JAIST Repository
https://dspace.jaist.ac.jp/

Title	語義の箱埋め込み学習とその応用
Author(s)	小田, 康平
Citation	
Issue Date	2024-03
Type	Thesis or Dissertation
Text version	author
URL	http://hdl.handle.net/10119/18879
Rights	
Description	Supervisor: 白井 清昭, 先端科学技術研究科, 修士(情報科学)

JAIST
JAPAN
ADVANCED INSTITUTE OF
SCIENCE AND TECHNOLOGY

Japan Advanced Institute of Science and Technology

修士論文

語義の箱埋め込み学習とその応用

小田 康平

主指導教員 白井 清昭

北陸先端科学技術大学院大学
先端科学技術研究科
（情報科学）

令和６年３月

Abstract

In general, a word is polysemous, i.e., has two or more senses. It is important to handle polysemous words appropriately to understand human language by a machine. Word Sense Disambiguation (WSD) is a well-known problem on polysemy of words. It is a task to select one appropriate sense of a word in a context from a predefined set of senses. WSD has been studied for many years in the field of natural language processing. Recent trends in WSD apply supervised learning of classification models based on neural networks using word sense tagged sentences as training data. However, most of the conventional WSD researches suppose that word senses in test data are always known, that is, appear in the training data. However, senses of words are changed and new senses are generated day by day. Therefore, it is required to judge whether the target word has an unknown sense that does not appear in training data (new sense), rather than simply selecting the appropriate sense from a predefined set of senses. It is also important to provide some useful information about a new sense to help human to understand it. For example, a superordinate sense of a new sense can be helpful to grasp a general meaning of the new sense.

The goal of this study is to learn box embeddings of word senses. The box embedding of a sense is an abstract meaning of a sense represented by a region in a vector space. Unlike previous WSD researches that represent a sense as a single vector, a box embedding can represent how broad or narrow a concept of a sense is. Box embeddings enable us to determine whether a word in a given context has a new sense and to predict a superordinate sense of a new sense. In this thesis, we extend an existing WSD method, MetricWSD, to learn box embeddings of senses. In addition, we propose two policies to create a small dataset used for computation of loss in training of the model of box embeddings. Our proposed method is applied to three tasks: WSD, new sense classification, and superordinate sense prediction of a new sense, then the performance of the proposed method is empirically compared with conventional methods that represent a sense as a single vector.

Box embedding is formulated as a pair of vectors: one represents the center of the box and the other represents the size of the box. Our proposed model produces a box embedding for each word in a sentence that represents the meaning of the word in that context. We call it "contextual box embedding" of an instance of a sense (a sense appeared in a particular sentence). The box embedding of a sense is obtained by averaging the contextual box embeddings of instances of the sense.

In MetricWSD, Bidirectional Encoder Representations from Transformers (BERT) was used as a model to obtain the contextual embedding of the target word. In this research, a fully connected layer (FCL) is attached to the final layer of BERT

to obtain the contextual box embedding of the target word. The input of the FCL is an embedding of the target word in the final layer of BERT, and the output is a vector representing the contextual box embedding. The half dimensions of the output vector represent the center of the box embedding, and another half represent the size of the box embedding.

The loss function for training the model is computed by considering the overlap of box embeddings of two senses. Intuitively, the loss function is designed so that the overlap of box embeddings of a sense and its superordinate sense becomes greater, and less otherwise. A dataset to be used for calculation of the loss, called episode, is prepared for each sense. In each episode, the model is trained by the following procedures. (1) Select N_C senses from all senses appearing in the training data. (2) For each sense, randomly select N_S sense instances (sentences including the sense) as the support set. (3) For each sense, randomly select N_Q sense instances, which are not chosen in the support set, as the query set. (4) Obtain a prototype representation of each sense by averaging contextual box embeddings of the instances in the support set. (5) Calculate the loss based on the output of the model (contextual box embeddings) for the query set and the prototype representations. (6) Update the model parameters to minimize the loss.

In the above step (1), we propose two policies to select N_C word senses. The first policy is S_r. The target sense and its superordinate senses are selected first, then the rest is selected randomly. The second policy is S_n. The target sense, its superordinate senses, its subordinate senses, and sibling senses are chosen first, then the rest is selected randomly. The policy S_n emphasizes that the box embedding of a sense does not overlap with that of its subordinate and sibling senses.

In the experiment to evaluate the proposed methods, our methods are applied to three tasks: WSD, new sense classification, and superordinate sense prediction of a new sense. To evaluate the ability of the model to learn box embeddings of different number of senses, three datasets with different sizes are prepared: $\mathcal{D}_{\text{living_thing.n.01}}$, $\mathcal{D}_{\text{artifact.n.01}}$, and $\mathcal{D}_{\text{entity.n.01}}$. living_thing.n.01, artifact.n.01, and entity.n.01 are senses in WordNet (called synset), and each dataset contains sentences of subordinate senses of one of the three root senses. $\mathcal{D}_{\text{entity.n.01}}$ consists of senses of all nouns, while $\mathcal{D}_{\text{living_thing.n.01}}$ and $\mathcal{D}_{\text{artifact.n.01}}$ consist of subsets of nouns. Two baselines are prepared: BERT-NN , which based on a pre-trained BERT, and MetricWSD.

As for the WSD task, the proposed methods outperformed the baselines for small datasets ($\mathcal{D}_{\text{living_thing.n.01}}$ and $\mathcal{D}_{\text{artifact.n.01}}$). However, for the dataset consisting of many senses of all nouns ($\mathcal{D}_{\text{entity.n.01}}$), the proposed methods performed worse than the baselines. It indicates that the box embeddings of senses trained by the proposed model are not always good when the box embeddings of too

many senses are trained simultaneously. As for the new sense classification task, the proposed methods outperformed the baselines on the dataset consisting of many senses of all nouns, which is inconsistent with the results of WSD. As for the task of superordinate sense prediction of a new sense, the proposed method outperformed the baseline in all three datasets. The parameter used for choosing candidates of superordinate senses was set to 0.5, 0.7, or 0.9, and it was found that the best thresholds were different for three datasets and two policies (S_r and S_n). To sum, the box embeddings of the senses learned by the proposed method was better representation than an ordinary single vectors for several sense-related tasks in some conditions. Especially, the box embeddings has ability to represent hypernym-hyponym relation of senses appropriately.

概要

単語が複数の意味 (語義) を持つことを単語の多義性という．機械が人間の言語を理解する上で単語の多義性を適切に取り扱うことは重要な問題である．単語の多義性に関する代表的な研究に語義曖昧性解消 (Word Sense Disambiguation; WSD) がある．WSD とは，ある文脈における単語の意味として適切なものを，あらかじめ定義された語義の集合の中から 1 つ選ぶタスクである．WSD は，自然言語処理分野において長年にわたり研究されてきた．近年の WSD の研究は，正解の語義が付与された用例集合を訓練データとして，ニューラルネットワークにより語義の分類モデルを教師あり学習することが主流である．ただし，従来の WSD の研究のほとんどは，訓練データに出現する語義の中からテストデータにおける単語の語義を選択することを想定している．しかし，語義は日々変化し，新しい語義も生まれている．そのため，単にあらかじめ定義された語義集合の中から該当する語義を選ぶのではなく，対象単語が訓練データに出現しない未知の語義 (新語義) であるのかを判定することが求められる．さらに，新語義に対し，人間の理解を助けるような何らかの手がかりを提示することも重要な課題である．例えば，新語義の上位概念を提示することで，人間がその新語義の大まかな意味を理解するのを助けることができる．

本研究は，語義の箱埋め込みを学習することを目的とする．語義の箱埋め込みとは，ベクトル空間における領域によって表される語義の抽象表現である．語義を箱埋め込みで表現することで，語義を単一のベクトルで表現していた従来の WSD の研究とは異なり，語義の概念的な大きさを表現できる．それにより，ある文脈における単語が新語義であるのかを判定したり，新語義の直接の上位概念 (上位語義) を推定することが可能となる．本論文では，既存の WSD の手法である MetricWSD を語義の箱埋め込みを学習するよう拡張する．さらに，モデルの損失を計算するために用いる小さいデータセットを作成する 2 通りの方法を提案する．提案手法を WSD，新語義の判定，新語義の上位語義の推定の 3 つのタスクに適用し，語義を単一のベクトルで表現する従来手法と実験的に比較する．

