卒 業 研 究 報 告 題 目

WordNetによる概念距離を利用したWord2vecモデルの学習

Training Word2vec model using concept distance from WordNet

指 導 教 員　　出口 利憲 教授

岐 阜 工 業 高 等 専 門 学 校　電気情報工学科

2016E24 加藤 笙

令 和 3 年（ 2 0 2 1 年 ）　　2 月 1 5 日 提 出

Abstract

In this study, I learn how to build own Word2vec model and how to train it, and compare own model with an existing model to see how you can create a better model in terms of Spearman's rank correlation coefficient. To train the model, skip-gram is used with only nouns extracted from Japanese WordNet, and programming to find the distance between words developed in this laboratory by WordNet. As a result of comparing the model developed in this way with the existing models, it was discovered that skip-gram models better than CBOW and better to create based on large data with a large number of words. In addition, the developed model showed a high quality, but it became insufficient because it was a model made only of nouns. In order to improve this, it is necessary to learn other part of speech by additional learning.

目 次

Abstract

第 1 章 序論 **1**

第 2 章 基礎知識 **2**

2.1 自然言語処理 2

2.2 ニューラルネットワーク 2

2.3 形態素解析 4

2.3.1 形態素解析 4

2.3.2 MeCab 6

第 3 章 実験で使用した技術 **8**

3.1 日本語 WordNet 8

3.1.1 WordNet とは 8

3.1.2 日本語 WordNet とは 8

3.2 WordNet での単語距離計算方法 8

3.3 Keras 9

3.4 PlaidML 10

3.5 Word2vec 11

3.5.1 ONE-HOT ベクトル 11

3.5.2 CBOW 12

3.6 Skip-Gram 13

3.7 Skip-Gram with Negative Sampling(SGNS) 15

第 4 章 実験 **17**

4.1 実験内容 17

4.2 実験準備 17

4.2.1 MeCab の動作確認 17

4.2.2 青空文庫による簡単なモデルを作成 18

4.2.3 学習データの作成 19

4.2.4 モデル作成に必要な辞書を作成 19

4.2.5 Keras でのモデル作成 19

4.2.6 Keras で作ったモデルを Word2vec に変換 21
4.2.7 世に出ているモデルについて 21
4.2.8 それぞれの日本語 Word2vec モデルの評価 22
第 5 章 実験結果とその考察 **24**
第 6 章 結論 **30**
参考文献 **31**

第１章　序論

日本には、およそ 2000 年前の弥生時代から使い続けている日本語が存在している。時代を重ねるにつれて日本語は進化し文字となり、それぞれの時代の生活に合った言葉を新しく生み出していった。そして、今この時にも新しい言葉が次々と生まれている。日本語の言葉の１つ１つには意味があり、必ず類似性の高い言葉が存在する。例をあげると「熱」という言葉には、太陽や夏、火などの類似性の高い言葉が存在する。これらの言葉の関係をベクトルに表し機械に学習させることによって、それぞれの言葉の距離と方向から類似性を求め言葉の意味を解析することが可能になった。そしてこれからも、自然言語処理の技術はさらに進歩していくだろう。

本研究では、数多く存在する日本語をベクトルで表現し、それぞれの言葉のベクトルの距離から言葉の近さを求めることが可能な Word2vec モデルの作成を行った。また、現在テキストマイニングなどで使用されている様々な Word2vec モデルと作成したモデルではどのような違いが出るのか比較し実験を行った。

第2章　基礎知識

2.1　自然言語処理

自然言語処理とは、私達が普段使う人間の言葉 (自然言語) をコンピュータに理解させるための技術である。私たちの言葉をコンピュータに理解させることはとても難しい問題であり、重大なテーマでもある。実際にこの自然言語処理の技術により私達の生活は大きく変わった。Web 検索や機械翻訳、音声アシスタントなど、世の中に大きな影響を与えた技術の根幹には、自然言語処理の技術が使われている。このように私達の生活で活躍している自然言語処理の技術だが、ディープラーニングにおいても極めて重要な位置を占めている。実際に Google の機械翻訳でもディープラーニング手法によって大きな進化を遂げている。[1)]

2.2　ニューラルネットワーク

ニューラルネットワークとは、人間の脳内にある神経細胞ニューロンとその繋がりの神経回路網を参考に数式的なモデルで機械によって表現した技術である。ニューラルネットワークは主に、入力層、出力層、隠れ層 (Figure 2.1) から構成される。入力データを受け取る層が入力層、予測値を返す層が出力層、入力層と出力層に挟まれたその他の層が中間層である。中間層と出力層では、それぞれ活性化関数という特殊な変換する関数を持っている。また、各層にはまるで神経細胞のように、前の層から受け取った情報を処理して次の層へ流すユニット（図中の○）があり、ユニット同士が結んでいるのがエッジ (図中の→）で、それぞれ重みという情報を持っている。[2),3)]

活性化関数には、しきい値関数、シグモイド関数、Rectifier 関数などが使われている。目的や計算の容易さなどで活性化関数は選ばれる。次にそれぞれの関数について説明する。[4)]

- しきい値関数

  しきい値関数は一定値を超えると 1 になる 0 か 1 で表すことができる関数である。式は (2.1) で表せられる。

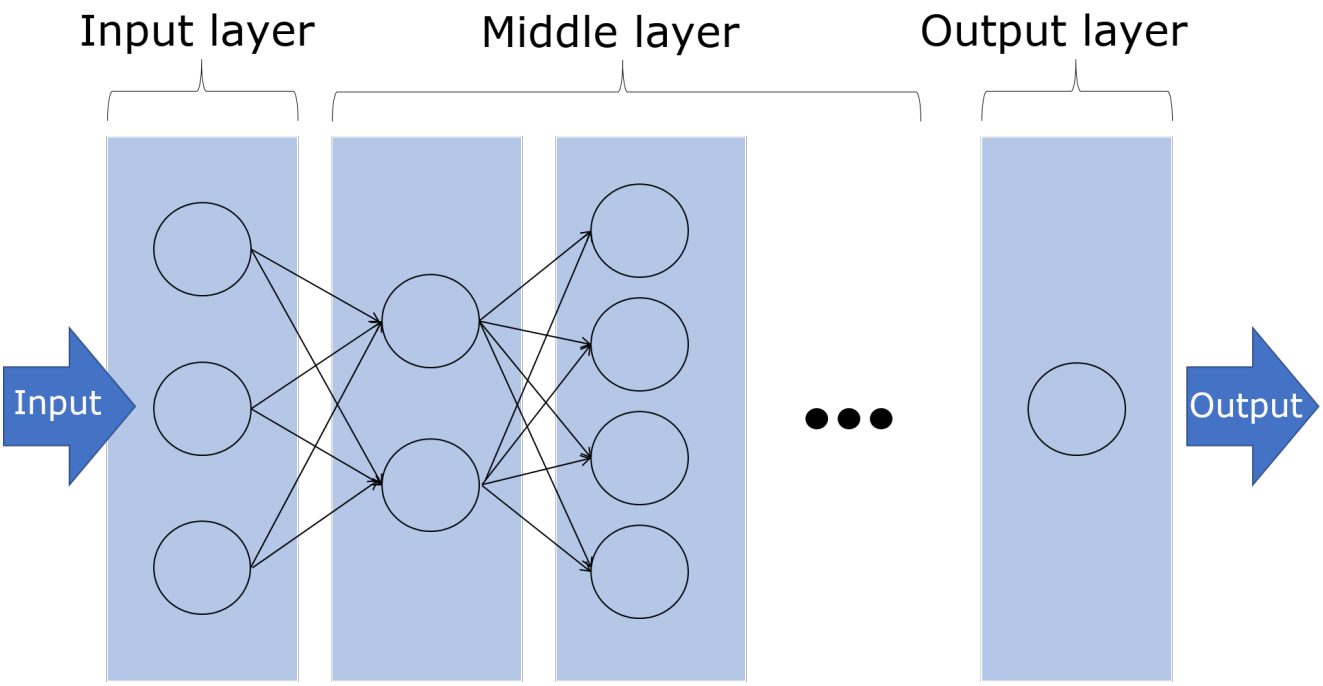

Figure 2.1　Structure of neural network.

$$f(x) = \begin{cases} 1 & (x > 0) \\ 0 & (x \leq 0) \end{cases} \tag{2.1}$$

- シグモイド関数

  シグモイド関数は 0 から 1 まで滑らかに変わる関数である。式は (2.2) で表せられる。

$$f(x) = \frac{1}{1 + \exp(-x)} \tag{2.2}$$

- Rectifier 関数

  Rectifier 関数は負の時 0 に正の時は入力値をそのまま返す関数である。式は (2.3) で表せられる。

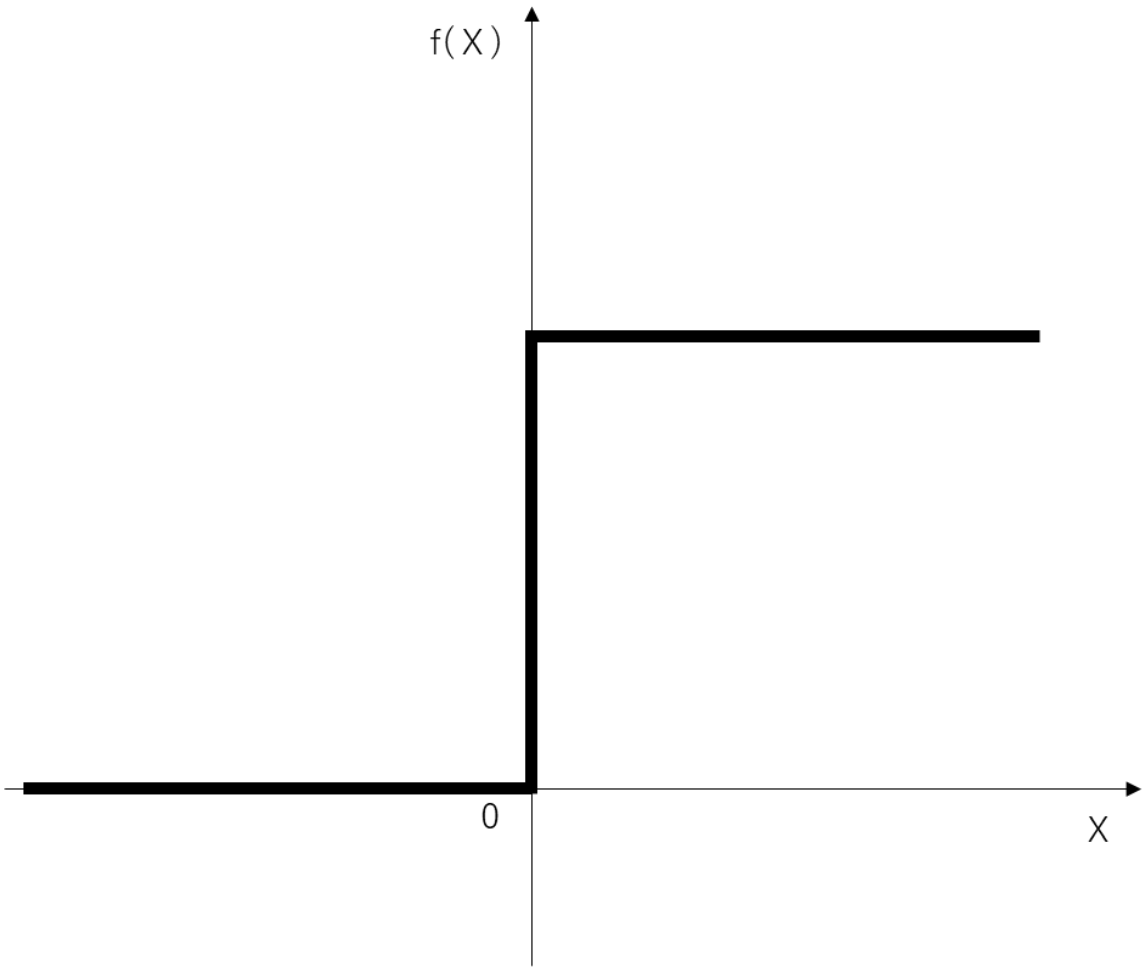

Figure 2.2　Threshold function.

