Paragraph Vectorのための効率的なパラメータサーチの検討

原　裕貴† 新妻　弘崇†† 太田　学 †††

† 岡山大学工学部情報系学科　〒 700–8530 岡山県岡山市北区津島中三丁目 1 番 1 号
†† ,††† 岡山大学大学院自然科学研究科　〒 700–8530 岡山県岡山市北区津島中三丁目 1 番 1 号
E-mail: †pxmn2dj7@s.okayama-u.ac.jp, ††niitsuma@cs.okayama-u.ac.jp, †††ohta@de.cs.okayama-u.ac.jp

あらまし　文章の特徴を表現する手法として Paragraph Vector が提案されている．Paragraph Vector を用いると文章の特徴を数百次元の特徴ベクトルで表現することができ, 文章分類を高精度で行うことができるとされている．しかし, 高精度な文章分類を行うには, Paragraph Vector の学習において多数のパラメータを適切な値に設定することが必要である．そこで, 我々は適切なパラメータ選択を効率的に行う手法として二分法とランダムサーチを組み合わせたパラメータサーチ手法を提案する．これを用いることで, パラメータ探索を低コストで行いつつ, 高精度な文章分類を実現できることを実験により示す．本稿では Brown-corpus, 及び Stanford Sentiment Treebank Dataset の英文の分類実験により, 提案手法の有効性を評価した．

キーワード　word2vec, Paragraph Vector, ニューラルネットワーク

1. はじめに

Mikolov らによって提案された word2vec [1] [2] は単語を数値ベクトルとして表現する手法として多くの応用で利用されている．word2vec には, 意味の近い単語から生成されたベクトルは類似したベクトルとなる特徴がある．word2vec は文章中のある単語を単語の前後関係から予測するという問題を, ニューラルネットワークに学習させ, そのニューラルネットワークの中間層の値を単語の特徴ベクトルとして出力する手法である．文章中の単語を予測する方法としては, 中心の単語から周辺の単語を予測するもの, 周辺の単語から中心の単語を予測するものがある．前者は Skip-gram モデル, 後者は Continuous Bag-of-Words (CBOW) モデルと呼ばれる [1] [2].

単語ではなく, 文章の特徴を表現する手法としては, TF-IDF や Bag-of-Words (BOW) のように単語の出現頻度を用いる方法が使われることが多い．しかし、これらの方法には語順を考慮できない, 同義語が似たベクトルに必ずしもならない, といった問題があった．word2vec はこれらの問題を解決するのに利用できる可能性はあるが, 文章の特徴ベクトルを直接生成はできない．そこで, word2vec と同じ原理で文章の特徴ベクトルを生成できるよう word2vec を拡張した Paragraph Vector [3] が提案された．映画のレビュー文章から, レビュアがつけた評価値を推定する問題 [4] において, Paragraph Vector は従来の BOW などを使った手法よりも高精度な推定を実現できたことが報告されている [3].

橋戸らは文章に付随する情報に着目し, Paragraph Vector を拡張した ScoreSent2Vec を提案した [5]．ScoreSent2Vec は, Paragraph Vector に付随情報を予測するニューラルネットワークを追加するという拡張によって, 付随情報の影響を受けた文章の特徴ベクトルを生成する手法である．橋戸らはテキスト分類の問題において, ScoreSent2Vec により, 従来の Paragraph Vector よりも高精度な分類が可能であることを示した [5]．また, 付随情報のエントロピーと分類精度の関係を示すことで, Paragraph Vector のパラメータが分類精度に与える影響について考察した [6]．しかし, この実験において調べられたパラメータの組合せは 100 以下程度であり，調査が不十分だった．

本稿では, 約 5,000 通りのパラメータの組合せを試すことで, 先行研究 [6] では十分に調べることができなった問題を調査する．また Paragraph Vector のパラメータをグリッドサーチなしで, 自動的に適切な値に効率的に設定する手法を提案する．具体的には, 二分法とランダムサーチを組み合わせた手法を提案する．

評価実験には Paragraph Vector の実装の一つである gensim ライブラリ [7] の doc2vec class を用いる．評価実験では, doc2vec の計算する Paragraph Vector を用いて Brown Corpus [8] 及び Stanford Sentiment Treebank Dataset [9] の英文の分類問題を扱う．先行研究よりも大規模なパラメータサーチによって, 橋戸らの文章の付随情報に着目する手法よりも高精度な分類が可能になることも示す．

2. 関連研究

ここでは, 本稿で利用する Paragraph Vector と, その基礎となっている word2vec について説明する．word2vec は単語を特徴ベクトルとして表し, Paragraph Vector は文章を特徴ベクトルとして表す手法である．

Mikolov らは word2vec で生成した単語ベクトル間の差は, 単語間の関係を表現していると主張した [1] [2]．以下の式を例に単語ベクトルについて説明する．

$$\text{king} - \text{man} + \text{woman} \tag{1}$$

式 (1) は king, man, woman の単語ベクトルを用いた演算であり, この式は man と king の関係を woman に適用したものを表す．Mikolov らは, word2vec で生成した単語ベクトルを用

