AI in Kakao

카카오 번역기가 양질의 대규모 학습 데이터를 확보하는 방법

2017년 10월, 카카오가 번역 서비스를 출시했다. 신경망 번역 기술이 적용됐고, 블라인드 테스트 결과에 근거하면, "경쟁력이 높다"는 결과가 나왔다. 이 높은 성능을 위해서는 좋은 모델이 전제되어야 하지만, 학습 데이터 역시 매우 중요하다. 이 글에서는 카카오의 번역 서비스가 양질의 대규모 학습 데이터(병렬 말뭉치)를 확보하기 위하여 사용한 기술 중 Ableualign 툴을 소개하려 한다.

글 | 오형석 hulk.oh@kakaocorp.com

새로 만드는 것은 무엇이든 흥미가 있어서 다양한 것들을 제작하다가 프로그래머가 되었습니다.
현재는 카카오의 AI부문에서 연구 및 개발을 하고 있으며, 이 분야 내에서 새로운 것을
창조해보려 노력 중입니다.

본론에 앞서 우선 BLEU가 어떤 의미인지 이해해야 한다. 이 개념을 간단히 설명하면, 원문에 대한 사람의 번역 결과와 번역기의 결과가 얼마나 유사한지를 수치화 한 것이다. 수치화 하는 방식은 문장을 구성하는 단어들을 n-gram으로 만들고 그 n-gram들이 얼마나 서로 매칭되는 지를 보는 것이다[1].

> n-gram은 문장을 어떤 단위로 쪼갰을 때 인접한 n개를 모은 것을 의미한다.
> 단위는 여러가지가 가능한데 띄어쓰기를 단위로 하여 3-gram을 구해보자.
> 예) input: "I am a boy"
> 3-gram: "I am a", "am a boy"
>
> 이런 식으로 문장을 쪼개는 이유는, 문장간의 유사도를 측정할 때, 문장의 일부로 매칭을 해보기 위함으로, 본 글에서는 유사도를 sliding window방식으로 비교하기 위한 것이라 생각해도 된다.

이 방식은 문법 구조, 유의어 등을 보지 않기 때문에 한계를 가지긴 하지만 다른 더 나은 방법론이 아직 없기 때문에 번역의 품질 평가에도 여전히 많이 사용되고 있다.

데이터 확보에 왜 BLEU를 쓰는가 라는 의문이 들 수 있을 것이다. 병렬 말뭉치(parallel corpus)의 raw data는 대부분 글(article) 단위이고 문장(sentence) 학습을 하는 신경망 기반 기계번역(neural machine translation, NMT) 모델을 위해서는 문장 단위로 정렬을 해주는 툴이 필수적이다. 카카오에서는 정렬(alignment)을 위하여 초기에 규칙 기반 툴을 만들어서 사용했는데 한계가 있음을 경험했다. 규칙을 프로그래밍 하기가 너무 힘들었고 도메인이 바뀌면 기존 규칙이 적합하지 않는 문제가 발생하였다. 이처럼 규칙은 복잡하지만 사람이 하면 쉬운 문제의 경우 해결할 수 있는 적절한 방법이 있는데, 바로 딥러닝을 적용하는 것이다. 결국 이미 딥러닝을 통해 학습된 번역기를 활용하면 쉽게 align을 할수 있을 것이란 결론에 도달하였고 Bleualign을 사용하게 된 것이다. Ablealign은 Bleualign을 개선한 버전인데 비교적 간단한 아이디어로 많은 효과를 보았다. 이제 본론으로 들어가 보자.

Bleualign

Bleualign[*2]은 BLEU score를 이용해서 두 말뭉치들을 정렬하는 툴이다. 번역기의 학습을 위해서는 병렬 말뭉치가 필요한데, Bleualign에 두 가지 언어로 된 글을 던지면 그 안의 문장 단위 번역 쌍을 뽑아준다. 카카오 번역의 모델을 학습시킬 때 쓸 데이터를 만들기 위해 Bleualign을 사용하였다. 이 툴의 동작 방법을 간단히 살펴보자.

