Latent Dirichlet Allocation using Gibbs Sampler

0. Intro

LDA의 사후 분포를 유도하는 방법은 아래와 같다.

- Variational Inference
  - Traditional Variational Inference: hyper parameter을 설정하고 mean-field assumption을 통해서 사후 분포와 KL Divergence가 최소인 variational distribution을 사후 분포의 approximation이라고 본다.
  - Variational EM: hyper parameter을 미리 정하는 것이 아니로 이것도 log likelihood을 통해서 추정하여, hyper parameter에 대한 Empirical Bayes Estimator을 생성한다. E step에서 VI을 통해서 variational parameters을 추정하고 이를 고정시켜 ELBO을 최대로 만드는 Empirical Bayes Estimator을 도출한다.
  - Online Variational Inference:
- Gibbs Sampler
  - Collapsed Gibbs Sampler: Gibbs sampler을 변형으로, 관심없는 모수에 대해서는 integrate out한다.
  - Fast Collapsed Gibbs Sampler: 속도가 더 빠른 Collapsed Gibbs Sampler

여기서는 Collapsed Gibbs Sampler로 LDA를 구현해보고 Online Variational Inference로 LDA를 구현한 python의 scikit learn LDA와 결과를 비교해보고자 한다.

1. Collapsed Gibbs Sampler

Collapsed Gibbs Sampler는 나머지 변수는 고정시킨 채 한 변수만을 변화시키되, 불필요한 일부 변수를 샘플링에서 제외(integrate out)하는 방법으로, LDA에서 불필요한 변수는 ϕ, θ이고 필요한 변수는 z이다. 따라서 깁스 샘플러 과정을 나타낸 수식은 아래와 같다.

$$p(z_i = j \mid z_{-i}, w) \tag{1}$$

(1)이 의미하는 바는 다음과 같다. z_{-i}, w을 조건으로 하여, z_i 단어가 토픽 j에 할당될 확률을 의미하는데, 여기서 w는 단어들이고 z_{-i}는 i 번째 단어의 토픽 정보를 제외한, 나머지 단어들의 토픽

정보이다.

몇 가지 유도 과정을 거치면 d번째 문서의 i번째 단어의 토픽 $z_{d,i}$가 j번째 에 할당될 확률은 다음과 같다.

$$p(z_{d,i} = j \mid z_{-i}, w) = \frac{n_{d,k} + \alpha_j}{\sum_{i=1}^{K}(n_{d,i} + \alpha_i)} \times \frac{v_{k,w_{d,n}} + \beta_{w_{d,n}}}{\sum_{j=1}^{V}(v_{k,j} + \beta_j)} \tag{2}$$

(2)의 notation을 정리하면 아래와 같다.

표기	내용
$n_{d,k}$	k번째 토픽의 d번째 문서의 단어 총 개수
$v_{k,w_{d,n}}$	전체 코퍼스에서 k번째 토픽에 단어 $w_{d,n}$이 나타난 총 횟수
$w_{d,n}$	d번째 문서에서 n번째로 나타난 단어
α	문서의 토픽 분포 생성을 위한 디리클레 분포 parameter(hyper parameter)
β	토픽의 단어 분포 생성을 위한 디리클레 분포 parameter(hyper parameter)
K	미리 지정하는 토픽의 수
V	코퍼스에서 등장하는 unique 단어 수

표 1

2. Python Code and Explanation

파이썬 코드에서 iteration 부분에 대한 추가 예시이다.

Collapsed Gibbs Sampler에서는 (1)의 조건부 분포를 유도하는 것이 목적이다. 이미 언급했듯이, 단어들과 d번째 문서에서 i번째 단어의 주제를 제외하고, 나머지 단어들의 주제를 조건으로 할 때의 확률인데, 구체적으로 어떻게 제외하고 확률을 유도하는지 알아보자.

그런데 확률을 계산하기 위해서 처음에 initial 토픽이 있어야 하므로, 처음에는 토픽 초기값을 랜덤하게 준다.

Doc1의 단어들에 초기 토픽을 랜덤하게 배정하고 어느 정도 iteration이 지난 후의 Doc1 단어별로 해당되는 토픽이 아래와 같다고 가정해보자. $z_{d,n}$은 d번째 문서의 i번째 단어의 토픽이고 $w_{d,n}$은 d번째 문서의 i번째의 관찰된 단어이다.

