平成 29 年度茨城大学工学部情報工学科
卒業研究論文

word2vec のパラメーター調整による nwjc2vec の効果的な fine-tuning

平成 30 年 2 月 5 日

茨城大学 工学部 情報工学科
14T4027Y 熊谷 佳奈
指導教員：新納 浩幸 教授

平成 29 年度茨城大学工学部情報工学科卒業研究

word2vec のパラメーター調整による nwjc2vec の効果的な fine-tuning

氏名：14T4027Y 熊谷 佳奈
指導教員：新納 浩幸 教授

論文要旨

本論文では，日本語分散表現データ nwjc2vec を効果的に fine-tuning する際に，分散表現構築プログラムのパラメータをどのように設定すべきかを調査する．

nwjc2vec は国語研日本語ウェブコーパス（以下 NWJC）から構築された大規模な分散表現データである．NWJC は超大規模コーパスであるため，そこから構築された nwjc2vec は非常に高品質であると考えられる．実際，いくつかの報告でこの点が確認されている ．また NWJC は様々なコーパスを含んでいるために，広い範囲の領域で利用できると考えられる．ただし nwjc2vec であっても領域適応の問題が生じていることが示され利用領域に応じて fine-tuning することが提案されている．

nwjc2vec を fine-tuning することは有益であるが，それを行うためにどの程度の規模のコーパスが必要なのかは明らかではない．理想的には小さなコーパスで効果的に fine-tuning できることが望まれる．ここでは小規模なコーパスを用いて分散表現構築プログラムのパラメータの調整だけで，どの程度の fine-tuning できるかを調査する．なおここでは分散表現構築プログラムとして word2vec を用いる．

分散表現の評価には，LSTM を用いて，得られた言語モデルにより行う．具体的には LSTM を用いて言語モデルを構築する際に，分散表現の学習も同時に行うが，実験では分散表現の学習は行わずに評価対象の分散表現を利用して言語モデルを学習する．利用した分散表現が高品質であれば，得られる言語モデルも高品質であると考えて評価を行う．

結果，nwjc2vec の効果的な fine-tuning には word2vec のパラメータのうちバッチサイズが最も影響していること，適切でないパラメータでは fine-tuning が逆効果になることが判明した．また fine-tuning にはコーパスの量が本質的であることも確認できた．

目 次

表 目 次

図 目 次

第1章　はじめに

1.1　概要

本論文では，日本語分散表現データ nwjc2vec を効果的に fine-tunig する際に，分散表現構築プログラムのパラメータをどのように設定すべきかを調査する．

nwjc2vec は国語研日本語ウェブコーパス（以下 NWJC）[1] から構築された大規模な分散表現データである [8]．NWJC は超大規模コーパスであるため，そこから構築された nwjc2vec は非常に高品質であると考えられる．実際，いくつかの報告でこの点が確認されている [9][6][7]．また NWJC は様々なコーパスを含んでいるために，広い範囲の領域で利用できると考えられる．ただし nwjc2vec であっても領域適応の問題が生じていることが示され利用領域に応じて fine-tuning することが提案されている [5]．

nwjc2vec を fine-tuning することは有益であるが，それを行うためにどの程度の規模のコーパスが必要なのかは明らかではない．理想的には小さなコーパスで効果的に fine-tuning できることが望まれる．ここでは小規模なコーパスを用いて分散表現構築プログラムのパラメータの調整だけで，どの程度の fine-tuning できるかを調査する．なおここでは分散表現構築プログラムとして word2vec を用いる．

分散表現の評価には，LSTM を用いて，得られた言語モデルにより行う．具体的には LSTM を用いて言語モデルを構築する際に，分散表現の学習も同時に行うが，実験では分散表現の学習は行わずに評価対象の分散表現を利用して言語モデルを学習する．利用した分散表現が高品質であれば，得られる言語モデルも高品質であると考えて評価を行う．

結果，nwjc2vec の効果的な fine-tuning には word2vec のパラメータのうちバッチサイズが最も影響していること，適切でないパラメータでは fine-tuning が逆効果になることが判明した．また fine-tuning にはコーパスの量が本質的であることも確認できた．

第2章　分散表現

2.1　概要

単語の意味や単語間の関係を低次元の密なベクトルで表現したものが分散表現である．意味が似ている単語同士は近い値のベクトルで表される．ベクトルで表現することの利点は演算が行えることである．例えば「王様」–「男」＋「女」＝「女王様」や「おにぎり」–「米」＋「パン」＝「サンドウィッチ」というように，演算を行うことで単語間の関係が求められる．そして単語間の関係を示すベクトルも意味が似ている物同士は近い値で表される．先の演算の例からも分かるように,「男」と「女」の関係と「王様」と「女王様」の関係はよく似ている．したがって「男」と「女」の関係を表すベクトルと「王様」と「女王様」を示すベクトルの値も図 2.1 のように近い値になる．

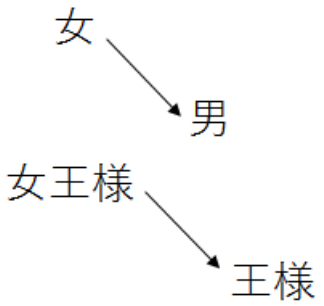

図 2.1: 単語間のベクトル

こうすることで例えば日本における東京はロシアにとっての何か，というように単語同士の関係から導くことができるようになる．

2.2　one-hot 表現

単語をベクトルで表現する手法は，分散表現の他に one-hot 表現という手法がある．one-hot 表現は要素のうちある 1 つのみを”1” とし，それ以外を”0” としたベクトルである．例えば単語の集合を『犬・猫・鳥・牛・馬』の 5 つとした時，” 犬” を表すベクトルは図 2.2 のようになる．

