パラグラフベクトルを用いた教師なし語義曖昧性解消の考察

堺澤 勇也 * 小町 守

首都大学東京

1 はじめに

自然言語処理の主な難しさは，人間の言語の中で生じる多くの曖昧性を解決することにある．例えば，語義曖昧性解消（Word Sense Disambiguation: 以下 WSD ）がそれに挙げられる．

この WSD の古典的なアルゴリズムとして Lesk アルゴリズムがある [5]．これを応用したもので Simplified Lesk アルゴリズムがよく知られている [3]．しかし，このアルゴリズムは WSD する対象後の語釈文に含まれている単語が与えられた文脈に存在しない場合，分類に失敗する．このスパース性を解消するために多くの先行研究が存在している．

最近では，単語の分散表現を用いて，WSD を行う研究も行われている．単語の分散表現とは，単語を固定長のベクトルで表現することを指す．単語のベクトルは，類似する単語が類似したベクトルを持つように学習される．生成されたベクトルを単語ベクトルと呼ぶ．

また，単語のみではなく文，段落自体も分散表現する研究も進んでいる．これらも単語ベクトルと同様に，類似する文・段落が類似したベクトルを持つように学習される．生成されたベクトルを，それぞれセンテンスベクトル，パラグラフベクトルと呼ぶ．

本稿では，パラグラフベクトルを用いて対象語が出現する文脈と知識ベースに記述された語釈文を分散表現することにより，Simplified Lesk アルゴリズムのスパース性を解消する方法を提案する．Simplified Lesk アルゴリズムは，語釈文と与えられた文脈間の単語の重複によって語義を判定する．この方法では文を表すベクトルの次元は語彙の数と等しくなり，それぞれの文の表現は対応した単語の次元でのみ値を持つことになるので疎なベクトルで表現される．それに対して，パラグラフベクトルは密ベクトルであり，パラグラフを表すのにすべての次元が表現に関連している点で異なる．この表現方法により，語彙数次元あったスパースな表現から固定次元に次元を圧縮することによりスパース性を解消する．

2 関連研究

Simplified Lesk はスパース性の問題を抱えている．このスパース性の解消の為に，多くの研究で語釈文の拡張がなされてきた．

例えば，Wilks et al. (1990) は WSD を行う単語の語釈文を拡張するために，共起行列を使用して決定される類似単語の語釈文を利用した [8]．また，Banerjee and Pedersen(2002) は，WordNet の中で関連しているシンセット内の語義と文脈内の内容語両方の語釈文を使用して WSD の対象となっている単語の語釈文を拡張した [1]．

多くの研究がオーバーラップを増加させるために語釈文を拡張する手法について焦点を当てている．これらは本稿の目的と同様に，スパース性を解消するためであるが，本稿では語釈文の拡張はせずにスパース性の解消を試みる．

最近では，分散表現で得られた単語の類似性を利用した WSD も研究されている．Chen et al. (2014) は，単語の分散表現によって得られた単語ベクトルを利用して意味ベクトルを生成した [2]．WordNet に入っている語義毎に，その語釈文内の内容語で類似度の高い単語ベクトルの平均を意味ベクトルと定義した．Neelakantan et al. (2014) は，1 つの単語に対して複数の意味ベクトルがあるものと仮定した．意味ベクトルは，周囲の単語ベクトルの平均と一番近い単語ベクトルが選択されるようにする [7]．

これらの研究では，単語の分散表現を用いて WSD を行っている．しかし，文や段落をベクトル化したパラグラフベクトルを使用しているものはない．今回の実験では，WSD を行う分散表現としてパラグラフベクトルを利用した．

*sakaizawa-yuya@ed.tmu.ac.jp

3 Simplified Lesk アルゴリズム

Lesk アルゴリズムの応用として，Simplified Lesk アルゴリズムが知られている [3]．それは多義語の各語義の語釈文と与えられた文脈で重複する語を数え，最も重複する語が多い語釈文の語義を正解として出力するというものである．

例として，“bank” という語について考える．“bank” の語釈文として以下の文が与えられているとする．

語義 1. a financial institution that accepts deposits and channels the money into lending activities

語義 2. sloping land (especially the slope beside a body of water)

この時，次の文が入力として与えられた場合，以下のように処理する．

入力. I went to the bank to deposit the money.