いてこのような演算を行った場合, 答は queen を表すベクトルになると主張した.

word2vec は, コーパス中の単語の前後関係をニューラルネットワークで学習し，このニューラルネットワークの中間層の値を単語の特徴を表すベクトルとする．Mikolov らは word2vec に用いるコーパスの単語が多い場合, 単語のベクトルの次元数も大きくすることにより, 式 (1) のような演算により得たベクトルが quenn を表すベクトルになる確率が上がることを示した [2]．このように, より高精度な分類を行うためには単語ベクトルの次元数をコーパスの単語数に応じて調整する必要がある．これは後述する Paragraph Vector でも同様である．word2vec と Paragraph Vector によるベクトル生成では, ベクトルの次元数以外にも低頻度の単語を考慮しないための出現頻度の閾値, 重みと閾値の調整の度合を決める学習率など調整の必要なパラメータが存在する.

2.1 word2vec

word2vec を実現するニューラルネットワークの構造として Skip-gram モデルと CBOW モデルが提案されている [1] [2]. 図 1 及び図 2 は, window size のパラメータを c としたときのニューラルネットワークである．window size は同じ文脈として考慮する前後の単語数を示すパラメータである.

Skip-gram モデル: 図 1 に Skip-gram モデルのニューラルネットワークを示す． このモデルは入力層, 中間層, 出力層の 3 層からなり, 文章中のある単語 $w(t)$ を入力, その前後の単語 $w(t-c),...,w(t-1),w(t+1),...,w(t+c)$ を出力とするニューラルネットワークである.

CBOW モデル: 図 2 に CBOW モデルのニューラルネットワークを示す．このモデルは Skip-gram モデルと同様, 入力層, 中間層, 出力層の 3 層からなるが, 入出力が Skip-gram モデルの逆となっている．出力は中心の単語 $w(t)$ であり, 入力をその前後の単語 $w(t-c),...,w(t-1),w(t+1),...,w(t+c)$ とするニューラルネットワークである．すなわち, Skip-gram モデルとは反対に, 周辺の単語から中心にある単語を推定する問題をニューラルネットワークに学習させるモデルである.

2.2 Paragraph Vector

Paragraph Vector は word2vec の拡張で, 単語ではなく文章の特徴ベクトルを生成する手法である．word2vec の単語ごとのニューラルネットワークを, 文章ごとのニューラルネットワークに書き変えることで, 文章の特徴を計算するのが Paragraph Vector である．Paragraph Vector も word2vec と同様に二つのモデルがある．Skip-gram モデルを拡張した Paragraph Vector with Distributed Bag of Words （PV-DBOW）モデルと CBOW モデルを拡張した Paragraph Vector with Distributed Memory （PV-DM）モデルである．以下では, word2vec と同様に window size のパラメータを c としたときのニューラルネットワークである図 3 と図 4 を例に説明する.

PV-DBOW モデル: 図 3 に PV-DBOW モデルのニューラルネットワークを示す．このモデルは word2vec の Skip-gram モデルを拡張したものである．ただし Skip-gram モデルと異なり, 単語 $w(t)$ ではなく, 文章の ID をネットワークの入力とする．ここで, 文章の ID とは学習データの文章につけた番号である．この文章の ID に対するニューラルネットワークの重みを word2vec と同様の方法で学習することで, 文章の特徴ベクトルを得ることができる.

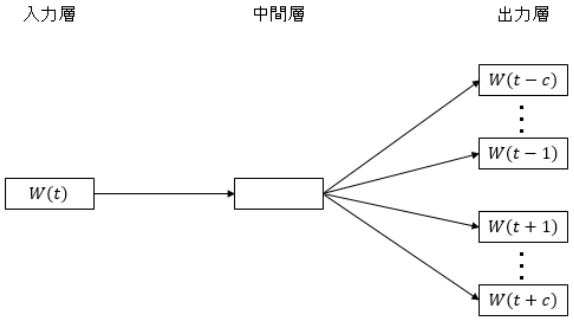

図 1 Skip-gram モデル

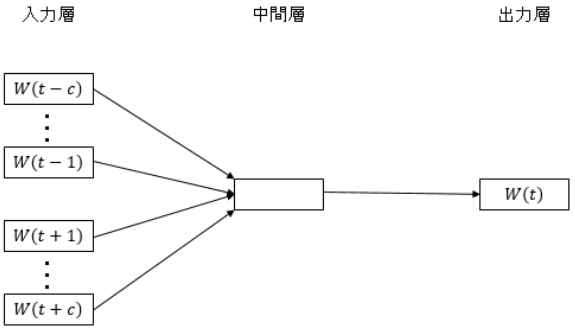

図 2 CBOW モデル

PV-DM モデル: 図 4 に PV-DM モデルのニューラルネットワークを示す．このモデルは, word2vec の CBOW モデルを拡張したモデルである．初めに word2vec の CBOW モデルを使って学習したニューラルネットワークを作成する．これは, 図 2 のようなニューラルネットワークである．次に, 文章の ID を入力するネットワークを入力層に追加する．この追加されたネットワーク部分のみを word2vec と同様の方法で目的の文章に対して学習することで, 文章の特徴ベクトルを得ることができる.

