#1

시작페이지

#2

안녕하세요 발표를 맡게 된 김호중, 송이령입니다. 저희는 2주차BERT를 맡게 되었습니다. ALBERT, RoBERTa, ELECTRA, SpanBERT Knowledge Distillation BERTSUM 에대해 발표할예정이며 목차에 대해 살펴보겠습니다.

#3

목차는 다음과 같이 구성되어 있습니다. Derivation 1 derivation 2 Burtsum hidden 그리고 Q&A 입니다. 그러면 이제 발표를 시작하겠습니다.

#4

BERT의 주요 문제점은 수백만개의 변수로 구성되어 있어 추론시에 시간이 많이 걸리며 리소스 제한이 발생하는 것 입니다. 이문제를 해결하기위해 탄생한 것이 ALBERT입니다. LABERT는 기존 BERT와는 다르게 Cross layer parameter sharing과 factorized embedding layer parameterization 을 활용합니다.

#5

먼저 크로스 레이어 변수공유에 대해서 살펴보면 BERT모델의 변수개수를 줄이는 방법으로 기존의 방법과는 다르게 첫 번째 레이어의 변수만 학습한 다음 첫 번째 인코더 레이어의 변수를 다른 모든 인코더 레이어와 공유합니다. 변수를 공유하는 방법에 따라 ALL shared, Shared feedforward network, Shared attention 으로 불리며 각각 살펴보면

ALL shared: 첫번째 인코더의 하위 레이어에 있는 모든 변수를 나머지 인코더와 공유

Shared feedforward network: 첫번째 인코더 fpdldjm이 치드포워드 네트워크의 변수만 다른 인코더 레이어의 피드포워드 네트워크와 공유

Shared attention: 첫번째 인코더 레이어의 멀티 헤드 어텐션의 변수만 다른 인코더 레이어와 공유

#6

다음은 factorized embedding layer parameterization(팩토라이즈 임베딩 변수화) 입니다. 워드피스 임베딩과 은닉 레이어 임베딩은 모두 학습이 진행될 때 학습이 이루어 집니다. 이럴경우 은닉레이어 임베딩 크기와 동일하게 설정하면 학습해야 할 변수 역시 늘어나는 데요. 이런 형상을 방지하기위해 행렬분해를 통해 어휘벡터를 은닉공간V\*H에 직접 투영하는 대신 낮은 차원의 임베딩 공간V\*E으로 투영하고 그다음에 낮은 차원의 임베딩을 은닉공간 E\*H로 투영하는 방법을 사용합니다. 정리하면 사전크기 V를 은닉레이어 H에 직접 투영하는 것이 아닌 V를 E에 투영하고 다시 E를 H에 투영합니다. 결론적으로 V\*H는 v\*E와 E\*H로 분해 됩니다.

#7

NSP를 사용하는 것이 실제로는 유용하지 않고 MLM대비 난도가 높지 않다는 점을 들어 ALBERT에서는 BERT와는 다르게 NSP 다음문장 예측대신 SOP 문장 순서 예측을 사용한다. SOP는 NSP와는 다르게 문장순서가 바뀌었는지 여부를 예측하며 모델 학습이 이루어집니다.

#8

간단한 예시를 살펴보면 두문장 1 she cooked pasta 2 it was delicious를 배열을 교차해 보았을 때 she cooked pasta it was delicious에서는 positive를 it was delicious she cooked pasta에서는 false를 결과값으로 내보내는 것을 알 수 있다.

#9

RoBERTa는 BERT의 파생모델중 하나로 BERT의 불완전한 부분을 해소하고자 만들어 졌다. BERT모델을 사전 학습할때에 다음 4가지 항목에 변화를 주었다.

MLM 태스크에서 정적 마스킹이 아닌 동적 마스킹 방법을 적용했다.

NSP 태스크를 제거하고 MLM 태스크만 학습에 사용했다.

배치크기를 증가해 학습시켰다.

토크나이저로 BBPE(byte level BPE)를 사용했다.

#10

먼저 정적 마스크에 대해서 알아보면 기존의 MLM은 입력토큰에 무작위로 마스크를 씌우고 Epoch별로 동일한 마스킹을 예측하도록 모델학습이 이루어진다. 이와는 다르게 동적 마스킹은 히나의 문장을 복사하여 각각 확률적으로 마스킹을하여 각각의 answkqddp 대하여 학습을 진행합니다.