箱埋め込みは，箱の中心と辺の長さを表すベクトルの組から構成される．本研究で学習するモデルは，入力文の各単語に対し，その単語の文中における意味を表す箱埋め込みを出力する．以下，これを「文脈箱埋め込み」と呼ぶ．語義の箱埋め込みは，各語義が付与された単語の文脈箱埋め込みの平均により得られる．箱埋め込みの平均とは，箱の中心と辺の長さを表すベクトルのそれぞれについて平均ベクトルを計算することで求められる箱埋め込みである．

MetricWSD では，対象単語の文脈埋め込みを得るためのモデルとして Bidirectional Encoder Representations from Transformers (BERT) を用いていた．本研究では，対象単語の文脈箱埋め込みを得るために，BERT の最終層の後に全結合層をつなげたモデルを用いる．この層の入力は BERT の対象単語の最終層のベクトル表現であり，出力は文脈箱埋め込みである．全結合層の出力を半分に分割し，それぞれ箱の中心と辺の長さを表すベクトルとする．

語義の箱埋め込みを出力するモデルを学習するための損失関数は 2 つの語義の箱埋め込みの重なりから計算する．直感的には，ある語義の箱埋め込みは，自身もしくは上位概念の語義の箱埋め込みと重なるように，それ以外とは重ならないように学習される．損失を計算するためのデータセット (エピソード) を各語義ごとに用意し，各エピソードでは以下の手順でモデルを学習する．(1) 訓練データに出現する語義の集合の中から N_C 個の語義を選択する．(2) それぞれの語義について，その語義の用例をランダムに N_S 個選択し，サポートセットとする．(3) 対象語義について，サポートセットに含まれていない用例の中からランダムに N_Q 個の用例を選択し，クエリセットとする．(4) サポートセットに対するモデルの出力の平均から，各語義のプロトタイプ表現を得る．(5) クエリセットに対するモデルの出力とプロトタイプ表現を基に損失を計算する．(6) 損失を最小化するようにモデルのパラメータを更新する．

上記の手順 (1) において，N_C 個の語義を選択する 2 つの戦略を提案する．戦略 S_r では，まず対象語義とその上位語義を選び，残りをランダムに選ぶ．戦略 S_n では，対象語義とその上位語義に加えて下位語義と兄弟関係にある語義を選び，その後残りの語義をランダムに選ぶ．戦略 S_n では，ある語義の箱埋め込みが，特に下位語義や兄弟関係にある語義の箱埋め込みと重ならないことを重視している．

提案手法を WSD，新語義の判定，新語義の上位語義の推定の 3 つのタスクに適用し，その結果を評価する．実験に用いるデータセットとして，箱埋め込みを学習する語義の数による違いを調査するために，既存の WSD のデータセットから $\mathcal{D}_{\text{living_thing.n.01}}$，$\mathcal{D}_{\text{artifact.n.01}}$，$\mathcal{D}_{\text{entity.n.01}}$ という 3 つのデータセットを作成する．living_thing.n.01，artifact.n.01，entity.n.01 は WordNet における語義 (synset) であり，各データセットはこれらの synset より下位の概念をもつ用例から構成される．$\mathcal{D}_{\text{entity.n.01}}$ は名詞全体であり，$\mathcal{D}_{\text{living_thing.n.01}}$ と $\mathcal{D}_{\text{artifact.n.01}}$ は名詞のサブセットである．ベースラインとして，事前学習済みの BERT である BERT-NN と MetricWSD の 2 つを用意する．MetricWSD は，ベースモデルとして BERT を用いているため，BERT-NN をファインチューニングした手法といえる．

WSD の実験では，一部の名詞から構成されデータセットに出現する語義の数が少ないデータセット ($\mathcal{D}_{\text{living_thing.n.01}}$ と $\mathcal{D}_{\text{artifact.n.01}}$) では，提案手法はベースラインを上回った．しかし，名詞全体から構成され語義の数が多いデータセット ($\mathcal{D}_{\text{entity.n.01}}$) では，提案手法はベースラインを下回った．これは，語義の数が多くなると，語義の箱埋め込みが必ずしも適切に学習されないことを示唆するものであった．新語義の判定の実験では，WSD の結果とは異なり，名詞全体から構成され語義の数が多いデータセットでは，提案手法はベースラインを上回った．新語義の上位語義推定の実験では，提案手法は全体的にベースラインを上回った．上位語義の候補を絞り込む際に用いる閾値は 0.5，0.7，0.9 のいずれかと設定したが，データセットや戦略によって最良の結果が得られる閾値の設定は異なることがわかった．以上の結果から，提案手法によって学習された語義の箱埋め込みは，実験条件によっては単一のベクトルで表現された語義の埋め込み表現よりも優れていること，ま

た語義の上位下位関係を適切に表現できる能力を持つことが確認された,

目 次

第 1 章　はじめに **1**
1.1 背景 1
1.2 目的 2
1.3 本論文の構成 2

第 2 章　関連研究 **4**
2.1 BERT 4
2.2 WSD 6
2.2.1 語義の定義文を利用した WSD 6
2.2.2 意味関係を利用した WSD 9
2.2.3 系列変換モデルを利用した WSD 10
2.2.4 不均衡データ問題 10
2.2.5 MetricWSD 11
2.3 タクソノミ拡張 13
2.4 本研究の特徴 15

第 3 章　提案手法 **17**
3.1 語義グラフの構築 17
3.2 箱埋め込み 21
3.3 語義の箱埋め込みの学習 22
3.4 エピソードの作成 24
3.5 語義の箱埋め込みの応用 25
3.5.1 WSD 25
3.5.2 新語義の判定 26
3.5.3 新語義の上位語義の推定 26

第 4 章　評価 **28**
4.1 データセット 28
4.2 ベースライン 30
4.3 モデルの学習 31
4.4 結果と考察 32
4.4.1 WSD の実験結果 32
4.4.2 新語義判定の実験結果 34

4.4.3　新語義の上位語義推定の実験結果 36
4.5　語義の箱埋め込みの可視化 37

第 5 章　おわりに **42**
5.1　まとめ 42
5.2　今後の課題 42

図 目 次

1.1 語義の箱埋め込みによる上位語義の推定 2

2.1 BERT による文脈埋め込みの例 [11] 5
2.2 BEM の概要 [7] 7
2.3 ESCHER の概要 [4] 8
2.4 EWISE の概要 [21] 10
2.5 Prototypical Networks の概要 12
2.6 出現頻度別の 3 つの WSD 手法の比較 [10] 12
2.7 BERT-classifier と MetricWSD による語義の埋め込みの可視化 [10] . 13
2.8 TaxoExpan の概要 [37] 14
2.9 mini-path の例 [42] 14
2.10 STEAM における上位概念の予測 [42] 15

3.1 animal.n.01 を根とする語義グラフ 20
3.2 箱埋め込みの概念図 21
3.3 アイテム間の 3 つの関係 21
3.4 文脈箱埋め込みと語義の箱埋め込み 23
3.5 文脈箱埋め込みを得るためのモデル f_θ 24
3.6 損失計算における正例と負例（赤が正例，青が負例） 24

4.1 上位下位関係にある語義間の学習度スコア 33
4.2 閾値 α の分布 35
4.3 語義 animal.n.01（赤）と dog.n.01（青）の箱埋め込み（$\mathcal{D}_{\text{living_thing.n.01}}$，ProtoBox S_r） 38
4.4 語義 structure.n.01（黄）と house.n.01（青）の箱埋め込み（$\mathcal{D}_{\text{artifact.n.01}}$，ProtoBox S_r） 38
4.5 語義 animal.n.01（赤）と dog.n.01（青）の箱埋め込み（$\mathcal{D}_{\text{entity.n.01}}$，ProtoBox S_r） 39
4.6 語義 structure.n.01（黄）と house.n.01（青）の箱埋め込み（$\mathcal{D}_{\text{entity.n.01}}$，ProtoBox S_r） 39
4.7 語義 tree.n.01（赤）と dog.n.01（青）の箱埋め込み（$\mathcal{D}_{\text{living_thing.n.01}}$，ProtoBox S_r） 39

4.8 語義 art.n.01（黄）と house.n.01（青）の箱埋め込み（$\mathcal{D}_{\text{artifact.n.01}}$，ProtoBox S_r） 39
4.9 語義 cat.n.01（赤）と dog.n.01（青）の箱埋め込み（$\mathcal{D}_{\text{living_thing.n.01}}$，ProtoBox S_r） 40
4.10 語義 hotel.n.01（黄）と house.n.01（青）の箱埋め込み（$\mathcal{D}_{\text{artifact.n.01}}$，ProtoBox S_r） 40
4.11 語義 cat.n.01（赤）と dog.n.01（青）の箱埋め込み（$\mathcal{D}_{\text{living_thing.n.01}}$，ProtoBox S_n） 41
4.12 語義 hotel.n.01（黄）と house.n.01（青）の箱埋め込み（$\mathcal{D}_{\text{artifact.n.01}}$，ProtoBox S_n） 41

表 目 次

2.1 GlossBERT の入力と正解ラベルの例 [19] 6

3.1 WordNet における gloss と用例の例 18

4.1 訓練データの統計 29
4.2 開発データの統計 29
4.3 テストデータの統計 29
4.4 WSD タスクのテストデータの用例数 29
4.5 新語義判定タスクのテストデータの用例数 30
4.6 新語義の上位語義推定タスクのテストデータの用例数 30
4.7 WSD の結果 (正解率) 34
4.8 新語義の判定の結果 (F1 スコア) 34
4.9 開発データに出現しない単語に対する α 36
4.10 新語義の上位語義推定の結果 37

第1章　はじめに

本章では，本研究の背景と目的を述べる．1.1 節で背景，1.2 節で目的を述べる．最後に，1.3 節で本論文の構成を述べる．

1.1　背景

人間の言語は曖昧である．多くの単語は，その単語が出現する文脈により複数の意味（語義）をもつ．例として，以下の 2 つの文を考える．

文 1 I can hear <u>bass</u> sounds.

