	k=fashion
P(k | ice)	1.9×10^{-4}	6.6×10^{-5}	3.0×10^{-3}	1.7×10^{-5}
P(k | steam)	2.2×10^{-5}	7.8×10^{-4}	2.2×10^{-3}	1.8×10^{-5}
P(k | ice) / P(k | steam)	8.9	8.5×10^{-2}	1.36	0.96

즉, 단어 'ice'와 단어 'steam'의 차이를 표현하기 위해서 다양한 프로브 워드 k를 이용하여 동시 발생 확률(co-occurrence probability)의 비율(ratio)을 구한 후, 이를 이용해서 임베딩 벡터를 업데이트하는 과정을 거친다.

FastText[*4]는 Word2Vec의 변형으로, 각 단어에 대한 임베딩 벡터를 도출하는 Word2Vec과 달리, 하나의 단어를 여러 n-gram의 그룹으로 보고, 각 n-gram[*5]의 임베딩을 계산한다. 단어의 임베딩은 해당 단어가 포함하는 n-gram들의 임베딩의 조합으로 계산된다.

단어 '안녕하세요'를 예로 들면, n=3일 경우 해당 단어는 그룹 {'<안녕', '안녕하', '녕하세', '하세요', '세요>'}으로 정의되고, Word2Vec의 방식을 활용하여 각 n-gram에 대한 임베딩 벡터를 계산하게 된다. 각 단어의 임베딩 벡터는 해당 단어를 이루는 n-gram들의 임베딩 벡터의 조합으로 결정된다. 따라서 FastText 방식은 훈련 시 말뭉치에 존재하지 않았던 단어에 대해서도 알려진 n-gram으로 나누어 분석함으로써 임베딩 벡터를 도출할 수 있다. 이는 문법적 요소가 n-gram을 통해 분석되므로 한국어의 존댓말 표현과 같은 문법적 유사도를 파악하는 데 큰 강점이 있으나, 의미적 유사도를 파악하는 데에는 Word2Vec와 비교하여 성능이 약간 떨어지게 된다.

카카오미니에서의 발화 처리는 문법적인 요소보다 의미적인 요소가 훨씬 중요하게 작용하므로 미니의 문장 분류기에 사용된 단어 임베딩 벡터의 훈련에는 GloVe 알고리즘이 사용되었으며, 임베딩 벡터는 형태소 단위로 훈련되었다. 평가를 위해 9,736 쌍의 단어 유추(word analogy) 평가 데이터 셋이 자체 구축되었는데, 이러한 평가 셋은 {단어 A}:{단어 B} = {단어 C}:{단어 D}의 형태를 띄며, w(B) - w(A) + w(C) 가 w(D)와 가장 가까운 단어 벡터인지를 판별한다. 평가 데이터 셋의 예로는, 가평:경기 = 군산:전북(도시와, 도시가 속한 도의 관계), 밥:진지 = 딸:따님(높임말 관계) 등이 있다.

또한, 영어와는 다른 한국어 어순의 특징 때문에(영어: 주어-동사-목적어, 한국어: 주어-목적어-동사), 영어에서는 쉽게 판별되는 주어-동사간의 동시 발생 정보가 한국어에서는 주어-동사간의 거리가 인접하지 않으므로 판별력이 떨어진다. 이 문제를 해결하기 위해 카카오에서 자체 보유한 한국어 의존 문법 기반 구문 분석기(dependency parser)를 이용하였다. 기존 GloVe 알고리즘에서는 문장 내에서 단어와 단어 간의 거리를 측정하였다면, 의존 구문 문법 파서를 이용한 방법은 의존 구문 트리 내에서의 거리를 측정한다.

[표 2]는 다음 뉴스 2017년 1월부터 6월까지(총 14G)를 이용하여 훈련한 단어 벡터에 대한 단어 유추 셋의 평가 결과이다. TOP 20%의 빈도(frequency)를 가지는 단어에 대해서만 수치를 계산하였다.