しかし，このようなベクトル表現では，内積を計算しても 0 しか得られず，単語間の類似度を計ることができない．また，1 つの単語に 1 次元を割り当てるため単語の数だけ次元を必要とする結果，高次元で疎なベクトルになってしまう．

図 2.2: one-hot 表現

2.3 Bag of Words

one-hot 表現以外にも分散表現とは異なるベクトル表現の手法がある．Bag of Words(以下 BoW) は，文中のある単語の周辺単語により単語の値を決める手法である．値の決め方には出現したかどうかを one-hot 表現により決めるものと，出現頻度により決めるものがある．前者の場合は先に述べたようにベクトル同士の演算を行っても，単語同士が等しいか否かしか求められない．後者であれば周辺単語の頻度の違いから，単語同士の類似性を求めることができる．

BoW の次元数は辞書として設定した単語の数だけ次元を必要とする．よって BoW も高次元で疎なベクトルである．

2.4 word2vec

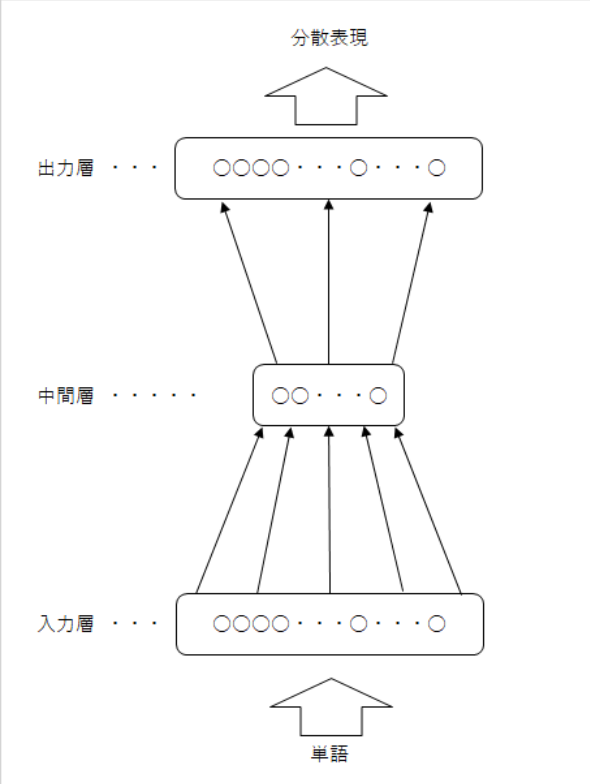

図 2.3: ニューラルネットワーク

分散表現を構築するためのツールの一つが word2vec である．構築はコーパスを入力として行われ，初期値に分散表現を利用でき，出力はコーパス中に出現した単語の分散表現である．そして構築する際にはベクトルの次元数やトレーニング反復回数などのパラメータを設定することができる．本論文でこのパラメーターの設定を変更することで，少ないコーパス量での効果的な fine-tuning を図る．

word2vec には学習のアルゴリズムとして Skip-gram モデルと Continuous Bag of Words モデルがある．

本論文では Skip-Gram モデルによる word2vec を利用する．Skip-Gram モデルとは入力層・中間層・出力層からなるニューラルネットワークであり，単語を入力データとして重み付けをして分散表現を出力する (図 2.3 参照)．

入力層から中間層への遷移時に重みによって学習を行う．

word2vec のシステムプログラムはオープンソース化されており，ウェブから利用ができる．[1]

2.5 領域適応

分散表現を構築する際に利用したコーパスの領域と，処理対象とする領域が異なる場合，性能が大きく劣化してしまう現象が起きる．これが領域適応問題である．一般的に単語には複数の意味が存在し，使用領域によってどの意味として使われるかは大きく左右される．例えば「犬」という単語には「動物」としての語義と「スパイ」としての語義が存在する．

この問題を解決する方法として，処理用域に合ったコーパスで学習させた分散表現を利用する方法が挙げられる．

2.6 nwjc2vec

NWJC から word2vec を利用して構築されたのが nwjc2vec である．NWJC は文の収集を約 1 億 URL のウェブ上の日本語テキストから行っており，その単語数は約 258 億語と新聞記事にすると約 1200 年分に相当する非常に大規模なコーパスである．

また内包する分野も多岐にわたっており，これらのことから nwjc2vec は非常に高品質であると考えられている．しかし nwjc2vec のような高品質な分散表現であっても，領域適応の問題が生じている．そこで利用領域に応じたコーパスで fine-tuning するといった，この問題の改善が提案されている．

[1] https://github．com/svn2github/word2vec

第3章　関連研究

一般に分散表現の優劣は評価法によって異なり，タスクに応じて分散表現をチューンニングすべきことが指摘されている [3].