$$f(x) = \begin{cases} x & (x > 0) \\ 0 & (x \leq 0) \end{cases} \tag{2.3}$$

2.3　形態素解析

2.3.1　形態素解析

形態素解析とは、自然言語処理の一部で、自然言語で書かれた文を言語上で意味をもつ意味を持つ最小単位 (＝形態素) に分け、それぞれの品詞や変化などを判別することである。例えば「庭には二羽ニワトリがいる」という文章を、

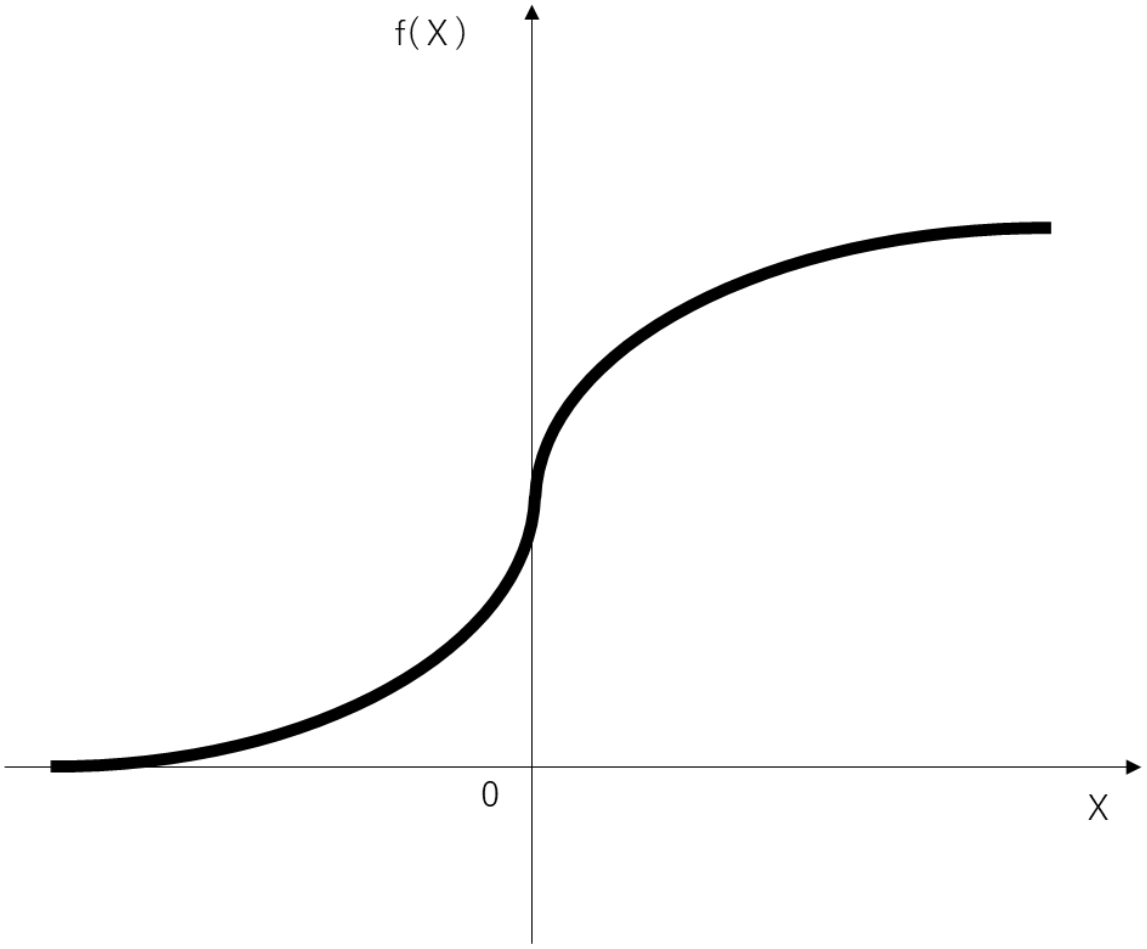

Figure 2.3　Sigmoid function.

庭（名詞）/に（助詞）/は（助詞）/二（数詞）/羽（助数詞）/ニワトリ（名詞）/が（助詞）/いる（動詞）

のように形態素に分解し、意味を割り出す。また、大量のテキストの文法上の属性を明らかにすることで、そのテキストの意味理解に使うことができる。形態素解析は、分かち書き、品詞分け、原型付与の三つのツールがある。それぞれの三つの機能のそれぞれの定義は、分かち書きの場合、文章を形態素で分ける。品詞分けの場合、名詞や動詞などに分類する。原型付与の場合、単語の基本形 (例:食べた→たべる、た) をだすことである。[5)]

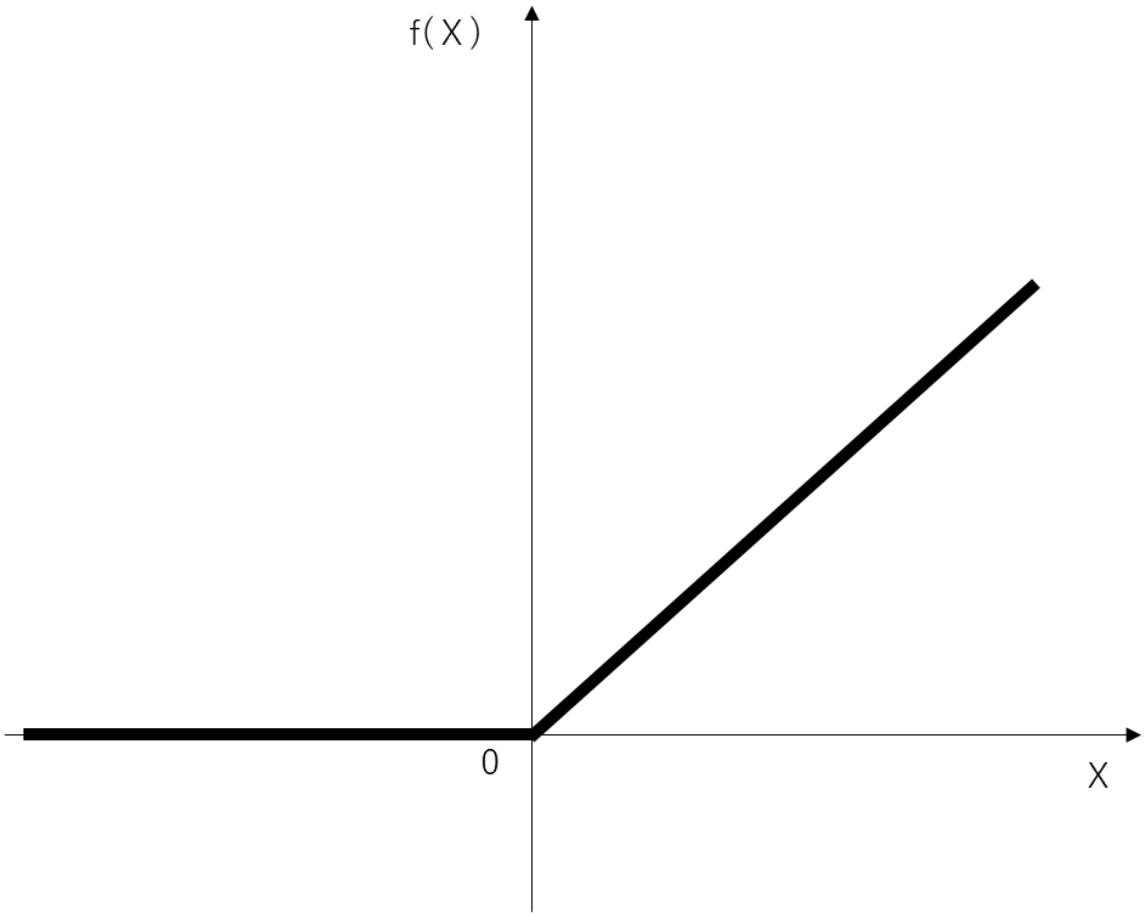

Figure 2.4　Rectifier function.

2.3.2　MeCab

MeCab とは、形態素解析のエンジンであり、辞書やコーパスに依存しない汎用的な設計なため、連結可能な辞書も、IPAdic、NAIST jdic、UniDic など多数あり、追加学習も可能である。また、他の日本語形態素解析システムの Chasen や KAKASI に比べ高速でもあり、各種スクリプト言語 (perl/ruby/python/java/C#) で使用することが可能である。下図に実際に MeCab で「すももももももものうち」という文を分かち書きをした結果を示す。[6)]

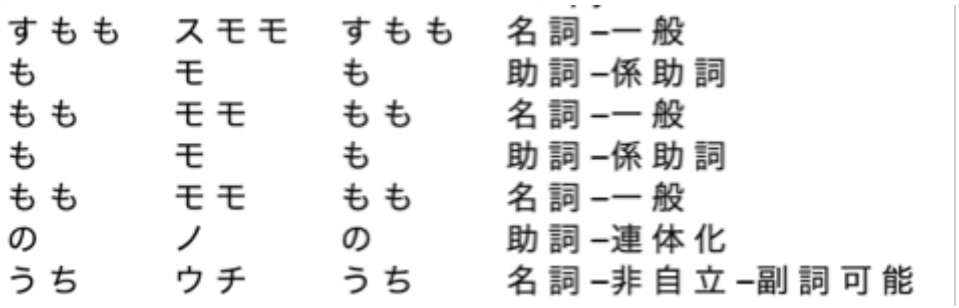

Figure 2.5　Word separating by MeCab.