文 2 They like grilled <u>bass</u>.

2 つの文に現れる bass という単語は，文 1 では「低周波の音」，文 2 では「魚の一種」とそれぞれ異なる意味で用いられている．このように，1 つの単語が複数の意味をもつ性質を「多義性」といい，機械が人間の言語を理解する上で大きな問題となっている．単語の多義性に関する代表的な研究に語義曖昧性解消（Word Sense Disambiguation; WSD）がある．WSD とは，ある文脈における単語の意味として適切なものを，あらかじめ定義された語義の集合の中から 1 つ選ぶタスクである [29]．先ほどの例の場合，bass の複数ある語義の中から，文 1 における bass の意味として「低周波の音」，文 2 における bass の意味として「魚の一種」を選ぶことを目標とする．WSD は，情報検索，感情分析，機械翻訳などの前処理として行われている [36, 16, 9]．

WSD は自然言語処理分野において長年にわたり研究されてきた．近年の WSD の研究は，正解の語義が付与された用例集合を訓練データとして，ニューラルネットワークにより語義の分類モデルを教師あり学習することが主流である．ただし，従来の WSD の研究のほとんどは，訓練データに出現する語義の中からテストデータにおける単語の語義を選択することを想定している．しかし，語義は日々変化し，新しい語義も生まれている．そのため，単にあらかじめ定義された語義集合の中から該当する語義を選ぶのではなく，対象単語が訓練データに出現しない未知の語義（新語義）であるのかを判定することが求められる．さらに，新語義に対し，人間の理解を助けるような何らかの手がかりを提示することも重要な課題である．例えば，新語義の上位概念を提示することで，人間がその新語義の大まかな意味を理解するのを助けることができる．

1.2 目的

本研究は，語義の箱埋め込みを学習することを目的とする．語義の箱埋め込みとは，ベクトル空間における領域によって表される語義の抽象表現である．語義を箱埋め込みで表現することで，語義を単一のベクトルで表現していた従来の WSD の研究とは異なり，語義の概念的な大きさを表現できる．それにより，ある文脈における単語が新語義であるのかを判定したり，新語義の直接の上位概念（上位語義）を推定することが可能となる．図 1.1 の例では，x の箱埋め込みが cat や dog の箱埋め込みと重ならないためにこれらとは異なる語義であることがわかり，animal の箱埋め込みに包含されていることから上位語義は animal であることが推定できる．このように，語義の箱埋め込みは，テキスト内の新語義の自動検出や，WordNet[26] などの概念辞書の自動拡張に応用できる．

本論文では，既存の WSD の手法である MetricWSD[10] を語義の箱埋め込みを学習するよう拡張する．さらに，モデルの損失を計算するために用いる小さいデータセットを作成する 2 通りの方法を提案する．提案手法を WSD，新語義の判定，新語義の上位語義の推定の 3 つのタスクに適用し，語義を単一のベクトルで表現する従来手法と実験的に比較する．

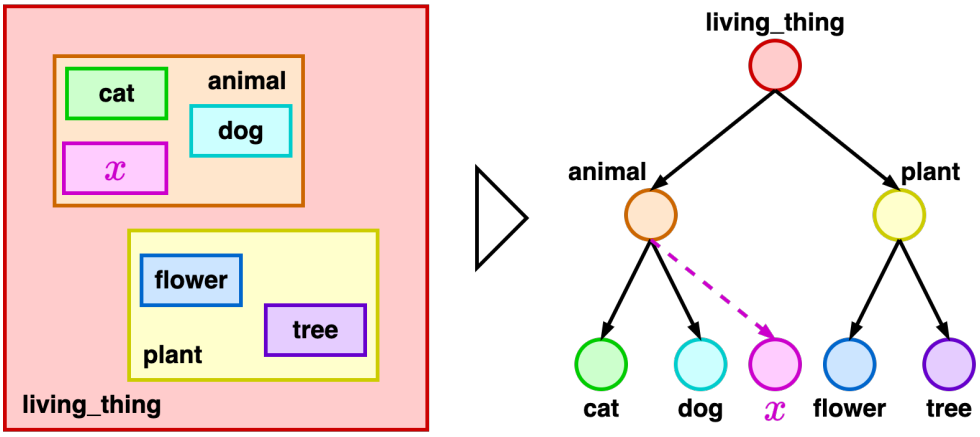


図 1.1: 語義の箱埋め込みによる上位語義の推定

1.3 本論文の構成

本論文の構成は以下の通りである．第 2 章では，本研究に関連する研究について述べる．また，それらと本研究を比べることにより，本研究の特徴を示す．第 3 章では，本研究の提案手法を詳述する．まず箱埋め込みの基本的な知識を述べ，次に語義の箱埋め込みを学習したりそれを応用タスクに適用する方法を提案する．第 4 章では，評価実験について説明する．実験に用いるデータセットを述べ，い

くつかのタスクで実験を行い，その結果を考察する．第 5 章では，本研究のまとめと今後の課題を述べる．

第2章　関連研究

本章では，本研究の関連研究について述べる．2.1 節では，汎用的な言語モデルである BERT を紹介する．BERT は様々な自然言語処理タスクに応用され，本研究の関連研究でも使われることから，まずこれについて紹介する．2.2 節では，WSD に関する研究について述べる．2.3 節では，タクソノミ拡張の研究について述べる．2.4 節では，先行研究と本研究の違いを論じ，本研究の特徴を述べる．

2.1　BERT

Bidirectional Encoder Representations from Transformers (BERT)[14] は，系列変換モデルである Transformer[40] のエンコーダ部分からなる言語モデルである．Transformer のエンコーダとデコーダはそれぞれ 6 層の Transformer Unit からなるが，BERT はエンコーダのみで 12 層（$\mathrm{BERT}_{\mathrm{BASE}}$）もしくは 24 層（$\mathrm{BERT}_{\mathrm{LARGE}}$）の Transformer Unit から構成される．BERT は膨大なテキストデータを用いてあらかじめ言語の基本的な知識を学習（Pre-training; 事前学習）しているため，感情分析や質問応答といった特定のタスクに対し，少量のデータで学習（Fine-Tuning; ファインチューニング）するだけで高い性能を発揮する．

事前学習では，膨大なテキストデータを用いてマスクの穴埋めタスクと次文予測タスクを行う．マスクの穴埋めタスクとは，文の一部をマスク処理（[MASK] トークンに置換すること）し，マスク処理された部分に元々あった単語を予測するタスクである．一方，次文予測タスクは，2 つの文があったとき，一方の文がもう一方の文の次に出現するかを予測するタスクである．上記の事前学習により，BERT は言語の基本的な知識を学習していると考えられる．

ファインチューニングでは，BERT を適用したいタスク（「下流タスク」と呼ばれる）のラベル付きデータを用いて，事前学習された BERT モデルのパラメタを下流タスクに合わせて更新する．例えば，感情分析タスクは，ある文が与えられたとき，その文の感情が positive，negative，neutral のうちどれに該当するかを予測するタスクである．感情分析タスクに BERT をファインチューニングするときは，入力文の先頭に新しく [CLS] トークンを追加し，BERT モデルの最終層の後に，[CLS] トークンに対する埋め込み表現を入力，positive/negative/neutral に対応する 3 次元ベクトルを出力とする全結合層を追加し，この出力と正解ラベルとの差を損失としてモデル全体のパラメタを更新する．別の下流タスクの例として，

自然言語推論タスクは，前提と仮定の 2 つの文が与えられたとき，それらの関係が含意，中立，矛盾のうちどれに該当するかを予測するタスクである．そのため，前提と仮定の 2 つの文を [SEP] トークンを用いて連結したトークン列を入力とし，感情分析と同様に，[CLS] トークンの最終層の後に 3 次元のベクトルを出力とする全結合層を追加することで含意/中立/矛盾の 3 つのクラスに対する分類確率を予測する．下流タスクが分類問題のときは BERT の最終層の後に全結合層を追加する場合が多いが，BERT に全結合層を追加せずにファインチューニングが行われることもある．例えば，文の類似度推定タスクでは，[CLS] トークンの最終層のベクトル表現を文の意味表現とし，類似している文の意味表現同士がベクトル空間上で近づくようにパラメタを学習する [17].