最も単純なチューンニングの方法は，本論文で行ったように学習済みの分散表現を初期値として，追加コーパスを用いて再度，分散表現を学習する fine-tuning である．この場合，大規模な追加コーパスを必要とするが，辞書などの外部知識を組み入れて分散表現を改善する試みもある．論文 [4] では分散表現を学習する目的関数の部分に，事前知識を利用した形に変更することで，分散表現を改善している．また論文 [2] では大量のコーパスから構築した分散表現を，外部知識を使って再学習する retrofitting と呼ばれる手法により分散表現をチューニングしている．

第4章　提案手法

4.1　nwjc2vec の fine-tuning

分散表現を構築する際の word2vec のパラメータが複数存在する．まずそれらパラメータの基準値を設定する．その基準値のパラメータと追加コーパスにより nwjc2vec を fine-tuning する．次に基準値の中で，ウィンドウサイズのみを変更して，追加コーパスにより nwjc2vec を fine-tuning する．同様にして今度は基準値の中で，バッチサイズのみを変更して，追加コーパスにより nwjc2vec を fine-tuning する．同様にして最後に基準値の中で，エポック回数のみを変更して，追加コーパスにより nwjc2vec を fine-tuning する．

追加コーパスとしては，毎日新聞 ’93 年度版から’99 年度版の 7 年分の記事からランダムに抽出した 10 万文を用いる．

4.2　分散表現の評価

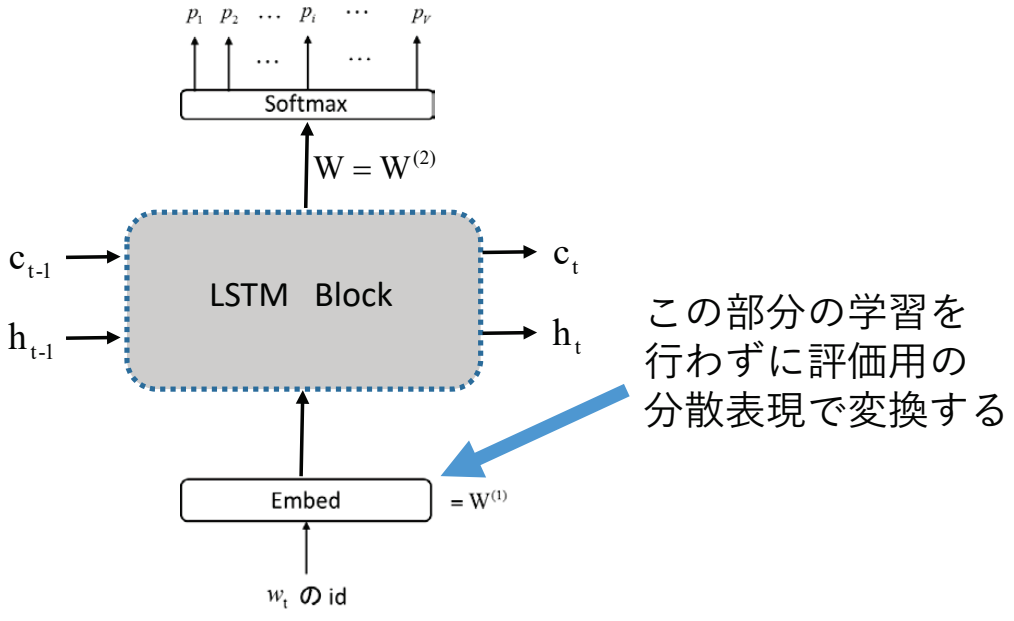

図 4.1: LSTM の時刻 t 時の入出力

fine-tuning により得られた分散表現を評価するには，類似度が付与された単語ペアのデータを利用するのが一般的であるが，ここでは論文 [7] で用いられた手法を用い

る．論文 [7] では LSTM による言語モデルを用いて分散表現の評価を行っている．通常，LSTM により言語モデルを学習する場合，分散表現も同時に学習するが，ここではその学習を行わずに（図 4.1 参照），単語から分散表現への変換は評価対象の分散表現を用いて行う．LSTM で利用する訓練コーパスが同一の場合，得られた言語モデルの優劣により利用した分散表現の優劣を表すと考える．言語モデルの評価にはパープレキシティを用いる．

パープレキシティは言語モデルのエントロピーにより求められる．

言語モデル M のエントロピー H は以下の式で表される．

$$H = \frac{1}{|D|} \sum_{i=1}^{|D|} - \log 2(P(W_i|M))$$

このときの言語モデル M のパープレキシティは 2^H となる．

パープレキシティはモデルの複雑さを表しており，値が小さいほどモデルの品質が良いと判断できる．

また言語モデルの訓練用コーパスと言語モデルの評価用コーパスは，ともに毎日新聞 ’93 年度版から’99 年度版の 7 年分の記事からランダムに抽出した 10 万文及び 1 万文である．またこれらは nwjc2vec を fine-tuning するため利用した追加コーパスとは重複していない．

第5章　実験

5.1　実験設定

fine-tuning する際の word2vec のパラメータの基準値を表 5.1 に示す．

表 5.1: word2vec のパラメータの基準値

ユニット数	200
ウィンドウサイズ	5
バッチサイズ	10
エポック回数	10
使用モデル	skip-gram

まず表 5.1 の基準値を用いて，nwjc2vec の fine-tuning を行い，分散表現を構築する．ここで構築できた分散表現を base_emb と名付ける．次に表 5.1 の値からウィンドウサイズのみを 8 に変更し，nwjc2vec の fine-tuning を行い，分散表現を構築する．ここで構築できた分散表現を win_emb と名付ける．同様にバッチサイズのみを 20 に変更し，nwjc2vec の fine-tuning を行い，分散表現を構築する．ここで構築できた分散表現を batch20_emb と名付ける．また同様にバッチサイズのみを 100 に変更し，nwjc2vec の fine-tuning を行い，分散表現を構築する．ここで構築できた分散表現を batch100_emb と名付ける．また同様にエポック回数のみを 20 に変更し，nwjc2vec の fine-tuning を行い，分散表現を構築する．ここで構築できた分散表現を epch_emb と名付ける．

