目 次

第1章　序論

高品質なソフトウェアを効率的に開発していく上で,.... が重要である [1]．そこで本論文では.... を行う．

本論文の構成は以下の通りである．まず，第 2 章でソフトウェアメトリクスについて概説し，第 3 章で○○に基づいた△△を提案する．そして，第 4 章でオープンソースソフトウェアを対象とした評価実験を行い，提案法の有効性について考察する．最後に第 5 章で本論文のまとめと今後の課題について述べる．

第2章　ソフトウェアメトリクス

2.1　緒言

本章では本論文での議論の準備としてソフトウェアメトリクスについて概説する．以下，**??** 節でソフトウェアメトリクスの定義とその大まかな分類を示す.

2.2　Word2Vec

Word2Vec [2] とは，学習の際にテキストデータを解析し，単語の意味を考慮して各単語をベクトル化する手法である．これにより，単語同士の意味の類似度を計算したり，意味を足したり引いたりすることが可能になる．Word2Vec の学習方法として，Continuous Bag-of-Words と Skip-gram が挙げられる．Word2Vec はどちらかの方法に従い，ニューラルネットワークを用いて学習を行うことにより，単語をベクトル化する．それぞれの学習方法の概要を以下に示す.

- Continuous Bag-of-Words

  Continuous Bag-of-Words(CBoW) [2] は，周辺の単語から注目する単語を予測できるように学習を行う．例えば，“I have a cat at home” という文章の “cat” に着目して “I have a ○○ at home” という文章を考える．CBoW では，この○○にどういった単語が入るかを予測するという考え方の下で学習を行う．○○に入る可能性のある単語を考えると，“cat” 以外にも “dog” や “hamster” などが挙げられる．CBoW は，周辺の単語である “I”，“have”，“a”，“at”，“home” から○○に入る単語が “cat” だと予測できるように学習する.

- Skip-gram

  Skip-gram [2] は，注目している単語から周辺の単語を予測できるように学習を行う．先ほどの文章を例に挙げると，“cat” という単語からその前後に登場する単語 “have” や “home” などを予測できるように学習する.

実際に “I have a cat at home” という文章に対して Skip-gram による学習の流れを概説する．まず，文字列の状態では計算することができないため，文字列をベクトル化するために one-hot ベクトルを考える．one-hot ベクトルとは，文字列中の単語の種類数に等しい次元のベクトルを考え，その単語に該当する要素を 1，その他を全て 0 とするベクトルである．例えば，各単語の one-hot ベクトルは以下に示すベクトルとなる．

- I : (1,0,0,0,0,0)
- have : (0,1,0,0,0,0)
- a : (0,0,1,0,0,0)
- cat : (0,0,0,1,0,0)
- at : (0,0,0,0,1,0)
- home : (0,0,0,0,0,1)

このとき，入力が “cat” のときのニューラルネットワークによる学習のイメージを図 2.1 に示す．ここでは隠れ層のユニット数を 3 としている．隠れ層とは，データが入力される入力層と，結果を出力する出力層の間の層である．

