A Biterm Topic Model for Short Text

1. Introduction

tweets나 instant message와 같이 짧은 텍스트에서 topics을 발견하는 것은 최근들어 그 중요도가 커지고 있다. 이러한 형태의 데이터에 대해서, LDA나 PLSA와 같은 conventional topic models은 잘 작동하지 않는다. 왜냐하면, 내재하는 topics을 밝히기 위하여 document 단위의 word co-occurrence을 보기 때문이다.

본 논문에서는 새로운 방법인 biterm topic model (BTM)을 제시한다. BTM은 *document* 단위에서 word의 generating 과정을 보는 것이 아니라, 전체 *corpus* 단위에서 word co-occurrence의 패턴 또는 생성 (biterms)을 직접적으로 모델링한다. 즉, word co-occurrence 패턴을 직접적으로 모델링하여 topic learning의 질을 향상시키고, 동시에 document level이 아니라 corpus level에서 이를 수행함으로써 short text의 큰 단점인 sparsity 문제를 해결하는데 주요 목적이 있다.

short text에 대하여 다음과 같은 어려움이 있다. 1) 짧은 문서에서 나타나는 단어들은 충분히 많은 단어들이 있는 문서에서보다 덜 discriminative한 역할을 수행한다. 2) 말 그대로 '짧기' 때문에 충분한 정보를 담아내지 못한다. 이러한 short text 문제를 극복하기 위해서 몇 가지 시도들이 있었다. 하나는 short text을 합쳐서 마치 하나의 문서로 보는, psedo-documents을 topic modeling 이전에 만드는 것이다. 하지만 이러한 heristic data aggregation은 data마다 방법이 완전히 달라지므로, general한 접근법은 아니다. 또 한 가지 시도는 데이터에 좀 더 강한 가정을 두는 것이다. 전형적인 방법은, short text가 하나의 topic으로 구성된다는 가정이다. 이는 마치 unigram의 가정과 유사한데, 따라서 unigram의 한계점을 그대로 가진다. 즉, short text라도 여러 개의 topics으로 구성될 수 있으므로 덜 flexible하다는 것이다.

2. BTM Approach

2.1 Biterm Extraction

시작하기에 앞서, Biterm의 정의와 예시를 살펴보자.

- Biterm: An unordered word-pair co-occurred in a short context
- 예시 : "I visit apple stor"라는 문장에서 stop word인 I를 제외하면, "visit apple", "visit store", "apple store"의 세 biterm이 있다.

2.2 Biterm Topic Model

BTM의 핵심 아이디어는, 전체 corpus에서 biterms을 모아서 하나의 short text가 가 지고 있는 sparsity 문제를 해결하는데에 있다. LDA는 문서 하나를 mixture of topics으로 보는 반면에, BTM은 문서 하나가 짧으므로 전체 corpus를 mixture of topics으로 본다. 또한 LDA는 하나의 문서를 specific topic에서 뽑힌 words가 구성한다고

보는 반면에, BTM은 전체 corpus를 specific topic에서 독립적으로[1] 뽑힌 biterm이 구성한다고 본다. 이와 같이 LDA는 문서 하나가 어느 정도 길이가 된다고 가정하므로, 문서 시각으로 본다. 따라서 LDA의 model 또한 blei 논문에서 'generative process of a document'이라고 많이 나오는 것이다. 반면에, BTM은 short text을 타겟으로 한다. 따라서 하나의 text(문서) 보다는 전체 corpus에 주목하고 (sparsity 해결을 위해) 따라서 본 논문에서도 'generative process of the corpus'라는 표현으로 아래와 같은 과정이 소개된다.

1. For each topic z,
   (a) draw a topic-specific word distribution. $\phi_z \sim Dir(\boldsymbol{\beta})$
2. Draw a topic distribution $\boldsymbol{\theta} \sim Dir(\boldsymbol{\alpha})$ for the whole collection.
3. For each biterm b in the biterm set B
   (a) draw a topic assignment $z \sim Multi(\boldsymbol{\theta})$
   (b) draw two words: $w_i, w_j \sim Multi(\phi_z)$

위의 과정을 따라가면, biterm $b = (w_i, w_j)$에 대해 아래와 같은 joint probability을 구할 수 있다.