Paragrh Vector を用いた研究は多数報告されている．中野ら [10] は, 提題表現に基づく重要段落の抽出に Paragraph Vector を用いた．提題表現とは, 文の主題を取り上げる表現であり,「〜は」の形式を典型とする．中野らの提案手法においては, 文章中の語をベクトル化し, 段落ごとに各ベクトルの内積を計算することで重要段落を得る．毎日新聞コーパス (1998-99 年) 及び日経新聞コーパス (1998 年) のうち, ニュース報道記事に該当するものを実験データとしたとき, 毎日新聞記事で 61.2 %, 日経新聞記事で 77.9 %の精度で重要段落を抽出した.

佐藤ら [11] はウェブ上の文書が有害であるか否かという分類

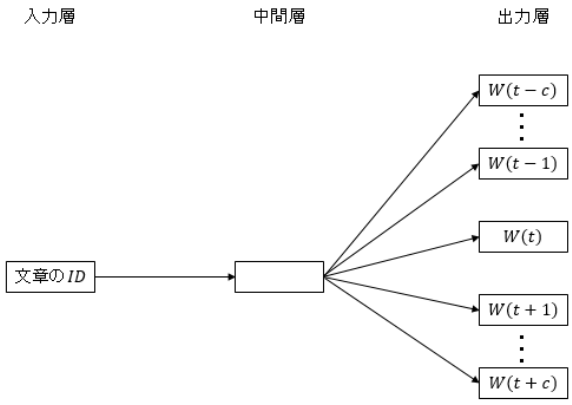

図 3　PV-DBOW モデル

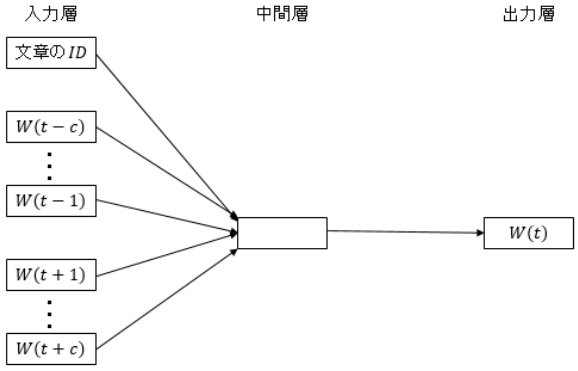

図 4　PV-DM モデル

に Paragraph Vector を用いた．実験では, 有害文書と無害文書をそれぞれ 10 万件ずつ用いて評価実験を行った．佐藤らは PV-DM モデルを拡張した PV-CBOW (Continuous Bag-of-Words of Paragraph Vectors) モデルを提案した．この提案したモデルが最も高精度であり, ベクトルの次元数が 200 の時に F-measure が 0.9431 であった．

3. 提案するパラメータ選択手法

本稿では doc2vec の以下の 4 種類のパラメータに注目する．

- alpha
  学習率を示す．
- iter
  反復学習の回数を示す．
- min_count
  出現回数がこの値より小さい単語を除去する．
- window
  前述の window size を示す c であり window とも呼ばれる．

doc2vec には, これ以外にも生成されるベクトルの次元を決めるパラメータ size, PV-DM モデルと PV-CBOW モデルを切り替えるパラメータ dm なども存在するが, 本研究では他のパラメータはデフォルトとする．doc2vec のパラメータで調節範囲が広く, 生成されるベクトルの性質に大きく影響するのは, この 4 つのパラメータであり, 本研究ではこの 4 つのパラメータに注目する．これらのパラメータ調節は各研究者が経験的に行っており, 新たな学習データに対してはパラメータの再調整が必要である．そこで, 本論文では二分法とランダムサーチを組み合わせたパラメータサーチを用いることで探索を効率化する手法を提案する．

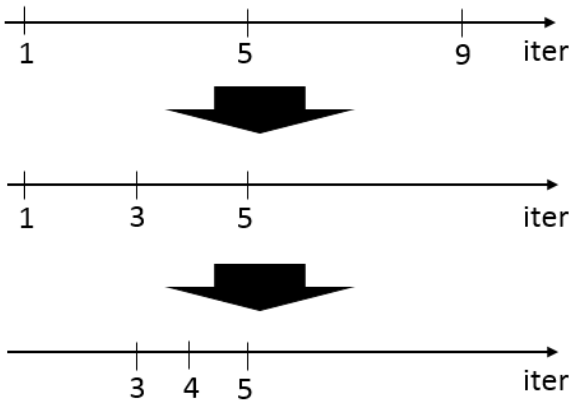

図 5　区間下限・上限及び中間点の選択例

提案手法は, まず二分法で優先的に探索したいパラメータを一つ選ぶ．ここでは提案手法を, 二分法で優先的に探索したいパラメータを iter とした場合を例に挙げて説明する．この場合は, alpha, min_count, window の 3 種のパラメータはランダムサーチで選択する．以下では, 図 5, 及びアルゴリズムの疑似コード Algorithm 1 を用いて説明する．まず, 探索範囲となる区間下限 (low = iter = 1) と区間上限 (high = iter = 9) を決め, それぞれ残りのパラメータとの組み合わせでランダムサーチを複数回行う．以下では, このランダムサーチの回数を 1 パラメータ値あたりのランダムサーチの回数と定義する．これを Algorith 1 では run と記述している．ランダムサーチでは, 二分法で優先的に探索したいパラメータ iter 以外のパラメータ alpha, min_count, window については表 1 の中からランダムに選択したパラメータで正解率を計算する．このランダムな選択は Algorithm 1 で指定した run の回数だけ行う．例えば iter = 9 で run = 10 の場合は,