このアルゴリズムでは，語釈文と与えられた文脈（この例では文）を比較し，単語の重複によって語義を特定する．この例では，入力文は “deposit”，“money” が重複しているので，入力文の中にある “bank” は語義 2 であることがわかる．

4 パラグラフベクトルの生成

本稿では，パラグラフベクトルを利用して Simplified Lesk アルゴリズムを拡張する [4]．パラグラフベクトルの利点は，教師なしでパラグラフの意味を含んだベクトルが生成可能なところにある．これにより，与えられたパラグラフを分散表現することによって，Simplified Lesk アルゴリズムで問題になっていたスパース性の問題を解消する．生成されたパラグラフベクトルは，固定長で表現されており，類似する文・段落が類似したベクトルを持つように学習される．

本稿では，WSD を行う対象語が出現する文脈と語釈文をベクトル化し，それらの類似性を基に意味を推定する．最初のセクションでは，パラグラフベクトルを作るのに必要な単語ベクトルの学習について説明していく．次のセクションで，パラグラフベクトルの学習について説明する．

4.1 単語ベクトルの学習

単語ベクトルの学習では，図 1 で示すようなフレームワークを使用する．単語ベクトルの生成は，図 1 の四角で囲まれた部分が表している．このフレームワークがするタスクは周辺の単語が与えられたときに中心の単語を予測するものである．

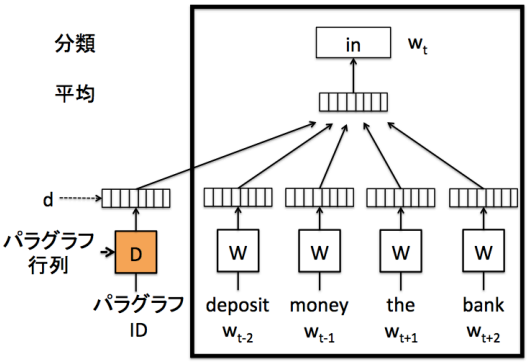

図 1: 四角内：単語ベクトル学習のためのフレームワーク．　全体：パラグラフベクトル学習のためのフレームワーク

図 1 では，各単語は単語行列 W 中の列ベクトルによって表現されている．この列は語彙中の単語の位置によってインデックスが貼られている．周辺の単語ベクトルの平均が，中心の単語ベクトルを予測するための素性として使用されている．

より形式的に書くと，単語系列 $w_1, w_2, w_3, \ldots, w_T$ が与えられたとき，単語ベクトルモデルは以下の平均対数尤度を最大化するように学習される．

$$\frac{1}{T}\sum_{t=k}^{T-k}\log p(w_t|w_{t-k},...,w_{t+k})$$

予測タスクは，通常 softmax のようなマルチクラス分類を通して行われるので，以下の式で行われる．

$$p(w_t|w_{t-k},...,w_{t+k}) = \frac{\exp\left(y_{w_t}\right)}{\sum_i \exp\left(y_i\right)}$$

各 y_i は各出力単語 w_i に対する正規化されていない対数確率なので以下のように計算できる．

$$y = b + Uh(w_{i-k},...,w_{i+k};W)$$

U, b は softmax のパラメータである．h は W から抽出された単語ベクトルの平均によって構築される．

4.2 パラグラフベクトルの学習

ここでは，単語ベクトルと同様に，パラグラフから多くの文脈のサンプルが与えられたとき，周辺の単語から中心の単語を予測する問題に有用になるようにパラグラフベクトルが生成される．パラグラフベクトルの学習のフレームワークは図 1 全体で示される．

形式的に表すと，以下のような式で与えられる．

$$y = b + Uh(d, w_{i-k}, ..., w_{i+k}; W, D)$$

このフレームワークは単語ベクトルのフレームワークと一点しか違いがない．最後の式の h が W だけでなく，単語行列 W とパラグラフ行列 D から構築されるようになるだけである．単語ベクトルと同様に各パラグラフベクトルは行列 D 中の列ベクトル d によって表現されている．

学習において，パラグラフが異なれば違うパラグラフベクトルになる．しかし，すべてのパラグラフで単語ベクトルは共有される．つまり，“powerful” のベクトルは，すべてのパラグラフで同じである．またパラグラフベクトルは，単語ベクトルと同様に，誤差逆伝搬によって学習される．