$$\begin{aligned} P(b) &= \sum_z P(w_i, w_j, z) \\ &= \sum_z P(z)P(w_i, w_j \mid z) \\ &= \sum_z P(z)P(w_i \mid z)P(w_j \mid z) \ \because w_i, w_j \textit{ are independent given topic} \\ &= \sum_z \theta_z \phi_{i|z} \phi_{j|z} \end{aligned} \tag{1}$$

따라서 전체 corpus에 대한 likelihood는 다음과 같다.

$$P(B) = \prod_{(i,j)} \sum_z \theta_z \phi_{i|z} \phi_{j|z} \tag{2}$$

이로부터, word co-occurrence pattern을 직접적으로 모델링함을 알 수 있다. 왜냐하면 LDA는 한 문서에 대해서 likelihood을 세웠던 반면에, likelihood를 biterm에 대해서 세웠기 때문이다. 또한 하나의 단어가 두 단어의 동시 출현에 대한 정보는 하나의 단어가 출현할 때의 정보보다 더 많은 것을 내포할 것이다. 또한 이러한 biterm은 하나의 문서가 기준이 아니라, 하나의 corpus가 기준이므로 정보를 short text에서 제한된 정보를 더 사용하는 방법인 것이다. LDA와의 차이점을 명확히 보기 위해 아래 그림을 살펴보자.

[1] 논문에서는 indenpendently에 대한 설명을 다음과 같이 한다. 엄밀히 말해서, 한 문서에서 two biterms은 독립이 아닐 것이다. 하지만 이러한 간단한 가정으로 인해서 BTM 모델의 computation을 쉽게 할 수 있다고 한다.

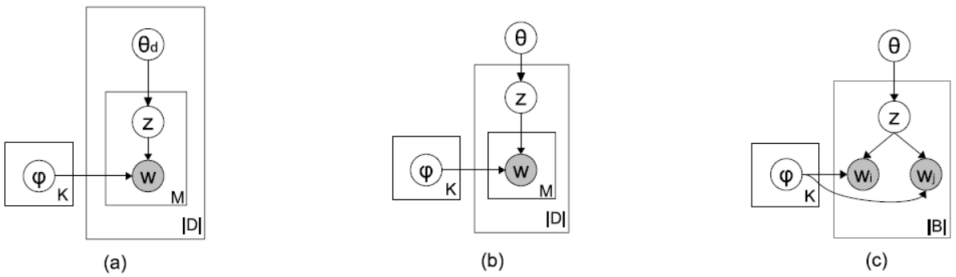

그림 1: (a)는 LDA, (b)는 mixture of unigrams, (c)는 BTM의 graphical representation

(a)에서 볼 수 있듯이, LDA는 각각의 문서는 document-level($\boldsymbol{\theta}_d$에 subscript로 d가 있는 것으로 보아, 문서 단위의 모수라고 볼 수 있다.) topic distribution인 $\boldsymbol{\theta}_d$를 뽑고, 문서 내의 단어 w마다 topic assignment인 z을 뽑는 구조이다. Dirichlet prior에서 $\boldsymbol{\theta}_d$를 뽑고, 바로 이 동일한$\boldsymbol{\theta}_d$를 기반으로 $Multinomial(\boldsymbol{\theta}_d)$에서 word w에 대한 topic을 뽑는 구조이다. 따라서 LDA에서 생성되는 단어들은 문서 단위의 모수인 $\boldsymbol{\theta}_d$를 기반으로 하므로, LDA는 문서 단위의 word co-occurrenece을 잡는다고 볼 수 있다.
반면에, (b)는 $\boldsymbol{\theta}$에 subscript가 없다. 따라서 문서 단위가 아니라 corpus 전체에 공통적으로 해당되는 모수라고 볼 수 있다. 전체 corpus의 정보를 leverage함으로써, sparsity 문제를 어느정도 해결할 수 있지만 한 문서의 모든 단어가 공통 주제에서 나왔다는 가정은 너무 strong하다.
(c)는 (a), (b)의 장점을 합치면서 단점은 보완한다. 즉, corpus level의 모수 $\boldsymbol{\theta}$을 통해서 data sparsity을 해결함과 동시에 문서를 biterms로 쪼갬으로써 (b)의 단점을 보완한다.