Doc1의 i번째 단어의 토픽 ($z_{1,i}$)	1	3	1	2	2
Doc1을 구성하는 단어 ($w_{1,n}$)	statistics	very	fun	and	practical

표 2

또한 토픽별로 코퍼스에 등장하는 모든 단어들이 어떤 토픽에 속해 있는지를 랜덤하게 배정하고 어느 정도 iteration이 지난 후의 토픽별 단어들의 분포가 아래와 같다고 가정하자.

	Topic1	Topic2	Topic3
statistics	5	1	0
very	2	3	1
fun	0	3	6
and	0	3	1
practical	4	2	0

표 3

(1)의 이제 깁스 샘플링을 통해서 $z_{1,1}$의 조건부 확률인 $p(z_i = j \mid z_{-i}, w)$을 구해보자. z_{-1}은 첫 번째 단어의 토픽 정보를 지운 상태를 의미하고, w은 주어진 단어들을 의미한다. 이러한 조건을 표로 다시 나타내면 아래와 같다.

Doc1의 i번째 단어의 토픽 ($z_{1,i}$)	?	3	1	2	2
Doc1을 구성하는 단어 ($w_{1,n}$)	statistics	very	fun	and	practical

표 4

$z_{1,1}$, 즉 Doc1의 첫 번째 단어인 statistics의 토픽 정보가 ?로 가려진 상태에서 $n_{1,1} = 1,\ n_{1,2} = 2,\ n_{1,3} = 1$이다($n_{d,k}$는 k번째 토픽의 d번째 문서의 단어 총 개수임을 표 1에서 명시하였다.)
이제 statistics의 토픽 정보를 지운 상태를 표 3에도 적용해야 하므로 단어별 토픽 분포 표에서 $v_{1,statistics}$가 1이 줄어들 것이다.

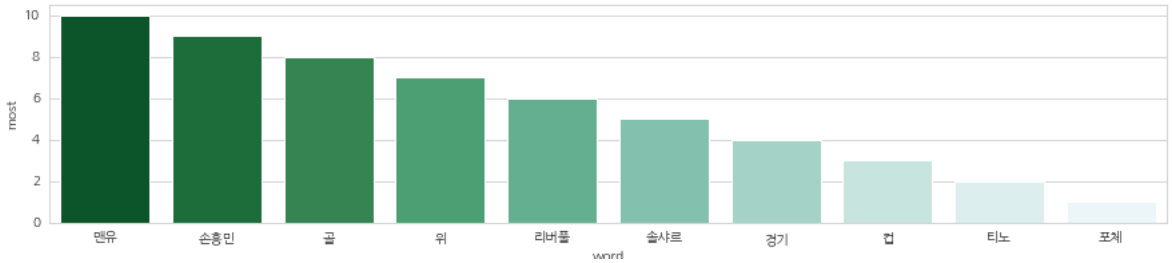
	Topic1	Topic2	Topic3
statistics	**5-1**	**1**	**0**
very	2	3	1
fun	0	3	6
and	0	3	1
practical	4	2	0

표 5

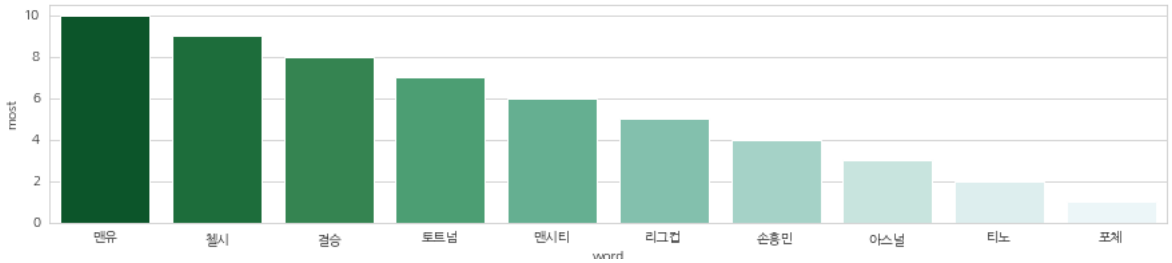
즉, $v_{1,statistics} = 4$, $v_{2,statistics} = 1$, $v_{3,statistics} = 0$ 이다. 여태까지의 논의를 종합하여 (2) 의 $p(z_{1,statistics} = j \mid z_{-statistics}, w)$ 을 계산할 수 있다.