5.2　実験結果

上記 5 つの分散表現（base_emb，win_emb，batch20_emb，batch100_emb，epch_emb）を用いて，10 万文からなる訓練用コーパスを用いて，LSTM による言語モデルの構築を行った．LSTM の学習での各 epoch 終了時に得られている言語モデルのパープレキシティを 1 万文からなる評価用コーパスを用いて測定した．この結果を表 5.2 と図 5.1 に示す．表 5.2 と図 5.1 には fine-tuning を行わず nwjc2vec を用いた場合のパープレキシティも示している．

batch100_emb だけが fine-tuning した効果があった．他の分散表現はどれも fine-tuning の効果はなく，むしろ性能が下がっている．つまり不適切なパラメータを使ってしまったら，fine-tuning が逆効果になる危険性があることが示された．

表 5.2: パラメータ設定変更時のパープレキシティ

epoch	nwjc2vec	base_emb	win_emb	batch20_emb	batch100_emb	epch_emb
1	91.03	93.70	95.36	91.51	89.69	95.06
2	73.20	75.21	75.71	73.43	72.36	75.89
3	68.65	70.21	70.52	68.69	67.54	70.30
4	**67.43**	68.85	**69.33**	**67.56**	**66.23**	68.46
5	67.52	**68.84**	69.51	67.70	66.35	**68.17**
6	68.17	69.55	70.20	68.37	67.13	68.54
7	69.08	70.37	71.11	69.37	68.17	69.29
8	70.06	71.48	72.22	70.56	69.37	70.36
9	71.09	72.71	73.40	71.80	70.58	71.49
10	72.18	73.92	74.66	73.06	71.82	72.68

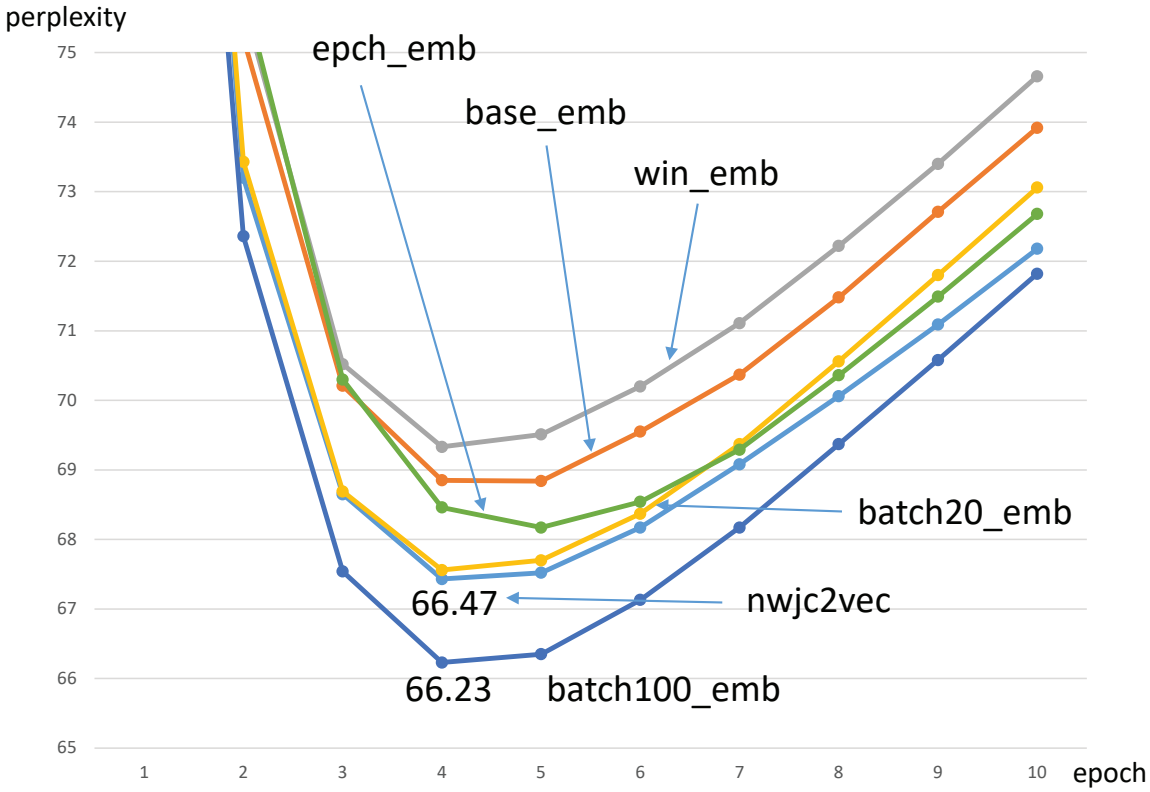

図 5.1: パラメータ設定変更結果

第6章　考察

実験では最初に設定した基準値が適切ではなかったために効果的なパラメータを得ることはできなかった．バッチサイズの基準値を 100 に設定しなおして，再度，同様の実験を行えばよいと考えている．ただし表 5.2 と図 5.1 からウィンドウサイズやエポック回数よりもバッチサイズが最も fine-tuning の効果に影響していることが分かるので，batch100_emb の性能辺りが限度だと予想している．

