Sentence Latent Dirichlet Allocation を用いたエンティティリンキング

山本 和慶　鶴岡 慶雅
東京大学　工学部電子情報工学科
{k-yamamoto,tsuruoka}@logos.t.u-tokyo.ac.jp

1　はじめに

エンティティリンキングは、図 1 のように、テキストに出現するキーワードを Wikipedia などの知識データベースに紐付けるタスクである。機械にとっては文章はそのままではただの文字列でしかないが、エンティティリンキングを行うことで Entity (実体情報) を付与することが出来る、言葉の曖昧性の問題を解消することが出来るなどのメリットがある。

エンティティリンキングを行う手法の一つとして、Neil らの LDA を用いる方法がある [1]。LDA は、文書中の語は潜在的なトピックから生成されると考えるモデルである。各キーワードに対して潜在トピックを推定して割り当てる。単語に割り当てられたこの潜在トピック情報を用いることで単語と知識データベースの紐付けを行う。

また、別の手法として、Tristram らの Support Vector Machine (SVM) を用いる方法がある [2]。これは、いくつかの特徴量を元に、SVM を用いて単語と知識データベースの紐付けを行うものである。

本研究では、Sentence-LDA (SLDA) を用いてエンティティリンキングを行うことを提案する。平均化パーセプトロンを用いた単純な手法をベースライン手法として、SLDA によるトピック情報を特徴量として追加したエンティティリンキングを実装し、他の手法と精度を比較する実験を行った。現時点では、SLDA によるトピック情報はエンティティリンキングには有効に働かず、むしろマイナスに働くという結果が得られた。ただし、実験の設定や SLDA の利用法にはまだ改良の余地が残っており、その有効性についてはさらなる検討が必要である。

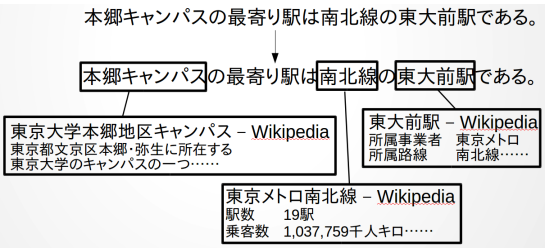

図 1. エンティティリンキングの概要

2　関連研究

2.1　Latent Dirichlet Allocation (LDA)

LDA は、Blei ら [3] によって提案された、文書が潜在的なトピックに基づいて、確率的に生成されると仮定するモデルである。文書集合には経済、政治、スポーツなどの様々なトピックが存在し、そのトピック毎に単語の出現頻度が異なると仮定することが出来る。ただし、文書集合にはトピックの情報が明示的に与えられているわけではないため、トピックは観測出来ない潜在トピックとして扱う。統計的に共起しやすい単語の集合を、潜在トピックという確率変数によって定式化している。LDA では、一つの文書には複数のトピックが潜在していると仮定する。そして、そのトピックの分布を離散分布によって表す。

文書数を M, 文書 d の単語数を n_d とする。文書 d でトピック k が出現する確率 $\theta_{d,k}$ を用いて、全トピック数を K として、トピック分布を $\boldsymbol{\theta}_d = (\theta_{d,1},\cdots,\theta_{d,K})$ とする。同様に、トピック k における単語 v の出現確率 $\phi_{k,v}$ を用いて、トピック k に対する単語の出現分布を $\boldsymbol{\phi}_k = (\phi_{k,1},\cdots,\phi_{k,V})$ とする。$\boldsymbol{\theta}_d$ と $\boldsymbol{\phi}_k$ は確率ベクトルであるので、Dirichlet 分布 (以下 $Dir()$ と表記する) による生成を仮定する。Dirichlet 分布は多項分布の共役事前分布である。

$$\theta_d \sim Dir(\alpha)\ (d = 1,\cdots,M) \quad (1)$$
$$\phi_k \sim Dir(\beta)\ (k = 1,\cdots,K) \quad (2)$$

ただし、$\boldsymbol{\alpha} = (\alpha_1,\cdots,\alpha_K)$ は K 次元ベクトル、$\boldsymbol{\beta} = (\beta_1,\cdots,\beta_V)$ は V 次元のベクトルであり、どちらも Dirichlet 分布のパラメータである。