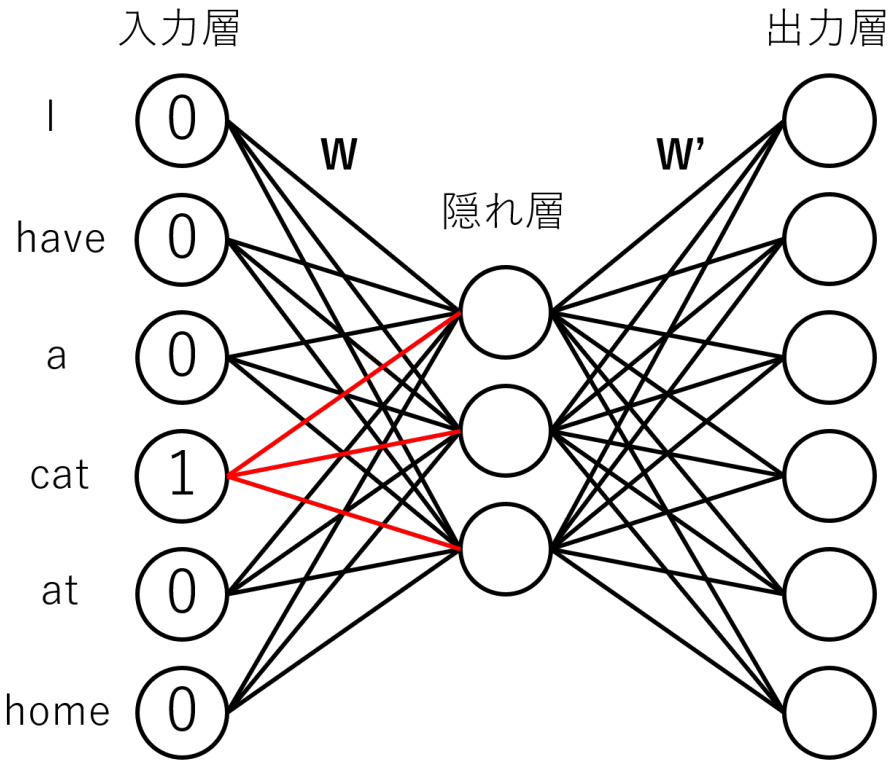

図 2.1　学習のイメージ図

次に入力層から隠れ層，隠れ層から出力層への重み行列を導入する．ここで入力層から隠れ層への重み行列を **W**，隠れ層から出力層への重み行列を **W'** とする．また，隠れ層のユニット数は 3 としているため，重み行列の次元数は 3 である．初期状態では，重み行列の内容はランダムに生成される．(2.1) 式のような重み行列 **W** を考える．

$$\boldsymbol{W} = \begin{bmatrix} 0.1 & 0.2 & 0.5 \\ 0.2 & 0.8 & 0.4 \\ 0.3 & 0.5 & 0.7 \\ 0.4 & 0.4 & 0.6 \\ 0.5 & 0.9 & 0.1 \\ 0.6 & 0.1 & 0.1 \end{bmatrix} \tag{2.1}$$

このとき，単語 “cat” を入力したときの計算が (2.2) 式で表され，重み行列の 4 行目が隠れ層の値となる．

$$[0 \quad 0 \quad 0 \quad 1 \quad 0 \quad 0] \cdot \begin{bmatrix} 0.1 & 0.2 & 0.5 \\ 0.2 & 0.8 & 0.4 \\ 0.3 & 0.5 & 0.7 \\ 0.4 & 0.4 & 0.6 \\ 0.5 & 0.9 & 0.1 \\ 0.6 & 0.1 & 0.1 \end{bmatrix} = [0.4 \quad 0.4 \quad 0.6] \tag{2.2}$$

これらの値と重み行列 **W'** との積を計算することで出力層の値が決まる．重み行列 **W'** を (2.3) 式，隠れ層の値と重み行列 **W'** の積を (2.4) 式に示す.

$$\boldsymbol{W'} = \begin{bmatrix} 0.1 & 0.2 & 0.3 & 0.4 & 0.5 & 0.6 \\ 0.2 & 0.8 & 0.5 & 0.4 & 0.9 & 0.1 \\ 0.5 & 0.4 & 0.7 & 0.6 & 0.1 & 0.1 \end{bmatrix} \tag{2.3}$$

$$[0.4 \quad 0.4 \quad 0.6] \cdot \begin{bmatrix} 0.1 & 0.2 & 0.3 & 0.4 & 0.5 & 0.6 \\ 0.2 & 0.8 & 0.5 & 0.4 & 0.9 & 0.1 \\ 0.5 & 0.4 & 0.7 & 0.6 & 0.1 & 0.1 \end{bmatrix} = [0.42 \quad 0.64 \quad 0.74 \quad 0.68 \quad 0.62 \quad 0.34] \tag{2.4}$$

これに対して，正解の単語は周辺の単語であるため，その one-hot ベクトルが教師データとなる．Word2Vec では，以上のようにして得られる出力が正解に近づくように重みの更新を繰り返していく．そして最終的に得られた重みを用いることで，単語をベクトルとして表現し，定量的に扱うことができる．