#11

다음은 NSP 태스크 제거에 대하여 알아보겠습니다. 연구진들은 다음실험을 통해 NSP 태스크가 BERT 모델 사전 학습에 유용하지 않다는 사실을 발견했습니다.

Segment -pair +NSP NSP를 사용해 BERT를 학습 입력은 512개 이하의 토큰 쌍

Segment -pair +NSP NSP를 사용해 BERT를 학습 입력은 512개 이하의 토큰 쌍이며 한문서의 연속된 부분 또는 다른 문서에서 추출한 문장을 쌍으로 구성한다.

Full sentences NSP를 사용하지 않고 BERT를 학습시키며 입력값은 하나 이상의 문서에서 지속적으로 샘플링한 결과를 사용한다. 하나의 문서 마지막까지 샘플링을 한 이후에는 다음 문서에서 샘플링 작업을 이어 나간다.

DOC Sentences NSP를 사용하지 않고 BERT를 학습시킨다. FULL Sentence와 전체적으로 유사하나 입력값은 하나의 문서에서만 샘플링한 결과만 입력한다. 하나의 문서 마지막까지 사용한 후에 다음 문서 내용을 사용하지 않는다.

실험 결과 4가지 설정중 NSP를 수행하지않은 DOC Sentences Full sentences 더 높은 성능을 보여줍니다.

#11

ELECTRA 는 BERT의 파생모델중 하나로 replaced token detection이라는 태스크를 사용합니다. MLM과 굉장히 유사한 방법으로 마스킹 대상인 토큰을 다른 토큰으로 변경한 후 이토큰이 실제 토큰인지 아니면 교체한 토큰인지를 판별하는 형태로 학습을 진행합니다.

#13

The chef cooked the meal 이라는 문장에서 the-.a로 cooked-ate로 대체 하였고 Bert에 입력하였을때 토큰이 원본인지 아니면 교체되었는지 여부를 출력한다.

#14

조금더 자세히 알아보면 the chef cooked the meal이라느 문장에서 확률기반으로 마스킹을 진행하고

#15

생성자가 마스킹된 토큰에 대해 워드를 확률화하여 예측된 단어을 판별자에게 전달하며 판별자는 이를 대체된 단어 또는 그렇지 않은 단어 2가지로 판단합니다.

#16

생성자에 대해서 조금 더 알아보면 생성자는 MLM태스크를 수행하며 생성기에서 마스크된 토큰을 통해 학습을 진행합니다. 무작위로 일부 토큰을 마스킹하고 생성기에 입력해 각 토큰의 결과값을 얻는다. H1,h2,h3…을 생성기를 통해 얻은 각 토큰의 표현이라하며 이 값을 통해 마스크된 단어를 확률적으로 추론합니다. 위 그림에서는 x1은 a로 x3는 ate로 예측됩니다.

#17 다듬자

이제 판별자에 대하여 알아보면 입력토큰을 생성자에 의해 예측된 토큰으로 변경하고 이를 판별자가 입력값으로 받게 되는데 판별자는 주어진 토큰이 생성자에 의해 만들어진 토큰인지 아니면 원래 토큰인지를 판별합니다. 판별자는 생성자를 거친 토큰들에서 토큰에 대한 값을 얻고 시그모이드 함수를 가지고 있는 피드포워드 네트워크 형태의 분류기에 넣음으로서 연산을 진행합니다.

#18

이런 연산과정속에서 BERT 형태의 판별자를 바로 ELECTRA 모델이라고 하며 손실함수를 살펴보면 다음과 같다 X\_corrupt는 마스크된 토큰이 생성자에 의하여 손상이 된 것을 표현하며 ELECTRA 모델의 최종 손실함수는 아래와 같습니다.

#19

SPANBERT는 텍스트 범위를 예측하는 질문 응답과 같은 태스크에 주로 사용됩니다. 아키텍처를 살펴보면 your are expected to know the laws of your country 위 문장을 토큰화 시키면 다음과 같습니다. 여기에 기존의 BERT와는 다르게 마스킹을 범위로 하게 되는데요 다음 예제에서는 the laws of your이 단체로 마스킹되었습니다. 이처럼 토큰의 위치가 아닌 토큰의 범위로 마스킹하는 것이 SPANBERT의 특징입니다.