予測時では，新しく与えられたパラグラフに対してパラグラフベクトルを計算する必要がある．この時には，他のすべてのパラメータを固定することによって，新しいパラグラフベクトルを計算する．

5 実験

本稿では，WSD の実験を行い，パラグラフベクトルによるスパース性の解消を評価する．Simplifed Lesk アルゴリズムをベースラインとし，それと合わせて MFS も加えて評価を行う．

5.1 データ

単語ベクトルの学習では，2014 年 12 月の Wikipedia のデータ [1] を使用した．

評価データは， Senseval-3 English lexical sample task [2] で提供された訓練データのうち品詞が名詞のものを用いた．このデータセットは，WSD のターゲットとする多義語が，文脈（数文からなるパラグラフ）内に出現する事例の集まりであり，各々の事例には，WordNet の語義で正解語義が割り当てられている．

前処理として，Wikipedia，WordNet 3.0 に記述されている語釈文，対象語が出現する文脈に対してすべての文字を小文字化した．その後，NLTK 2.0.4 で記述されているストップワードを取り除いたあとで，各単語に Porter Stemmer でステミングを行い，単語ベクトル・パラグラフベクトルの学習をした．

[1] http://dumps.wikimedia.org/enwiki/20141208
[2] http://www.senseval.org

5.2 設定

本稿では，sentence2vec[3] を使用して，与えられた文脈と語釈文のパラグラフベクトルを生成した．sentence2vec では，単語ベクトル・パラグラフベクトルとも任意の次元で表現できる．今回の実験では，各ベクトルとも 20, 50, 100 次元の設定で実験を行った．また，与えられた文脈全体と語釈文を比較するものと与えられた文脈のうち対象の単語を含む文のみ抽出し語釈文と比較したもの両方を示す．本稿では，それぞれパラグラフ Lesk ，センテンス Lesk と呼ぶ．

定義文と与えられた文脈の類似度は，Simplifed Lesk アルゴリズムでは単語のオーバーラップ，提案手法では各ベクトルの類似度を cos 類似度によって計算した．また，パラグラフベクトルと比較するため，文脈中・文中に出現するすべての単語ベクトルの平均を一つのベクトルと定義し，類似度の測定をした．実験では，類似度に対する閾値を 0 から 1 まで値 0.1 ずつで刻み，一番高い類似度の値が閾値を超えたときその意味を採用した．閾値を超えない場合，単語の意味には MFS が割り当てられる．また，今回の実験では一つの語義のみ出力する形式になっているため，精度は正解率で表される．

5.3 実験結果

各単語に対して WSD を行い，正解率を測定した．その実験結果を表 1 に示す．この結果は，そのすべての単語が出した正解率の平均を示している．表 1 からわかるように，閾値を上げていくとすべての単語に MFS を割り当てたときの正解率の値に収束する．

今回，与えられた文脈や語釈文を拡張していないため単語の重複が少なく，Lesk アルゴリズムの結果は低い閾値で MFS に到達している．

5.4 考察

表 1 から適切な閾値を求めることで一部，精度が MFS を上回ることがわかる．

パラグラフベクトルを使用した結果，以下のような文脈（文）の意味を捉えることができている．

I will inform the bank in writing if I wish to cancel this instruction.

これは，ベースラインは建物の語義の “bank” と分類するが，パラグラフベクトルは文脈の類似性を捉え，銀行の “bank” の語釈文との類似度が最も高くなる．

[3] https://github.com/klb3713/sentence2vec

表 1: Senseval-3 Lexical Sample の名詞に対する WSD の正解率．W： 単語の分散表現のみ，S：センテンス Lesk，P： パラグラフ Lesk，- ：上の値と同じもの