本研究では，Skip-gram を用いるため，CBoW についての説明は割愛する．

2.3　Doc2Vec

単語をベクトル化する Word2Vec に対して，これを文書レベルへ拡張した手法が Doc2Vec [3] である．Word2Vec は単語のみを入力として学習する手法である．これに対して Doc2Vec では，単語と文書の ID を入力として学習を行う．Doc2Vec の学習方法は，Distributed Memory Model of Paragraph Vector と Distributed Bag of Words version of Paragraph Vector の 2 種類がある．それぞれの学習方法の概要を以下に示す．

- Distributed Memory Model of Paragraph Vector

  Distributed Memory Model of Paragraph Vector（PV-DM）[3] は，Word2Vec の CBoW 同様に中心の単語について学習する方法である．CBoW と異なる点は，CBoW が前後の単語のみを入力データとするのに対し，PV-DM では文書の ID も入力データとして加えて学習するところにある．

- Distributed Bag of Words version of Paragraph Vector

  Distributed Bag of Words version of Paragraph Vector（PV-DBoW）[3] は，Word2Vec の skip-gram と似た手法であり，文書の ID を入力データとして，その文書に登場する単語を予測するように学習を行う．PV-DBoW は，PV-DM とは異なり，文書の ID のみが入力データであるため，語順を考慮しない学習方法となっている．

2.4　畳み込みニューラルネットワーク

畳み込みニューラルネットワーク（Convolutional Neural Network: CNN）とは，2 次元の入力データから特徴を抽出できる手法である [4]．当初は画像認識の分野で開発されたが，未知データに対しても高い識別率を持つことから自然言語処理の分野でも使用されている．CNN の概要を図 2.2 に示す．

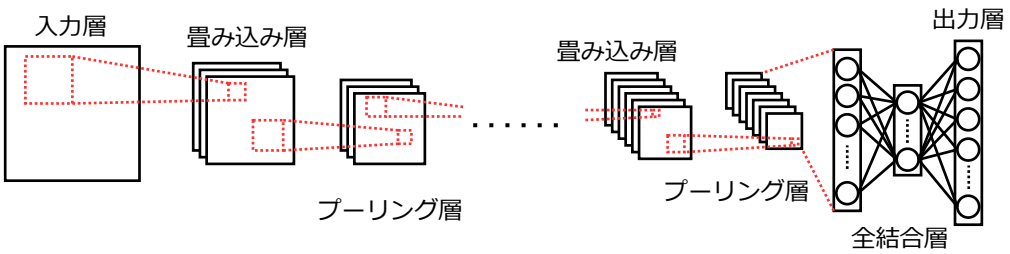

図 2.2　CNN の概要

図 2.2 のように CNN は畳み込み層とプーリング層と呼ばれる層が交互に重ねて構成され，全結合層へとつながる．これらの層の概要を以下に示す．

- 畳み込み層

  畳み込み層は，データに対していくつかのカーネル (フィルター) を適用することで特徴を抽出する層である．カーネルは，事前に学習してデータ上に特徴を見つけた際に活性化させることができるようなカーネルを見つけて使用する．畳み込みによって得られた複数の特徴を特徴マップという．特徴マップの数は使用するカーネルの数に等しくなる．畳み込みの一例を図 2.3 に示す．ここでは，簡単のため入力データとカーネルの数値は 0 と 1 の 2 値とする．