文書 d の i 番目の単語 $w_{d,i}$ とそれに対応する潜在トピック $z_{d,i}$ は離散値であるので、多項分布 (以下 $Multi()$ と表記する) による生成を仮定する。各文書 $d(= 1,\cdots,M)$ に対して

$$z_{d,i} \sim Multi(\boldsymbol{\theta_d}), w_{d,i} \sim Multi(\phi_{z_{d,i}})\ (i = 1,\cdots,n_d) \quad (3)$$

これをグラフィカルモデルを用いて表すと、図 2 のようになる。

実際の LDA を用いてテキストの分析を行う際には、テキストの各単語に対してその単語を生成したトピックを推定して割り当て、このトピック情報を用いるということが行われる。このトピック推定にはいくつかの方法があるが、

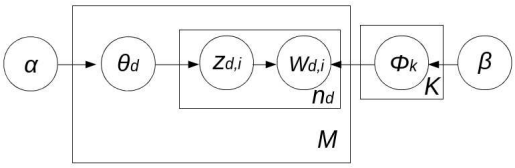

図 2. LDA の概要

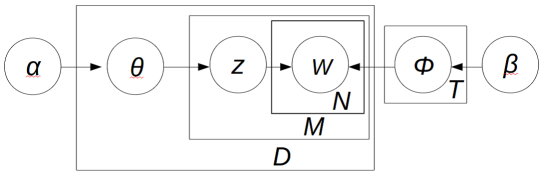

図 3. SLDA の概要

周辺化ギブスサンプリングを用いる方法が比較的実装が容易で有力な手法である。

2.2 LDA を用いたエンティティリンキング

Neil ら [1] は、LDA を用いてエンティティリンキングを行う手法を提案した。通常の LDA では、各トピックは関連語の集まりでしかないが、ここでは各トピックが Wikipedia の一記事に対応するモデルを考える。つまり、ある単語に一つのトピックを割り当てることはそのまま一つの Wikipedia の記事をその単語に紐付けすることに相当する。通常、LDA で扱うトピックの数は数百程度であるが、このモデルにおいては Wikipedia の記事の数だけトピックが存在するため、トピックの数が 4,000,000 程度と非常に多くなる。

2.3 Sentence-LDA

SLDA は、Yohan ら [4] の、LDA を拡張して作ったモデルである。通常の LDA ではトピックの割り当てを単語単位で行うのに対して、SLDA ではトピックの割り当てを文単位で行う。通常の LDA においては文書中の単語は Bag-of-words として扱われ、文書中で出現した単語とその数が考慮されるのみで語順は考慮されない。ギブスサンプリングを行い、ある単語にトピックの割り当てを行う際にはその文書中の他の単語にどのトピックが割り当てられているかということは考慮されるが、すぐ隣にある単語のトピックを、文書中の遠い場所にある単語のトピックより重視するなどといったことは出来ない。実際には、ある単語に割り当てられるトピックへの影響は、その単語の近くにある単語ほど大きいと考えられる。SLDA では文単位でトピックの割り当てを行うため、少なくとも同じ文中にある単語についてはそれ以外とは区別して扱うことが出来る。一方で、文単位でトピックを割り当てるため、一つの文の途中でトピックが急に変化している場合などは適切なトピック割り当てが行われない可能性がある。

SLDA では、ギブスサンプリングの際に文単位でトピックの割り当てを行うだけであり、トピックに対する単語の出現確率分布は単語毎に考える。ギブスサンプリングを行う際、i 番目の文に対して各トピックが割り当てられる確率は以下の式で表される [4]。

$$p(z_i = k|\boldsymbol{z_{-i}}, \boldsymbol{w}) \propto \frac{C_{dk}^{DT} + \alpha_k}{\sum_{k'=1}^{T} C_{dk'}^{DT} + \alpha_{k'}} \frac{\Gamma(\sum_{w=1}^{W} C_{kw}^{TW} + \beta_w)}{\Gamma(\sum_{w=1}^{W}(C_{kw}^{TW} + \beta_w) + m_i)} \prod \frac{\Gamma(C_{kw}^{TW} + \beta_w + m_{iw})}{\Gamma(C_{kw}^{TW} + \beta_w)} \tag{4}$$