- iter = 9, alpha = 0.025, min_count = 2, window = 8, 5 分割交差検定正解率 = 0.5
- iter = 9, alpha = 0.175, min_count = 6 , window = 3, 5 分割交差検定正解率 = 0.7
- iter = 9, alpha = 0.075, min_count = 4, window = 5, 5 分割交差検定正解率 = 0.6

…

など 10 通りのパラメータの組合せについて正解率を求め, この中で最良の正解率 = 0.7 を iter = 9 の場合の正解率とする．

次に, 区間下限 (low = iter = 1) と区間上限 (high = iter = 9) で正解率の低い方を中間点 (center = iter = 5) で置き換え

Algorithm 1 二分法 + ランダムサーチのアルゴリズム

```
function paramsearch(n = 区間下限もしくは区間上限を中間点で置き換える回数, low = 区間下限, high = 区間上限, run = ランダムサーチの回数, randomparamlist = ランダムサーチするパラメータの組み合わせ){
    param_list_l = {'iter' = [low], randomparamlist};
    score_l = 0;
    for (i = 0; i < run; i = i + 1) {
        score = param_list_l に対して行ったランダムサーチの正解率;
        if (score_l < score)
            score_l = score;
    }
    param_list_h = {'iter' = [high], randomparamlist};
    score_h = 0;
    for (i = 0; i < run; i = i + 1) {
        score = param_list_h に対して行ったランダムサーチの正解率;
        if (score_h < score)
            score_h = score;
    }
    for (x = 0; x < n; x = x + 1) {
        center = (low + high)/2;
        param_list_c = {'iter' = [center], randomparamlist};
        score_c = 0;
        for (i = 0; i < run; i = i + 1) {
            score = param_list_c に対して行ったランダムサーチの正解率;
            if (score_c < score)
                score_c = score;
        }
        if (score_h > score_l)
            score_l = score_c;
            low = center;
        else
            score_h = score_c;
            high = center;
    }
    if (score_l > score_h)
        best = score_l;
    else
        best = score_h;
    return best;
}
```

る. 図 5 の例では, 区間上限 (high = iter = 9) を中間点 (center = iter = 5) で置き換えて, 区間上限 (high = iter = 5), 区間下限 (low = iter = 1) と更新している. 以降, 同様に新たな中間点の正解率を計算して, 区間下限と上限の低い正解率の側を中間点で置き換えていく. 図 5 の例では, 次ステップで区間下限 (low = iter = 1) と区間上限 (high = iter = 5) が, 区間下限 (low = iter = 3) と区間上限 (high = iter = 5) に置き換えられる. この繰り返し処理は, 事前に指定した探索すべきパラメータ集合以外のパラメータが中間点に出現するまで行う. 表 1 に本研究で扱う探索すべきパラメータ集合を示す. 図 5 の例では, 次のステップとして中間点が iter = 4 の場合についての処理を行うが, さらに次のステップでは, 中間点は iter = 3.5 または iter = 4.5 であり, これは事前に指定した探索すべきパラメータ集合 {iter} = {1, 2, 3, 4, 5, 6, 7, 8, 9} 以外であるため, ここで中間点の探索を終了する. ここまでで iter = 1, 9, 5, 3, 4 の 5 つの場合について計算を行った. run = 10 の場合は, 5 つの場合それぞれで 10 回の 5 分割交差検定を行うので, 合計で 50 = 5 * 10 回の 5 分割交差検定をランダムに決めたパラメータで行うことになる.

4. 評 価 実 験

以下では, 提案手法とランダムサーチ, 及びグリッドサーチの文書分類の精度とパラメータ探索コストを比較することで, 提案手法の有効性を示す.

本実験では, 二つの実験で提案手法の有効性を示す. 一つは, Brown Corpus [8] に含まれる英文を news や religion といった

表 1　パラメーター一覧

パラメータ	探索するパラメータの値
alpha	0.025, 0.05, 0.075, 0.1, 0.125, 0.15, 0.175, 0.2, 0.225
iter	1, 2, 3, 4, 5, 6, 7, 8, 9
min_count	1, 2, 3, 4, 5, 6, 7, 8, 9
window	1, 2, 3, 4, 5, 6, 7, 8, 9

