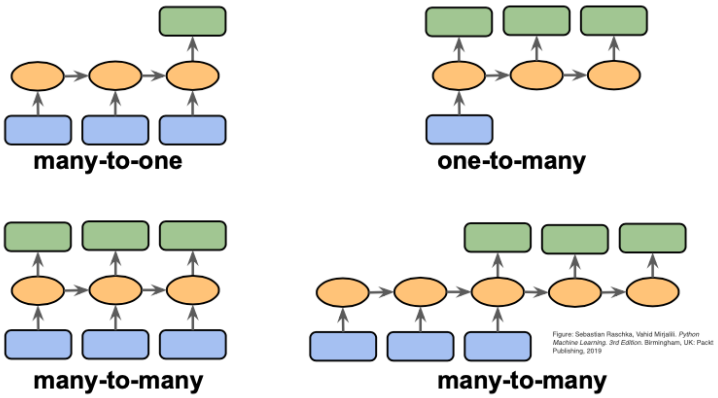
2/2 Recurrent Model (1)

Sequential Model

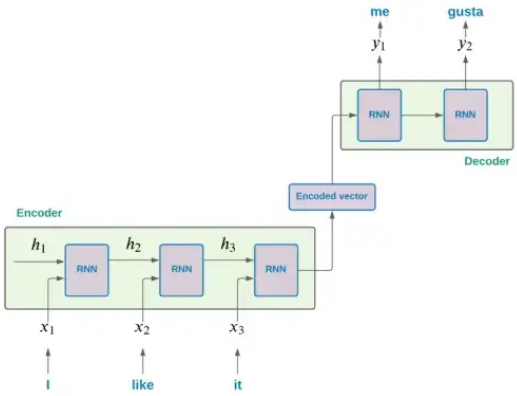
기존 지도학습 머신러닝 기법들은 데이터가 i.i.d. (independent and identically distributed; 독립 동일 분포)라고 가정하지만, 시퀀스 데이터에서는 순서가 존재한다.

시퀀스 데이터 적용 예시: image captioning, time series, NLP, machine translation, speech recognition, DNA sequence analysis ⇒ RNN, LSTM, GRU, auto-encoders, transformer 등 활용

풀고자 하는 문제에 따라 다양한 시퀀스 모델링 기법을 활용할 수 있다.



Ex) seq2seq 모델



인코더의 인풋 시퀀스: $x_1, x_2, \ldots, x_N$ (N=인풋 토큰의 개수)

디코더의 아웃풋 시퀀스: $y_1, y_2, \ldots, y_T$ (T=시퀀스에 있는 최대 토큰 개수)

각 아웃풋 토큰의 확률: $P(y_t|y_1, y_2, \ldots, y_{t-1}, x)$

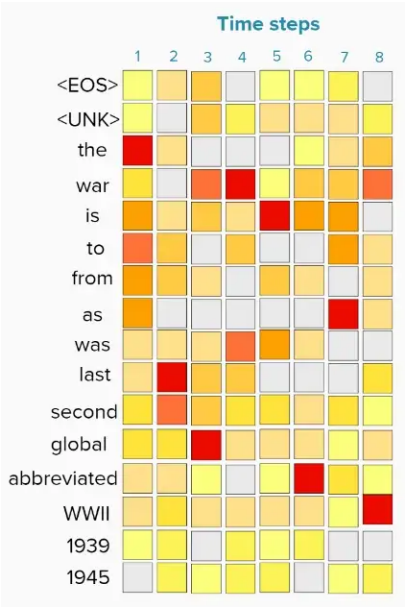
Greedy Search

: 코퍼스 V에서 조건부확률이 가장 높은 토큰을 사용하는 기법

$$y_t = argmax_{y \in V} P(y|y_1, y_2, \ldots, y_{t-1}, x)$$

Ex) V에 존재하는 단어들:

<EOS>, <UNK>, the, war, is, to, from, as, was, last, second, global, abbreviated, WWII, 1939, 1945