2.3 Inferring Topics in a Document

LDA와 BTM의 가장 큰 차이점은 LDA는 document generation process을 모델링하지만 BTM은 그렇지 않다는 것이다. 따라서 LDA에서는 각 문서별로 topic이 차지하는 비중을 확인할 수 있었지만 BTM에서는 topic learning process 중에, 직접적으로 topic proportions을 알 수 없다. BTM에서 topic proportions을 추론하기 위해서, 한 문서의 topic proportions은 문서로부터 생성되는 biterms의 topic proportions에 대한 기대값과 같다는 가정을 한다.

$$P(z \mid d) = \sum_{b} P(z \mid b) P(b \mid d) \tag{3}$$

(3)에서 $P(z \mid b)$는 Bayes Theorem을 통해서 계산할 수 있다.

$$
\begin{aligned}
P(z \mid b) &= \frac{P(z,b)}{P(b)} \\
&= \frac{P(z,w_i,w_j)}{\sum_z P(z)P(w_i \mid z)P(w_j \mid z)} \quad \because (2) \\
&= \frac{p(z)p(w_i,w_j \mid z)}{\sum_z P(z)P(w_i \mid z)P(w_j \mid z)} \\
&= \frac{P(z)P(w_i \mid z)P(w_j \mid z)}{\sum_z P(z)P(w_i \mid z)P(w_j \mid z)} \quad \because w_i, w_j \; are \; indep \; given \; topic \\
&= \frac{\theta_z \phi_{i|z} \phi_{j|z}}{\sum_z \theta_z \phi_{i|z} \phi_{j|z}}
\end{aligned} \tag{4}
$$

문제는 $P(b \mid d)$인데, 이는 아래와 같이 empirical distribution of biterms을 사용한다.

$$
P(b \mid d) = \frac{n_d(b)}{\sum_b n_d(b)} \tag{5}
$$

$$
where \; n_d(b) : \; frequency \; of \; biterm \; b \; in \; document \; d
$$

3. Parameters Inference

3.1 Inference by Gibbs Sampling

BTM의 gibbs sampling 알고리즘은 아래와 같다.

Algorithm 1: Gibbs sampling algorithm for BTM

Input: the number of topics K, hyperparameters α, β, biterm set B

Output: multinomial parameter ϕ and $\boldsymbol{\theta}$

initialize topic assignments randomly for all the biterms

for $iter = 1$ *to* N_{iter} **do**

  for $b \in B$ **do**

    draw z_b from $P(z|\mathbf{z}_{-b}, B, \alpha, \beta)$

    update n_z, $n_{w_i|z}$, and $n_{w_j|z}$

compute the parameters ϕ in Eq.(5) and $\boldsymbol{\theta}$ in Eq.(6)

BTM에서는 latent variables인 $z, \phi, \boldsymbol{\theta}$가 모수이므로 이에 대한 gibbs sampling을 수행해야 한다. 하지만 collapsed gibbs smapling을 이용하여, $\phi, \boldsymbol{\theta}$는 conjugate prior로 인해서 integrate out된다. 따라서 나머지 모든 변수를

조건으로 하여, 각각의 biterm에 대한 topic assignment인 z만을 sample하면 된다. 몇 가지 계산 과정을 거치면, 아래와 같은 conditional probability을 얻을 수 있다.

$$P(z \mid \boldsymbol{z}_{-b}, B, \alpha, \beta) \propto (n_z + \alpha)\frac{(n_{w_i|z} + \beta)(n_{w_j|z} + \beta)}{\left(\sum_w n_{w|z} + M\beta\right)^2} \tag{6}$$

여기서 n_z는 number of times of the biterm b assigned to the topic z, $n_{w|z}$는 number of times of the word w assigned to the topic z, M은 전체 단어의 갯수이다. LDA에서와 마찬가지로 hyper-parameter인 α, β는 symmetric Dirichlet priors로 준다.
마지막으로 $\phi, \boldsymbol{\theta}$를 아래와 같이 추정한다. $|B|$는 biterms의 총 갯수이다.

