Lstm에 대해 발표하겠습니다

시퀸스 데이터를 생성하는 방법은 이전 토큰을 입력으로 사용해서 시퀸스의 다음 토큰을 예측하는 것입니다. 예를들어 더 캣 이즈 온 더 ma 라는 토큰이 있으면 다음 토큰으로 t를 예측하는 식이죠. 이런식으로 이전 토큰으로 다음 토큰의 확률을 모델링할 수 있는 네트워크를 언어모델이라고 합니다.

이렇게 새로운 시퀸스를 생성하는 것을 샘플링이라고 하는데요. 초기 데이터, 즉 여기선 조건데이터라고 하는 데이터를 주입하면 다음 글자의 확률 분포를 계산해 새로운 토큰이 생성되고, 이 출력을 다시 입력 데이터로 써서 다른 토큰을 생성할 수 있습니다. 이런식으로 반복이 된다면 사람이 쓴 문장과 비슷한 시퀸스를 생성할 수 있게 됩니다.

샘플링을 할때 다음 글자를 선택하는 방법에 따라 결과물이 달라질 수 있습니다. 두가지 방법이 있는데 첫번째는 탐욕적 샘플링입니다. 반복적이고 예상 가능한 문자열을 만들어 논리적인 언어처럼 보이지는 않습니다. 두번째는 확률적 샘플링입니다. 확률적 샘플링은 실제 같은 새로운 단어를 만들어 재미있고 창의적으로 보이는 문장을 생성합니다. 탐욕적 샘플링을 확률적 샘플링으로 설명할 수 있는데요. 예를 들어 한 글자의 확률이 1이고, 나머지 글자가 모두 0인 경우죠. 하지만 단점은 무작의성의 양을 조절할수가 없다는 것입니다.

낮은 엔트로피를 가지면 더 실제같은 시퀸스를 생성하고 반대로 높은 엔트로피를 가지면 흥미로운 시퀸스를 생성할 수 있습니다. 샘플링을 할때 무작위성의 양을 바꾸어 최적의 엔트로피 값을 찾아야 좋은 데이터를 생성할 수 있습니다.

샘플링 과정에서 확률의 양을 조절하기 위해 소프트맥스 온도라는 파라미터를 사용합니다. 샘플링에 사용되는 확률분포의 엔트로피를 나타내는 값인데요. 온도가 낮으면 실제 있는 단어를, 온도가 높으면 흥미로운 단어들을 출력하지만 구조가 무너집니다.

케라스로 구현해보겠습니다. 처음은 데이터 전처리를 해야합니다. 사이트에서 텍스트 파일을 다운받은 뒤, lower() 함수를 사용해 소문자로 바꿔줍니다. 그 다음에 maxlen, 즉 60의 길이를 가진 시퀸스를 추출하겠습니다. 3 글짜씩 건너뛰면서 샘플링합니다. 그 아래는 시퀸스를 담을 리스트고요. For 문을 사용해 진행합니다. 텍스트 파일에서 고유한 글자를 찾아 글자의 인덱스와 매핑해줍니다. 아래는 글자를 원핫인코딩해주어 벡터화 시켜줍니다.

다음은 네트워크 구성입니다. 이 네트워크는 하나의 lstm 층과 그 뒤에 dense 분류기가 있습니다. 손실함수는 캐태고리컬 크로센트로피 손실을 사용하는데요. 이유는 타깃이 원핫 인코딩 되어있기 때문입니다.

이제 새로운 텍스트를 생성할 수 있습니다. 지금까지 생성된 텍스트를 주입후 모델에서 다음 글자에 대한 확률 분포를 뽑습니다. 그 다음 특정 온도로 이 확률 분포~~~~~~ 이와 같은 단계를 반복하면 새로운 텍스트를 생성할 수 있습니다.