6 カテゴリに分類する実験を行う．もう一つは，Stanford Sentiment Treebank Dataset [9] の英文を Very Negative, Negative, Neutoral, Positive, Very Positive の 5 つの評価ラベル毎に分類する実験である．以下では，これを Treebank のレビュー分類実験と呼ぶ．Paragraph Vector を使って文の特徴ベクトルを計算し，この特徴ベクトルを Logistic Regression で分類し，どれだけ正しく分類できるかを評価する．

Paragraph Vector の実装には gensim ライブラリ [7] の doc2vec class を使用する．Logistic Regression 及び，ランダムサーチ，グリッドサーチには scikit-learn [12] ライブラリの関数を用いた．二つのコーパスにおける実験では，ともに表 1 のパラメータリストについて探索する．

4.1　Brown Corpus のカテゴリ分類実験

Brown Corpus [8] に含まれる英文の分類実験を行う．Brown Corpus は，言語研究のための英文のコーパスで，1961 年にブラウン大学で作成された．Brown Corpus の英文は全て news, religion などのカテゴリに分類されている．このカテゴリの内，news, religion, hobbies, science fiction, romance, humor の計 6 カテゴリに含まれる 16,964 文を分類対象とする．表 2 に各カテゴリの文数を示す．4.1.1 項では，Logistic Regression の精度を 5 分割交差検定で評価し，パラメータサーチの手法の違いによる正解率の変化を調べる．

4.1.1　実験データの全てを用いた 5 分割交差検定

ここでは，16,964 文全てに対して表 1 のパラメータの組み合わせについて 5 分割交差検定を行う．ただし，これらのパラメータについては，交差検定においてテストデータの分類結果が最良となるように選択する．グリッドサーチを行ったところ，最良の正解率として 0.5927 を得た．この最良の正解率を与えるパラメータを表 3 に示す．この正解率は，橋戸らが提案した文章の付随情報を考慮する ScoreSent2Vec で Brown Corpus のカテゴリ分類実験を行った際の正解率 0.527 [5] よりも高い．橋戸らの実験においては，パラメータ探索範囲が本実験に比べ狭かったため，パラメータ探索の範囲は正解率への影響が大きいと言える．

しかし，グリッドサーチの場合は，その探索コストが問題となる．そこで提案手法である二分法とランダムサーチを組み合わせた手法において同様の実験を行った．図 6 に結果を示す．この実験では，1 種類のパラメータを二分法で選択している．本実験で行う二分法でのパラメータ探索を二分法で優先的に探索したいパラメータとして iter を選択した場合である図 5 を例に説明する．この場合，探索順序は区間下限 (iter = 1), 区間上限 (iter = 9), 中間点 (iter = 5), 区間上限更新後の中間点 (iter = 3), 区間下限更新後の中間点 (iter = 4) の順であり，計 5 値が探索される．また，二分法で選択した 1 パラメータあたりのランダムサーチ数 run が 10 であれば，iter = 1, 9, 5, 3, 4 について，それぞれ 10 回ランダムサーチをするので，50 回のランダムサーチの探索コストに相当する．これは，他のパラメータを二分法の対象とした場合も同様である．探索コストをこのように仮定して，同等の探索コストにおいて提案手法とランダムサーチのみの場合について比較する．ここで，全てのパラメータについて 50 回のランダムサーチのみを 3 回行った際の平均正解率は 0.5686 であり，4 種のパラメータをそれぞれ二分法の対象とし，ランダムサーチを合計 50 回行った実験 3 回の平均正解率は全てこの 0.5686 を上回った．よって，提案手法は単純なランダムサーチに比べて探索に必要な探索コストにおいて優れていると言える．

表 2　Brown Corpus のカテゴリ毎の文数

カテゴリ	文数
news	4,623
religion	1,716
hobbies	4,193
science fiction	948
romance	4,431
humor	1,053
合計	16,964

表 3　グリッドサーチによる最良のパラメータの組み合わせ (Brown Corpus)

パラメータ	変動値
alpha	0.075
iter	5
min_count	3
window	9

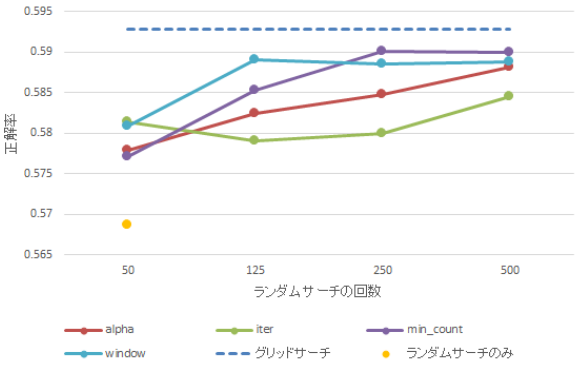

図 6　提案手法の正解率 (Brown Corpus)

4.1.2　未知データの分類

4.1.1 項の実験では，最適なパラメータの探索を，5 分割交差検定でのテストデータの正解率が最大になるように探索している．つまりテストデータの正解率が最大になるようにテストデータを使ってパラメータの探索をしてしまっている．そのため未知のデータをどれだけ予測できるかの評価にはなっていない．そこで全データ 16,964 文を，学習データとテストデータが