#20

마스크된 토큰을 예측하기 위해 MLM과 새로운 목적 함수인 SMO span boudry object를 사용해 SpanBERT를 학습 시킵니다. 기존MLM에서 마스크된 토큰을 예측하기 위해 마스크된 토큰의 표현을 사용하는 것을 떠올려 보면 마스크된 x7을 예측하기 위해 R7을 사용해서 마스크된 토큰을 예측하며 이때 R7을 분류기에 입력하면 사전에 있는 모든단어에 대해 마스크된 토큰이 될 확률을 얻을 수 있습니다.

#21

이와는 다르게 SBO는 마스크된 토큰을 예측하기 위해서 해당하는 마스크된 토큰의 표현을 사용하는 대신에 스팬경계에 있는 토큰의 표현만 사용합니다. 조금 더 자세히 살펴보면x5 x10이 스팬경계의 토큰을 나타내고 R5 R10이 스팬 경계에 대한 표현이 됩니다. 이제이를 이용하여 마스크된 토큰 x7을 예측하게 되죠

#22

하지만 위 예시를 조금만더 살펴보면 이상한점이 하나 있습니다. 모델의 마스크된 토큰을 예측하기 위해 범위의 경계에 있는 토큰의 표현만 사용하는데, 이때 경계안에 있는 마스크된 토큰은 어떻게 구분하는 걸까요? 예를들어 X6와 x7을 예측한다고 했을때 모두 R5와 R10만 사용하는데 모델이 올바르게 예측할 수 있을까요? 이런 이유로 하여금 스팬경계 토큰과는 별도로 모델은 위치임베딩 값을 함계 사용합니다. 여기에서 위치 임베딩은 마스크된 토큰의 상대적인 위치를 의미합니다. 즉 x7을 예측한다고 했을 때 위치토큰인 P2또한 함께 사용됨으로서 위와 같은 문제를 해결합니다.

#23

다음은 지식 증류를 활용한 BERT 파생모델에 대해서 살펴보겠습니다.

#24

지식 증류는 사전 학습된 대형 모델의 동작을 재현하기 위해 소형 모델을 학습시키는 모델 압축기술이며. 교사 학생 학습이라고도 합니다. 예시를 하나 살펴보면 I completed writing my \_\_에서 빈칸에 들어갈 확률에 대해 나타내 보면 다음과 같습니다. Homework가 가장 높기 때문에 homework가 선택되며. Book과assignment는 다른 cake, car와 같은 단어에 비해 상대적으로 확률이 높죠 위와 같이 최종적으로 내보내는 결과물 homework만이 아니라 증류하는 과정에 있어 암흑지식이라 부르는 이런 확률관계또한 학생 네트워크가 학습하기를 바라죠. 일반적으로 가장 아래그림과 같이 모델은 정답을 1에 가까운 확률로 나타내어 줍니다. 하지만 이럴 경우 실측 단어를 제외하고 분포에 많은 정보가 없다는 것을 알 수 있습니다.

#25

이럴 때 사용하는 것이 바로 템퍼러쳐입니다. 템퍼러쳐를 높게 설정할수록 확률분포가 평활화되며 우리에게 필요한 암흑지식에 대해 더 많은 정보를 줍니다. 그림을 보면 알 수 있듯이 t=1일 때 t=2.t=3일 때를 각각 비교해보면 점점 확률값이 균등해져가는 것을 볼 수 있습니다.

#26

교사 네트워크가 소프트맥스 템퍼러처로 학습이 된 후 입력 값에 대한 교사와 학생이 출력한 확률 분포 값입니다. 이때 교사 버트의 출력을 소프트 타깃이라고 하며 학생 버트에서 소프트 타깃을 소포트 예측이라고 합니다.

#27

소프트 예측과 스프트 타깃간의 cross entropy loss를 증류 손실이라고 하며 이를 최소화 하기 위해서 학습을 진행합니다.

#28

학생 네트워크는 증류손실과 별도로 학생손실이라는 하나의 loss를 더 사용합니다. 전 슬라이드에서는 학생과 교사 네트워크 모두 t=5값을 사용하여 확률분포를 얻어 냈는데요. 학생 손실은 이와는 다르게 t=1값으로 고정하고 다음 그림과 같이 가장 확률이 높은 곳에 1을 설정하고 나머지 값에 대해서는 0을 설정하여 타깃을 만들어 냅니다. 학생 손실은 위와 같은 방법으로 만들어진 하드타깃과 하드 예측간의 cross Entropy loss입니다.