[표 2] 구문 분석 트리를 이용한 GloVe 임베딩 벡터 도출 시 성능 향상

방식	# Correct	# Known	# Total	Known %	Total %
GloVe	3,763	8,162	9,736	46.10	38.65
GloVe + parsetree	3,816			46.75	39.19

위의 결과에서 보듯이, 일반적으로 낮은 구문 분석기의 분석 성능에도 불구하고 실제 단어 유추 말뭉치(word analogy corpus)에서 약간의 성능 향상이 있음을 알 수 있다. 따라서 위의 가설을 더욱 잘 반영할 수 있는 동시 발생 빈도(co-occurrence frequency)를 계산하는 것이 현재 해결해야 할 숙제다. [표 3]은 사용되는 데이터의 분량을 늘렸을 때의 실험 결과이다.

[표 3] 말뭉치 크기에 따른 GloVe 임베딩 벡터 성능 향상 추이

데이터	# Correct	# Known	# Total	Known %	Total %
14G(2017~)	3,816	8,162	9,736	46.75	39.19
41G(2016~)	4,426			54.23	45.46
69G(2015~)	4,492			55.04	46.14

문장 분류를 위한 신경망(neural network)

실제 네트워크의 입력으로는 위의 단어 벡터 외에 아래와 같은 특성들의 임베딩 벡터가 추가로 입력된다.

1) Local word vector

GloVe로 얻어진 단어의 의미벡터는 일반적인 상황에서의 단어의 의미를 나타낸다. 그러나 단어의 의미는 풀고자 하는 문제의 종류 및 범위에 따라 달라질 수 있다. 예를 들어, 단어 '카카오'는 주식 관련 기능으로 분류될 때에는 회사의 명칭으로서의 의미를 가지지만, 교통 정보 관련 기능으로 분류될 때에는 장소 명칭으로서의 의미를 가지며, 음식 관련 기능으로 분류될 때에는 식재료 명칭으로서의 의미를 가진다.

GloVe로 얻어진 단어의 의미 벡터는 여러 의미들을 모두 포괄적으로 나타내고 있으므로 특정 상황에 맞는 의미를 더욱 강화하기 위해 별도의 Local word vector를 구축한다. 해당 벡터는 네트워크 모델 훈련 시 함께 훈련된다.

2) Entity tag vector

사용자의 발화 요청이 짧아서 발화 자체로부터 의미 있는 정보를 얻을 수 없을 경우, 대신 발화에 속하는 엔티티(entity)의 태그(tag) 정보를 이용하여 문장 분류를 진행할 수밖에 없다. 예를 들어, 발화 '아이유 좋은날'의 경우, '아이유'는 인물명이고, '좋은날'이 곡명이라는 사실을 알아야 해당 발화가 음악과 관련된 요청이라는 것을 알 수 있다.

3) POS vector

각 형태소의 품사 정보 또한 사용된다.

문장 훈련을 위해 사용된 네트워크는 EMNLP(2014) 논문[*6]에서 제안된 방법으로 구성했다. [그림 1]은 해당 네트워크의 구조를 간략히 보여준다.