各トピックに対してこの式の計算を行い、合計が 1 になるように正規化すれば各トピックが割り当てられる確率が計算可能である。各文書 d に対してトピック k の出現確率は以下の式で近似される。

$$\theta_{dk} = \frac{C_{dk}^{DT} + \alpha_k}{\sum_{k'=1}^{T} C_{dk'}^{DT} + \alpha_{k'}} \tag{5}$$

各トピック k から単語 w が生成される確率は以下の式で近似される。

$$\phi_{kw} = \frac{C_{kw}^{TW} + \beta_w}{\sum_{w'=1}^{V} C_{kw'}^{TW} + b_{w'}} \tag{6}$$

表 1. Sentence-LDA の式、図中の各記号について

D ： 文書数
M ： 文の数
N ： 単語数
T ： トピック数
V ： 語彙数 (出現する単語の種類数)
w ： 単語
z ： トピック
ϕ ： 単語の (各トピックに対する出現率の) 多項分布
θ ： トピックの (各文書に対する出現率の) 多項分布
α_k ： θ を生成する Dirichlet 分布のパラメータ
β_w ： ϕ を生成する Dirichlet 分布のパラメータ
z_i ： 文 i のトピック
$\boldsymbol{z_{-i}}$ ： 文 i を除く全ての文に割り当てられたトピック
$\boldsymbol{w}$ ： そのコーパスに出現する全種類の単語
C_{dk}^{DT} ： 文書 d 中でトピック k が割り当てられている文の数
C_{kw}^{TW} ： 単語 w のうち トピック k が割り当てられているものの数
m_i ： 文 i の単語数
m_{iw} ： 文 i 中の単語 w の数

2.4 SVM

サポートベクターマシンは、教師あり学習を用いてパターン識別器を構成する機械学習の手法である。サポートベクターマシンの他のパターン識別手法と比べた最大の特徴は、マージン最大化という処理を行っていることである。マージン最大化とは、2クラスを分類する境界に最も近いデータと境界との距離が最大となるように境界を設定する処理である。マージン最大化を行うことにより、未知のデータを正しく分類出来る可能性が上がる。また、カーネル関数を用いて高次元の空間へと写像することで、線形分離不可能な場合についても分類が可能となる。

2.5 SVM を用いたエンティティリンキング

Tristram らは、SVM を用いたエンティティリンキングを行った [2]。その手順は、まず Wikipedia の全ての記事をチェックし、アンカーテキスト (ハイパーリンクが貼られているテキスト) とリンク先の記事名を調べる。「アンカーテキスト ー リンク先の記事」の対応関係を全て列挙する。エンティティリンキングを行う単語のエンティティ候補として、その単語と同じアンカーテキストからハイパーリンクが貼られている記事を用いる。そして、それらの候補に対して SVM を使ってスコア付けを行い、最もスコアが高くなった候補に対してリンキングを行う。というものである。SVM においては、以下の 4 つの情報を特徴量として用いる。

(1) tf-idf vector similarity：その単語 (エンティティリンキングを行う単語) が出現した文書と、リンク先候補の記事に出現する単語がどの程度類似しているかという情報 (類似度が高いほどスコアが高い)
(2) Semantic Neighborhood Signature Vector Similarity：リンク先候補の記事から一定回数以下のハイパーリンクで結ばれている記事が、エンティティリンキングを行う単語の出現する文書の他のエンティティリンキング対象語のリンキング先候補としてどの程度出現しているかという情報
(3) Candidate Reference Probability：各リンキング先候補が、その単語のアンカーテキストに対するハイパーリンク先としてどれだけ現れるかという情報 (多く現れるほどスコアが高い)
(4) Page Rank：各リンキング先候補の記事のページランク (ページランクが高いほどスコアが高い)

3 提案手法

3.1 SLDA を用いたエンティティリンキング

SLDA をエンティティリンキングに用いることを考える。Neil ら [1] の、LDA を用いてエンティティリンキングを行う方法のように、一つのトピックに一つのが対応しているとする方法では、文単位でのトピックの割り当てしか出来ない SLDA では、2 つ以上のエンティティリンキングの対象となる語が同一文中にある場合や、文中の主な単語とエンティティリンキング対象となる語にあまり関連がない場合などに適切に機能しないことが想定される。そこで、Tristram ら [2] の方法と同様に、SVM もしくは平均化パーセプトロンを用いてエンティティリンキングを行う。そして、その特徴量として SLDA によるトピック情報を用いるという方法を考える。