이 함수는 모델의 예측이 주어졌을 때 새로운 글자를 샘플링하는 함수입니다. 이 반복문은 반복적으로 훈련해 텍스트를 생성합니다. 에포크마다 온도에 따른 텍스트를 생성합니다. 60 에포크동안 모델을 훈련합니다. 무작위로 시드 텍스트를 선택 후, 여러가지 샘플링 온도를 시도합니다. 온도마다 총 400개의 글자를 생성합니다. 지금까지 생성 된 글자를 원핫 인코딩으로 바꿉니다. 그 이후에 다음 글자를 샘플링합니다. 여기서 샘플링이 반복되는 것을 볼 수 있습니다.

결과를 보겠습니다. 시드 텍스트는 ~~입니다. 이것은 무작위로 선택된 것입니다. 온도가 0.2 일때 생성된 텍스트와 0.5일때 생성된 텍스트를 비교해보겠습니다. 겉으로 보기엔 큰 차이가 없는 것 같지만 자세히 읽어보면 온도가 0.5일때는 스펠링이 이상한 글자가 몇 있고 구조가 살짝 이상합니다.

온도 1.0을 보겠습니다. 온도 0.5보다 구조와 단어가 흥미로워진것을 확인할 수 있습니다.

온도 1.2는 1.0보다 더욱 더 구조와 단어가 이상해졌습니다.

이렇게 확인할 수 있듯이 낮은 온도는 반복적이도 예상되는 텍스트를 만드는 반면 높은 온도는 흥미롭고 있을법한 창의적인 단어를 만들어냅니다. 하지만 국부적인 구조가 무너지기 때문에 균형을 잘 맞추어야 합니다. 그래서 이러한 균형을 위해 소프트 맥스 함수를 사용합니다. 언제나 다양한 온도에서 적절한 값을 찾는 것이 제일 중요합니다.

다음은 딥드립입니다.

딥드림이란 합성곱 신경망이 학습한 표현으로 예술적인 이미지를 만들어내는 기법입니다. 옆에 보이는 사진이 딥드림이 적용된 사진입니다. 예술적으로 보이죠? 딥드림은 컨브넷 상위 층에 있는 특정 필터의 활성화를 극대화하기 위해 컨브넷의 입력에 경사 상승법을 적용합니다. 딥드림은 전체 증의 활성화를 최대화해 한꺼번에 많은 특성을 섞어 시각화합니다. 그리고 딥드림은 이미 가지고 있는 이미지를 사용해 왜곡시킵니다. 시각 품질 향상을 위해 여러 다른 스케일로 처리합니다.

사전 훈련된 모델을 로드하겠습니다. 여기서 보시면 0으로 처리해 모든 훈련 연산을 비활성화 시켜 모델을 훈련시키지 않습니다. 이건 최대화하려는 손실 층의 활성화가 기여할 양을 정하는 겁니다. 층 이름과 계수를 매핑한 딕셔너리입니다. 아래는 손실 텐서를 정의하는 코드입니다. 여기서 선택한 층의 활성화에 대한 노름의 가중치 합입니다.

이 코드는 경사 상승법 과정에 대한 코드입니다. 1. 생성된 딥드림 이미지를 저장하는 텐서입니다. 2. 여기선 손실에 대한 딥드림 이미지와 그래디언트를 계산합니다. 3. 그것을 정규화 시킵니다. 여기서는 케라스 펑션 객체를 만들고 (아래 def) 여기서 경사상승법을 여러번 반복하여 수행합니다.

마지막으로 딥드림 알고리즘을 설명하기 전에 옥타브를 먼저 설명드리겠습니다. 작은 이미지로 시작해서 딥드림을 입힌 뒤, 업스케일 즉 크기를 키웁니다. 이 과정에서 뭉개지거나 픽셀 경계가 나타나므로 원본 이미지를 사용해 디테일을 입힙니다. 이 과정을 반복하면 이미지의 사진은 점점 커지지만 화질은 나빠지지 않습니다.

여기선 싸이파이를 사용합니다.

설명

설명

설명

설명 + 결과를 보시면 위쪽은 제가 조금 잘랐지만 사이즈가 250에서 350으로 커진것을 볼 수 있습니다. 이것은 옥타브의 효과입니다.

처음 이미지와 딥드림이 적용된 이미지를 살펴보면 개인차가 있겠지만 예술적으로 보이는 이미지가 알고리즘으로 인해 탄생되었습니다. 감사합니다.