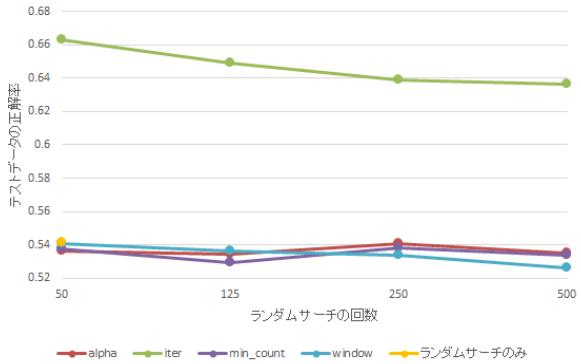

図 7　提案手法によるテストデータの分類精度 (Brown Corpus)

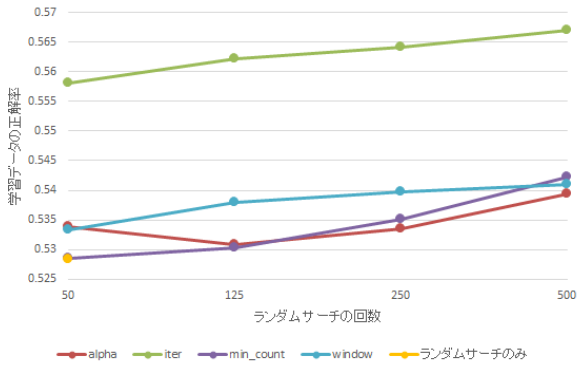

図 8　提案手法による学習データの分類精度 (Brown Corpus)

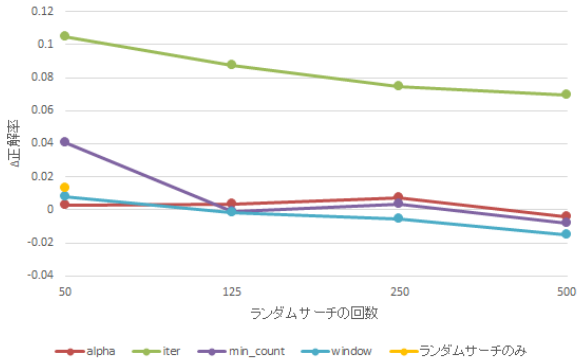

図 9　提案手法による未知データ分類の有効性 (Brown Corpus)

4 対 1 となるように分割する．ここでは，この学習データのみを利用して 4.1.1 項と同様の 5 分割交差検定を行い，最も高い正解率を示したパラメータの組み合わせを利用してテストデータの分類を行った．図 7 にこのテストデータの正解率を，図 8 に学習データの正解率を示す．ここで，学習データの正解率とは，全体の 4/5 の学習データに対する 5 分割交差検定の正解率である．図 7 は，パラメータ iter を二分法の対象とした際のテストデータの正解率が学習データの正解率を大きく上回ったことを示している．ここで，50 回のランダムサーチのみを行った際のテストデータの正解率は，実験 3 回の平均が 0.5415 であり，iter についてはこれを大きく上回った．図 9 は，図 7 のテストデータの正解率と図 8 の学習データの正解率の差を示している．この差を以下では Δ 正解率と呼ぶ．Δ 正解率は式 (2) のように定義する．

Δ 正解率
＝ テストデータの正解率 − 学習データの正解率　　(2)

Δ 正解率が 0 に近い，もしくは 0 を上回っている場合には，提案したパラメータサーチが未知データにも有効といえる．図 9 から，提案手法は iter を二分法で提案する場合，Brown Corpus の分類実験において有効であることが分かる．

4.2　Treebank のレビュー分類実験

ここでは，Stanford Sentiment Treebank Dataset [9] に含まれるレビュー文を評価ラベルごとに分類する実験を行う．Stanford Sentiment Treebank Dataset は，Amazon Mechanical Turk(注1) を利用して，映画のレビュー文に評価ラベルを付与した構文木コーパスである．Amazon Mechanical Turk は，ソフトウェアより人間が行うほうが効率的である作業を，Web で代行依頼できるサービスである．Stanford Sentiment Treebank Dataset には，レビュー各文に対して Very Negative, Negative, Neutral, Positive, Very Positive の 5 種類の評価ラベルが付与されている．レビュー文は事前に学習データ 8,544 件，テストデータ 2,210 件に分割されている．表 4 に学習データとテストデータのラベル毎の文数を示す．

本実験では，表 4 の学習データ及びテストデータそれぞれの先頭 1,000 件を利用する．4.1.2 項で述べた Brown Corpus の未知データ分類実験と同様に，学習データに対して 5 分割交差検定を行って得たパラメータの組み合わせを利用して，未知データに相当するテストデータを分類する．図 10 にテストデータ

表 4　Stanford Sentiment Treebank Dataset の文数

評価ラベル	学習データ	テストデータ
Very Negative	1,092	279
Negative	2,218	633
Neutral	1,624	389
Positive	2,322	510
Very Positive	1,288	399
all	8,544	2,210