(1) 다음과 같은 한국어 영어로 된 글이 있다고 하자. (문장 단위로 분리가 되어 있다고 가정한다)

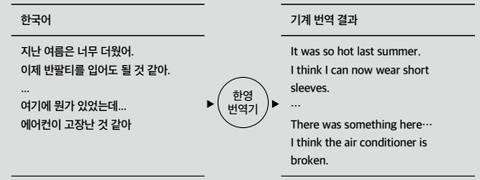
[그림 1] 한국어, 영어 글

한국어	영어
	I'm Hulk!
지난 여름은 너무 더웠어.	I will destroy everything!!!
이제 반팔티를 입어도 될 것 같아.	It was so hot last summer.
…	I think I can wear short sleeves now.
여기에 뭔가 있었는데…	…
에어컨이 고장난 것 같아	Shall we turn on the air-conditioner?
	I think the air conditioner is out of order.

같은 글에 대한 문장들이므로, 비슷한 문장들이 많이 존재하나 무조건 1:1 매칭이 되지는 않는다.

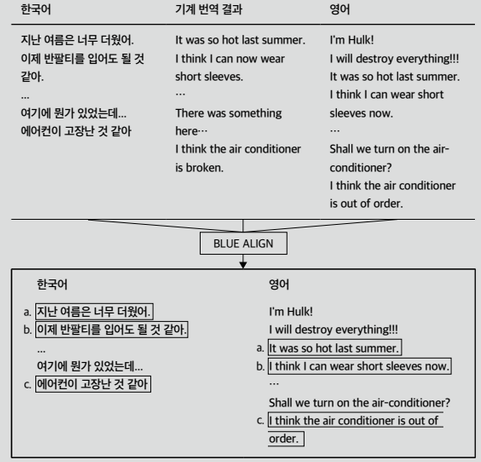
(2) 이 중 하나를 한영번역기(또는 영한번역기)를 통해 번역한다.

[그림 2] 한영번역기를 이용한 번역



(3) 이제 한국어, 영어, 기계 번역 결과를 동시에 입력값으로 Bluealign에 넣어주면 문장 단위의 병렬 말뭉치가 출력된다. a,b,c 박스는 각각 같은 문장의 다른 언어로 된 표현(representation)들이다.

[그림 3] Bluealigen 출력 결과



3번 단계에서 기계번역 결과와 영어 말뭉치를 통해서 같은 문장을 찾게되는데, 이 때, 비슷한 문장을 찾기위해 BLEU score를 사용한다. 학습 데이터를 얻기위해 Bleualign tool을 잘 사용하였는데, BLEU를 개선시키면 더 많은 병렬 말뭉치를 얻을 수 있지 않을까 하는 생각이 들었고, 그 결과 나온 것이 점수 함수(score function)를 ABLEU로 대체한 Ablealign이다. 이제 BLEU에 대해서 알아보고, 개선을 시켜보도록 한다.

BLEU

기계 번역의 품질을 평가하는 측정 기준(evaluation metric) 중의 하나로서, 간단하고 빠른 계산으로 인간이 생각하는 품질과 높은 상관관계(correlation)를 가지고 있어, 현재까지도 가장 널리 쓰이는 측정 기준이다.

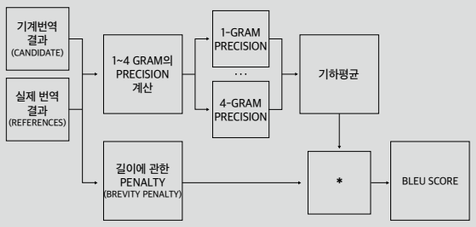
BLEU의 입력값(input)은 '기계 번역 결과'와 '여러 개의 실제 번역 결과'이며, 출력값(output)은 '0~1 사이의 값'이다. 출력값이 1에 가까울 수록 번역 결과가 좋은 것을 의미한다. 이 문서에서는 조금 더 쉽게 설명하기 위해 실제 번역 결과가 하나만 존재한다고 가정할 것이다.