BERT は，入力文の各単語に対し，その単語の文脈における意味を表すベクトル（単語の文脈埋め込み）を出力する．例えば，asylum と madhouse の 2 つの単語の文脈埋め込みを学習したいとする．図 2.1 は，asylum と madhouse を含む用例を BERT に入力したときの対象単語の文脈埋め込み（768 次元）を 2 次元に圧縮し可視化したものである [11]．asylum は政治的な避難の意味で用いられることが多いが，精神病患者の監禁施設の意味でも用いられる．そのため，図 2.1 に示すように，asylum の文脈埋め込みの一部は，同じような意味をもつ madhouse の文脈埋め込みとベクトル空間上での距離が近い．このように，BERT では，単語に対して 1 つの埋め込みを学習するのではなく，文脈に応じた異なる埋め込みを学習することができる．このような特長から，BERT の文脈埋め込みは固有表現認識や語義曖昧性解消などの単語を分類するタスクでよく利用される．

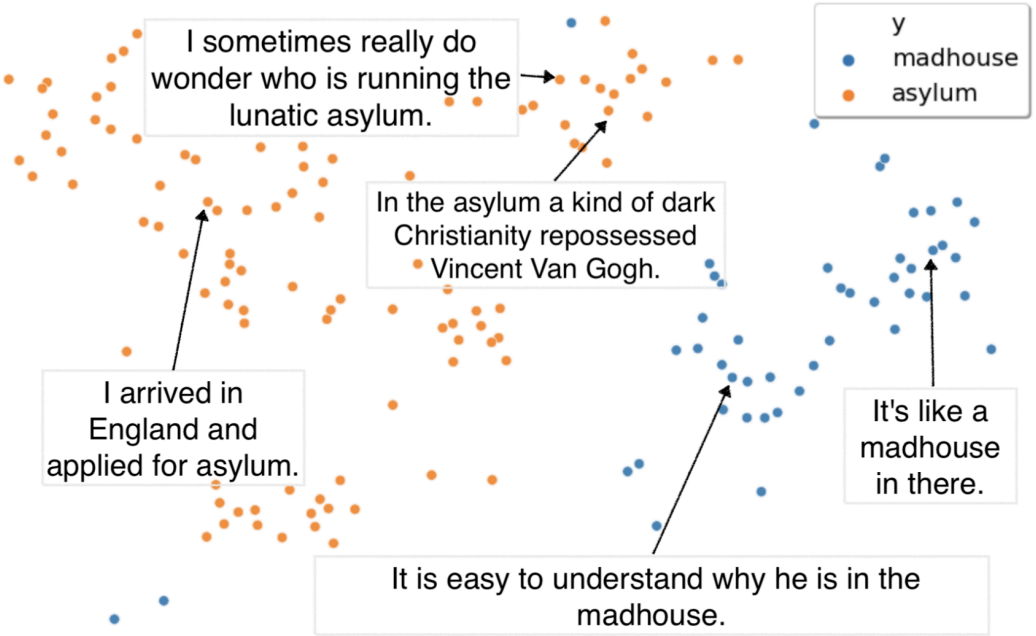


図 2.1: BERT による文脈埋め込みの例 [11]

2.2 WSD

WSD は長年にわたり研究されており，これまでに様々な手法が提案されてきた．最近では，WordNet[26] に記載された語義の定義文や語義間の上位下位関係といった用例以外の外部情報を利用することにより WSD の性能向上を目指す研究が主流である．2.2.1 項では語義の定義文を利用した WSD，2.2.2 項では意味関係を利用した WSD の手法について述べる．2.2.3 項では系列変換モデルを利用した WSD の手法を紹介する．WSD の研究には，上記の研究のように WSD モデルの学習を行う以外にも，従来の WSD 手法の問題点について調査した研究もある．2.2.4 項では，WSD で大きな問題とされる不均衡データ問題について述べる．2.2.5 項では，このような不均衡データ問題による影響を軽減する試みとして提案された MetricWSD という手法について述べる．

2.2.1 語義の定義文を利用した WSD

Huang らは，語義の定義文と BERT を用いて WSD を行う GlossBERT を提案した [19]．BERT への入力は，対象単語が含まれた用例と対象単語の語義の定義文を [SEP] トークンをセパレータとして連結したものである．正解ラベルは，対象単語の正解の語義の定義文が連結されている場合は Yes（正例），それ以外は No（負例）とする．GlossBERT の入力と正解ラベルの例を表 2.1 に示す．この例では research が対象単語であり，また research は 4 つの語義をもつ．また，この用例における正解の語義は 1 番目なので，1 番目の入力に対する正解ラベルは Yes，それ以外は No である．GlossBERT では BERT の最終層の後に 2 次元ベクトルを出力する全結合層を追加する．ファインチューニングの際は，全結合層の出力から語義の分類確率を計算し，正解の語義の確率との差から計算される損失を最小化するようにモデルのパラメータを更新する．テストの際は，対象単語が含まれた用例と対象単語の語義の定義文のすべてのペアを入力し，Yes の予測確率が最も大きいペアの語義を選択する．

表 2.1: GlossBERT の入力と正解ラベルの例 [19]

用例：
Your <u>research</u> stopped when a convenient assertion could be made.

入力（対象単語は research）	正解ラベル
[CLS] Your research ... [SEP] systematic investigation to ... [SEP]	Yes
[CLS] Your research ... [SEP] a search for knowledge [SEP]	No
[CLS] Your research ... [SEP] inquire into [SEP]	No
[CLS] Your research ... [SEP] attempt to find out in a ... [SEP]	No

Blevins と Zettlemoyer は，対象単語が含まれた用例をエンコードする Context Encoder と，対象単語の語義の定義文をエンコードする Gloss Encoder の 2 つのエンコーダを用いて WSD を行う Bi-Encoder Model (BEM) を提案した [7]．BEM の概要を図 2.2 に示す．対象単語を w，対象単語が含まれた用例のトークン列を $\mathbf{c} = [c_1, c_2, ..., c_i, ...c_n]$ とし，i 番目の単語を対象単語（つまり，$c_i = w$）とすると，用例 $\mathbf{c}$ における対象単語 w の文脈埋め込み $\mathbf{r}_w$ は，式 (2.1) に示すように，Context Encoder の出力 $T_c(\mathbf{c})$ の i 番目のベクトルとなる．

$$\mathbf{r}_w = T_c(\mathbf{c})[i] \tag{2.1}$$

さらに，対象単語 w の語義の定義文の集合を $\mathcal{G} = \{\mathbf{g}_1, \mathbf{g}_2, ..., \mathbf{g}_m\}$ とすると，各語義の定義文の意味表現 $\mathbf{r}_{g_i}$ は，式 (2.2) に示すように，Gloss Encoder の出力 $T_g(\mathbf{g}_i)$ の最初のベクトルとなる．Context Encoder と Gloss Encoder はともに BERT であるため，最初のトークンは [CLS] トークンであり，これに対応する埋め込み表現（ベクトル表現）が語義の定義文全体の意味表現として用いられる．

$$\mathbf{r}_{g_i} = T_g(\mathbf{g}_i)[0] \tag{2.2}$$

w と $\mathbf{g}_i$ の類似度 $\phi(w, \mathbf{g}_i)$ は，式 (2.3) に示すように，$\mathbf{r}_w$ と $\mathbf{r}_{g_i}$ の内積で計算される．

$$\phi(w, \mathbf{g}_i) = \mathbf{r}_w \cdot \mathbf{r}_{g_i} \tag{2.3}$$

w の語義の定義文が $\mathbf{g}_i$ である確率は式 (2.4) で計算される．

$$P(w, \mathbf{g}_i) = \frac{\exp(\phi(w, \mathbf{g}_i))}{\sum_{j=1}^{m} \exp(\phi(w, \mathbf{g}_j))} \tag{2.4}$$