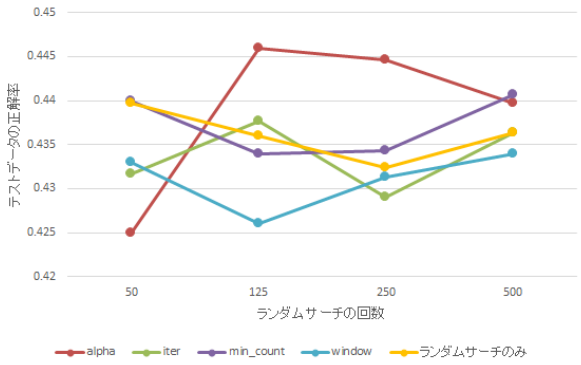

図 10　提案手法によるテストデータの分類精度 (Treebank)

(注1)：https://www.mturk.com/mturk/welcome

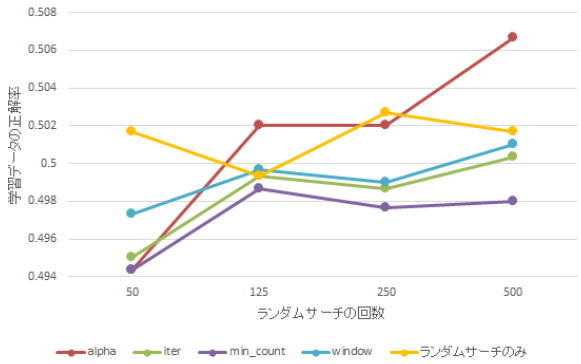

図 11　提案手法による学習データの分類精度 (Treebank)

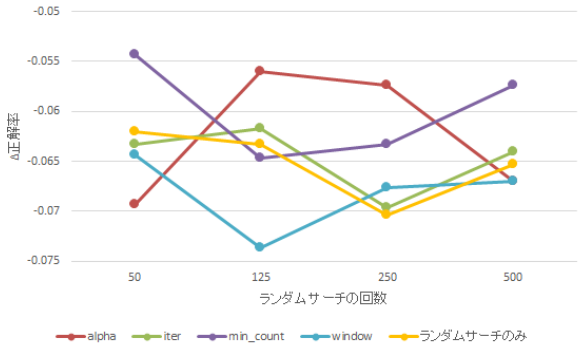

図 12　提案手法による未知データ分類の有効性 (Treebank)

の正解率, 図 11 に学習データの正解率を示す. ここで, 50 回のランダムサーチのみを行った際のテストデータの正解率は実験 3 回の平均が 0.4396 であり, 図 10 で同等のコストで比べると, min_count を二分法の対象とした際の正解率のみがこれを上回った. また, 500 回のランダムサーチのみを行った際のテストデータの正解率は実験 3 回の平均で 0.4363 であり, 図 10 では alpha や min_count を二分法の対象とした際の正解率がこれを上回り, iter を二分法の対象とした時の正解率と同じであった. よって, テストデータ正解率については, 提案手法はランダムサーチとほぼ同等の精度であることが分かる. 図 12 は, テストデータの正解率と学習データの正解率の差 (Δ正解率) を示している. ここで, 50 回のランダムサーチのみ行った実験のΔ正解率は-0.0620 で min_count を二分法の対象とした場合はこれを上回った. 250 回のランダムサーチのみを行った実験のΔ正解率は-0.0703 であり, 全てのパラメータについてこれを上回った. よって正解率の差についても, 提案手法はランダムサーチとほぼ同等であると言える.

5. 考　　察

4 節の実験では, doc2vec のパラメータのうち, alpha, iter, min_count, window について探索した. ここでは, Brown Corpus 及び Treebank の分類において有効なパラメータ値と分類精度の関係について考察する. また, 実行時間の短縮についても考察する.

5.1　有効なパラメータと分類精度

5.1.1　Brown Corpus のカテゴリ分類実験

図 6 では, グリッドサーチの正解率 0.5927 の-1 ポイント以内の正解率を示したパラメータの組み合わせが合計で 13 回確認できる. 以下に二分法の対象とした際のパラメータ値の選択傾向を示す.

- alpha
  0.075, 0.1, 0.125 のいずれかであった.
- iter
  3 もしくは 4 が 12 回あった.
- min_count
  3 が 11 回あり, 残り 2 回は 2 であった.
- window
  7, 8, 9 のいずれかであった.

以上より, window に関しては比較的広範囲の単語を同時に参照する必要性が示されている. また, window を二分法の対象とすると, ほぼ全ての値が 9 となったため, 10 や 11 といったより大きな値についても調べる必要があることが分かった. また, 図 7 に示したテストデータ分類では, iter を二分法の対象とすると値は全て 9 となった.

5.1.2　Treebank のレビュー分類実験

図 12 に示した未知データ分類の有効性については, 提案手法がランダムサーチ 250 回に相当する探索コストの場合, ランダムサーチに比べてわずかに有効であることが確認された. Treebank のレビュー分類実験においては, 必ずしも学習データに対する 5 分割交差検定で良い正解率を残したパラメータの組み合わせが未知データ分類でも有効という訳ではなかった.