第3章　実験で使用した技術

3.1　日本語 WordNet

3.1.1　WordNet とは

WordNet とは、ブリンストン大学で研究されている英語の概念辞書で自然言語処理の分野において最も有名である。英文 15 万 5 千語を収容する、同義語や上位互換、下位概念等の整理された概念辞書である。たとえば dog の意味として 8 つ (名詞 7 つ、動詞 1 つ) 定義されている。動物、としての犬のほか、英語で使われる「面白くない女性」や「男のインフォーマルな表現」などの意味が挙げられている。さらに、それぞれの意味概念単位について、上位概念 (例:イヌ科の動物、家畜)、下位概念 (例:コーギー、ダルメシアン、チワワなどの様々な犬種)、部分関係、論理的含意などの関係でお互いにリンクされた構造になっている。この WordNet を使えば、類義語を取得したり、単語ネットワークを利用したりすることができる。また、単語ネットワークを使うと単語間の類似度を算出することが可能である。[1),7)]

3.1.2　日本語 WordNet とは

日本語 WordNet とは、NanYang 大学の F.Bond らが英語の WordNet3.0 に日本語が対応する訳を追加する形、つまり、概念構造は英語で作られたそのままを使い、日本語の語と概念とのマップを追加して作られた日本語 WordNet が提供されている。現在はほかの多くの言葉を含めて Open Multilingual WordNet に公開されている。さらに、NLTK の WordNet アクセスのための WordNetCorpusReader がこの Open Multilingual WordNet に対応する形になっているため、簡単な入力で日本語 WordNet にアクセスすることができる。[7)]

3.2　WordNet での単語距離計算方法

WordNet での距離法と段数法について、下図を用いて説明する。下図での A,B は対象概念であり、a,b は対象概念のカテゴリである。また、対象概念を A,B としたときの距離法, 段数法の類似度を RD(A,B),RL(A,B) とする。なお、対象概念は、そのカテゴリの下位に存在すると考え、その分も 1 段として計算を行う。[8)]

- 距離法　距離法の式は (3.1) で表し類似度を算出する。

$$R_D(A,B) = \frac{1}{D_{AB}+1} \tag{3.1}$$

ここで、D_{AB} は図中の矢印のように概念 A,B 間のシソーラス (言葉を同義語や意味上の類似関係、包含関係などによって分類した辞書) 上の距離、つまり、カテゴリ間の枝の数のことである。図中の例では、距離が 5 であるので $R_D(A,B)$ は 1/6 となる。

- 段数法

段数法の式は (3.2) で表し類似度を算出する。本実験では段数法を使用する。

$$R_L(A,B) = \frac{C_{AB} \times 2}{L_A + L_B} \tag{3.2}$$

ここで、L_A, L_B は根カテゴリ (図中のカテゴリ T) を 1 段とし、そこから、1 つの下位カテゴリになるごとに 1 を加算してカウントした概念 A, B のシソーラス上での段数である。また、C_{AB} は対象概念 A, B の共通の上位ノード (図中の C) の段数である。下図の例では、L_A は 5、L_B は 4、C_{AB} が 2 なので、$R_L(A,B)$ は 4/9 となる。

3.3 Keras

Keras とは、Python のディープラーニングフレームワークであり、ほぼあらゆる種類のディープラーニングモデルを定義して訓練するための便利な手段を提供している。当初は、実験を素早く行えるようにすることを目的に研究者のために開発された。次に Keras の主な特徴を取り上げる。

- CPU でも GPU でも同じコードをシームレスに実行できる。
- ディープラーニングモデルのプロトコルを簡単にすばやく作成できユーザーフレンドリな API を備えている。
- 畳み込みネットワーク、リカレントネットワーク、およびそれらの組み合わせを組み込みでサポートしている。
- 複数入力/複数出力モデル、層の共有、モデルの共有など、任意のネットワークアーキテクチャをサポートしている。つまり、敵対的生成ネットワークからニューラルチューリングマシンまで、ほぼ全てのディープラーニングモデルの構築に適している。

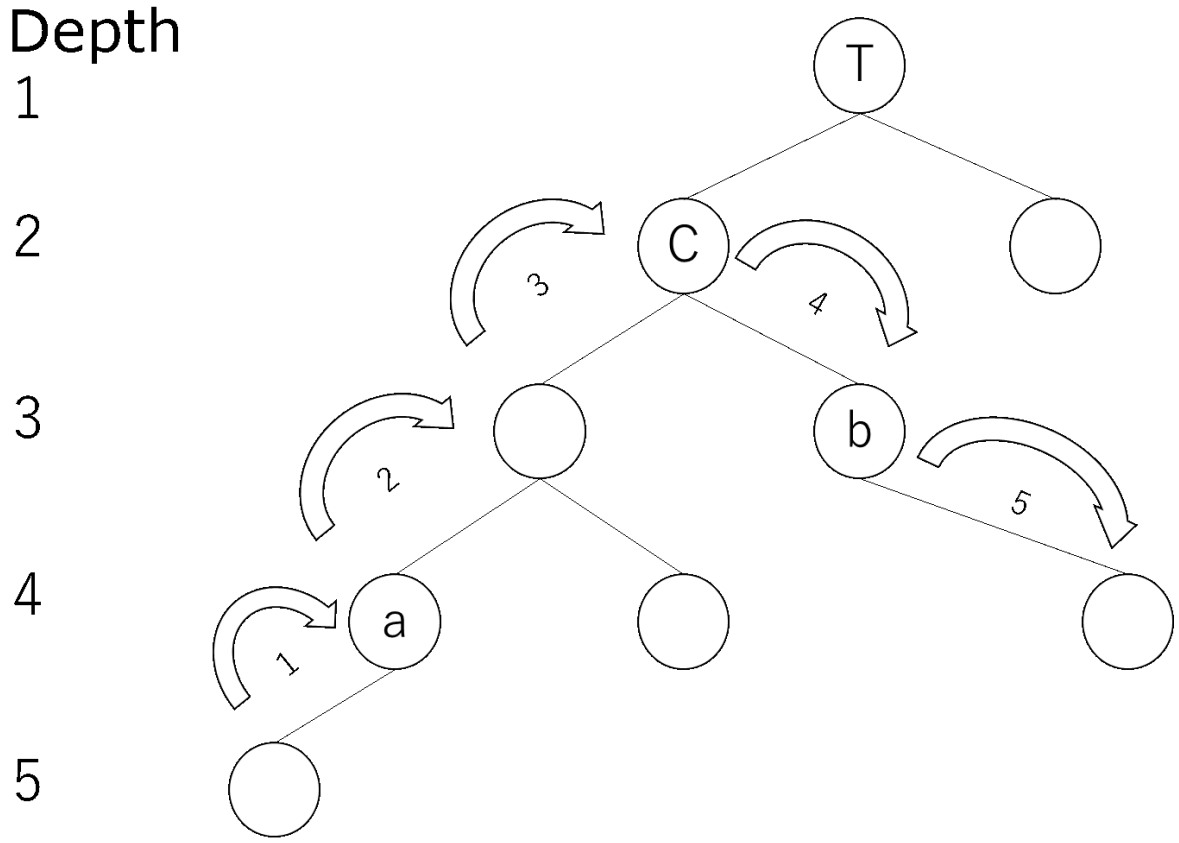

Figure 3.1　Similarity calculation by distance and depth methods.

そして、Keras は MIT ライセンスで配布されているため、営利目的のプロジェクトでも無償で利用することが可能である。また Python2.7 から 3.8 までのバージョンと互換性があり幅広く対応している。

Keras はモデルレベルのライブラリであり、テンソルの操作や微分といった演算を直接扱うのではなく、Keras のバックエンドエンジンとして専用の最適化されたテンソルライブラリを利用する。また、テンソルライブラリをどれか 1 つ選択して Keras の実行をそのライブラリに結びつけるのではなく、問題をモジュール方式で処理するため、数種類のバックエンドエンジンを Keras にシームレスに接続することが可能である。現在では、tf.keras となって TensorFlow に取り組まれている。[9),10)]

3.4　PlaidML

PlaidML とは、NVIDIA 以外の GPU を機械学習に使用できるようにした、Python 向けのフレームワークである。Intel が中心となって開発されており、Intel 製 GPU、AMD

製 GPU、OpenCL に対応している。NVIDIA にも対応しているため、PlaidML を使って書いたプログラムは Mac, Windows, Linux のどの OS でも、GPU を使った機械学習をすることができる。クラウドを使わずにクライアントパソコンで機械学習をする一つのメリットとして、途中経過がその場で確認できるということがある。クラウド上で機械学習をする場合は、学習処理に影響を与えないようにファイルをコピーする必要があり、手間がかかる。学習する画像の中に不適切なデータが含まれていないかどうか、クライアントパソコンであれば数百枚を一覧表示して目視で確認することができる。動画、音声のようなメディアデータの機械学習処理は、クライアントパソコンで前処理したデータを数十分から 1 時間かけて学習し、ある程度成果の目算をつけ、プログラム・データ・モデルに自信を持ってから、クラウドで一括処理をすると両者のいいとこ取りができる。citebib11

3.5 Word2vec

Word2vec とは、大量のテキストデータを解析し、各単語の意味をベクトル表現化する手法である。単語をベクトル化することで、単語同士の意味の近さを計算することと、単語同士の意味を足したり引いたりすることが可能になります。例えば、Word2vec により「群馬」「茨城」「温泉」「海」という言葉を次のようにベクトル化することが可能である。

- 群馬 (0.6 , 0.3)
- 茨城 (0.7 , 0.5)
- 温泉 (-0.1 , 0.0)
- 海 (0 , 0.1)

このベクトル表現から次のこと考えられる。

- 群馬と茨城の距離を計算すると、近いところにあるので２つの意味は近い
- 「群馬」 -「温泉」 +「海」 =「茨城」　になる

このように単語ベクトルが行えると単語の関係が可視化できるため、そこからいろいろと分析することが可能になる。[3),4)]

3.5.1 ONE-HOT ベクトル

ONE-HOT ベクトルとは、$(0, 1, 0, 0, 0, 0)$ のように 1 つの成分が残りの成分が全て 0 であるようなベクトルのことである。例えば犬を 1、猫を 2、ウサギを 3 の数字に変換する

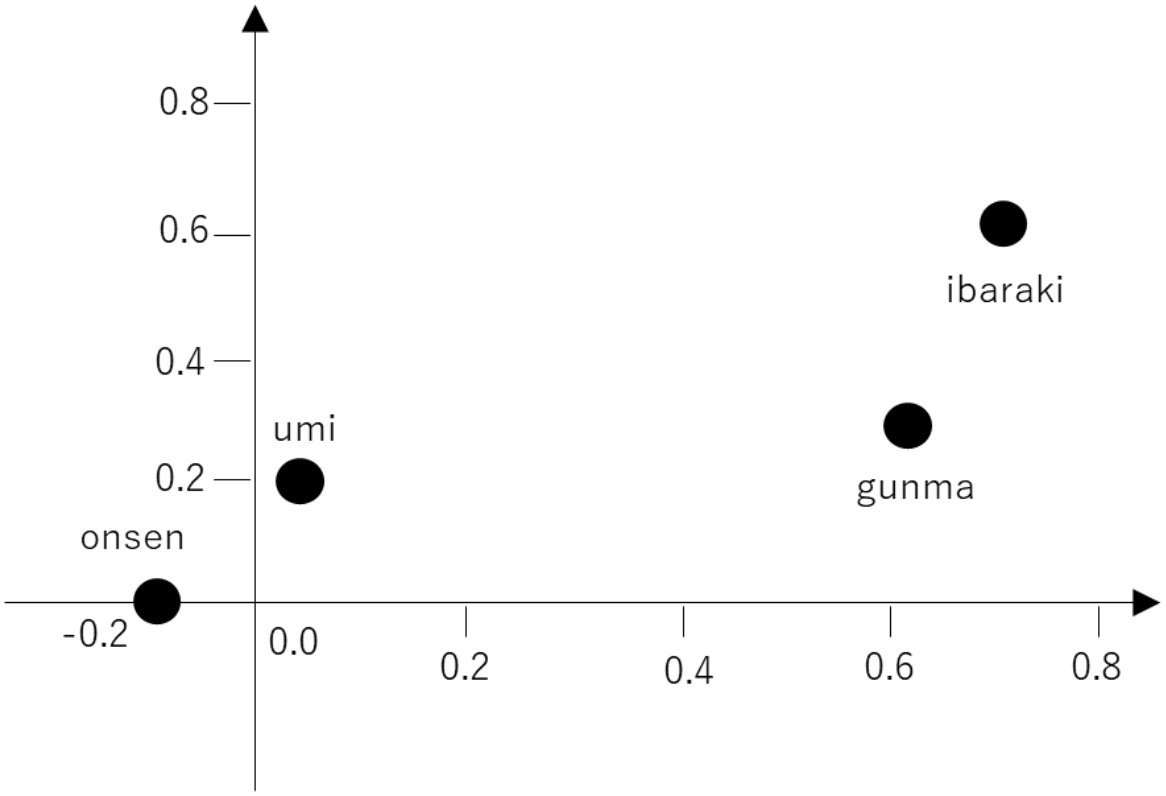

Figure 3.2　Vector representaion.