#29

그림으로 정리해보면 다음과 같죠 교사 네트워크와 학생 네트워크의 t=5갑을 통하여 증류 손실을 얻어 내고 g다 타깃과 학생네트워크에 t=1을 해준 값을 비교하여 학생 loss를 만들어 냅니다.

#30

이제 본격적으로 교사 학생 아키텍쳐를 적용한 BERT 파생 모델인 Tiny BERT에 대해 알아보겠습니다. Tiny Bert는 출력 레이어에서 학생에게 지식을 전달하는 것 외에 임베딩 및 여러 인코더 레이어에서 지식을 전달합니다. 그림과 함께 살펴보면 출력 레이어의 로짓 인코더 은닉 상태 인코더의 어텐션 행렬 임베딩 레이어의 출력 등을 학생 BERT에게 학습 시킵니다. 정리하면 Tiny Bert는 출력 레이어에서 학생 BERT로 지식을 전달하는 것 외에도 중간 레이어의 지식을 전달합니다.

#31

버트의 지식증류과정을 그림과 함께 살펴보면 다음과 같습니다. 교사 BERT의 N번 index를 학생 BERT의 m번 index로 매핑하며 이 과정은 매핑함수 g로 표현가능합니다. 예를 들어 g(0)=0 이식은 교사의 0번째레이어에서 학생의 0번째레이어로 지식을 이전하는 것을 의미합니다.

#32

다음 식은 어텐션 기반 증류 손실함수입니다. 식에서 h는 어텐션 헤드의 수 A si는 학생 BERT의 i번째 헤드의 어텐션 행렬을 A Ti는 교사 BERT의 i번째 헤드의 어텐션 행렬을 나타냅니다. 결국 각각의 어텐션 행렬을 MSE하여 나타낸 것이 어텐션 루스입니다.

#33

그림으로 표현하면 다음과 같죠

#34

다음은 은닉상태 기반 증류입니다. 은닉 상태는 인코더의 출력 표현 벡터로 loss 식을 살펴보면 학생의 은닉 상태에 교사와의 차원을 맞추기 위한w값을 연산하여 나온 H s Wh값과 교사의 은닉상태 H t사이의 편균제곱오차를 취한 값입니다.

#35

마찬가지로 그림으로 보면 다음과 같은 형태입니다.

#36

Hidden state based Distillation과 마찬가지로 교사와 학생의 연산에 차원을 맞춰주기위해 We를 학생에게 곱한값을 MSE취해줍니다.

#37

예측 레이어 증류는 교사 BERT가 생성한 최종 출력 레이어의 로짓값을 학생 BERT로 전달해 진행 합니다. 소프트 타깃과 소프트 예측간의 교차엔트로피 손실을 최소화해 예측 레이어 증류를 수행하며 예측 레이어 손실함수는 다음과 같습니다.

#38

모든 레이어의 loss function은 다음과 같이 정리 할 수 있습니다. M=0일때는 임베딩 레이어 손실을 m이 0보다 크고 M보다 작을 때는 트랜스포머 레이어를 의미하므로 은닉상태 손실과어텐션 레이어 손실의합으로 표현합니다. M=m+1일때는 예측레이어 손실을 사용합니다.

#39

다음은 데이터 증식 방법에 대해 살펴보겠습니다.

X[i]가 단일 단어인지 확인후 단일 단어인 경우 mask 토큰으로 x masked[i]를 마스킹 합니다. 그 후 Bert bhase를 사용하여 마스크 된 단어를 에측 하고 가장 가능성이 높은 첫 k개릐 단어를 예측하여 candidate 리스트에 저장합니다.

X[i]가 단일 단어가 아닐 경우 마스킹하는 대신에 글로브 임베딩을 사용하여 X[i]와 가장 유사한 단어 k를 확인하고 candidates 리스트에 저장합니다. 그후 균일 분포에서 p를 추출 임계값을 정하고 이후 과정을 진행합니다.

이후 X[i]에 대한 확률값을 비교하며 결과에 따라 임의의 단어로 교체하거나 실제단어 그대로 두게 됩니다.