5.2　実 行 時 間

本稿ではこれまで探索コストをパラメータサーチを行った回数, すなはち 5 分割交差検定を行った回数としてきた. これは, 一般的にはパラメータサーチは並列計算で行うためである. そこで, 提案手法のパラメータサーチについて並列計算を前提として, 実行時間の短縮について考察する.

パラメータサーチの並列計算を最適化するにあたり, iter の値に応じたプロセスの振り分け調整が必要となる. これは, iter が反復学習の回数を決定するパラメータであり, パラメータそのものが実行時間に影響を与えるからである. 例えば, iter = 9 のときにかかる実行時間は iter = 1 の時の 9 倍になる. 並列計算を行う際には, 探索するパラメータの組み合わせを, iter の合計値が例えば各 CPU コアでほぼ同等となるように調整することで, 実行時間の短縮ができる. 2 コア CPU で四つのパラメータの組み合わせについて探索する場合を例に説明する. iter = 1, iter = 3, iter = 4, iter = 9 の順にパラメータが選択されたとする. まず, コア A で iter = 1, コア B で iter = 3 のプロセスが実行される. 次にコア A での実行が終わり次第, iter = 4 のプロセスが実行され, コア B でも同様に前のプロセスが終われば iter = 9 のプロセスが実行される. この場合には, コア A で iter = 7 相当の時間が余る. これをコア A に iter = 3 と iter = 4, コア B に iter = 1, iter = 9 を割り当てることで, コア A

でプロセスが実行されない時間を iter = 3 相当まで減らすことができる.

6. ま と め

本稿では doc2vec にて低コストで効率的なパラメータサーチを行う手法を提案し, Brown Corpus と Stanford Sentiment Treebank Dataset の英文をカテゴリ毎に分類する実験を行った.

Brown Corpus の分類実験では, 提案した二分法にランダムサーチを組み合わせた手法が, ランダムサーチに比べパラメータ探索コストにおいて優位性があることを確認した. グリッドサーチと正解率を比較しても, 低コストでこの正解率に近づくことが分かった. またこの実験から, 探索を行った doc2vec の 4 種のパラメータと分類正解率の関係が明らかになり, 考慮する前後の単語数を決める window のパラメータに関しては, 実験しなかったもう少し大きな値についても調べる必要があることが分かった.

Treebank コーパスの分類実験においては, データセットが大きかったためグリッドサーチで最適解を求められなかった. そのため, 実験ではデータセットの一部を使用し, 提案手法とランダムサーチを比較したところ, 同等のパラメータ探索コストで, 両者はほぼ同等の正解率であった.

今後の課題としては, 他のコーパスの分類実験や並列処理による実行時間の短縮などがあげられる.

文 献

[1] Mikolov, T., Sutskever, I., Chen, K., Corrado G. and Dean, J.: Distributed representations of words and phrases and their compositionality, *Advances in Neural Information Processing Systems*, pp. 3111–3119, 2013.

[2] Mikolov, T., Chen, K., Corrado G. and Dean, J.: Efficient estimation of word representations in vector space, *arXiv preprint arXiv:1301.3781*, pp. 1–12, 2013.

[3] Le, Q. and Mikolov, T.: Distributed Representations of Sentences and Documents, *CoRR*, abs/1405.4053, pp. 1–9, 2014.

[4] Socher, R., Perelygin, A., Wu, J. Y., Chuang, J., Manningand, C. D., Ng, A. Y. and Potts, C.: Recursive Deep Models for Semantic Compositionality Over a Sentiment Treebank, *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, Stroudsburg, PA, Association for Computational Linguistics, pp. 1631–1642, 2013.

[5] 橋戸拓也, 新妻弘崇, 太田学: Paragrph Vector への追加情報の効率的な埋め込み, 情報処理学会研究報告, Vol. 2015-DBS-162, No. 11, pp. 1-8, 2015.

[6] 橋戸拓也, 新妻弘崇, 太田学: Paragraph Vector へ埋め込む有効な付随情報の検討, DEIM Forum 2016, C5-6, 2016.

[7] Radim Řehůřek and Petr Sojka.: Software Framework for Topic Modelling with Large Corpora, Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks, pp. 45–50, 2010.

[8] Francis, W. N., Kucera, H.: Brown Corpus Manual, Brown University, 1979.

[9] Socher, R., Perelygin, A., Wu, J. Y., Chuang, J., Manning, C. D., Ng, A. Y., Potts, C.: Recursive deep models for semantic compositionality over a sentiment treebank, *Conference on Empirical Methods in Natural Language Processing*, pp. 1–12, 2013.

[10] 中野滋徳, 足立顕, 牧野武則: 提題表現に基づく重要段落抽出, 情報処理学会研究報告. NL, 自然言語処理研究会報告, Vol. 162, pp. 159–166, 2004.

[11] 佐藤元紀, 伊藤孝行: Paragraph Vector と多層パーセプトロンを用いた有害文書の分類手法, 情報処理学会第 77 回全国大会, pp. 165–166, 2015.

[12] Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., Duchesnay, É.: Scikit-learn: Machine Learning in Python, *The Journal of Machine Learning Research*, Vol. 12, pp. 2825–2830, 2011.