w の正解の語義の定義文が $\mathbf{g}_i$ だとすると，損失は式 (2.5) で計算される．

$$L = -\log P(w, \mathbf{g}_i) \tag{2.5}$$

この損失を最小化するように Context Encoder と Gloss Encoder の両方のパラメータを更新する．

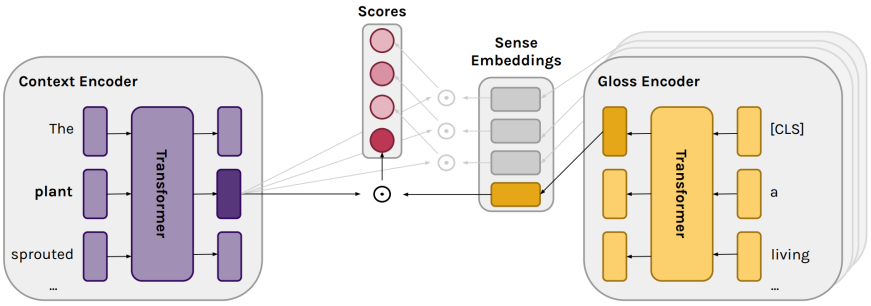


図 2.2: BEM の概要 [7]

Barba らは，対象単語が含まれた用例と対象単語のすべての語義の定義文を連結したものを入力とし，正解の語義の定義文のスパンを抽出することで WSD を行う ESCHER を提案した [4]．ESCHER の概要を図 2.3 に示す．対象単語を $\hat{w}$，対象単語が含まれた用例を $\mathbf{w} = [w_1, w_2, ..., \hat{w}, ..., w_n]$ とし，対象単語の語義の定義文の集合を $\mathcal{D} = \{d_1, d_2, ..., d_k\}$ とすると，入力は以下のようになる．

$$
\begin{aligned}
m = &<s>\ w_1\ w_2\ ...\ <t>\ \hat{w}\ </t>\ ...\ w_n\ </s> \\
&w_1^{d_1}\ w_2^{d_1}\ ...\ w_{|d_1|}^{d_1}\ w_1^{d_2}\ w_2^{d_2}\ ...\ w_{|d_2|}^{d_2}\ ...\ w_1^{d_k}\ w_2^{d_k}\ ...\ w_{|d_k|}^{d_k}\ </s>
\end{aligned}
$$

$<s>$ と $</s>$ は用例と語義の定義文を分けるための特殊トークンであり，$<t>$ と $</t>$ は用例中の対象単語を識別するための特殊トークンである．モデルの構造は Transformer[40] をベースとしたモデルの後に 2 つの全結合層を追加したものである．これらは，正解となる語義の定義文のスパンの開始位置と終了位置を予測するために用いられる．入力系列の長さを l，上記の 2 つの全結合層の出力を $Z^s = [Z_{11}, Z_{12}, ..., Z_{1l}]$ と $Z^e = [Z_{21}, Z_{22}, ..., Z_{2l}]$ とし，正解の語義の定義文の開始位置を $start$，終了位置を end とすると，それらの予測確率はそれぞれ式 (2.6) と式 (2.7) で計算される

$$
P(w_i = start|Z^s) = \frac{\exp(Z_i^s)}{\sum_{j=1}^{l} \exp(Z_j^s)} \tag{2.6}
$$

$$
P(w_i = end|Z^e) = \frac{\exp(Z_i^e)}{\sum_{j=1}^{l} \exp(Z_j^e)} \tag{2.7}
$$

損失はこれら 2 つの確率の負の対数尤度の平均で計算され，それが最小となるようにモデルのパラメータを更新する．

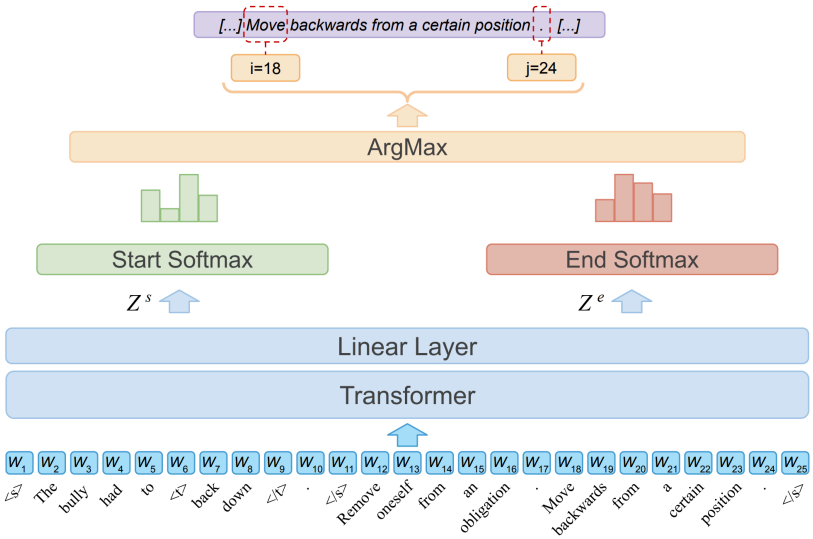


図 2.3: ESCHER の概要 [4]

2.2.2 意味関係を利用したWSD

Kumar らは，語義の定義文だけでなく語義間の意味関係も利用して WSD を行う EWISE を提案した [21]．EWISE の概要を図 2.4 に示す．EWISE は (i) 対象単語を含む用例をエンコードするためのモデル（図の Attentive Context Encoder），(ii) 対象単語の語義の定義文をエンコードするためのモデル（図の Definition Encoder），(iii) 知識グラフの埋め込み表現を得るためのモデル（図の Knowledge Graph Embedding）の 3 つのモデルから構成される．(i) と (ii) は Bi-directional LSTM (BiLSTM) であり，(iii) は ConvE[13] と呼ばれる知識グラフの埋め込みモデルである．知識グラフは head，relation，tail の 3 つ組 (h, l, t) の集合としてしばしば表現される．head と tail はグラフのノードであり，relation は head と tail を結ぶエッジである．EWISE の場合，head と tail には語義が該当し，relation には上位下位関係や part_of 関係が該当する．ConvE は，multi-layer convolutional network を用いて知識グラフの埋め込みを学習する手法であり，entity ならびに relation の埋め込み表現を出力する．モデル (i) の学習は，モデル (ii) と (iii) の学習とは別に行われる．

まず，モデル (i) の学習について説明する．対象単語を w，対象単語が含まれた用例を $\mathbf{c}$，対象単語 w の語義の定義文の集合を $\mathcal{G} = \{\mathbf{g}_1, \mathbf{g}_2, ..., \mathbf{g}_m\}$ とし，用例 $\mathbf{c}$ における対象単語 w の意味表現（モデル (i) の出力ベクトル）を $\mathbf{r}_w$，語義の定義文それぞれの意味表現（モデル (ii) の出力ベクトル）を $\mathbf{r}_{g_i}$ とすると，モデル (i) は 2.2.1 項で述べた BEM[7] と同じ方法で学習される．

次に，モデル (ii) と (iii) の学習について説明する．対象単語の正解の語義を h，h の定義文に対するモデル (ii) の出力ベクトルを $\mathbf{q}$，ある語義 t と relation l に対するモデル (iii) の出力ベクトルを $\mathbf{e}_t$ と $\mathbf{e}_l$ とすると，h と t の relation が l であるスコアは式 (2.8) で計算される．

$$\psi_l(h, t) = f(\mathrm{vec}(f([\overline{\mathbf{q}}; \overline{\mathbf{e}_l}] * \omega))\mathbf{W})\mathbf{e}_t \tag{2.8}$$

$\mathbf{q}$ だけでなく $\mathbf{e}_t$ もモデル (ii) の出力により得ることができるが，モデル (ii) の計算グラフのサイズを抑えるためにモデル (iii) の出力を利用している．$\overline{\mathbf{q}}$ と $\overline{\mathbf{e}_l}$ は $\mathbf{q}$ と $\mathbf{e}_l$ を 2 次元に縮退したものであり，ω は畳み込みのフィルタである．確率は $\psi_l(h, t)$ にシグモイド関数 σ を適用した式 (2.9) で計算される．

$$P_l(h, t) = \sigma(\psi_l(h, t)) \tag{2.9}$$

損失は式 (2.10) の binary cross-entropy で計算される．

$$L = -\delta \cdot \log P_l(h, t) - (1 - \delta) \cdot \log(1 - P_l(h, t)) \tag{2.10}$$