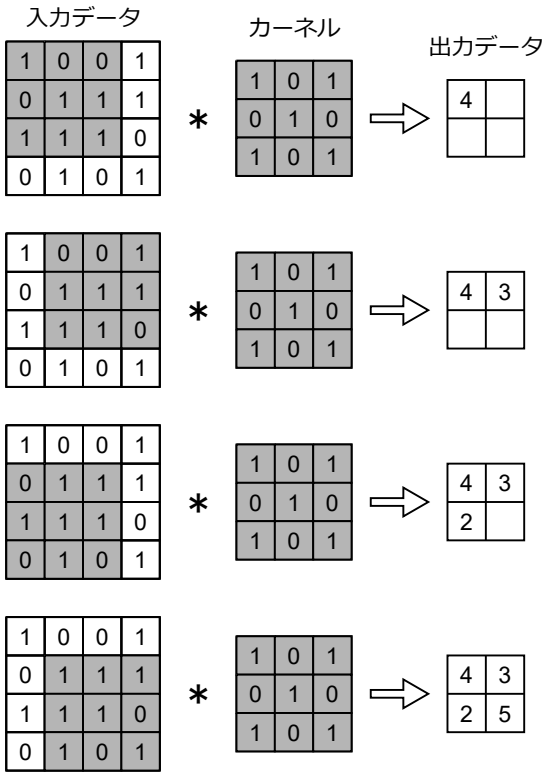

図 2.3　畳み込みの一例

- プーリング層

  プーリング層は，データ量を減らす変換を行う層である．これにより，後に続く層の計算量を削減することができる．本研究では，近傍の入力値の最大値を出力する MAX プーリングを使用する．MAX プーリングの一例を図 2.4 に示す.

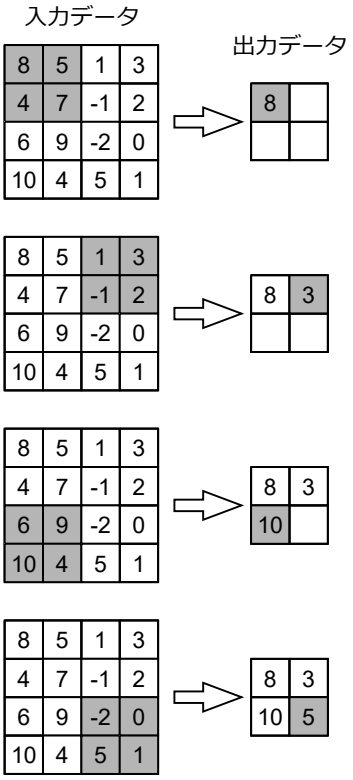

図 2.4　MAX プーリングの一例

- 全結合層

  全結合層は，通常のニューラルネットワークと同様に重みとバイアスを用いて入力値を変換する層である．最後にソフトマックス関数を適用することで確率の表現に変換する.

通常のニューラルネットワークでは全てのニューロンが完全に結合しているため，少しだけずれたわずかに違うデータでもまったく別のデータとして学習してしまい柔軟性や汎化能力に欠けるが，CNN ではニューロン同士の結合はカーネルの大きさに制限されるため，わずかに違うデータにも対応できるといった利点がある.

2.5　結言

本章ではソフトウェアメトリクスについて概説した．ソフトウェアメトリクスにより.....が可能である．そこで次章では○○に基づいた△△を提案していく．

第3章　○○に基づいた△△

3.1　諸言

3.2　データの前処理

本節ではデータの前処理に使用する抽象構文木とそれを実装するために用いたライブラリである JavaParser について説明する．

3.2.1　抽象構文木

抽象構文木（Abstract Syntax Tree : AST）とは，ソースコードの文法構造を木構造で表したものである．

AST には，ファイル全体を表すルートノードがあり，その子としてノードが複数接続されている．例えば，import 文やクラスの宣言などが挙げられる．そして，さらにクラスの宣言はフィールドやメソッドを表すノードに接続されている．AST の概要を図 3.1 に示す．図 3.1 のルートノードに当たる CompilationUnit は，構築された AST を受け取るクラスである．

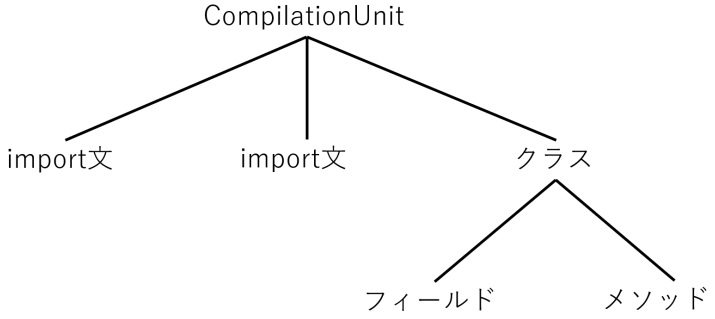

図 3.1　AST の概要

AST を利用することで，元の字句だけでなく，その字句の意味も同時に取得できる．例えば，“int a;” という命令は a が int 型の変数，int が PrimitiveType という意味を持ち，それを取得することができる．