1) BLEU Overview

BLEU를 간략하게 나타내보면 다음 그림과 같다.

(1) 기계번역 결과(이하 candidate)와 실제 번역 결과(이하 reference)에서 1-gram ~ 4-gram에 대한 정확도를 계산한다.
(2) 4 개의 결과를 기하평균을 내서 precision을 구한다.
(3) 길이(length)에 대한 패널티(brevity penalty)를 곱해서 BLEU값을 계산한다.

[그림 4] BLEU score 계산에 대한 Overview



각 방법에 대해서 자세히 알아보자.

2) n-gram의 정확도 구하기

쉽게 설명하기 위해 예시를 들어서 띄어쓰기를 단위로 한 1-gram의 정확도를 계산하겠다. Candidate로 '나는 밥을 먹었다', reference로 '나는 밥을 버렸다'가 들어왔다고 가정하자. 여기서 candidate의 1-gram은 '나는', '밥을', '먹었다'이며, 2-gram은 '나는 밥을'과 '밥을 먹었다' 이다. 1-gram의 정확도(precision)는 [수식 1]로 구해진다.

[수식 1]

$$P = \frac{m_{max}}{w_t}$$

· m_{max}: candidate의 단어가 한 reference에서 나온 최대 갯수
· w_t: candidate에 있는 총 단어 갯수

즉 위의 예제에서 각각은 다음과 같은 값을 갖게되어 1-gram precision은 2/3가 된다.
'나는'의 m_{max}=1/ '밥을'의 m_{max}=1/ '먹었다'의 m_{max}=0

왜 굳이 reference에서 나온 최대 갯수로 clipping을 하는지에 대해서 궁금할 것이다. 이는 번역기 모델이 반복적으로 단어를 뱉어내는 경향이 있는데, 정확도를 계산할 때 반복된 단어들을 전부 넣어버리면 점수가 올라가므로 이를 방지하기 위해서다.
[표 1] 은 BLEU: a method for automatic evaluation of machine translation[3]에서 발췌한 예제이다. [표 1]의 경우 max 값으로 clipping 하지 않으면 7/7로 정확도가 1이 돼버린다.

[표 1] 번역은 안좋은데 unigram precision이 잘 나오는 경우

Candidate	the	the	the	the	the	the	the
Reference1	the	cat	is	on	the	mat	
Reference2	there	is	a	cat	on	the	mat

또한 위처럼 정확도를 계산하면 짧은 번역문을 선호하는 문제가 또 있다. 예를 들어, candidate로 '나는'이 나왔다고 하면,

[수식 2]

$$P = \frac{1}{1} = 1$$

[수식2] 와 같이 된다. 따라서 candidate와 reference의 길이를 비교하여 패널티를 주어 보정한다.

3) Brevity penalty

문장의 길이를 이용해 penalty를 준다. 다음 [수식 3]을 precision에 곱해줘서 페널티를 줄 수 있다.

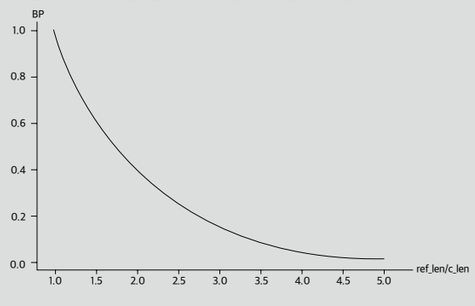
[수식 3]

$$e^{(1-\frac{r}{c})}$$

· r: reference의 길이
· c: candidate의 길이

[그림 5] 는 reference 길이를 candidate 길이로 나눈 값에 따른 brevity penalty이다. Candidate가 reference 길이의 절반만 되더라도 정확도가 반 이상으로 떨어지게 됨을 알 수 있다.