また nwjc2vec の効果的な fine-tuning で最も重要な要因は追加コーパスのサイズであると考えられる．この点を確認するため，先の実験では追加コーパスとして 10 万文からなるコーパスを用いたところを，コーパスのサイズを変え，20 万文のコーパス，30 万文のコーパスを用いて，同様の実験を行った．なおこの際の word2vec のパラメータは表 5.1 の値のうちバッチサイズを 100 に設定したものを用いた．

この結果を表 6.1 と図 6.1 に示す．表 6.1 と図 6.1 から追加コーパスのサイズが大きいほど fine-tuning の効果が高いことが分かる．

表 6.1: 追加コーパスのサイズ

epoch	10 万文 (batch100_emb)	20 万文	30 万文
1	89.69	89.55	87.94
2	72.36	71.50	70.28
3	67.54	66.96	65.83
4	**66.23**	65.65	**64.61**
5	66.35	**65.62**	64.75
6	67.13	66.27	65.44
7	68.17	67.32	66.45
8	69.37	68.46	67.56
9	70.58	69.64	68.78
10	71.82	70.89	69.92

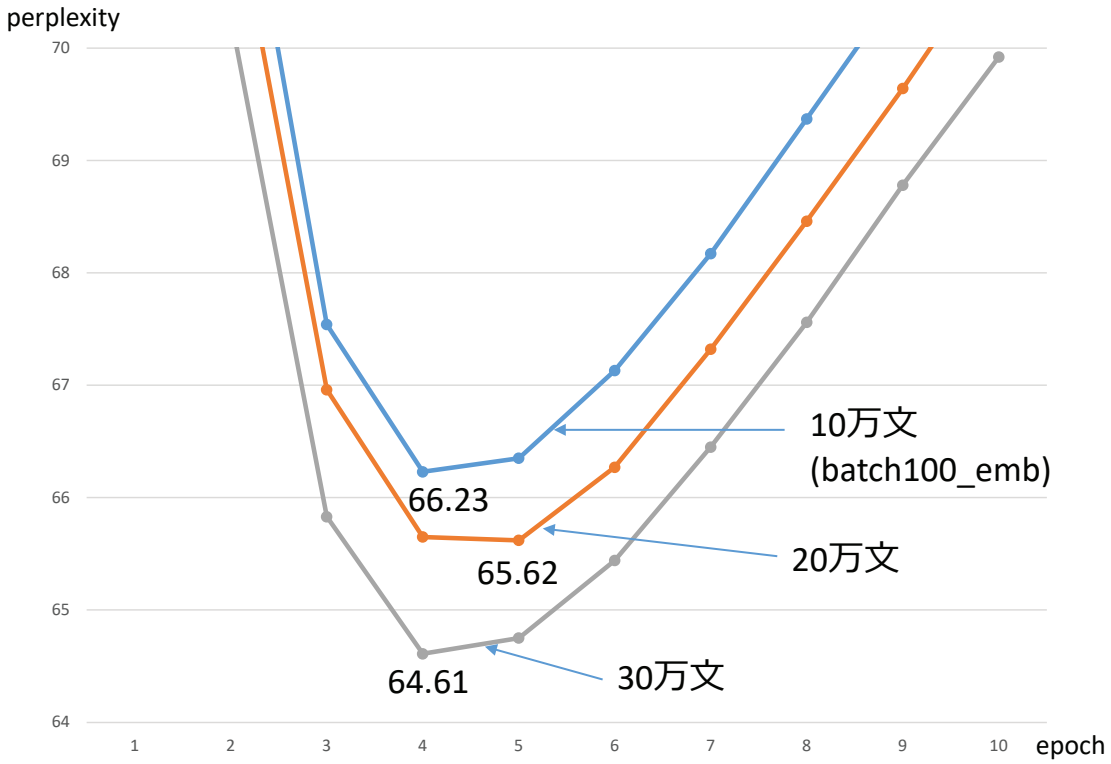

図 6.1: コーパス量変更結果

本論文で行ったチューニングの手法は，分散表現のチューニングとしては最もシンプルな方法である．学習には新たに大規模な追加コーパスを必要とする．追加コーパスを用いる代わりとして，辞書などの外部知識を融合する方式がある．その方式を利用して，nwjc2vec を改善する手法について今後調査していきたい．

第7章　おわりに

本論文では nwjc2vec に対して少量の追加コーパスを用いて fine-tuning を行う際の，word2vec の最適なパラメータを調査した．

その結果，バッチサイズの調整が最も fine-tuning には影響があることと不適切なパラメータの設定では fine-tuning が逆効果になることが判明した．また実験の結果から効果的な fine-tuning のためには，追加コーパスのサイズが本質的に重要であると予想し，同様の実験を行った．その結果，予想通りコーパスサイズが大きいほど fine-tuning の効果が高いことが確かめられた．

nwjc2vec を fine-tuning するには大規模な追加コーパスが必要と考えられる．大規模な追加コーパスの代用として外部知識を利用する方式について，今後調査を行おうと考えている．

謝辞

本研究を進めるにあたり，たくさんのご指導ご協力を頂いた指導教員の新納浩幸教授をはじめとして，自然言語処理研究室の皆様に心より感謝申し上げます.