方法があるとする場合に、これを線形回帰などで分析を行う際に猫はウサギと犬の中間などと、解釈されてしまう可能性がある。そこで犬を $(1,0,0)$、猫を $(0,1,0)$、ウサギを $(0,0,1)$ のようにそれぞれを三次元の OEN-HOT ベクトルで表現すれば、変数の全ての値を平等に扱えることが可能である。[12)]

3.5.2　CBOW

Word2vec では 2 つのモデル (CBOW と Skip-Gram) が提案されておりその内の 1 つが CBOW である。CBOW とは、下図のように周辺の単語 (コンテキスト) から中央の単語 (ターゲット) を、予測することを目的としたニューラルネットワークである。CBOW への入力はコンテキストのみであり、このコンテキストは ['you', 'goodbye'] のような単語のリストで表される。これを ONE-HOT 表現に変換することで、CBOW モデルが処理できるように調整する。Figure 3.4 が、CBOW モデルのネットワークである。入力層が 2 つあり、中間層を経て、出力層へとたどり着く。入力層から中間層の重みは、同じ全結合層 (Win) によって行われ、中間層から出力層の変換は、別の全結合層 (Wout) によって

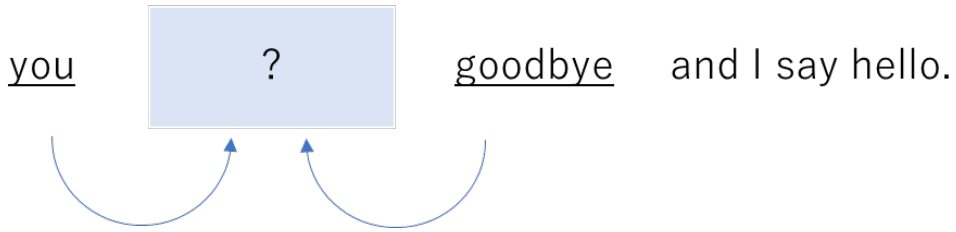

Figure 3.3　CBOW model.

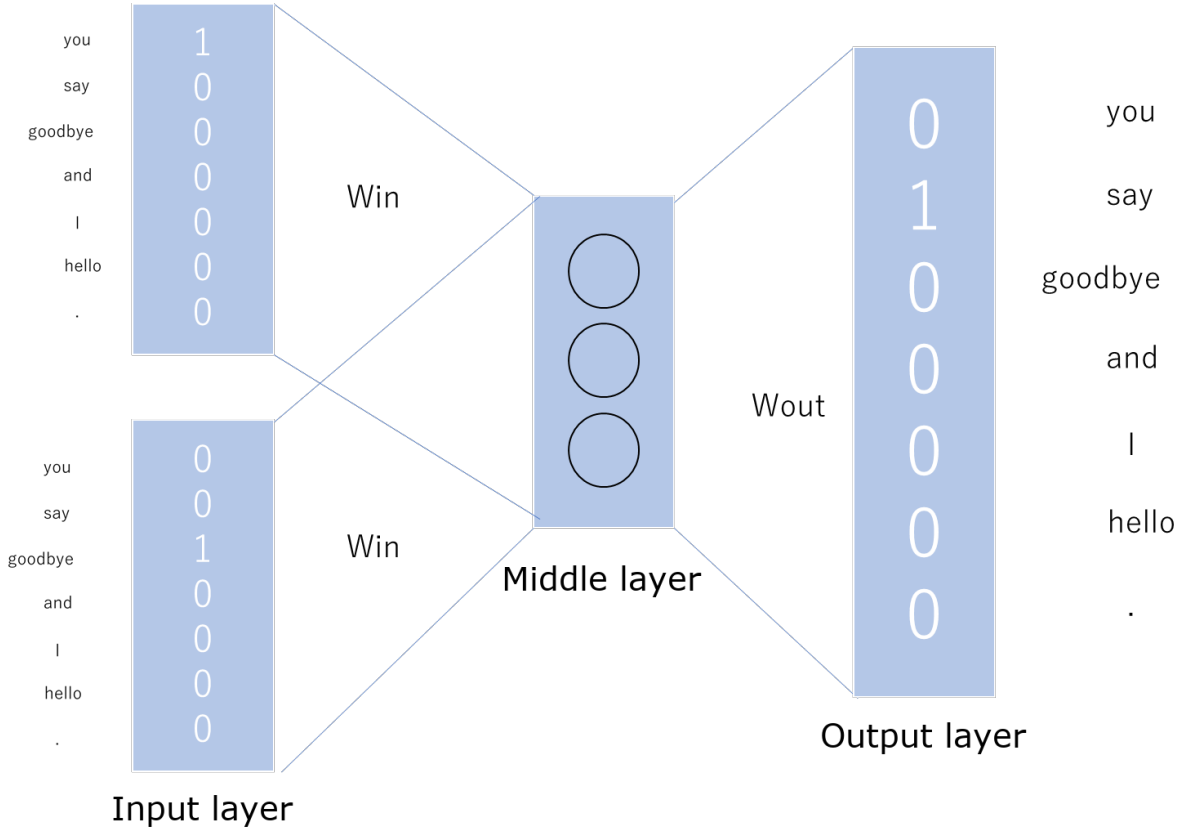

Figure 3.4　Network structure of CBOW model.

行われる。[1),4)]

また CBOW モデルのニューラルネットワークは Figure 3.4 のようになる。

3.6　Skip-Gram

Skip-Gram とは、CBOW とは反対にひとつの中央の単語 (ターゲット) から、その周囲の複数ある単語 (コンテキスト) を予測するニューラルネットワークであり、CBOW モデ

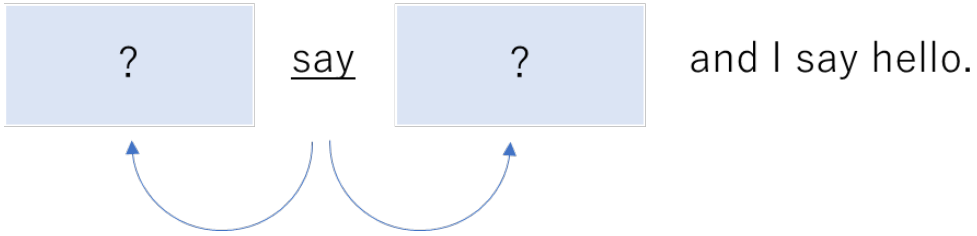

Figure 3.5　Skip-Gram model.

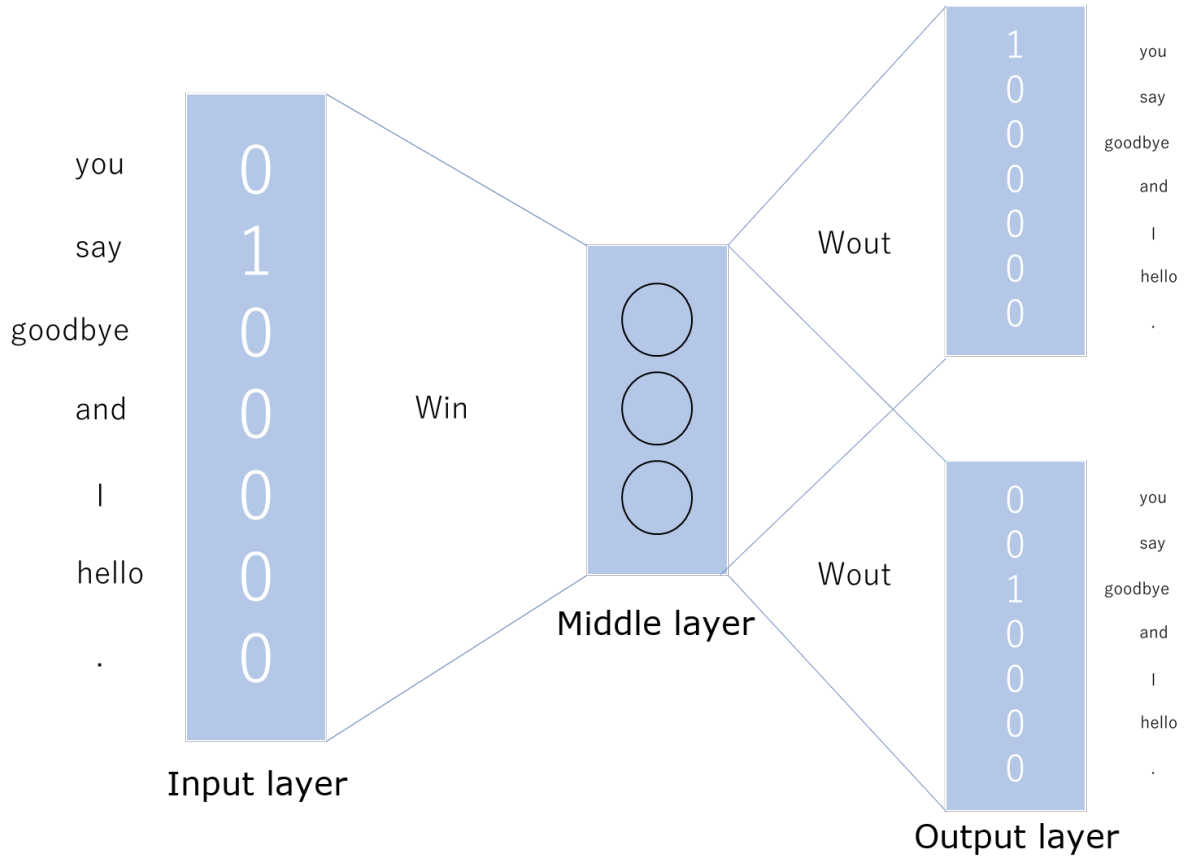

Figure 3.6　Network structure of Skip-Gram model.

