第4章

発言内容の類似度計算

4.1　序言

本章では発言内容，すなわち文字列の意味的類似度を計算する手法を提案する．以下に本章の構成を示す．まず，4.2 節では類似度の精度を上昇させるために行う前処理について説明する．4.3 節では発言内容の類似度計算の手法について述べる．4.4 節では本章のまとめを示す．

4.2　前処理

分散表現による類似度計算で精度を上昇させるためには発言内容から余分な単語を取り除き重要な単語を抽出する，または極めて短く要約することが重要である．提案手法では形態素解析エンジン MeCab と**??**節で説明した okapiBM25 と LexRank を用いて発言の文章から重要単語を抽出し，抽出した単語の類似度を分散表現によって計算する．

4.2.1 MeCab

MeCab(めかぶ)[?] は京都大学情報学研究科-日本電信電話株式会社コミュニケーション科学基礎研究所共同研究ユニットプロジェクトを通じて開発されたオープンソース形態素解析エンジンである．以下の図 4.1 で MeCab に対して「MeCab はオープンソース形態素解析エンジンである。」と入力した際の結果を示す．

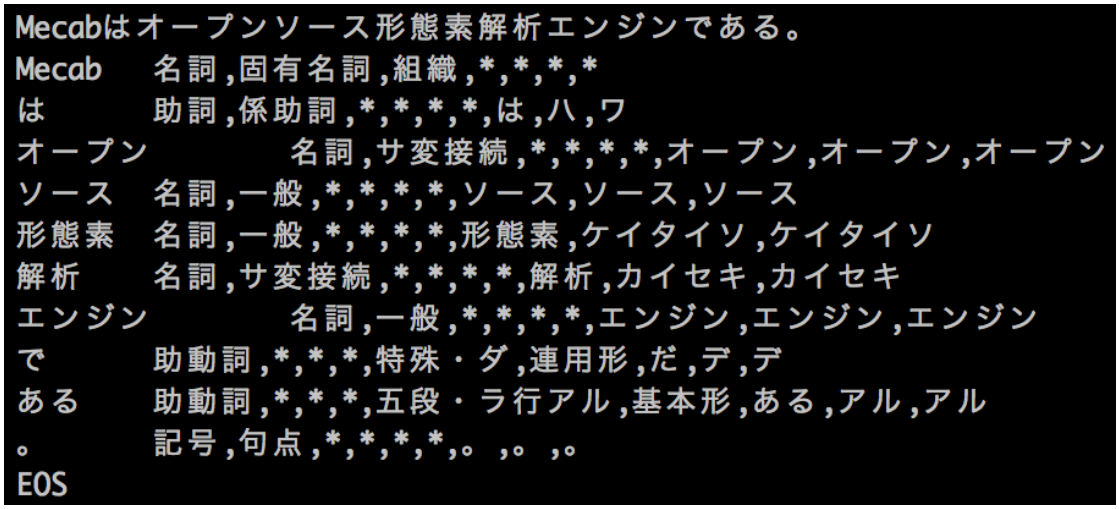

図 4.1: 解析結果

出力フォーマットは次の形式となっている．

表層形 \t 品詞, 品詞細分類 1, 品詞細分類 2, 品詞細分類 3, 活用型, 活用形, 原形, 読み, 発音

Mecab では形態素解析を行う時，辞書データに基いて図 4.2 に示すように全ての可能な出力をグラフで表現した構造を作成し，Conditional Random Fields[?] と呼ばれるコスト推定アルゴリズムを用いて最も可能性の高い経路を推定し，出力としている．また,「読み」と「発音」は図 4.1 の”MeCab”のように不明であるものには付与されない．