閾値\手法	Lesk	Lesk-cos	W-20	W-50	W-100	S-20	S-50	S-100	P-20	P-50	P-100
0.0	39.26	39.26	31.63	35.16	36.93	32.98	35.90	38.88	33.81	35.72	38.71
0.1	55.36	53.13	31.63	35.16	36.93	33.42	36.22	39.66	34.38	36.17	39.79
0.2	-	55.19	31.63	35.16	36.93	34.56	37.86	42.30	34.87	37.61	43.69
0.3	-	55.32	31.61	35.16	36.93	36.43	40.82	48.16	37.01	41.82	50.69
0.4	-	55.36	31.59	35.25	37.34	39.00	46.64	54.14	41.36	48.10	54.32
0.5	-	-	31.91	36.61	39.58	44.99	52.52	55.11	47.17	53.78	**55.42**
0.6	-	-	32.64	39.48	44.16	49.28	55.15	**55.39**	52.19	**55.39**	55.33
0.7	-	-	36.31	45.64	51.10	53.79	55.36	55.36	54.99	55.30	55.36
0.8	-	-	42.94	52.92	55.31	55.16	-	-	55.27	55.36	-
0.9	-	-	53.77	**55.39**	55.26	55.36	-	-	55.36	-	-
1.0	**55.36**	**55.36**	**55.36**	55.36	**55.36**	**55.36**	**55.36**	55.36	**55.36**	55.36	55.36

また，分類に失敗している事例として以下のような文が挙げられる．これは，単語 “interest” に入っている一事例をとってきている．

Walker, which bought out GKN’s steel stockholding interests two years ago, has a market share of 20 percent.

Simplified Lesk アルゴリズムでは，法律上での権利という正解語義を導くことができている．しかし，パラグラフベクトルでは，利息としてこの “interest” を解釈している．これは，文中に出現する “20” や “percent” など利息と同じような文脈で使用されるような単語の出現に起因するものだと考えられる．

今回の実験ではパラグラフベクトルを生成するときに，語釈文の拡張をしていない．そのため，前処理をすると一文が 3 単語になってしまっているものもある．それにより，パラグラフベクトルの学習で十分にウィンドウ幅をとることができず，文脈の類似性がベクトルに反映されていない可能性がある．

また，今回の実験では多義語が単一のベクトルで表現されてるので，銀行の意味の “bank” と土手の意味の “bank” は同じベクトルで表されている．これらに同じベクトルを割り当てるのは不自然である．[7] のように，語義毎に意味ベクトルを生成してからパラグラフベクトルを学習することで，より WSD に適したパラグラフベクトルの生成が可能であると考えられる．

6 おわりに

本稿では，パラグラフベクトルを用いて対象語が出現する文脈と知識ベースに記述された語釈文をベクトル化することにより，Simplified Lesk アルゴリズムのスパース性を解消する方法を考察した．

今後の課題としては，今回の実験は名詞にのみにこの手法を適応したので，動詞・形容詞などにの品詞にこれを適応することが挙げられる．また，今回生成したパラグラフベクトルを直接評価するのではなく，教師あり学習の素性として使用することも考えられる．

参考文献

[1] Satanjeev Banerjee and Ted Pedersen. An Adapted Lesk Algorithm for Word Sense Disambiguation Using WordNet. In *CICLing*, pp. 136–145, 2002.

[2] Xinxiong Chen, Zhiyuan Liu, and Maosong Sun. A Unified Model for Word Sense Representation and Disambiguation. In *EMNLP*, pp. 1025–1035, 2014.

[3] Adam Kilgarriff and Joseph Rosenzweig. Framework and Results for English SENSEVAL. *Special Issue on SENSEVAL. Computers and the Humanties*, pp. 15–48, 2000.

[4] Quoc V. Le and Tomas Mikolov. Distributed Representations of Sentences and Documents. In *ICML*, pp. 1188–1196, 2014.

[5] Michael Lesk. Automatic Sense Disambiguation Using Machine Readable Dictionaries: How to Tell a Pine Cone from an Ice Cream Cone. In *SIGDOC*, pp. 24–26, 1986.

[6] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient Estimation of Word Representations in Vector Space. In *ICLR*, pp. 1–12, 2013.

[7] Arvind Neelakantan, Jeevan Shankar, Alexandre Passos, and Andrew McCallum. Efficient Non-parametric Estimation of Multiple Embeddings per Word in Vector Space. In *EMNLP*, pp. 1059–1069, 2014.

[8] Yorick Wilks, Dan Fass, Cheng Ming Guo, James E. McDonald, Tony Plate, and Brian M. Slator. Providing machine tractable dictionary tools. *Machine Translation*, Vol. 5, No. 2, pp. 99–154, 1990.