ル (Skip-Gram とは反対に周囲の単語から中央の単語を予測するモデル) と比較すると、CBOW モデルならば容易に目的の単語を予測することが可能であるが、Skip-Gram モデルの場合は様々な候補が考えられる。そういった点で Skip-Gram の方がタフな問題に鍛えられることができるため、より優れた単語の分散表現を得ることが可能である。

また Skip-Gram モデルのネットワーク構成は Figure 3.6 のようになる。

Figure 3.6 に示すように、Skip-Gram モデルの入力層はひとつのみである。そして、出

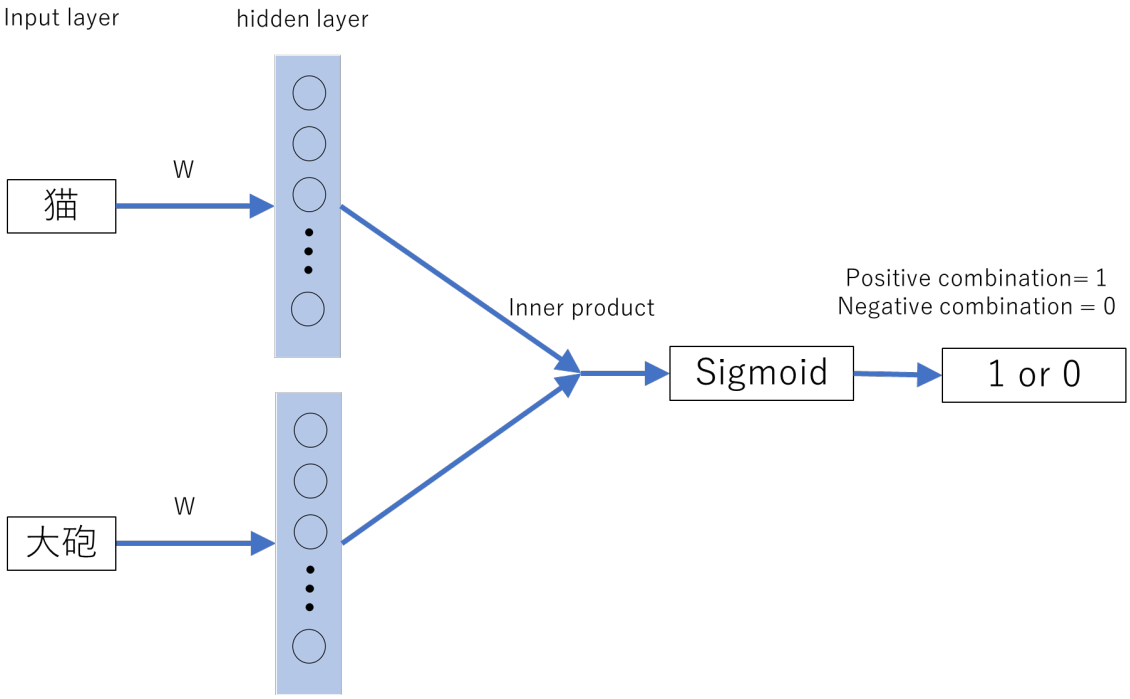

Figure 3.7　Skip-Gram with Negative Sampling(SGNS)

力層はコンテキストの数だけ存在する。そのため、それぞれの出力層では個別に損失を求め、それらを足し合わせたものを最終的な損失としている。[1),4)]

3.7　Skip-Gram with Negative Sampling(SGNS)

通常の Skip-Gram では、単語数の多値分類になるので、単語数が増えれば出力が 0 に近づける学習が非常に多く、効率が悪い。例えば、周りの 10 単語を学習するにしても出力が 1 となるのはその 10 単語だけで、それ以外の単語は全て 0 として学習される。そこで、周辺単語以外の単語をサンプリングして、二値分類にしてしまうのが SGNS である。

本実験では、SGNS を使用している。正の組み合わせだけでなく負の組み合わせを用いることで、負例をモデルに学習させ、通常の Skip-Gram より高速に学習することが可能となる。

本実験では使用していないが、周辺単語以外の単語のサンプリング方法は、単語頻度によって重みをつける方法である。ただし、そのまま頻度を使用すると頻出しない単語がほとんど出てこないため、単語の個数に 3/4 乗するとよいと言われている。つまり、それぞれの単語 w_i が選ばれる確率は次の式 (3.3) になる。

$$P(w_i) = \frac{|w_i|^{3/4}}{\sum_{w_j} |w_j|^{3/4}} \tag{3.3}$$

これにより、通常の Skip-Gram より高速に学習することができる。[14)]

第4章　実験

4.1　実験内容

日本語 WordNet から名詞だけを抜き出した単語群を元に Word2vec での Skip-Gram による学習を行い、モデルを作成し世に出ているモデル 3 種類の chiVe、エンティティベクトル (WikiEntVec)、シロヤギ (Japanese Word2vec Model Builder) とで日本語の評価データセット (JWSAN) を利用して、それぞれの日本語 Word2vec の評価を行った。

4.2　実験準備

それぞれの日本語 Word2vec モデルを評価する前に、以下の操作を行った。

- MeCab の動作確認
- 青空文庫による簡単なモデルを作成
- 学習データの作成
- モデル作成に必要な辞書を作成
- Keras でのモデル作成
- Keras で作ったモデルを Word2vec に変換
- 世に出ているモデルについてそれぞれの日本語 Word2vec モデルの評価

4.2.1　MeCab の動作確認

MeCab をダウンロードし、どのように動き、コマンドを打ち込めばどのような出力結果が得られるか調べた。MeCab を起動し簡単に文字を「これはテストです。」と打ち込んで分かち書きをしてみた。すると以下の Figure 4.1 の出力が得られた。

また、MeCab には出力フォーマットを変更する-O オプションを使用することができ、

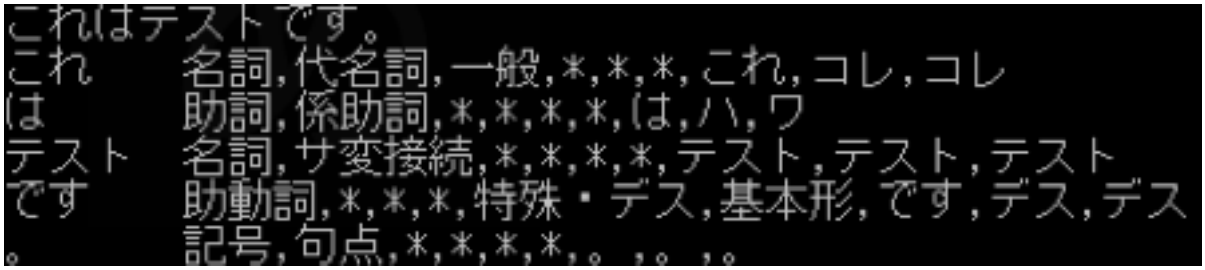

Figure 4.1　Output from Mecab

```
  results = model.most_similar(positive="父", topn=10)
繰り返す          0.7443997859954834
とか      0.737993597984314
結婚      0.7214841842651367
以後      0.7190877795219421
覚悟      0.705105185508728
きっと    0.6902227997779846
つもり    0.6901242733001709
様子      0.6876108646392822
自信      0.6840754151344299
注意      0.682325005531311
```

Figure 4.2　Words which are similar to 「父」.

-Oyomi では読みを付与し、-Ochasen では ChaSen の形式で出力し、-Odump では全情報を出力させることができた。とても便利なオプションではあるが今回のモデル作成ではシンプルな分かち書きを行う-Owakati だけを使用して実験を行った。

4.2.2　青空文庫による簡単なモデルを作成

a 簡単なモデルを作成するためにインターネットの電子図書館である青空文庫 (https://www.aozora.gr.jp/) から、公開中の全ての作品をダウンロードした。ダウンロードした文章から本文以外の不要な文章をプログラミングで削除し、それぞれのテキストデータを 1 つのテキストデータにまとめるために cat コマンドを使用し、テキストデータを 1 つにまとめた。まとめたテキストデータをそのままの状態でモデル作成することは難しいため、MeCab よりまとめたテキストデータに対して分かち書きを行った後に、Word2vec を使ってモデル学習させた。作成したモデルに対して「父」にベクトルが近い言葉 Top10 を調べてみると、以下の Figure14 のように結果が出力された。

Figure 4.2 の結果をみると、「繰り返す」、「とか」「結婚」の順に類似度がつけられているが、それほど父に意味が近い言葉を出力させることが出来なかった。しかし、モデルを作成し実際にモデルを利用して類似度を求めることが出来ることが分かった。

4.2.3　学習データの作成

本実験での学習データは、日本語 WordNet から名詞のみを抜いた単語群に、研究室で開発された日本語 WordNet で単語間の距離を求めることが出来るプログラム (段数法) を利用して、一定の単語間の距離を求めたデータを 1 つのテキストファイルにまとめて学習データを作成した。またこのプログラムでは、単語の距離範囲を指定してデータを求めることが、出来るため極めて距離が近い 0〜0.1 の間のデータを指定してデータを取得した。また、本実験の学習データにはネガティブサンプリングを取り入れるため、単語間の距離が 0〜0.1 の正のデータに対して、単語間の距離が 0.6〜0.7 の負のデータを 2 対 1 の割合で繋いで学習データを作成した。

4.2.4　モデル作成に必要な辞書を作成

実験で使用する単語に数字をペアにして並べる辞書には、日本語 WordNet から名詞を抜き出した単語群を使用して作成した。辞書をつくるために単語群と同じ数の空の配列をそれぞれ用意した。単語群のデータをひとつひとつ配列に追加し、データの配列の長さを len コマンドで取り出し、その長さデータを配列に追加し { ' 頭金': 0, ' 大砲': 1, ' スチーム': 2, ' マーティニ': 3, … } のような単語に数字をペアにした辞書を作成した。また、Word2vec でのモデル作成時に単語と番号が逆に置かれた辞書が必要なため逆辞書を作成した。次に、上記で作成した単語と数字が保存されている辞書に単語の類似度を集めたデータの２つの単語をあてはめ該当する２つの数字と言葉の類似度を追加した辞書を作成した。下記にプログラム内容を示す。

4.2.5　Keras でのモデル作成

Keras でのモデル作成を行うには、Sequential モデルを用いる方法と FunctionalAPI を用いる２つの方法がある。本実験では、Sequential モデルより柔軟にモデルを作れる FunctionalAPI でモデルを作成していく。はじめに Input クラスのインスタンスを作成し入力層を辞書の配列 [1,2] の２つを定義した。次にノードの数と次元数を決めてレイヤーを追加した。次に追加したレイヤーについて説明する。

- Embedding

  Embedding レイヤーは、入力に対して、単語をベクトル表現にして返すレイヤーである。

```
def build_dataset(words):
    """Process raw inputs into a dataset."""
    count = words
    dictionary = dict()
    for word in count:
        dictionary[word] = len(dictionary)
    reversed_dictionary = dict(zip(dictionary.values(), dictionary.keys()))
    return len(words), dictionary, reversed_dictionary

def collect_data():
    filename = "sori.txt"
    with open(filename, "r" , encoding="utf-8_sig") as f:
        file = f.read().splitlines()
        f.close()
    vocabulary = file
    count, dictionary, reverse_dictionary = build_dataset(vocabulary)

    filename1 = "output.txt"
    with open(filename1, "r" , encoding="utf-8_sig") as f:
        file1 = f.read().splitlines()
        file1 = [re.split("[ ]",file) for file in file1]
        f.close()
    vocabulary1 = file1
    vocabulary1 = [[dictionary[i[0]],dictionary[i[1]],1 if (float(i[2])<=0.1) else 0] for i in vocabulary1]
    del vocabulary
    return count, vocabulary1
```

Figure 4.3　Program for making dictionary.