3.2.2 JavaParser

Java 言語で書かれたソースコードを解析し，その AST を得るために JavaParser[5] というライブラリを用いる．これにより，メソッドをトークン化したものを容易に取得できる．JavaParser を使用して図 3.2 に示す Sample.java を解析したときの出力結果を図 3.3 に示す．本研究では空白，改行，区切り文字は使用しないため，無視している．

```
1 public class Sample {
2    public void main(String[] arg){
3       for (int i = 0; i<10; i++) {
4          System.out.println(i);
5       }
6    }
7 }
```

図 3.2　Sample.java

```
1  public
2  class
3  sample
4  public
5  void
6  main
7  String
8  arg
9  for
10 int
11 i
12 =
13 0
14 i
15 <
16 10
17 i
18 ++
19 System
20 out
21 println
22 i
```

図 3.3　出力結果

図 3.3 のように意味のある文字列だけを取得し，ソースコードをトークン化することができる．トークンを取得した後，for なら ForStatement，int なら PrimitiveType といった，それぞ

れのトークンの意味を追加する．

この処理をメソッドの本体に対して行うことで，本研究に用いるデータセットを作成する．

第4章　実験

表 4.1 に表の例を示す.

4.1　定義

一般に，定義 4.1 における（4.1）式の関数 d は距離関数と呼ばれる(注1)．距離関数が定義された空間――正確にいえばベクトル空間――を距離空間という(注2).

定義 4.1 (距離)

2 つのベクトル $\boldsymbol{x}, \boldsymbol{y}$ が与えられたとき，これらの距離 $d(\boldsymbol{x}, \boldsymbol{y})$ を次式で定義する：

$$d(\boldsymbol{x}, \boldsymbol{y}) = ||\boldsymbol{x} - \boldsymbol{y}|| \tag{4.1}$$

□

上の例のように定義や定理の終わり，箇条書きの終わりに`\QED` と書くと，右寄せで □ が表示される．数学では “証明終わり” の意味で使われるが，工学の論文ではそれ以外でも一つの区切りであることを明記するために使うことがある.

表 4.1　簡単な表の例です

左寄せ	中央揃え	右寄せ
1	JabRef[?]	1.50
2	XYZ	3.91
3	ZZZ	1.83

多くの学術論文では表がよく使われるが，その際には縦の罫線を極力使用しない傾向にある．特に両端の縦罫線は省略されることが多い．学会によっては横罫線も可能な限り省略することが推奨される.

(注1) 信じないでね.
(注2) これまた，信じないでね

なお,（これも学会によるが,）横罫線の一部を太くするように指示されることがある．本サンプルでは表の始まりと終わり，並びに項目名の下については太くしてある．太くする場合は `\hline` の代わりに `\Hline` を使う．

第5章　結論

ダブルクォーテーションは “このように” 書く.

特殊記号は α という感じで前後に半角スペースを入れて書くことを推奨する.

謝辞

例年の卒論を真似してお礼を書く.

参考文献

［1］ 白鳥則郎, 高橋薫, 神長裕明：ソフトウェア工学の基礎知識, 昭晃堂, 東京 (1997).

［2］ Mikolov, T., Chen, K., Corrado, G. S. and Dean, J.: Efficient Estimation of Word Representations in Vector Space (2013).

［3］ Le, Q. V. and Mikolov, T.: Distributed Representations of Sentences and Documents, in *International Conference on Machine Learning* (2014).

［4］ Matsugu, M., Mori, K., Mitari, Y. and Kaneda, Y.: Subject independent facial expression recognition with robust face detection using a convolutional neural network, *Neural Networks*, Vol. 16, No. 5, pp. 555 – 559 (2003), Advances in Neural Networks Research: IJCNN '03.

［5］ JavaParser, For processing Java code, https://javaparser.org/.

付 録A　その他

ここでは番号がすべてアルファベットに変わる.