3. Results

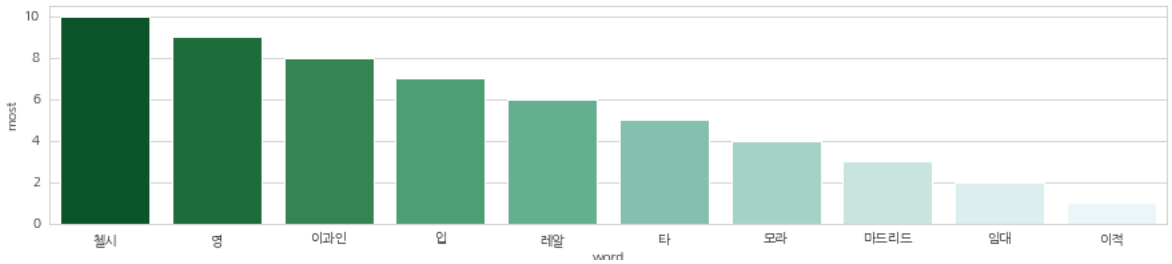
python의 scikit learn에 구현되어 있는 LDA(Online VI 이용) 와 깁스 샘플링을 통해 구현한 LDA의 결과를 비교해보았다. 특히, 각 토픽별로 가장 많이 나온 단어들을 아래와 같이 살펴보았다.



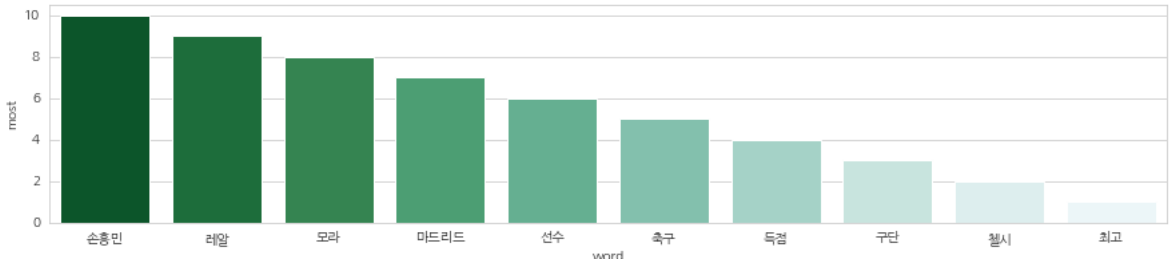
깁스 샘플링을 이용한 LDA Topic 1



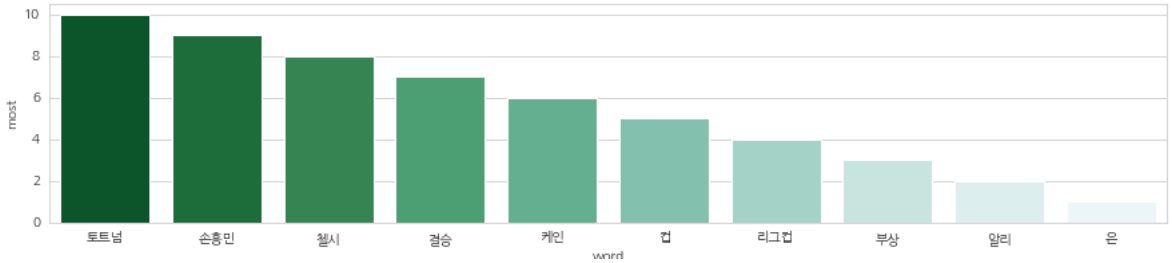
scikit learn을 이용한 LDA Topic 1



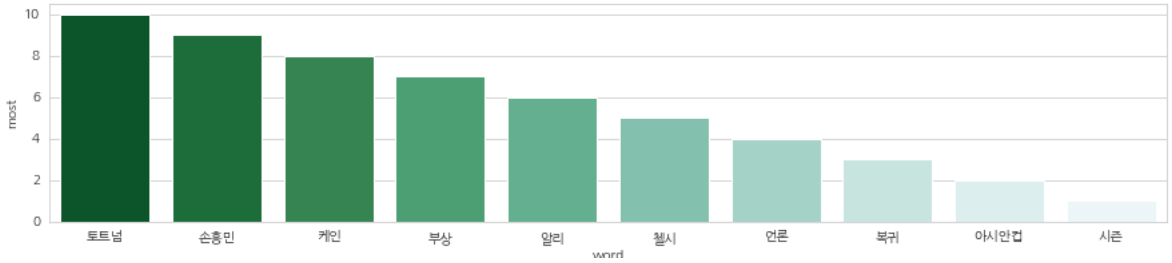
깁스 샘플링을 이용한 LDA Topic 2



scikit learn을 이용한 LDA Topic2



깁스 샘플링을 이용한 LDA Topic3



scikit learn을 이용한 LDA Topic3