- Reshape

  Reshape は、データの順番と総数をそのままに、配列の形を変えるレイヤーである。本実験では配列の形を 1 次元に変えている。

- Dot

  Dot は、２つのデータから dot 積を計算するレイヤーである。このレイヤーには引数が axes,normalize,**kwargs の３つあり、axes とは、0 の場合に axis(一次元) を使い、1 の場合には axes(二次元) を使うか選択する引数である。Normalize とは dot 積を取る前に正規化するかどうか決められる引数である。**kwargs は、 標準的なレイヤーのキーワード引数である。本実験では使用していない。

- Dence

  Dence は、次のノード数を決め入力値に活性化関数を適用させてから出力する引数である。ニューラルネットワークを決め、モデルをコンパイルしモデルを作成した。

```
word_target, word_context, labels = zip(*vocabulary1)
word_target = np.array(word_target, dtype="int32")
word_context = np.array(word_context, dtype="int32")
labels = np.array(labels, dtype="int32")
input_target = Input((1,))
input_context = Input((1,))

embedding = Embedding(vocab_size, vector_dim, input_length=1, name='embedding')
target = embedding(input_target)
target = Reshape((vector_dim, 1))(target)
context = embedding(input_context)
context = Reshape((vector_dim, 1))(context)

# setup a cosine similarity operation which will be output in a secondary model
similarity = dot([target, context], axes=0 , normalize = True)

# now perform the dot product operation to get a similarity measure
dot_product = dot([target, context], axes=1 , normalize = False)
dot_product = Reshape((1,))(dot_product)
# add the sigmoid output layer
output = Dense(1, activation='sigmoid')(dot_product)
# create the primary training model
model = Model(input=[input_target, input_context], output=output)
model.compile(loss='binary_crossentropy', optimizer='rmsprop')

model.fit(x = [word_target,word_context] , y = labels , epochs = 1 , shuffle = true)
model.save_weights('kato91.h5')
```

Figure 4.4　Program for making model on Keras.

4.2.6　Keras で作ったモデルを Word2vec に変換

Word2vec のモデルは Keras のモデルとは構造が違う。Word2vec のモデルは、ニューラルネットワークより求めた、隠れ層の重みを単語ベクトルとしたデータの塊であるが、Keras のモデルはニューラルネットワークの形状である。そのため、Keras で求めた隠れ層の重みを抜き出し、Word2vec モデルの katomodel を作成した。

4.2.7　世に出ているモデルについて

- chiVe

  Sudachi の開発元である Works applications が公開され、国立国語研究所の日本語ウェブコーパス（NWJC）で学習されたモデルである。モデルは Skip-Gram で

```
w = embedding.get_weights()[0].tolist()
with open("katomode9l.txt","w") as f:
    f.write('%d %d¥n'%(len(w),vector_dim))
    for i,x in enumerate(w):
        f.writelines(reversed_dictionary[i]+''+''.join([str(j) for j in x])+'¥n')
```

Figure 4.5　Program for converting into Word2vec model.

作成されており、次元数は 300 であり、データの分かち書きには Sudachi を使用している。[13)]

- エンティティベクトル (WikiEntVec)

  Wikipedia でモデルを学習させ、モデルはおそらく Skip-Gram で作成されており、次元数は 100,200,300(本実験では 100 を使用) であり、分かち書きには MeCab を使用している。

- シロヤギ (Japanese Word2Vec Model Builder)

  Wikipedia でモデルを学習させ、モデルはおそらく CBOW で作成されており、次元数は 50 であり、分かち書きには MeCab を使用している。

4.2.8　それぞれの日本語 Word2vec モデルの評価

上記で作成した日本語 Word2vec モデルと、世に出ているモデル 3 種類の chiVe、エンティティベクトル (WikiEntVec)、シロヤギ (Japanese Word2Vec Model Builder) とで形容詞、動詞、名詞どうしのペア (1400 組と 2145 組) に対して類似度と関連度の両方を収録している日本語単語類似度・関連度データセット (JWSAN) を利用して、日本語 Word2vec の評価を行った。評価プログラムでは、JWSAN における類似度と Word2vec の類似度を比較するため、両者のスピアマンの順位相関係数を求めている。スピアマンの順位相関係数とは、データを一定の幅で振り分けることで一定間隔のデータとして取り扱い比較するデータと実データで飛ぶ抜けたデータがあっても相関係数の結果にはさほど左右されない特徴を持った関数である。全ての品詞を含んだ 1400 単語、全ての品詞を含んだ 2145 単語、名詞だけの 1096 単語、名詞だけの 1576 単語のそれぞれの評価データを使用して、それぞれのモデルの評価を行った。本実験で利用させていただいた日本語単語類似度・関連度データセットのサイトを次に示す。

https://blog.hoxo-m.com/entry/2020/02/20/090000

次に利用させていただいた評価プログラムが置かれているサイトを次に示す。

https://github.com/shihono/evaluate_japanese_w2v

第5章　実験結果とその考察

実験を行った結果、下図のように結果を求めることができた。Figure 5.1, Figure 5.2, Figure 5.3, Figure 5.4 はそれぞれのモデルを評価する際に、それらのモデルには含んでいない単語 (損失単語) の数を表したグラフである。Figure 5.1、Figure 5.2 をみると katomodel の損失単語がとても多いことが目立つが、katomodel は名詞だけでモデルを学習したため名詞以外の品詞を含んだ評価データを使用しているため、損失単語の数がとても多い結果となることは必然であった。Figure 5.3、Figure 5.4 をみると、評価データから名詞以外の品詞を削除してから使用したため、どのモデルでも損失単語が少なく、多くても損失単語が 14 単語のみの結果となった。名詞のみの評価データを使用することで、本実験で作成したモデルと既存のモデルを完全に同列として比較することが出来ないが、少なからずは既存のモデルとの比較ができると考えられる。

次の Figure 5.5, Figure 5.6, Figure 5.7, Figure 5.8 では、それぞれの類似度評価を表す。類似度評価とは、単語ペアと、その単語ペアの類似度 (0〜1) で構成された評価である。

Figure 5.5, Figure 5.6 をみると、katomodel がどちらの図でも類似度評価が高いことが分かる。これは、Figure5.1, Figure 5.2 の結果から損失単語が多いことと繋がっていると考えられる。なぜなら、評価する単語自体が減っているため、評価が上がると考えられる。また chiVe の類似度評価が既存のモデルの中で一番高いことが分かった。chiVe が特に損失単語が多いことはないので katomodel とは他に類似度評価が高くなる理由があると考えられる。考えられる理由として、モデルを作成するにあたって使用したモデル (Skip-Gram , COW) と単語を取得してきたデータが影響をしていると考えられる。WikiEntVec とシロヤギの元となるデータは、Wikipedia と同じであるが Skip-Gram を利用して作成した WikiEntVec が、シロヤギよりも類似度評価が高い結果となった。また同じ Skip-Gram で作成した chiVe と WikiEntVec では元データが日本語ウェブコーパスを使用している chiVe が、WikiEntVec よりも類似度評価が高い結果となった。このことから、タフな問題に鍛えられることができる Skip-Gram を使用し、ウェブを母集団として 100 億語規模を目標として構築した日本語ウェブコーパスを元のデータとして使用したモデルなため類似度評価が高いと考えられる。Figure 5.6, Figure 5.7 の結果をみると katomodel が他のモデルよりも類似度評価が高い結果となった。このことから、katomodel はタフな問題に鍛えられることができる Skip-Gram を使用しており、日本語 WordNet から名詞の

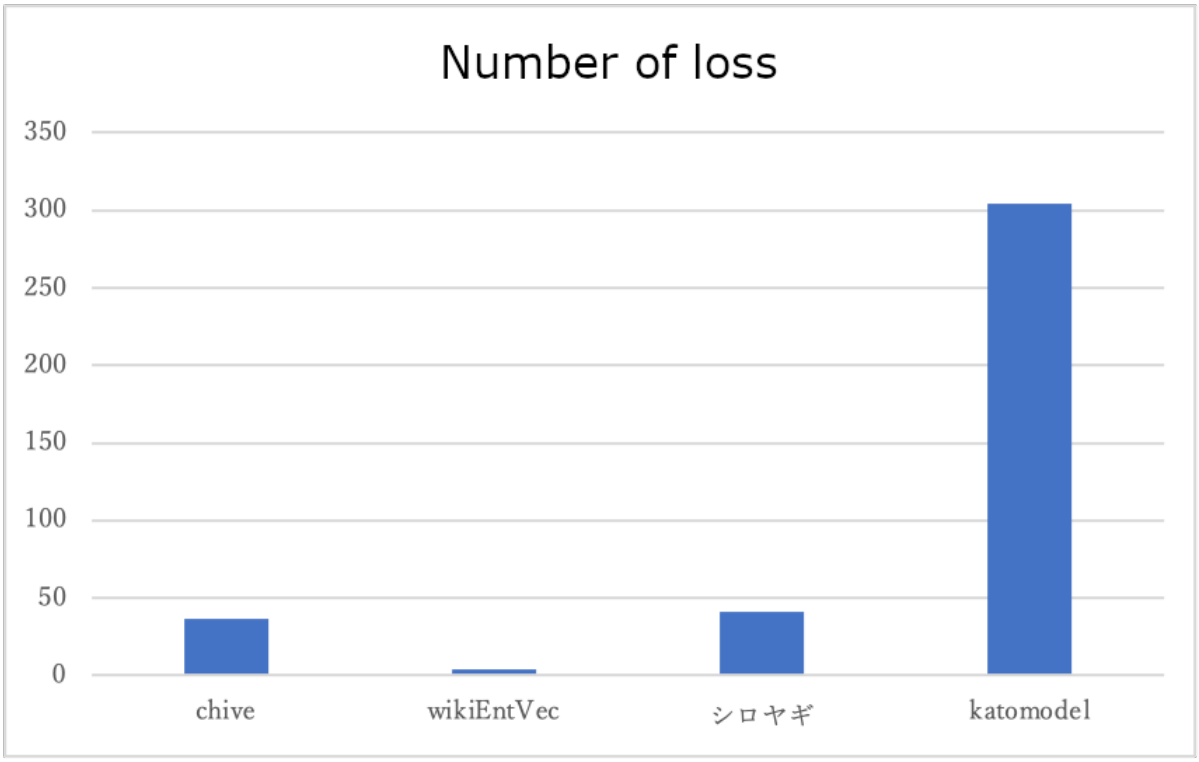

Figure 5.1　Number of loss in 1400 words.