参考文献

[1] Masayuki Asahara, Kikuo Maekawa, Mizuho Imada, Sachi Kato, and Hikari Konishi. Archiving and analysing techniques of the ultra-large-scale web-based corpus project of ninjal, japan. *Alexandria: The Journal of National and International Library and Information Issues*, Vol. 25, No. 1-2, pp. 129–148, 2014.

[2] Manaal Faruqui, Jesse Dodge, Sujay K. Jauhar, Chris Dyer, Eduard Hovy, and Noah A. Smith. Retrofitting Word Vectors to Semantic Lexicons. In *Proceedings of NAACL*, 2015.

[3] Tobias Schnabel, Igor Labutov, David M Mimno, and Thorsten Joachims. Evaluation methods for unsupervised word embeddings. In *EMNLP 2015*, pp. 298–307, 2015.

[4] Mo Yu and Mark Dredze. Improving Lexical Embeddings with Semantic Knowledge. In *ACL (2)*, pp. 545–550, 2014.

[5] 新納浩幸, 古宮嘉那子, 佐々木稔. nwjc2vec の fine-tuning. 国語研言語資源活用ワークショップ, pp. PB–4, 2017.

[6] 新納浩幸, 古宮嘉那子, 佐々木稔. 順方向多層 LSTM と分散表現を用いた教師あり学習による語義曖昧性解消. 情報処理学会第 232 回自然言語処理研究会, pp. NL–232–4, 2017.

[7] 新納浩幸, 浅原正幸, 古宮嘉那子, 佐々木稔. nwjc2vec:国語研日本語ウェブコーパスから構築した単語の分散表現データ. 自然言語処理, Vol. 24, No. 5, pp. 705–720, 2017.

[8] 浅原正幸, 岡照晃. nwjc2vec:『国語研日本語ウェブコーパス』に基づく単語の分散表現データ. 言語処理学会第 23 回年次大会発表論文集, pp. 94–97, 2017.

[9] 山木翔馬, 新納浩幸, 古宮嘉那子, 佐々木稔. 教師データを用いた語義の分散表現の構築. 言語処理学会第 23 回年次大会発表論文集, pp. 78–81, 2017.

付録 ソースリスト

ソースコード 1： word2vec による分散表現の構築 (myw2v.py)

```
#!/usr/bin/env python

"""Sample script of word embedding model.

This code implements skip-gram model and continuous-bow model.
Use ../ptb/download.py to download 'ptb.train.txt'.
"""
import argparse
import collections
import time
import pickle

import numpy as np

import chainer
from chainer import cuda
import chainer.functions as F
import chainer.links as L
import chainer.optimizers as O

parser = argparse.ArgumentParser()
parser.add_argument('--gpu', '-g', default=-1, type=int,
                    help='GPU ID (negative value indicates CPU)')
#parser.add_argument('--unit', '-u', default=100, type=int,
parser.add_argument('--unit', '-u', default=200, type=int,
                    help='number of units')
parser.add_argument('--window', '-w', default=5, type=int,
                    help='window size')
parser.add_argument('--batchsize', '-b', type=int, default=100,
                    help='learning minibatch size')
parser.add_argument('--epoch', '-e', default=10, type=int,
                    help='number of epochs to learn')
parser.add_argument('--model', '-m', choices=['skipgram', 'cbow'],
                    default='skipgram',
                    help='model type ("skipgram", "cbow")')
parser.add_argument('--out-type', '-o', choices=['hsm', 'ns', 'original'],
# default='hsm',
                    default='ns',
                    help='output model type ("hsm": hierarchical softmax, '
                    '"ns": negative sampling, "original": no approximation
                        )')
#parser.add_argument('--test', dest='test', action='store_true')
parser.set_defaults(test=False)

args = parser.parse_args()
if args.gpu >= 0:
    cuda.check_cuda_available()
xp = cuda.cupy if args.gpu >= 0 else np