まず、ベースライン手法として、平均化パーセプトロンを用いてエンティティリンキングを行う単純な方法を実装した。特徴量として、「 (エンティティリンキングを行う) その単語」、「前後 5 単語」のみを用いる。1 つのクラスが、1 つのリンキング先候補の記事に対応しているものとする。次に、SLDA を用いた方法として、ベースライン手法に特徴量として「SLDA によってその単語に割り当てられたトピック情報」を加えたものを実装した。

4 実験

4.1 実験設定

本実験は実際に SLDA を用いたエンティティリンキングを実装し、その精度を評価することを目標とした。CoNLL-Aida データセット [5] を用いて評価を行った。データセットを 3 分割し、946 個の文章を機械学習に、216 個を test-a (development) に、231 個を test-b (blind evaluation) に用いた。ベースライン手法と提案手法および Tristram らの手法 (Weasel)[2] の精度を Precision (「全ての割り当てられたエンティティ」のうち「正しく割り当てられたエンティティ」の割合) を指標として比較した。

SLDA については、まず学習用データの全てに対してサンプリングを行い、トピックを割り当てる。各トピックの出現確率を特徴ベクトルとしてそのまま学習に用いる。開発データおよびテストデータのトピックのサンプリングは学習が終わった後で行う。ベースライン手法、提案手法共に、平均化パーセプトロンを用いて学習データを元に学習を行い、学習後の識別器によって開発データおよびテストデータに対して分類を行い、その精度を評価した。ベースライン手法においては「その単語」、「前後 5 単語」を、提案手法では「その単語」、「前後 5 単語」、「その単語に割り当てられたトピック」を特徴量として用いる。SLDA に用いる

表 2. 実装したプログラムのパラメータ設定

トピック数 K=20
サンプリング回数 S=100
ハイパーパラメータ α=1/K
ハイパーパラメータ
β=1/”全データ中に出現した単語の種類”

表 3. CoNLL-AIDA データセットを用いた実験結果

	Precision
Weasel	0.60
ベースライン	0.29
提案手法	0.17

パラメータは表 2 のように設定した。

4.2 実験結果

実験結果は表 3 のようになった。ベースライン手法は単純な特徴量しか用いていないためか、Tristram ら [2] の手法と比べて低い精度となった。また、ベースライン手法に SLDA のトピック情報を追加した提案手法については精度がベースライン手法と比べて低くなる結果となった。SLDA によって割り当てられたトピック情報は、平均化パーセプトロンに用いる特徴量として有効に働かないという結果が得られた。特徴量として、正しいエンティティを見分ける手がかりになるよりはノイズとなることの方が多いためと考えられる。ただし、実験の設定や SLDA の利用法にはまだ改良の余地が残っている。今後は平均化パーセプトロンの特徴量として用いる以外の利用法が有効であるかどうかを検討することを考えている。

謝辞

本研究は、JST、CREST の支援を受けたものである。

参考文献

[1] Neil Houlsby and Massimiliano Ciaramita. A scalable gibbs sampler for probabilistic entity linking. In *Advances in Information Retrieval*, pages 335–346. Springer, 2014.

[2] Felix Tristram, Sebastian Walter, Philipp Cimiano, and Christina Unger. Weasel: a machine learning based approach to entity linking combining different features. In *Proceedings of 3th International Workshop on NLP and DBpedia, co-located with the 14th ISWC 2015, USA*, 2015.

[3] David M Blei, Andrew Y Ng, and Michael I Jordan. Latent dirichlet allocation. *the Journal of machine Learning research*, 3:993–1022, 2003.

[4] Yohan Jo and Alice H Oh. Aspect and sentiment unification model for online review analysis. In *Proceedings of the fourth ACM international conference on Web search and data mining*, pages 815–824. ACM, 2011.

[5] Johannes Hoffart, Mohamed Amir Yosef, Ilaria Bordino, Hagen Fürstenau, Manfred Pinkal, Marc Spaniol, Bilyana Taneva, Stefan Thater, and Gerhard Weikum. Robust disambiguation of named entities in text. In *Proceedings of the Conference on EMNLP*, pages 782–792.