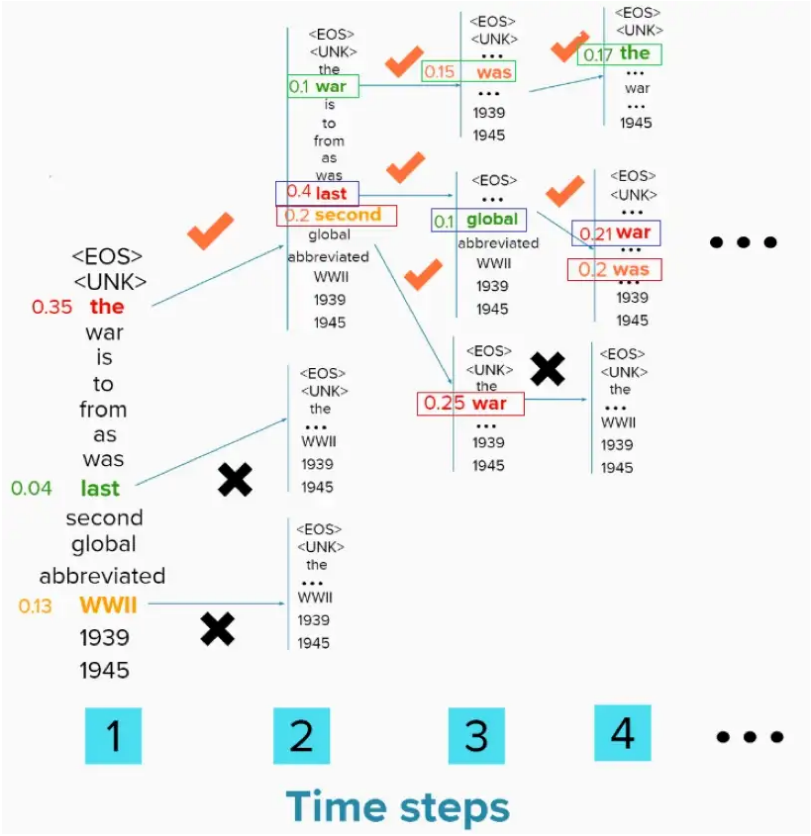
- 각 time step에서 조건부 확률이 가장 높을수록 붉은색으로 표시
- 각 time step마다 확률이 가장 높은 단어 선택
- 디코더의 시퀀스 아웃풋: ***the last global war is abbreviated as WWII.***

단점: 가장 높은 확률이 나온 토큰으로만 밀고가서 다른 조합을 고려하지 않을 수 있음

Beam Search

: Greedy search 기법에 가장 높은 조건부확률을 가진 토큰의 개수를 나타내는 파라미터 `beam_size`를 도입함

Ex) `beam_size`=3



Sequence 1 — ***the last global war*** — *(0.35 * **0.4** * 0.1 * 0.21) = 0.0029*

Sequence 2 — ***the second war was*** — *(0.35 * **0.2** * 0.25 * 0.2) =**0.0034***

Sequence 3 — ***the war was the*** — (0.35 * **0.1** * 0.15 * 0.17) = 0.00089

- Greedy search에서는 가장 높은 확률을 가진 *last*만 고려하여 sequence 1 선택. Beam search는 더 높은 확률을 가진 sequence 2를 찾아냄

단점: `beam_size`를 크게 설정할수록 성능이 좋아지지만 디코더 속도가 느려지고, 일정 값을 넘어가면 더이상 개선되지 않는다. 또한 가장 높은 점수를 가진 시퀀스를 찾지 못할 수도 있다.

N-gram

: n개의 연속적인 단어 나열을 하나의 토큰으로 간주하여 다음에 나올 단어를 예측하는 언어모델. 조건부 확률을 이용하는 통계적 언어모델 (Statistical Language Model)의 일종이다.

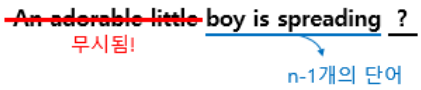
N-gram의 마지막 단어 w_i를 예측하기 위해 이전 시퀀스인 $w_{i-(N-1)}^{i-1}$: $w_{i-(N-1)}, \ldots, w_{i-1}$를 활용하여 확률 $P(w_i|w_{i-(N-1)}, \ldots, w_{i-1})$를 계산한다.

Ex)

```
An adorable little boy is spreading smiles
```

- **uni**grams : an, adorable, little, boy, is, spreading, smiles
- **bi**grams (w_{i-1}^{i}) : an adorable, adorable little, little boy, boy is, is spreading, spreading smiles
- **tri**grams (w_{i-2}^{i}): an adorable little, adorable little boy, little boy is, boy is spreading, is spreading smiles
- **4-**grams : an adorable little boy, adorable little boy is, little boy is spreading, boy is spreading smiles

n=4일때, spreading 다음에 올 단어는 그 앞의 세 단어로 판단



조건부확률

$$P(w|\text{boy is spreading}) = \frac{\text{count}(\text{boy is spreading } w)}{\text{count}(\text{boy is spreading})}$$

이때 가지고 있는 코퍼스에 따라서 boy is spreading insults가 boy is spreading smiles보다 많다면, 조건부확률에 따라 다음 단어를 insults로 예측해버린다.