```

```
49 print('GPU: {}'.format(args.gpu))
50 print('# unit: {}'.format(args.unit))
51 print('Window: {}'.format(args.window))
52 print('Minibatch-size: {}'.format(args.batchsize))
53 print('# epoch: {}'.format(args.epoch))
54 print('Training model: {}'.format(args.model))
55 print('Output type: {}'.format(args.out_type))
56 print('')
57
58
59 class ContinuousBoW(chainer.Chain):
60
61     def __init__(self, n_vocab, n_units, loss_func):
62         super(ContinuousBoW, self).__init__(
63             embed=F.EmbedID(n_vocab, args.unit),
64             loss_func=loss_func,
65         )
66
67     def __call__(self, x, context):
68         h = None
69         for c in context:
70             e = self.embed(c)
71             h = h + e if h is not None else e
72
73         return self.loss_func(h, x)
74
75
76 class SkipGram(chainer.Chain):
77
78     def __init__(self, n_vocab, n_units, loss_func, ev):
79         super(SkipGram, self).__init__(
80             embed=L.EmbedID(n_vocab, n_units, initialW=ev),
81             loss_func=loss_func,
82         )
83
84     def __call__(self, x, context):
85         loss = None
86         for c in context:
87             e = self.embed(c)
88
89             loss_i = self.loss_func(e, x)
90             loss = loss_i if loss is None else loss + loss_i
91
92         return loss
93
94
95 class SoftmaxCrossEntropyLoss(chainer.Chain):
96     def __init__(self, n_in, n_out):
97         super(SoftmaxCrossEntropyLoss, self).__init__(
98             W=L.Linear(n_in, n_out),
99         )
100
101     def __call__(self, x, t):
102         return F.softmax_cross_entropy(self.W(x), t)
103
104
105 def calculate_loss(model, dataset, position):
106     # use random window size in the same way as the original word2vec
107     # implementation.
108     w = np.random.randint(args.window - 1) + 1
109     context = []
```

```
110     for offset in range(−w, w + 1):
111         if offset == 0:
112             continue
113         c_data = xp.asarray(dataset[position + offset])
114         c = chainer.Variable(c_data)
115         context.append(c)
116     x_data = xp.asarray(dataset[position])
117     x = chainer.Variable(x_data)
118     return model(x, context)
119
120
121 if args.gpu >= 0:
122     cuda.get_device(args.gpu).use()
123
124 index2word = {}
125 word2index = {}
126 counts = collections.Counter()
127 dataset = []
128 #with open('ptb.train.txt') as f:
129 with open('tg-corpus.dat') as f:
130     for line in f:
131         for word in line.split():
132             if word not in word2index:
133                 ind = len(word2index)
134                 word2index[word] = ind
135                 index2word[ind] = word
136             counts[word2index[word]] += 1
137             dataset.append(word2index[word])
138
139 n_vocab = len(word2index)
140
141 ev = np.zeros(n_vocab * 200).reshape(n_vocab, 200)
142
143 wdic = {}
144 f = open('nwjc2vec-norm1.db','r')
145 #f = open('NWCJ-W2V.db','r')
146 wdic = pickle.load(f)
147 f.close()
148 ## embed[w] -> v
149
150 for i in range(n_vocab):
151     word = index2word[i]
152     vec = wdic[word]
153     for j in range(200):
154         ev[i,j] = vec[j]
155
156 # if args.test:
157 # dataset = dataset[:100]
158
159 print('n_vocab: %d' % n_vocab)
160 print('data length: %d' % len(dataset))
161
162 if args.out_type == 'hsm':
163     HSM = L.BinaryHierarchicalSoftmax
164     tree = HSM.create_huffman_tree(counts)
165     loss_func = HSM(args.unit, tree)
166 elif args.out_type == 'ns':
167     cs = [counts[w] for w in range(len(counts))]
168     loss_func = L.NegativeSampling(args.unit, cs, 20)
169 elif args.out_type == 'original':
170     loss_func = SoftmaxCrossEntropyLoss(args.unit, n_vocab)
```

```
else:
    raise Exception('Unknown output type: {}'.format(args.out_type))

if args.model == 'skipgram':
    model = SkipGram(n_vocab, args.unit, loss_func, ev) ####
elif args.model == 'cbow':
    model = ContinuousBoW(n_vocab, args.unit, loss_func)
else:
    raise Exception('Unknown model type: {}'.format(args.model))

if args.gpu >= 0:
    model.to_gpu()

dataset = np.array(dataset, dtype=np.int32)

optimizer = O.Adam()
optimizer.setup(model)

begin_time = time.time()
cur_at = begin_time
word_count = 0
skip = (len(dataset) - args.window * 2) // args.batchsize
next_count = 100000
for epoch in range(args.epoch):
    accum_loss = 0
    print('epoch: {0}'.format(epoch))
    indexes = np.random.permutation(skip)
    for i in indexes:
        if word_count >= next_count:
            now = time.time()
            duration = now - cur_at
            throuput = 100000. / (now - cur_at)
            print('{} words, {:.2f} sec, {:.2f} words/sec'.format(
                word_count, duration, throuput))
            next_count += 100000
            cur_at = now

        position = np.array(
            range(0, args.batchsize)) * skip + (args.window + i)
        loss = calculate_loss(model, dataset, position)
        accum_loss += loss.data
        word_count += args.batchsize

        model.zerograds()
        loss.backward()
        del loss
        optimizer.update()

    print(accum_loss)

with open('ft-w2v.model', 'w') as f:
    f.write('%d %d\n' % (len(index2word), args.unit))
    w = model.embed.W.data
    for i in range(w.shape[0]):
        v = ' '.join(['%f' % v for v in w[i]])
        f.write('%s %s\n' % (index2word[i], v))
```

ソースコード 2 : fine-tuning した分散表現の整形 (reconst.py)

```
#!/usr/bin/env python
# −∗− coding: utf−8 −∗−

import collections
import pickle
import numpy as np
import sys

argvs = sys.argv
argc = len(argvs)

wdic = {}
f = open('nwjc2vec-norm1.db','r')
wdic = pickle.load(f)
f.close()

f = open('ft-w2v.model','r')
line = f.readline()
line = f.readline()
while line:
    line = line.rstrip() ## <−− chomp
    a = line.split()
    v = np.array(a[1:], dtype=np.float32)
    sm = np.sqrt(np.sum(v∗∗2))
    wdic[a[0]] = v / sm
    line = f.readline()
f.close()

fw = open('nwjc2vec-ft-norm1.db','w')
pickle.dump(wdic,fw)
fw.close()
```