本研究では MeCab による形態素解析の結果，次の条件を満たす単語を除外している．

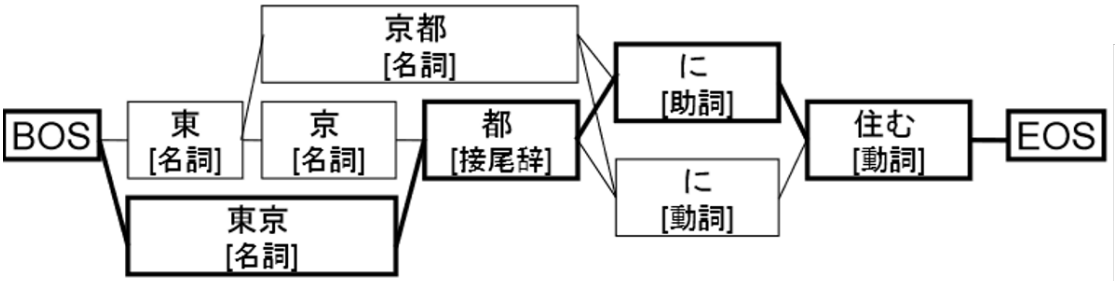

図 4.2: 解析結果

1. 品詞細分類に「数」を含む

2. 「読み」,「発音」が不明である

3. 品詞が「助詞」,「助動詞」,「記号」,「連体詞」のどれかである.

4. 1 文字のひらがなである

5. 品詞細分類に「接尾」または「非自立」を含む

上記の条件を満たす単語を除外した理由は 4.2.2 節で説明する重み付けにおいて重要な単語であると判定されやすいが，**??**節で説明する類似度計算において精度を下げてしまうからである．本研究は文章の前処理として，上記の 1 から 5 の条件に該当する単語の　クリーニング処理を実施した. 前処理を実施した理由は，類似度計算において，精度を向上させるためである．単語の除外は重み付けにおいて文章を単語に分割する際に行われる.

4.2.2 重み付け

提案手法では**??**節で説明した okapiBM25 と LexRank の 2 種類の重み付け手法を統合して発言の内容の文字列 *remark* 中の単語に対して重み付けを行う．アルゴリズムを **Algorithm1** に示す．

Algorithm 1 統合重みの計算アルゴリズム

```
1: Input : remark 発言内容の文字列
2: Output : combinedWeight remark 中の単語と重みを対応付けた連想配列
3: Array  sentList;  ▷ 以前に重み付けを行った最大 n 個前までの文章のリスト
4: procedure calcCombinedWeight(reamrk)
5:     bm25Weight = calcBM25Weight(remark)          ▷ 単語と重みの連想配列
6:     for Each sent ∈ remark do      ▷ remark を句点，改行コードで分割する
7:         sentList.append(sent)
8:     lexWeight = calcLexRank(sentList)
9:     for Each word ∈ bm25Weight.keys() do
10:        wordWeight = bm25Weight[word]
11:        if word is 固有名詞 then
12:            wordWeight *=2
13:        sentWeight = 0
14:        for Each sent ∈ remark do
15:            if word in sent then
16:                sentWeight+=lexWeight[sent]
17:        combinedWeight[word] = wordWeight*sentWeight
18:    return combinedWeight
```

本研究において，固有名詞は文章の中で重要な役割を果たす可能性が大きいと考え，12 行目で固有名詞の単語重みを倍にしている．そして，14 ～ 17 行目では word を含む全文章の重みの合計を求め，okapiBM25 による単語重みを掛け合わせたものを word の統合重みとしている．単語重みに単語を含む文章の重みを掛け合わせることで感嘆文のような文章そのものは重要でないが頻度の少ない単語を使用する文章中の単語が選ばれる可能性を下げている．

4.3 類似度計算

4.3.1 単語抽出

本研究では 4.2 節で計算された単語重みの値が大きいものの上位 n 個までの単語を発言文章 remark において重要度の高い単語であるとして抽出する．単語重みが等しいものが複数あった場合は単語を昇順に並び替えて順序を付ける．また，使用する分散表現モデルに登録されていない単語は除外している．

4.3.2 分散表現による類似度計算

分散表現による類似度計算は，4.3.1 節で述べた手法で抽出した単語集合の類似度から分散表現を用意て求める．それぞれの単語集合の単語ベクトルの平均を求め，**??**の図**??**で述べたように Cosine 類似度を 2 平均ベクトル間で取っている．

4.4 結言

本章では発言内容の類似度を計算する手法について説明した．本章では，分散表現による類似度計算の前処理である文章を単語に分割する手法，及び単語を除外する処理についても述べ ，また，okapiBM25 と LexRank を組み合わせた発言中の単語の重み付け手法，及び重み付けによって抽出した単語集合の類似度計算手法について説明した．