$$\phi_{w|z} = \frac{n_{w|z} + \beta}{\sum_w n_{w|z} + M\beta} \tag{7}$$

$$\theta_z = \frac{n_z + \alpha}{|B| + K\alpha} \tag{8}$$

3. Examples for Conditional Probability
그러면, CGS의 conditional probability인 (6)을 어떠한 방식으로 구현하는지 아래와 같이 실제 예시를 통해서 살펴본다. input으로 topic의 개수인 K, hyper-parameter인 α, β 그리고 Biterm Set인 B가 주어진다. Biterm은 각 문서 내에서 ${}_nC_2$개 만큼 Biterm을 생성한다.
BTM의 Collapsed Gibbs Sampler에서는 (6)의 조건부 분포를 유도하는 것이 목적이다. LDA의 CGS에서는 document index가 있었지만, BTM의 CGS에서는 그렇지 않다. 왜냐하면, 거듭 강조하지만 LDA는 document generative process를 모델링하지만, BTM은 corpus generative process을 모델링하기 때문이다. 이러한 이유로, LDA에서는 document을 나타내는 subscript가 계속 존재했지만, BTM에서는 그러한 index가 존재하지 않는다.
그런데 확률을 계산하기 위해서 처음에 initial 토픽이 있어야 하므로, 처음에는 토픽 초기값을 랜덤하게 준다.
가장 먼저 input으로 주어지는 Biterm Set인 B는 각 문서 내에서 단어들을 tokenization을 한 후에, 2개씩 고른 것이다. 이렇게 Biterm Set을 생성한 후에는 document의 구분 없이, B개의 Biterm Set을 고려하게 된다. 예를 들어 주어진 Corpus에서 B개의 biterm을 생성하고 이에 random으로 주제를 할당한 뒤에, 어느 정도 iteration을 하고 난 후, 전체 B개의 Biterm 중 일부 5개의 Biterm의 주제가 아래와 같다고 가정해보자.

주어진 Corpus의 b번째 Biterm의 토픽 (z_b)	**1**	**2**	**3**	**2**	**1**
주어진 Corpus의 b번째 Biterm(w_i, w_j)	**(statistics, very)**	**(statistics, DL)**	**(data science, coding)**	**(Bayesian, ML)**	**(very, useful)**

표 2

표2에서 LDA에서의 세팅과 비교해보면, document을 나타내는 subscript가 존재하지 않고, 단어 단위가 아니라, Biterm 단위로 topic이 배정되어 있다.

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

	Topic1	Topic2	Topic3
(statistics, very)	5	1	0
(statistics, DL)	2	3	1
(data science, coding)	0	3	6
(Bayesian, ML)	0	3	1
(very, useful)	4	2	0

표 3

(1)의 이제 깁스 샘플링을 통해서 z_b의 조건부 확률인 $p(z_b = j \mid \boldsymbol{z}_{-b}, B, \alpha, \beta)$을 구해보자. 여기서 $b = bth\ Biterm = (statistics, DL)$이라고 가정하자. z_{-b}는b번째 단어의 토픽 정보를 지운 상태를 의미하고, B는 주어진 Biterm 들을 의미한다. 이러한 조건을 표로 다시 나타내면 아래와 같다.

주어진 Corpus의 b번째 Biterm의 토픽 (z_b)	**1**	**?**	**3**	**2**	**1**
주어진 Corpus의 b번째 Biterm(w_i, w_j)	**(statistics, very)**	**(statistics, DL)**	**(data science, coding)**	**(Bayesian, ML)**	**(very, useful)**

표 4

z_b, 즉 Corpus의 b번째 Biterm인 (statistics, DL)의 토픽 정보가 ?로 가려진 상태에서 update해야하는 것은 (6)에서 n_z, $n_{w_i|z}$, $n_{w_j|z}$이다. 여기서 $z = 1, w_i = statistics, w_j = DL$이라고 생각하고 각각을 구해보자. 우선 b번째 Biterm인 (statistics, DL)의 토픽 정보가 가려진 상태에서, 표 3을 업데이트해야 한다. 이는 아래와 같이 수행한다.

	Topic1	Topic2	Topic3
(statistics, very)	5	1	0
(statistics, DL)	**2**	**3-1**	1
(data science, coding)	0	3	6
(Bayesian, ML)	0	3	1
(very, useful)	4	2	0

표 5

위에 굵게 표시된 부분을 보면, (statistics, DL)는 표3에서 알 수 있듯이 원래 topic2을 할당받았는데, 그 정보가 지워지는 바람에 3에서 1을 뺐다. 여기서, n_z의 정의가 number of times of the biterm b assigned to the topic z

라는 점을 고려할 때, $n_1 = 2, n_2 = 2, n_3 = 1$로 update 하게 된다.
BTM에서는 n_z 뿐만 아니라 $n_{w|z}$도 고려하게 된다. 따라서 Biterm에 대한 표뿐만 아니라 single-term에 대한 빈도도 고려해야할 것이다. z_{-b}인 상태에서, 즉 표 5에서 single-term이 각 topic에 속하는 빈도를 구하면 아래와 같다.