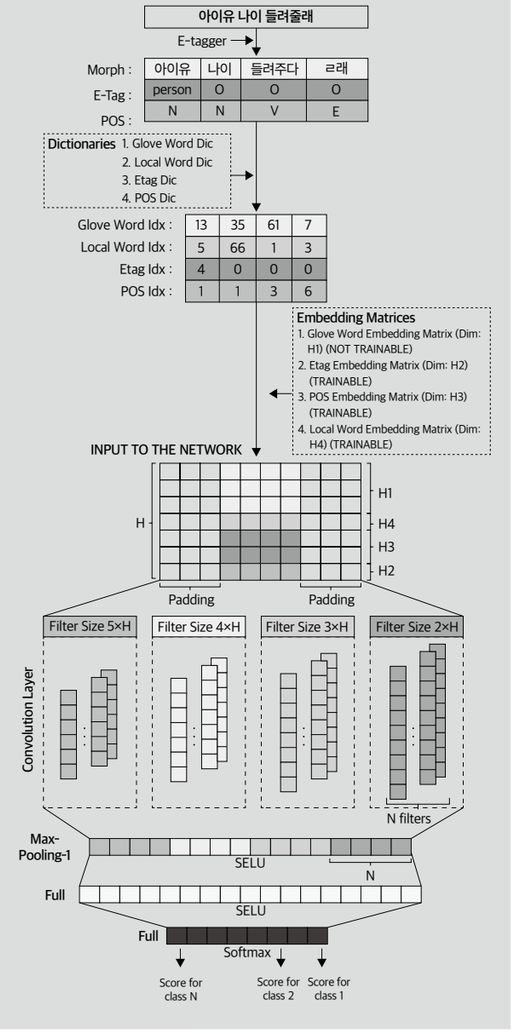
그 밖에 본문에서 자세히 설명되지 않은 내용으로 포지셔널 인코딩(positional encoding)[*7] 및 SELU(scaled exponential linear unit)[*8]이 사용되었다.

개선 방향

최근 하나의 문장에 다중의 인텐트를 포함하고 있는 발화들을 처리하는 방법이 구현되어 카카오미니에 적용되었다. 이런 다중 인텐트들의 조합을 가능하게 하고 처리 가능한 조합을 점점 늘려가면서 기존 기능에 악영향을 끼치지 않게 하는 것은 문장 분류 모듈에 있어 굉장히 중요한 도전이 될 것이다.

또한, 인공 신경망을 실제로 사용하기 위해서는 각종 신경망 패러미터(parameter) 튜닝에 많은 시간을 투자하여야 한다. 현재 자동으로 최적의 패러미터를 찾기 위한 작업[*9]이 수행되고 있다. 이 작업이 더 발전되면 자동으로 인공신경망 네트워크 구조를 구축하는 시스템[*10]으로의 발전도 기대할 수 있을 것이다.

[그림 1] 현재 카카오미니 문장 분류기 네트워크 구조



***1** 논문 | Lecun, Y., Bottou, L., Bengio, Y. & Haffner, P. (1998). Gradient-based learning applied to document recognition. Proceedings of the IEEE (Vol. 86, pp. 2278-2324). ***2** 참고 | 이수경, 이주진, 임성빈. (2018). Brain's Pick - 단어 간 유사도 파악 방법. 카카오 AI 리포트 (Vol. 10). ***3** 논문 | Pennington, J., Socher, R., Manning, C. (2014). Glove: Global Vectors for Word Representation. Proceedings of the EMNLP (pp. 1532-1543) ***4** 논문 | Bojanowski, P., Grave, E., Joulin, A., Mikolov, T. (2017). Enriching Word vectors with Subword Information. Transactions of the Association for Computational Linguistics (Vol. 5, pp. 135-146) ***5** 설명 | 일반적으로 대표적인 확률적 언어 모델의 하나로 n개 단어의 연쇄를 확률적으로 표현해 실제로 발성된 문장의 기록을 계산하는 방법(출처:TTA정보통신용어사전) 를 의미하지만, 해당 연구에서는 단어를 이루는 캐릭터들의 연쇄라는 의미로 사용됨. ***6** 논문 | Yoon, K. (2014). Convolutional Neural Networks for Sentence Classification. Proceedings of the EMNLP (pp. 1746-1751) ***7** 논문 | Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., Polosukhin, I. (2017). Attention is All you Need. Proceedings of NIPS ***8** 논문 | Klambauer, G., Unterthiner, T., Mayr, A., Hochreiter, S. (2017). Self-Normalizing Neural Network. Proceedings of NIPS ***9** 논문 | Bergstra, J. S., Bardenet, R., Bengio, Y., Kegl, B. (2011). Algorithms for Hyper-Parameter Optimization. Proceedings of NIPS ***10** 논문 | Zoph, B., Le, Q. V. (2017). Neural Architecture Search with Reinforcement Learning. arXiv preprint.