δ は 0 もしくは 1 の値をとる変数であり，h と t の relation が l である場合に 1，それ以外の場合には 0 をとる．この損失を最小化するようにモデル (ii) と (iii) の両方のパラメータを更新する．

なお，意味関係を利用した WSD の他の手法として EWISER[6] がある．EWISER は EWISE を改良した手法である．

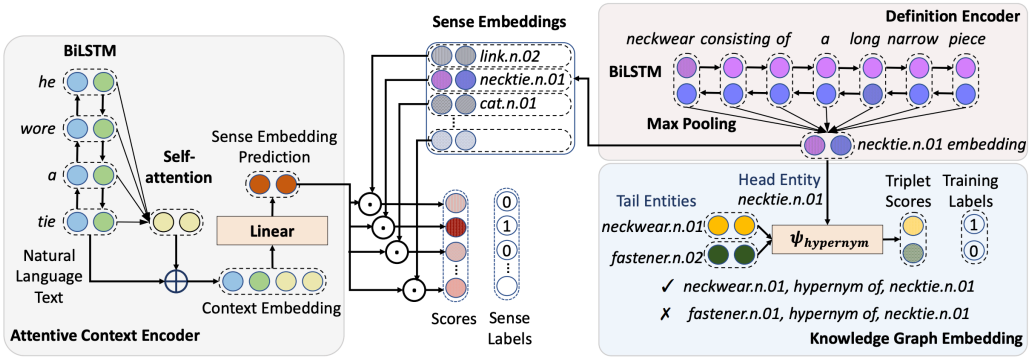


図 2.4: EWISE の概要 [21]

2.2.3 系列変換モデルを利用した WSD

Bevilacqua らは，系列変換モデルのひとつである BART[22] により語義の定義文を生成することで WSD を行う Generationary を提案した [5]．入力は用例中の対象単語を ⟨define⟩ と ⟨/define⟩ の 2 つの特殊トークンで挟んだものである．例えば，“I scooted them into the dog run” という用例があり，対象単語が scooted である場合，入力は “I ⟨define⟩ scooted ⟨/define⟩ them into the dog run” となる．出力は対象単語の定義文である．対象単語の語義を推定する際には，語義の定義文の集合 $\mathcal{G} = \{g_1, g_2, ..., g_m\}$ の中から，モデルが出力した語義の定義文 $\hat{g}$ と最も近いものを選び，それに対応する語義を選択する．そのために，2 文間の類似度を計算する必要がある．そこで，Sentence-BERT[35] によって $\mathcal{G}$ 内のすべての文と $\hat{g}$ のベクトル表現を取得し，それらのコサイン類似度が最大となる語義を選ぶ．この研究では，WSD の性能を向上させるだけでなく，既存の語義の定義文よりもさらに良い定義文を生成することや，新語義を説明することも目的としている．

2.2.4 不均衡データ問題

Maru らは，GlossBERT，BEM，ESCHER，EWISER，Generationary を含む 7 つの代表的な WSD の手法を取り上げ，それらの手法が WSD のデータセットにおける語義の偏りから受ける影響について調べた [25]．WSD のデータセットには，(i) 単語の出現頻度と (ii) 語義の出現頻度の 2 つの偏りがあることが知られている．単語の出現頻度の偏りは，高頻度の単語よりも低頻度の単語の方がはるかに多く，データセット中の単語の出現頻度の分布が長い尾のようになっていることを意味する．語義の出現頻度の偏りは，データセット中のある単語の語義のほとんどが

1 つの語義であることを意味する．その語義はたいてい，WordNet[26] で最初に書かれている（最もよく知られている）語義である．この研究では，7 つのモデルすべてで，WordNet で最初に書かれている語義でない語義（低頻度語義）が正解である場合に，正解率が大幅に低下することが報告されている．

2.2.5 MetricWSD

Chen らは，Prototypical Networks[38] とよばれるメタ学習手法を利用して WSD を行う MtericWSD を提案した [10]．Prototypical Networks は，分類問題を解くモデルを学習するためのメタ学習手法である．図 2.5 にその概要を示す．Prototypical Networks における学習の手順を以下に示す．

1. 各分類クラスの訓練データをサポートセットとクエリセットに分割する．
2. サポートセットに対するモデル f_θ の出力の平均から，各分類クラスのプロトタイプ表現を得る．
3. クエリセットに対するモデル f_θ の出力とプロトタイプ表現を基に損失を計算する．k 個の分類クラスのプロトタイプ表現を $\mathcal{C} = \{\mathbf{c}_1, \mathbf{c}_2, ..., \mathbf{c}_k\}$ とすると，クエリセット中の用例 x' が正解クラス i に分類される確率は式 (2.11) で計算され，損失は式 (2.12) で計算される．

$$P(x', i) = \frac{\exp(-d(f_\theta(x'), \mathbf{c_i}))}{\sum_{j=1}^{k} \exp(-d(f_\theta(x'), \mathbf{c_j}))} \tag{2.11}$$

$$L = -\log P(x', i) \tag{2.12}$$

4. 式 (2.12) の損失が最小となるようにモデル f_θ のパラメータを更新する．

推論の際は，訓練データからサポートセットをサンプリングすることでプロトタイプ表現を獲得し，テストデータをクエリセットとすることで，式 (2.11) が最大となるクラス i に分類する．

MetricWSD は Prototypical Networks を WSD に適用した手法である．分類クラスは語義であり，プロトタイプ表現が語義のベクトルに該当する．訓練データは語義が注釈された文（用例）の集合であり，モデル f_θ は用例中の対象単語の文脈埋め込みを生成する．MetricWSD ではモデル f_θ として BERT を用いている．MetircWSD は，語義の定義文や語義間の上位下位関係といった追加の語彙情報を用いない WSD の手法の中では，最も高い分類性能をもつ．

2.1 節で述べたように，BERT をファインチューニングするときは，BERT の最終層の後に全結合層を追加したモデルを用いることがある．この研究では，そのような標準的な BERT モデルによって語義の曖昧性を解消する手法（BERT-classifier）と MetricWSD との比較も行っている．図 2.6 は，ファインチューニングを行わ

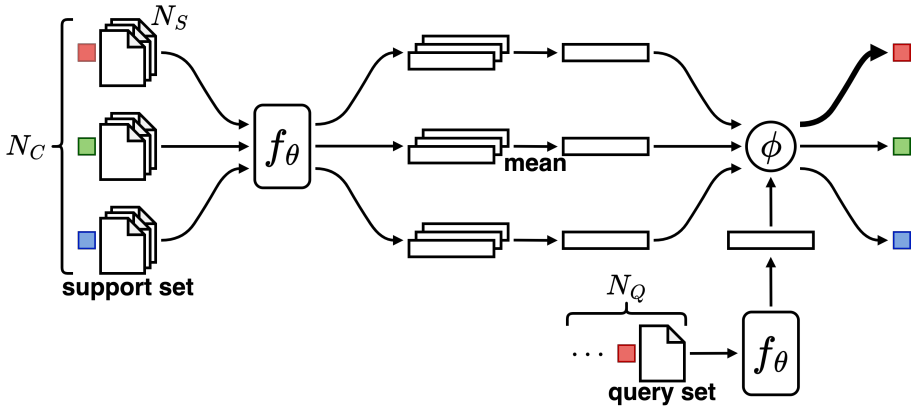


図 2.5: Prototypical Networks の概要

ずに事前学習された BERT モデルから得られる文脈埋め込みを用いた k-Nearest Neighbor 法で WSD を行う BERT-kNN，BERT-classifier，MetricWSD の 3 つの手法について，WSD の正解率を示している．また，単語の出現頻度ならびに語義の出現頻度によってテストデータを分け，高頻度または低頻度の単語または語義に対して 3 つの手法の性能を比較している．この図から，MetricWSD は基本的に BERT-classifier より高い分類性能をもつことがわかる．特に，出現頻度の低い単語や語義に対し性能の差が大きいこともわかる．また，BERT-kNN も出現頻度の低い単語や語義に限っては，BERT-classifier 以上の分類性能をもつことがわかる．BERT-classifier のように単に BERT をファインチューニングするだけでは出現頻度の低い単語や語義に対して高い正解率が得られないことは，Maru らの研究 [25] で述べられていた事実と一致する．

BERT-classifier と MetricWSD の対象単語の BERT の最終層のベクトル表現（768 次元）を 2 次元に圧縮して可視化したものを図 2.7 に示す．左 2 つのグラフは動詞 note，右 2 つのグラフは動詞 provide のベクトル表現であり，色の違いは語義の違いを表す．この図から，BERT-classifier に比べて，MetricWSD の方が，同じ語義のベクトル表現が近くなるようにモデルが学習されていることがわかる．