	Topic1	Topic2	Topic3
statistics	**7**	**1+(3-1)**	**1**
very	9	3	0
DL	**2**	**(3-1)**	**1**
data science	0	3	6
coding	0	3	6
Bayesian	0	3	1
ML	0	3	1
useful	4	2	0

표 6

표 6에서 Biterm의 각각의 단어, $w_i = statistics$, $w_j = DL$에 특별히 진하게 표시를 했다. $n_{w|z}$의 정의가 number of times of the word w assigned to the topic z이므로, $n_{w_i=statistics|z=1} = 7$, $n_{w_i=statistics|z=2} = 3$, $n_{w_i=statistics|z=3} = 1$이고 $n_{w_j=DL|z=1} = 2$, $n_{w_j=DL|z=1} = 2$, $n_{w_j=DL|z=1} = 1$이다.

4. Comparing Results

BTM의 CGS를 파이썬으로 구현한 뒤에, 이를 이전에 구현한 CGS를 통한 LDA, 그리고 Python의 Scikit learn에 구현된 Online VI을 통한 LDA의 결과와 비교해보았다. 이전에, 주목해야할 점은 LDA는 topic 별로 Top 단어가 나오지만, BTM은 topic 별로 Top Biterm이 나온다는 것이다. 그 이유는, LDA는 각 document의 word 확률을 모델링 하였고 BTM은 corpus의 Biterm 확률을 모델링하였기 때문이다.
사용한 데이터는 18년 10월~12월의 첼시 관련 네이버 뉴스 기사 제목이다. 따라서 document보다는, short text라고 보는 것이 더 타당하다. 이 시기에, 첼시팀과 관련해서는 크게 다음과 같은 이슈가 있었다. 1) 팀 내, 핵심 선수인 아자르의 레알 마드리드로의 이적설 2) 팀 내, 확실한 스트라이커의 부재 및 타 팀의 공격수와의 링크 3) 첼시의 유로파 리그 우승 4) 리그컵 준우승에서 토트넘을 만나서 손흥민의 활약에도 토트넘을 누르고 결승에 오름 (4번 이슈는 한국 한정, 첼시 관련 이슈이다.) 아래는 각각 scikit learn의 LDA, CGS의 LDA, BTM의 Topic Modeling 결과이다.

Topic1	Topic2	Topic3
첼시	첼시	첼시
사리	아자르	epl
감독	유로파리그	무승부
아자르	아스날	리버풀
유로파	결승	맨유
우승	레알	챔스
이적	아스널	프랑크푸르트
아스널	우승	아자르
uel	이적	토트넘
유벤투스	꺾고	ucl
램파드	uefa	번리
유럽	도움	진출
결승	vs	살라
레알	uel	정상

Topic1	Topic2	Topic3
맨유	영	토트넘
손흥민	입	손흥민
골	레알	첼시
위	포체	결승
컵	티노	케인
리버풀	첼시	리그컵
첼시	감독	부상
솔샤르	이과인	알리
경기	모라타	컵
토트넘	마드리드	언론
우승	임대	바오
도움	이적	좌절
선수	아자르	카라
살라	최고	승부차기

표 6, 7: scikit learn의 LDA 결과(좌), CGS의 LDA 결과(우)

Topic1	Topic2	Topic3
(아자르, 레알)	(아자르, 첼시)	(첼시, 에)
(아자르, 이적)	(첼시, 아자르)	(첼시, 4)
(사리, 감독)	(첼시, 유로파리그)	(첼시, 무승부)
(레알, 이적)	(첼시, 아스날)	(첼시, 위)
(이적, 료)	(첼시, 우승)	(맨유, 첼시)
(아자르, 에)	(첼시, 결승)	(첼시, 행)
(레알, 아자르)	(첼시, 꺾고)	(리버풀, 첼시)
(아자르, 마드리드)	(유로파리그, 우승)	(첼시, 사리)
(이적, 첼시)	(유로파리그, 첼시)	(첼시, 진출)
(유로파, 우승)	(EPL, 첼시)	(첼시, 은)

표 8: BTM 결과

Reference: Biterm Topic Modeling For Short Texts, Xiaohui Yan by Jiafeng Guo, Yanyan Lan, Xueqi Cheng