- 한계점
  - 문장의 모든 단어가 아닌 일부 단어를 고려함으로써 코퍼스에서 카운트 할 수 있는 확률을 높일 수 있지만, 위 예시처럼 여전히 의도한대로 문장을 만들지 못할 가능성이 있다.
  - n을 정하는데엔 trade-off가 존재한다 (최대 5를 넘지 않는게 권장사항)
  - MLE (maximum likelihood estimation)을 활용하기 때문에 n-gram이 훈련 코퍼스에 없을 경우 대응 불가능

Katz-Backoff Model

: 코퍼스에서 관측되지 않은 n-gram을 근사화하는 기법. n-1 gram으로 “backoff”을 반복하여 0이 아닌 카운트를 발견할시 해당 확률 사용

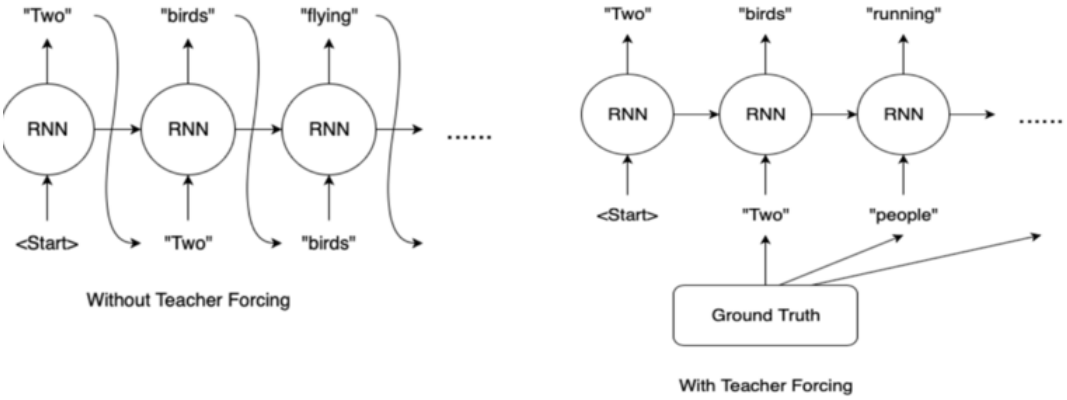
$$P_{Katz}(w_i|w_{i-N+1}^{i-1}) = \begin{cases} P^*(w_i|w_{i-N+1}^{i-1}), & \text{if } C(w_{i-N+1}^{i}) > 0 \\ \alpha(w_{i-N+1}^{i-1}) P_{Katz}(w_i|w_{i-N+2}^{i-1}), & \text{otherwise.} \end{cases}$$

$P*$: Good-Turing discounted probability

α: 정규화 상수 (관측되지 않은 N-gram으로 얼마나 가중치를 옮길 것인지)

smoothing: 말이 되는 시퀀스지만 코퍼스에 존재하지 않는 경우, 카운트가 0이 아닌 시퀀스의 확률을 가져와서 카운트가 1이상이 되게끔 일종의 discounting을 하는 것. (Laplace, Good-Turing, Simple Good-Turing)

Non-Teacher Forcing vs Teacher Forcing



Teacher Forcing: 이전의 time step의 output이 아닌 **ground truth** (target word)를 활용하여 RNN을 학습하는 기법.

Ex)

```
Mary had a little lamb whose fleece was white as snow
```

시퀀스의 시작과 끝을 표기하기 위해 토큰 입력

```
[START] Mary had a little lamb whose fleece was white as snow [END]
```

모델에 “[START]”를 넣고 다음에 올 단어를 생성하도록 함

```
X, yhat
[START], a
```

모델이 틀린 답인 “a”를 생성했으니, 이를 지우고 “Mary”를 입력해준다. 각 인풋-아웃 쌍에 대해 이 과정을 반복하면 올바른 시퀀스를 빠르게 학습할 수 있다.

```
X, yhat
[START], ?
[START], Mary, ?
[START], Mary, had, ?
[START], Mary, had, a,  ?
...
```

참고

https://deeplearningmath.org/sequence-models.html

https://medium.com/@jessica_lopez/understanding-greedy-search-and-beam-search-98c1e3cd821d

https://wikidocs.net/21692

https://rpubs.com/leomak/TextPrediction_KBO_Katz_Good-Turing

https://machinelearningmastery.com/teacher-forcing-for-recurrent-neural-networks/

https://blog.naver.com/sooftware/221790750668