ソースコード 3 : LSTM による言語モデルの構築 (eval-nsteplstm3.py)

```
#!/usr/bin/env python
# −∗− coding: utf−8 −∗−

import numpy as np
import chainer
from chainer import cuda, Function, gradient_check, Variable, optimizers, serializers,
    utils
from chainer import Link, Chain, ChainList
import chainer.functions as F
import chainer.links as L
import math
import sys
import pickle

## xp = cuda.cupy ## added

argvs = sys.argv

from collections import defaultdict
wdic = defaultdict(int)
# f = open('/home/shinnou/data/Mainichi/Mai93to99W2V-unidic.db','r')
# f = open('NWCJ-W2V.db','r')
# f = open('nwjc2vec-norm1.db','r')
f = open('nwjc2vec-ft-norm1.db','r')
wdic = pickle.load(f)
f.close()
# embed[w] −> v


vocab = {}
id2wd = {}

def load_data(filename):
    global vocab
# words = open(filename).read().replace('\n', '<eos>').strip().split()
    words = open(filename).read().replace('\n', "␣<eos>␣").strip().split()
    dataset = np.ndarray((len(words),), dtype=np.int32)
    for i, word in enumerate(words):
        if word not in vocab:
            vocab[word] = len(vocab)
        dataset[i] = vocab[word]
        id2wd[ vocab[word] ] = word
    return dataset

def load_data2(filename):
    global vocab
    words = open(filename).read().replace('\n', "␣<eos>␣").strip().split()
    dataset = np.ndarray((len(words),), dtype=np.int32)
    for i, word in enumerate(words):
        if word not in vocab:
            vocab[word] = len(vocab)
        id2wd[ vocab[word] ] = word
    pass

def softmax(x):
    return np.exp(x) / np.sum(np.exp(x), axis=0)

train_data = load_data('train2.dat')
eos_id = vocab['<eos>']

```

```
60 load_data2('test1.dat')
61
62
63 demb = 200
64 ebdw = Variable(np.zeros(len(vocab)*demb, dtype=np.float32).reshape(len(vocab),
       demb))
65 for i in range(len(vocab)):
66     if (i != eos_id):
67         vec = wdic[ id2wd[i] ]
68         for j in range(demb):
69             ebdw.data[i,j] = vec[j]
70
71 class MyLSTM(chainer.Chain):
72     def __init__(self, lay, v, k, dout):
73         super(MyLSTM, self).__init__(
74 # embed = L.EmbedID(v, k),
75             H = L.NStepLSTM(lay, k, k, dout),
76             W = L.Linear(k, v),
77         )
78     def __call__(self, hx, cx, xs, t):
79         accum_loss = None
80         xembs = []
81         for x in xs:
82             xembs += [ F.embed_id(x,ebdw) ]
83         xss = tuple(xembs)
84         hy, cy, ys = self.H(hx, cx, xss)
85         y = [self.W(item) for item in ys]
86         for i in range(len(y)):
87             tx = Variable(xp.array(t[i], dtype=xp.int32))
88             loss = F.softmax_cross_entropy(y[i], tx)
89             accum_loss = loss if accum_loss is None else accum_loss + loss
90         return accum_loss
91
92
93 #test_data = load_data('test3.dat')
94 test_data = load_data('test1.dat')
95 #test_data = train_data[0:1000]
96
97 #model = MyLSTM(2, len(vocab), demb, 0.5)
98
99 for fn in range(10):
100     print fn,
101     finename = "nsteplstm-" + str(fn) + ".model"
102     model = MyLSTM(1, len(vocab), demb, 0.5)
103     serializers.load_npz(finename, model)
104
105     ssum = 0.0
106     bc = 0
107     xs = []
108     t = []
109     s = []
110     wnum = 0
111     for pos in range(len(test_data)):
112         id = test_data[pos]
113         if (id != eos_id):
114             s += [ id ]
115         else:
116             bc += 1
117             next_s = s[1:]
118             next_s += [ eos_id ]
119             xs += [ np.asarray(s, dtype=np.int32) ]
```

```
120             t += [ np.asarray(next_s, dtype=np.int32) ]
121             s = []
122             if (bc == 5):
123     # hx = chainer.Variable(np.zeros((2, len(xs), demb), dtype=np.float32),
            volatile='on')
124     # cx = chainer.Variable(np.zeros((2, len(xs), demb), dtype=np.float32),
            volatile='on')
125                 hx = chainer.Variable(np.zeros((1, len(xs), demb), dtype=np.float32
                        ))
126                 cx = chainer.Variable(np.zeros((1, len(xs), demb), dtype=np.float32
                        ))
127 # sections = np.cumsum(np.array([len(x) for x in xs[:-1]], dtype=np.int32))
128 # xs = F.split_axis(F.embed_id(F.concat(xs, axis=0),ebdw), sections, axis=0)
129                 xembs = []
130                 for x in xs:
131                     xembs += [ F.embed_id(x,ebdw) ]
132                 xss = tuple(xembs)
133                 hy, cy, ys = model.H(hx, cx, xss)
134                 y = [ model.W(item) for item in ys]
135                 for i in range(len(y)):
136     # tx = Variable(np.array(t[i], dtype=np.int32), volatile='on')
137                     tx = Variable(np.array(t[i], dtype=np.int32))
138                     yv = y[i]
139                     for k in range(1, len(yv.data)):
140                         yv2 = softmax(yv.data[k])
141                         pi = yv2[t[i][k]]
142                         ssum -= np.log2(np.array(pi))
143                         wnum += 1
144                 xs = []
145                 t = []
146                 bc = 0
147
148     # if (pos % 100 == 0):
149     # print pos, "/", len(test_data)," finished"
150
151     print math.pow(2, ssum / wnum)
```