#40

예시와 함께 과정을 되짚어 보며 X\_masked =[Paris, is, a, beautiful, city]에 입력 리스트를 복사하고 과정을 진행합니다. I=0일때 X[0]에 해당하는 단어가 단일 단어인지 확인하고 단일 단어 일경우 mask토큰으로 교체합니다. 이 예제에서는 paris로 단일단어이며 이를 mask로 교체 하였습니다. 그후 마스크된 자리에 넣을 단어를 확률분포에 따라 k개 추출하여 candidates에 집어 넣습니다. 여기에서 k=3이라는 가정하여 paris it that 3가지가 후보지에 저장되었습니다. 이후 확률 값을 샘플링하여 임계값과 비교후에 만약 임계값보다 작다면 candidates에 존재하는 임의의 단어로 교체합니다. 위와 같은 과정을 반복하여 데이터를 증식하게 됩니다.

#41

이제 저희 발표의 마지막 파트인 BERT SUM입니다.

#42

이 파트에서 다룰 주 내용은 다음 2가지입니다. BERT를 활용한 추출요약 BERT를 활용한 생성요약 먼저 추출요약에 대해서 간략하게 알아 보겠습니다.

#43

추출 요약은 주어진 텍스트에서 중요한 문장만 추출해 요약하는 과정을 의미합니다. 즉 많은 문장이 포함된 긴 문서에서 문서의 본질적인 의미를 담고 있는 중요한 문장만 추출해 문서의 요약을 만들어 내는 것이죠 예시를 살펴보면 다음과 같습니다. 머신러닝에 대한 장문의 글중 핵심 문장을 추출하여 요약본을 완성했죠

#44

다음은 생성요약입니다. 생성요약은 텍스트를 의역하여 요약을 만드는데요. 텍스트에 대한 설명을 더 명확하게 설명할 수 있는 단어를 조합하여 문장으로 내뱉습니다. 다음 문장을 보면 알 수 있듯이 텍스트의 본질적인 의미를 담아낼 수 있는 문장을 생성하였죠

#45

추출요약 태스크를 수행하려면 우선 BERT 모델 입력데이터 형태를 맞춰 주어야 하는데요 토큰 임베딩 세그먼트 임베딩 위치 임베딩을 진행해주어야 합니다. 각각의 임베딩은 위치정보 토큰정보 문장구분정보를 담고 있습니다.

#46

임베딩을 바탕으로 추출요약을 진행하기 위해서는 문장의 중요도가 필요한데요 즉 모든 문장의 표현할 수 있다면 문장을 추출하고 그렇지 않다면 버리는 과정이 필요합니다. 이를 위하여 문장의 표현을 얻기 위해 CLS토큰을 문장의 표현으로 사용합니다. 그림과 함께 살펴보면 CLS토큰을 문장의 대표 표현으로 출력하는 모습을 확일 할수 있습니다. 첫 번째 문장의 CLS는 R1으로 두번째 세번째는 R2 R3로서 출력되는 것을 알 수 있습니다.

#47

문장의 표현을 추출하고 R1 R2 R3로 정리한후 요약 레이어를 통해 문장의 중요성을 파악합니다. 이때 시그모이드 함수를 사용하며 시그모이드 대신에 LSTM과 Transfomer를 사용할 수도 있습니다.

#48

BERTSUM은 세가지 형태로 사용할 수 있는데요 1 단일 분류기 2 문장간 트랜스포머를 활용한 BERTSUM LSTM을 활용한 BERTSUM 이 3가지 입니다. 다음 그림은 LSTM과 Transfomer를 사용한 아키텍쳐의 과정입니다.

#49

이번에는 Rouge에 대해서 알아 보겠습니다. Roudge는 텍스트요약 태스크를 평가할 때 사용하는 지표입니다. 여기에서는 Rouge n과 L에 대해서만 다루도록 하겠습니다.

#50

ROUGE N은 서로겹치는 n그램에 따라 재현율을 측정합니다. 후보요약과 참조요약 간의 n그램 재현율을 표현해주며 정리하면 예측요약과 실제요약간의 재현율을 알려줍니다.

#51

다음은 ROUGE-L입니다. Roudge는 가장 긴 공통 시퀀스 LCS를 기반으로 측정합니다. LCS는 두 시퀀스 사이의 최대길이를 가지는 공통 하위 시퀀스를 의미하며 F measure를 계산할시에 측정합니다. F measure 식에서 정밀도와 재현율의 중요도 가중치를 조절할 때 b를 사용하며 재현율과 정밀도의 식은 다음과 같습니다.

#52

이상으로 BERT 발표를 마칩니다. 감사합니다.