[그림 5] reference 길이를 candidate 길이로 나눈 값에 따른 brevity penalty



ABLEU

1) Idea

우리는 BLEU에서 n-gram precision을 계산하는데 정확히 매칭되는 것만 카운팅을 하고있는 것을 주목했다. '이쁘다' 와 '예쁘다' 는 분명히 비슷한 뜻인데 0,1로만 구분하여 집계(discrete counting)해서 매칭이 되지 않는 결과가 발생했다. 만약 비슷한 단어에 대해서 0과 1 사이의 어떤 유사도(similarity)를 연속적(continuous)으로 부여할 수 있다면, 이것이 훨씬 좋은 결과를 낼 수 있을 것이라 생각했다. 유사도를 구할 수 있다고 가정하고 ABLEU는 BLEU수식에서 단 한개만 바꾸면 된다.

[수식 4]

$$P = \frac{S_{max,m}}{w_t}$$

· $s_{max,m}$: candidate의 word m과 가장 높은 유사도

이제 우리가 할 일은 유사도를 계산하는 방법을 만드는 것이다.

2) Similarity

(1)Word2vec

Word2vec은 단어를 고정된 길이의 벡터로 embedding하는데 쓰인다. 따라서 word2vec으로 단어들을 embedding하고 그것의 유사도를 쓰면 될 것이라 생각했다. 단 한가지 문제점은 word2vec은 단어의 위치에 따른 유사도가 크기 때문에 정반대의 단어도 유사도가 높게 나온다는 단점이 존재했다. 예를 들어 word2vec의 경우 '잘생겼'과 가장 비슷한 단어들 10개를 뽑으면 [그림 6] 과 같은 결과가 나온다. '역겹'과 '잘생겼'이 비슷하게 나온 것을 보면 문장 내의 위치(position)가 유사도에 미치는 영향이 큰 것을 알 수 있다.

[그림 6] Word2vec으로 계산한 similarity

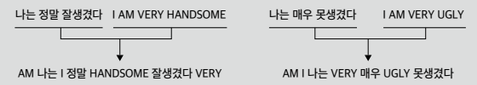
```
candidates = model.wv.most_similar([encode('잘생겼ㅓ')], topn=100)
for cand in candidates[0:10]:
    print(cand)
```

```
('사윙스럽ㅓ', 0.8485762476921082)
('우서웠ㅓ', 0.8454145193099976)
('귀엽ㅓ', 0.8109017610549927)
('시끼럽ㅓ', 0.8060537576675415)
('괜찮았ㅓ', 0.8029224872589111)
('역겹ㅓ', 0.8018471598625183)
('화났ㅓ', 0.7896434068679810)
('고통스럽ㅓ', 0.7856169939041138)
('무겁ㅓ', 0.7854277491569519)
('똑똑하ㅓ', 0.7840464711189270)
```

(2)Word2vec-Pair

이를 방지하기 위해 만든 방법은 굉장히 단순하며 비슷한 단어의 유사도를 잘 뽑아냈다. 그 방법은 우리가 가지고있는 병렬 말뭉치를 섞어서 word2vec을 학습(training)시키는 것이다. 두 문장을 섞어버리면 영어 단어와 한글 단어가 word2vec의 입력값으로 같이 들어가며, 비슷한 단어의 유사도를 높이는데 서로를 이용할 것이라는 생각이었다. [그림 7]을 보면, word2vec-pair의 경우 뜻이 비슷한 단어가 더 잘 뽑히는 것을 알 수 있다. 두 언어에 대해서 model을 학습시키기 때문에, 영어 단어도 뽑히게 된다.

두 언어쌍을 shuffle하여, 상대 언어의 단어를 이용하도록 하였다.