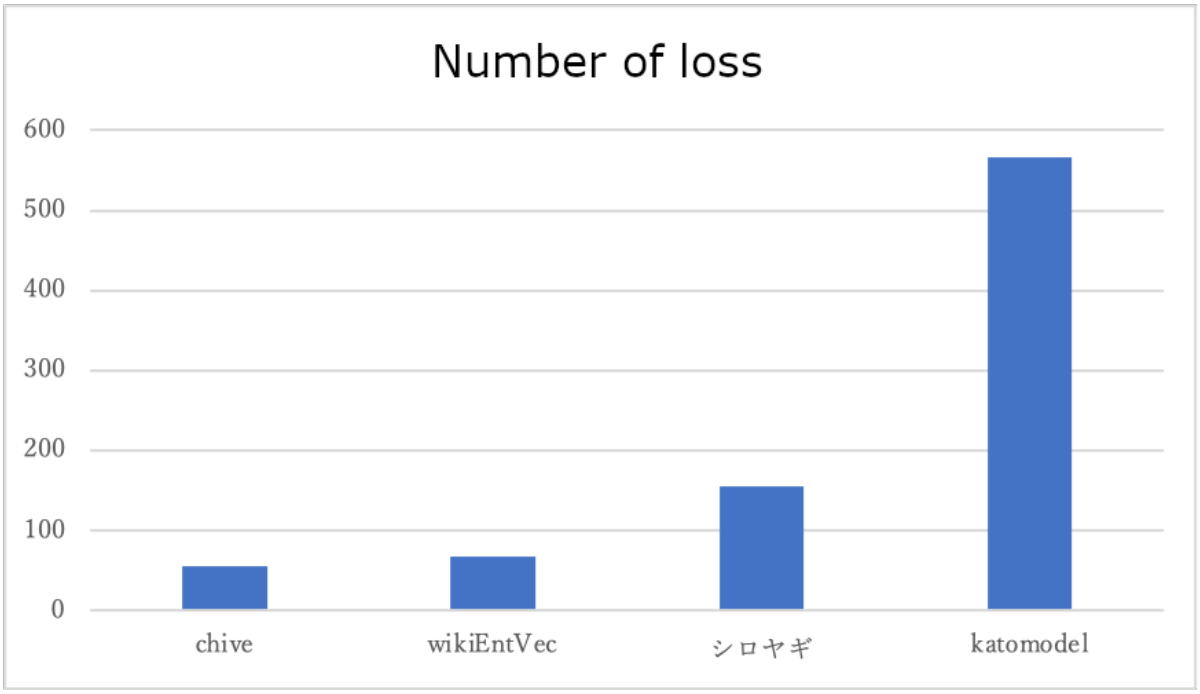

Figure 5.2　Number of loss in 2145 words.

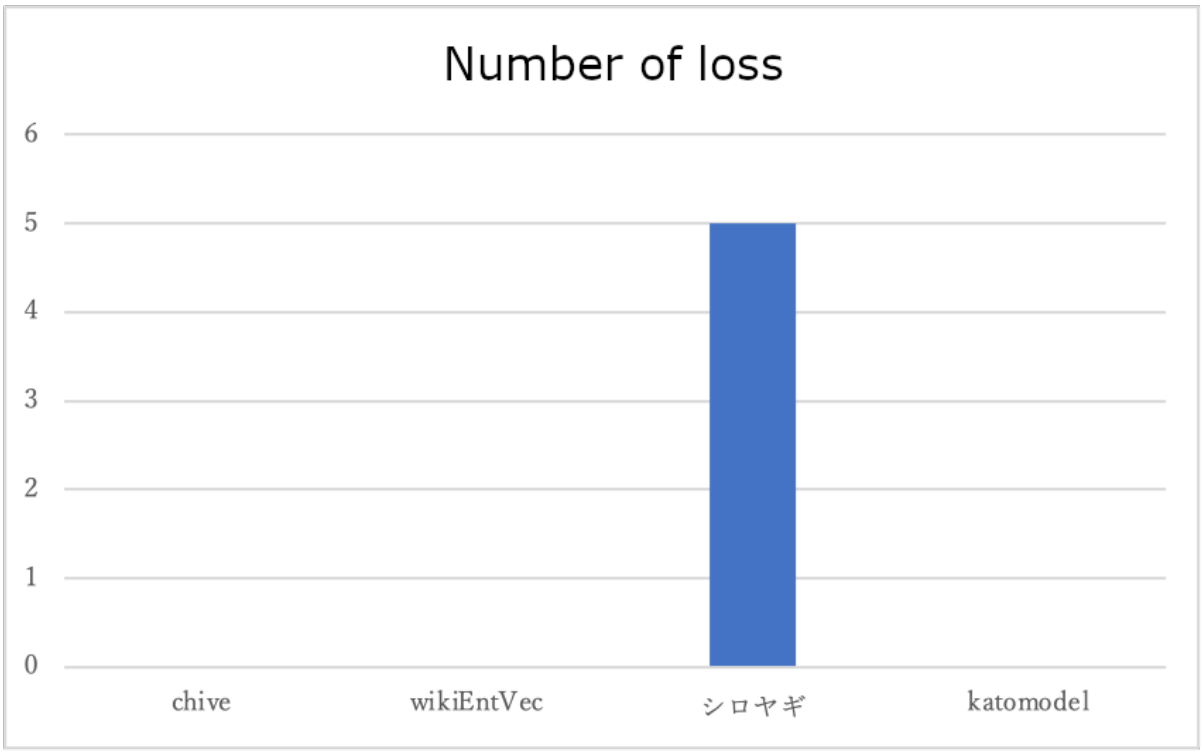

Figure 5.3 Number of loss in 1096 words derived by eliminating the others than noun.

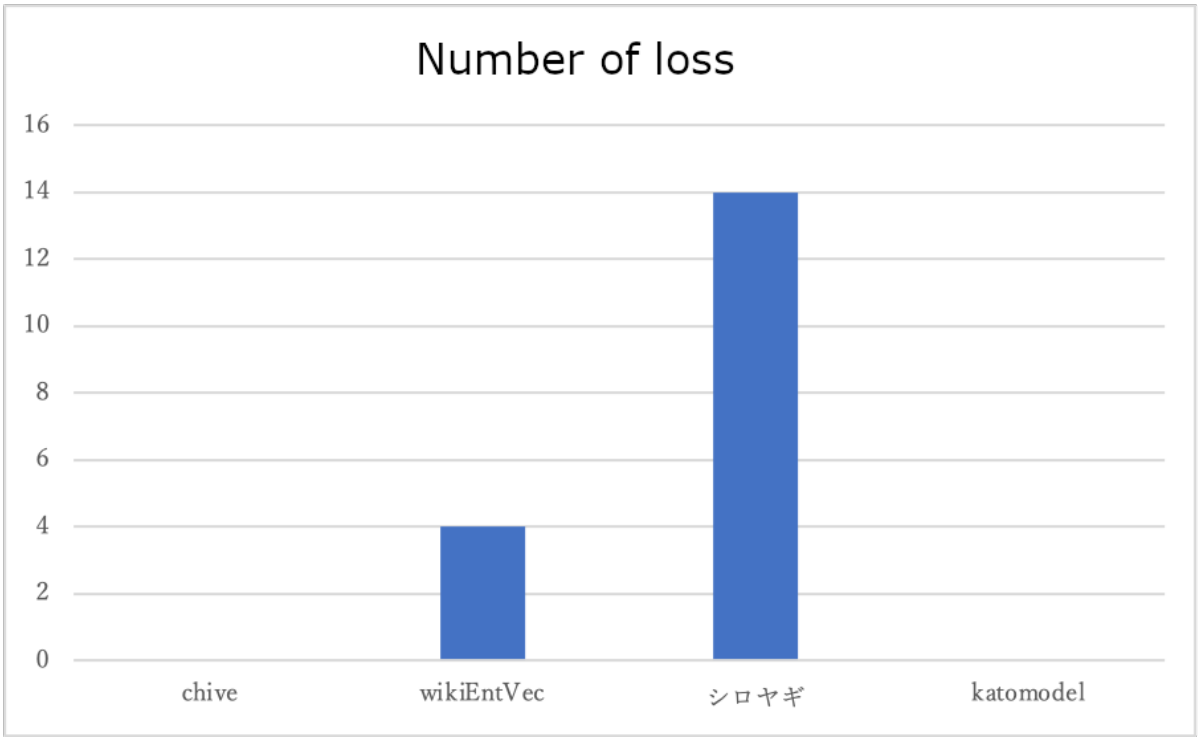

Figure 5.4 Number of loss in 1576 words derived by eliminating the others than noun.

みを抜いた単語群を元データにして作成したモデルであることから、やはり Skip-Gram で作成するモデルは優秀であり、日本語 WordNet から名詞のみを抜いた単語群は洗練されていること分かった。しかし名詞だけでの類似度評価であるため、同列に既存のモデルと比較できていないことが残念である。既存のモデルの評価は Figure 5.5, Figure 5.6 の結果と同じものになった。このことからさらに、Skip-Gram を使用し、ウェブを母集団として 100 億語規模を目標として構築した日本語ウェブコーパスを元のデータとして使用したモデルは類似度評価が高くなると考えられる。

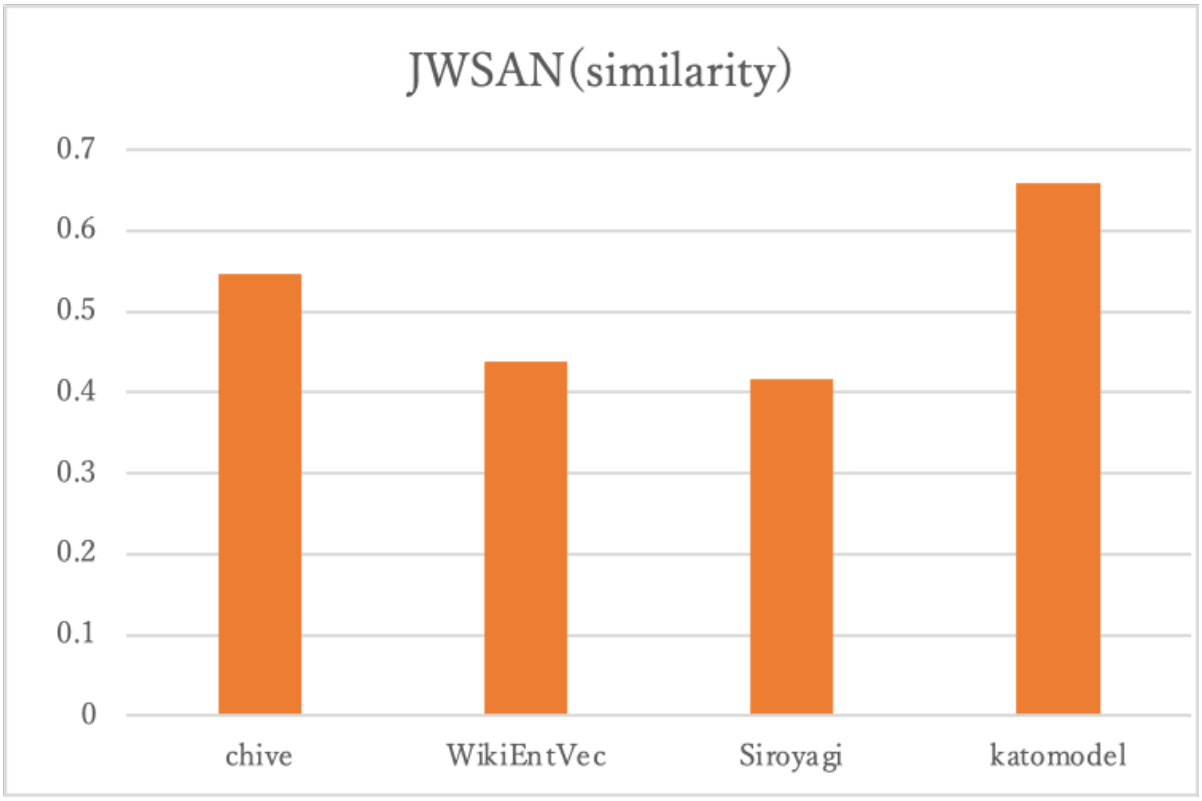

Figure 5.5 Similarity(Spearman's rank correlation coefficient) in 1400 words.