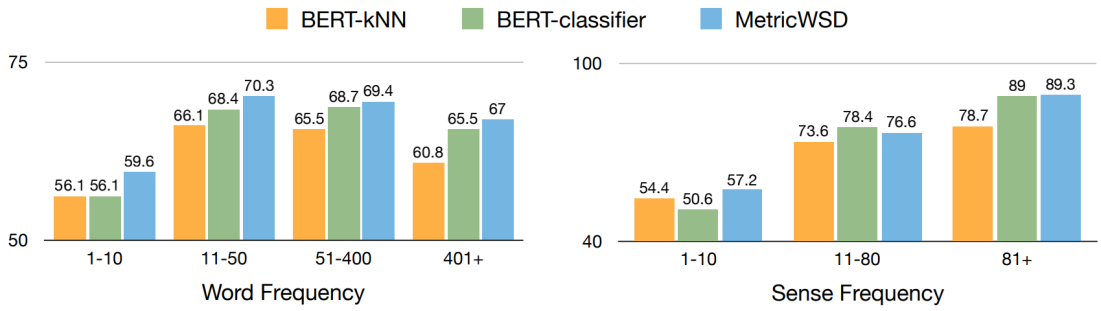


図 2.6: 出現頻度別の 3 つの WSD 手法の比較 [10]

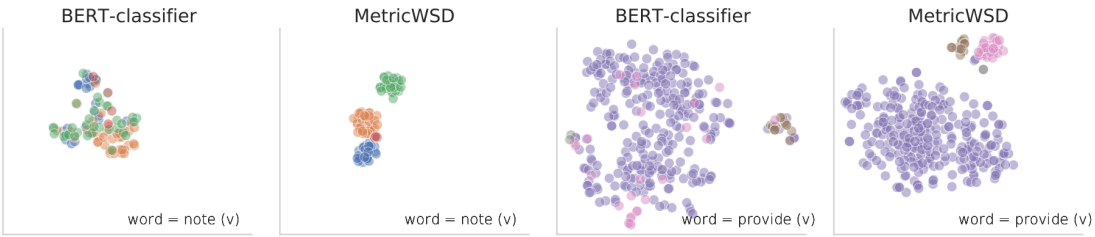


図 2.7: BERT-classifier と MetricWSD による語義の埋め込みの可視化 [10]

2.3 タクソノミ拡張

上位下位関係からなる知識グラフをタクソノミとよぶ．タクソノミは，知識の索引付けや整理に活用できるため，ウェブ検索や推薦システムなどで利用される [23, 18]．タクソノミ拡張（Taxonomy Expansion）とは，あるタクソノミと，そこに含まれない概念を含む文脈が与えられたとき，その概念のタクソノミにおける直接の上位概念（親）を特定するタスクである [8]．例えば，1.2 節で例に挙げた図 1.1 の右のグラフでは，新しい概念 x の親が概念 animal であると特定されている．WSD と同様に，タクソノミ拡張も盛んに研究されている．本節では，タクソノミ拡張の代表的な手法をいくつか述べる．

Shen らは，グラフニューラルネットワーク（Graph Neural Neteork; GNN）を用いてタクソノミ拡張を行う TaxoExpan を提案した [37]．タクソノミは有向非循環グラフであるため，グラフを扱うニューラルネットワークである GNN を自然に利用できる．TaxoExpan の概要を図 2.8 に示す．TaxoExpan は (i) 新しく追加される概念（クエリ）の親の近傍からなるグラフ（Egonet）をエンコードする GNN モデルと，(ii) あらかじめ学習されたクエリのベクトル表現 h_q とモデル (i) によりエンコードされた Egonet のベクトル表現 h_G との関連度を計算するモデル (Multilayer Perceptron (MLP))，といった 2 つのモデルから構成される．モデル (ii) は，図 2.8 における Matching Module に相当し，その入力は h_q と h_G を連結したものである．クエリを n^q，クエリの親を n^p，損失を計算する際に用いるノードの集合を $\mathcal{X} = \{n_1, n_2, ..., n_N\}$ とし，上記 2 つのモデルを用いてノード間の関連度を返す関数を f とすると，n^p を n^q の親と予測する確率は式 (2.13) で計算される．

$$P(n^p, n^q) = \frac{f(n^p, n^q)}{\sum_{i=1}^{N} f(n_i, n^q)} \tag{2.13}$$

モデルの学習は，式 (2.14) で計算される損失を最小化するようにモデル (i) と (ii) のパラメータを更新することで行われる．

$$L = -\log P(n^p, n^q) \tag{2.14}$$

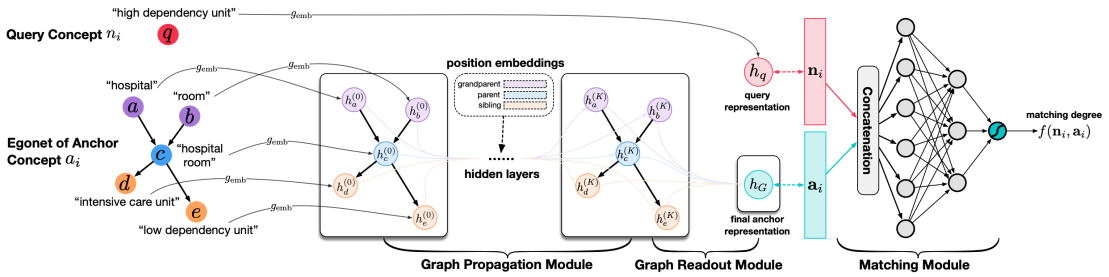


図 2.8: TaxoExpan の概要 [37]

Yu らは，葉ノードから根ノードまでの最短パス（mini-path）のうち対象ノードがどこに位置するかを学習することでタクソノミ拡張を行う STEAM を提案した [42]．図 2.9 に示すように，すべての mini-path からなる集合をあらかじめ取得しておき，それを学習と推論で用いる．

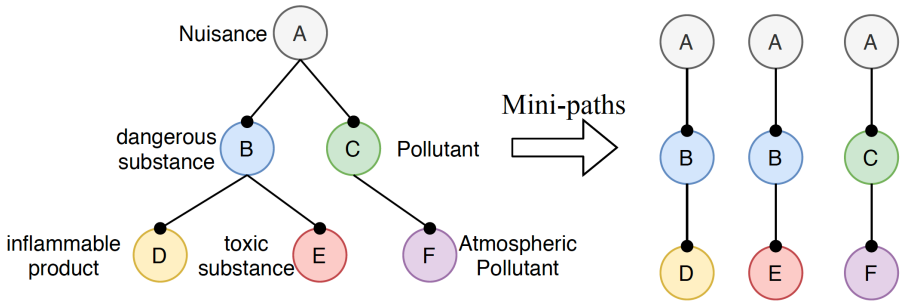


図 2.9: mini-path の例 [42]

学習の際は，まず，図 2.10 に示すように，いくつかの mini-path とそれらに含まれない概念 G を含む用例が与えられる．そして，mini-path 内の各概念について，その概念が G の親であるスコアを計算する．スコアは，mini-path 内の各概念や G について，その概念が含まれた用例の文脈情報や構文情報などを基に計算される．また，G の親が mini-path におけるどの概念にも該当しないときのスコアも計算する．最後に，それらのスコアから損失を計算し，その損失を最小化するようにモデルのパラメータを更新する．
推論の際は，新しく追加される概念（クエリ）q の親をすべての mini-path を用いて予測する．現在のタクソノミ内の概念 p について，p が含まれた mini-path の集合を $\mathcal{P}_p = \{P_1, P_2, ..., P_m\}$ とし，P_i において p が q の親となるスコアを y_{P_i} とすると，p が q の親となる最終的なスコアは式 (2.15) で計算される．

$$y_p = \frac{1}{|\mathcal{P}_p|} \sum_{i=1}^{m} y_{P_i} \tag{2.15}$$

式 (2.15) のスコアが最大である概念をクエリの親と予測する．

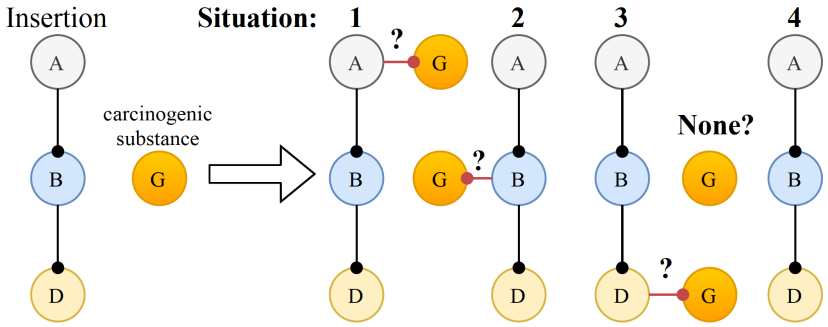


図 2.10: STEAM における上位概念の予測 [42]