[그림 7] Word2vec-Pair로 계산한 similarity

```
candidates = model.wv.most_similar([encode('잘생겼ㅓ')], topn=100)
for cand in candidates[0:10]:
    print(cand)
```

```
2017-09-26 11:43:28.552 : INFO : precomputing L2-norms of word weight vectors
('handsome.', 0.6068255305290222)
('똑똑하ㅓ', 0.5543735623359680)
('귀엽ㅓ', 0.5263066887855530)
('handsome', 0.5228010416030884)
('잘생긴', 0.5102168321609497)
('멋지ㅓ', 0.4998891022109985)
('예쁘ㅓ', 0.4981892108917236)
('good-looking', 0.4871434569358825)
('이쁘ㅓ', 0.4861024275594421)
('smart,', 0.4743206799030304)
```

위의 word2vec-pair를 통해 나온 embedding으로 서로의 유사도를 구했고, 이를 사용하여 ABLEU를 만들 수 있었다.

결과

[표 2] Bleualign과 Ableualign 의 수율 및 정확도

	Bleualign	Ableualign
수율(%)	35.64	47.1
정확도(%)	86	93.73

실험 결과 동일한 번역기를 사용할 때 Bleualign 대비 Ableuaign은 수율이 35.64%에서 47.10%로 약 11% 증가하였고 정확도도 86%에서 93.73%로 증가하였다(정확도는 sampling 하여 수동 평가로 측정하였다). 이 말은 ABLEU 적용으로 더 나은 품질의 데이터를 기존보다 30% 정도 더 추출했다는 말이다. Bluealign의 성능은 번역기의 성능에 의존적이기 때문에 더 나은 번역기를 사용할 경우 수율이 더 높아질 수 있다.

따라서 Ableualign을 사용해 번역기의 품질을 올리고, 이를 다시 Ableualign 번역 스텝에 사용하여 수율을 올리는 반복 작업이 가능했다. 실험이 진행되는 동안 카카오에서는 한번의 번역기 모델 개선이 더 이루어졌는데, 이 모델을 적용하여 최종적으로는 기존보다 58% 더 많은 병렬 말뭉치를 추출할 수 있었다.

결론

카카오는 번역기의 학습에 이용할 양질의 병렬 말뭉치를 얻기 위해 Bleualign을 사용하였다. Bleualign의 점수 측정 기준(score metric)을 더 좋게 바꾸면 더 질 좋은 데이터를 더 많이 얻을 수 있으리라 기대했으며 0, 1 로만 구분하여 집계하는 BLEU score의 discrete counting에 주목했다. 이를 연속적인(continuous) 유사도로 바꾸기 위해 word2vec을 언어쌍으로 학습하는 방법을 고안하였으며, 그 결과 Ableualign을 만들 수 있었다. 또한 Ableualign으로 더 많이 확보된 데이터를 통해 번역기의 품질을 높이고, 이 번역기를 다시 Ableualign의 번역 step에 넣어서 반복(iteration)을 통해 선순환 구조를 만드는 것이 가능했다.

[그림 8] Word embedding의 예시 - bridge와 가까운 단어들*4

[표 3] 단어 'bridge'와 다른 단어 간의 거리

단어	거리
bridges	0.408
road	0.554
tunnel	0.571
gate	0.590
canal	0.594
ferry	0.596
dam	0.613
tower	0.616
thames	0.633
railroad	0.634
harbour	0.634
link	0.648
arch	0.650
suspension	0.652
railway	0.656
roads	0.657
crossing	0.659
straits	0.660
bay	0.660
connecting	0.660
highway	0.662
river	0.663
gap	0.675
strait	0.676
tunnels	0.682
construction	0.686
avenue	0.687
boundary	0.688
erected	0.690
monument	0.693
peninsula	0.693
street	0.696
opened	0.699
connect	0.699
building	0.700
route	0.704
landmark	0.707
rail	0.709
connects	0.710
traffic	0.710
port	0.711
interstate	0.712
deck	0.714
span	0.715
routes	0.718

***1** 참고 | 보통 3~4-gram 까지 진행된다. ***2** 참고 | https://github.com/rsennrich/Bleualign ***3** 논문 | Papineni, K., Roukos, S., Ward, T., Zhu, W.J., (2002). BLEU : a method for automatic evaluation of machine translation, 02 Proceedings of the 40th Annual Meeting on Association for Computational Linguistics,(pp. 311-318). ***4** 참고 | http://projector.tensorflow.org