Figure 5.6 Similarity(Spearman's rank correlation coefficient) in 2145 words.

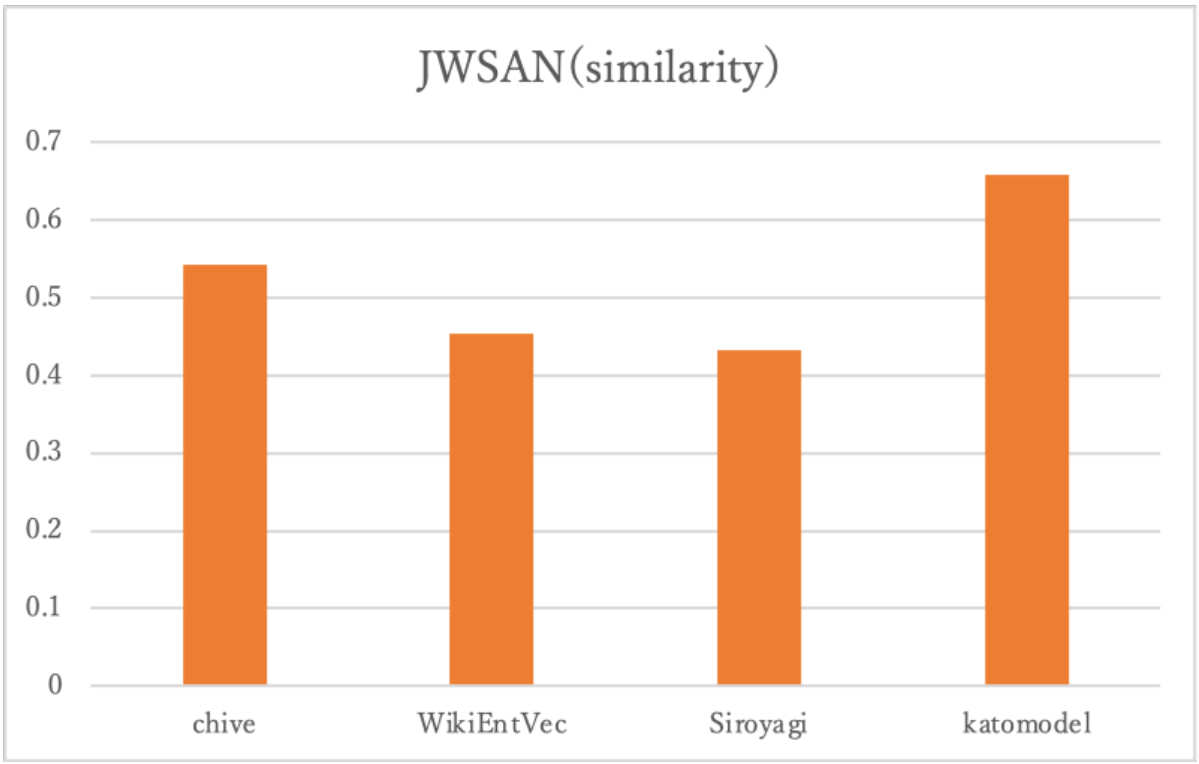

Figure 5.7 Similarity(Spearman's rank correlation coefficient) in 1096 words derived by eliminating the others than noun.

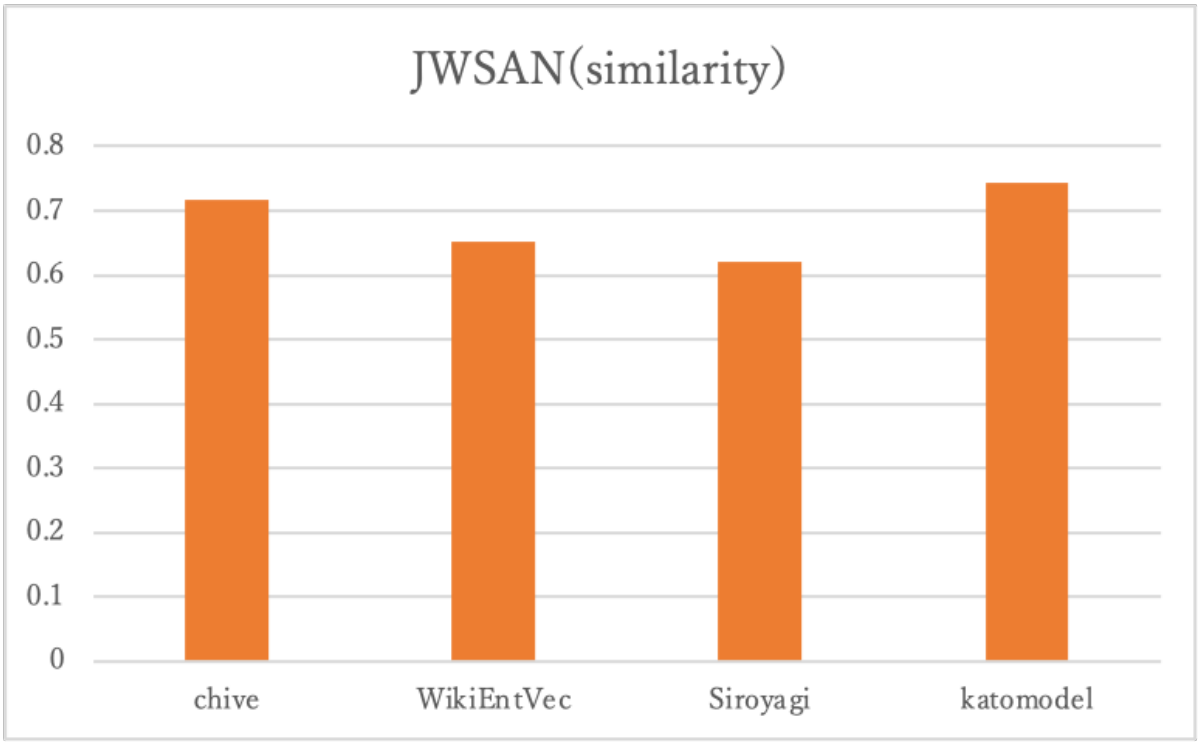

Figure 5.8 Similarity(Spearman's rank correlation coefficient) in 1576 words derived by eliminating the others than noun.

第6章　結論

結果から、モデルを作成するときはSkip-Gramによる学習と大きなデータを元に作成することが優秀なモデルを学習させるカギとなることが分かった。しかし、本実験で学習させたkatomodelは日本語WordNetから抜いた名詞のみで作成したため類似度評価は高い結果となったが、モデルとしては名詞だけで学習させたモデルであるため、不十分になってしまったため今後改善していく必要がある。

これを改善するためには、全ての品詞を学習させればよい。しかし、初めから学習データ、辞書を作り、モデルを長い時間かけて作成するのは極めて大変であるため、追加学習を行いたい。しかし、日本語WordNetは便利であるが簡単にモデルの追加学習をすることができない。そこで、Word2vecでモデル作成を行えば後から追加学習が容易に行えるためいままで、ネックだった追加学習を行うことができる。本実験では名詞だけでしかモデルを学習させていないがWord2vecでのモデル作成であったため後から追加で他の品詞も学習させることが可能である。つまり、このモデルを元にさらに優れたモデルへ学習させることが可能であると考えられる。また、既存のモデルでWord2vecを使用しているものならば、そのモデルの上から追加学習が行えるということである。追加学習させたモデルと現時点のモデルを比較してみると、本当に追加学習を行えばよりよいモデルを学習させることが出来るのか確かめられることができると考えられる。

参考文献

1) ゼロから作る Deep Learning2-自然言語処理編, 斎藤 康毅, オライリー・ジャパン, オーム社, 2018
2) ビジネス+IT, ニューラルネットワークの基礎解説：仕組みや機械学習・ディープラーニングとの関係は, https://www.sbbit.jp/article/cont1/33345
3) Python と実データで遊んで学ぶデータ分析講座, 梅津 雄一/中野 貴広, シーアンドアール研究所
4) Word2vec による自然言語処理, 西尾 泰和, オライリー・ジャパン,2018
5) Ledge.ai 形態素解析とは — 定義・用途・3 種のツールについて解説 https://ledge.ai/morpho_analysis_japan/
6) MeCab: Yet Another Part-of-Speech and Morphological Analyzer https://taku910.github.io/mecab/#feature
7) Python によるテキストマイニング入門, 山内長承, オーム社,2018
8) 人工知能学会論文誌 20 巻 5 号 B, 言葉の意味の類似性判別に関するシソーラスと概念ベースの性能評価,2005 https://www.jstage.jst.go.jp/article/tjsai/20/5/20_5_326/_pdf
9) Keras Documentation,Keras とは https://keras.io/ja/
10) Python と Keras によるディープラーニング,Francois Chollet, 巣籠悠輔, 株式会社クイープ,2018
11) さくらのナレッジ,DeepFake 使って見た!?Mac で GPU を使った機械学習 https://knowledge.sakura.ad.jp/26769/
12) 算数から高度な数学まで、網羅的に解説したサイト,One-hot ベクトル (ワンホット表現) の意味とメリット・デメリット https://mathwords.net/onehot
13) chiVe 2.0：Sudachi と NWJC を用いた実用的な日本語単語ベクトルの実現に向けて, 株式会社ワークスアプリケーションズ　香川大学大学院工学研究科　人間文化研究機構　国立国語研究所 https://www.anlp.jp/proceedings/annual_meeting/2020/pdf_dir/P6-16.pdf
14) Shingo の数学ノート,word2vec(Skip-Gram with Negative Sampling) の理論と実装 1 http://mathshingo.chillout.jp/blog18.html

謝辞

本研究を進めるにあたり、ご多忙の中にも関わらず親身に指導をしていただきました出口利憲先生に深く感謝申し上げます。また、つまずいた時に相談に乗っていただいた同研究室の皆様にも深く感謝申し上げます。