Jiang らは，箱埋め込みを用いてタクソノミ拡張を行う BoxTaxo を提案した [20]．箱埋め込みはベクトル空間上における領域であり，それらの重なりから概念間の上位下位関係を自然に表現できる．そのため，TaxoExpan[37] や STEAM[42] と比べ，より直感的な視点からタクソノミ拡張を行うことが可能となる．箱埋め込みは 2 つのベクトル $\mathbf{c}$ と $\mathbf{o}$ のペア $\mathbf{b} = (\mathbf{c}, \mathbf{o})$ で定義できる．$\mathbf{c}$ は箱の中心，$\mathbf{o}$ は箱の辺の長さを表すベクトルである．モデルは BERT の [CLS] トークンの最終層の後に 2 つの Multilayer Perceptron (MLP) をつなげたものであり，2 つの MLP の出力をそれぞれ $\mathbf{c}$ と $\mathbf{o}$ とする．この BERT への入力は，タクソノミ内の概念 e とその定義文 s をカンマによって連結したもの（“[CLS] e, s [SEP]”）である．学習の際は，箱埋め込みの大きさと重なりから損失を計算し，それを最小化するようにパラメータを更新する．学習が終わると，モデルは，上位概念の箱埋め込みが下位概念の箱埋め込みを包含するような箱埋め込みを生成する．

箱埋め込みと同じく，上位下位関係を表現できる埋め込みである双極埋め込みを用いてタクソノミ拡張を行う研究もある [3, 24]．双極埋め込みは，アイテムを，ユークリッド空間のようなまっすぐな空間ではなく，曲がった空間である双極空間で表現する埋め込みである．これらの研究では，双極空間のひとつであるポワンカレ球 [31] を双極埋め込みとして用いている．ポワンカレ球は，中心に近づくほど情報が荒く概念的・意味的に大きいものが埋め込まれ，中心から遠ざかるほど情報が細かく概念的・意味的に小さいものが埋め込まれる．

2.4 本研究の特徴

従来の WSD の研究のほとんどは，語義ないしは語義を含む用例をベクトルで表現し，これを基に語義の曖昧性を解消していた．これに対し，本研究では，語義の抽象表現として単一のベクトルではなく箱埋め込みを学習する点に新規性がある．語義の箱埋め込みを学習することで，語義間の上位下位関係を自然に表現することが可能となり，タクソノミ拡張など様々な場面に応用できる．また，語義の箱埋め込みを用いることで WSD の性能を向上させることも狙う．Jiang らの研

究 [20] も箱埋め込みを用いて概念を表現する点は本研究と共通しているが，本研究では，Prototypical Networks[38] に基づき，概念の文脈箱埋め込みの平均を概念の箱埋め込みとする点が異なる．

従来の WSD の問題設定では，訓練データに出現する語義の中からテストデータにおける単語の語義を選択することを想定していた．これに対し，本研究は，テストデータの単語の語義が新語義であるかを判定したり，新語義の上位語義を推定することも目的としている．Bevilacqua らの研究 [5] も新語義を説明することを目的としている点で本研究と近い．しかし，彼らの研究では定義文を生成することにより新語義を説明するが，本研究では上位語義を推定することにより新語義の大まかな概念を説明する点が異なる．

第3章　提案手法

本章では，語義の箱埋め込みを学習する提案手法について述べる．3.1 節では訓練データ内の語義から構成される語義グラフを構築する方法を述べる．3.2 節では箱埋め込みの基本的な概念を説明する．3.3 節では語義の箱埋め込みを学習するモデルのアーキテクチャや，そのモデルを学習する際の損失関数について述べる．3.4 節では損失を計算する際に用いるデータセットをサンプリングする方法を述べる．3.5 節では学習した語義の箱埋め込みを応用タスクに適用する方法を述べる．

3.1　語義グラフの構築

提案手法では語義グラフを必要とする．語義グラフとは，単語の語義をノード，それらの間の上位下位関係をリンクとするグラフである．語義の箱埋め込みは語義付きコーパス，すなわち正しい語義が付与された用例（例文）の集合から学習される．語義付きコーパスが与えられたとき，まず，そのコーパスに付与されている語義から構成される語義グラフを構築する．

WordNet[26] は，Princeton 大学によって開発された大規模な語彙データベースである．WordNet では，名詞（noun），動詞（verb），形容詞（adjective），副詞（adverb）の単語が同じ概念をもつ集合にまとめられ，その集合が示す概念の簡単な定義や他の集合との関係が記述されている．

WordNet では単語間の様々な関係が記述されているが，その主な関係のひとつに同義性がある．同義性の関係にある単語（同義語）は，同じ概念（語義）を示すため，多くの文脈で交換可能である．同義語の例として，shut と close，car と automobile などが挙げられる．また，WordNet では，同義語からなる集合を synset と呼ぶ．先ほどの例では，shut と close は close.v.01，car と automobile は car.n.01 という名前の synset に属している．ここで，synset の名前の中の「v」と「n」は，それぞれ動詞と名詞を意味している．

WordNet には，各 synset に対し，それが示す概念の簡単な定義（gloss）や使用例（用例）が記載されている．WordNet 3.1 における synset の gloss と用例の例を表 3.1 に示す．

WordNet では synset 間にも様々な関係が記述されているが，その主な関係は上位下位関係（hypernym-hyponym relation）である．ある synset に対し，それよりも上位の概念をその synset の hypernym，下位の概念を hyponym と呼ぶ．例えば，

表 3.1: WordNet における gloss と用例の例

synset	close.v.01
gloss	move so that an opening or passage is obstructed; make shut
用例	“Close the door”，“shut the window”
synset	car.n.01
gloss	a motor vehicle with four wheels; usually propelled by an internal combustion engine
用例	“he needs a car to get to work”

bed.n.01 は bunk.n.03 の hypernym であり，bunk.n.03 は bed.n.01 の hyponym である．上位下位関係にある synset をつなげることで，synset の集合は全体として階層構造をもつ．名詞の場合，名詞を表す synset の集合は，entity.n.01 を根とする 1 つの有向非循環グラフになっている．つまり，すべての名詞は，直接の上位概念（上位語義）を辿る操作を繰り返すことで，最終的に entity.n.01 に到達する．

本研究では，語義の上位下位関係を基に語義の箱埋め込みを学習するため，WordNet の synset をノードとする語義グラフを用いる．ただし，多くの場合，訓練データに出現する語義は，WordNet におけるすべての語義ではなく，その一部である．そのため，ある語義の直近の上位概念（上位語義）を参照したとき，その語義が訓練データに存在せず，何も情報が得られないといったことが起こりうる．そのような問題を避けるために，訓練データに出現する語義のみで語義グラフを構築する．その手順を Algorithm 1 に示す．簡単に説明すると，訓練データに出現する語義集合の各語義について，その上位語義を取得し，元の語義とその上位語義の間にエッジを張る操作を繰り返す．このとき，WordNet から取得した上位語義の中で，(i) 訓練データに存在する語義はそのままグラフに加える．(ii) 訓練データに存在しない語義はその語義の上位語義を WordNet から取得し，再び (i) と (ii) の処理を行う．

Algorithm 1 語義グラフの構築

```
S ← SYNSETS_IN_DATASET()                          ▷ 訓練データに出現する語義
G ← MAKE_GRAPH(S)                                 ▷ 語義グラフ（有向非循環グラフ）

function MAKE_GRAPH(S)
    G ← ∅
    for s ∈ S do
        H ← HYPERNYM_IN_DATASET(s)                ▷ s の上位語義を取得
        for h ∈ H do
            G ← G ∪ {(h, s)}                      ▷ グラフにエッジを追加
        end for
    end for
    return G
end function

function HYPERNYM_IN_DATASET(s)
    H ← ∅                                         ▷ 語義グラフにおける s の上位語義
    T ← WORDNET_HYPERNYMS(s)                      ▷ WordNet における s の上位語義
    for t ∈ T do
        if t in S then
            H ← H ∪ {t}                           ▷ t が訓練データにあれば H に追加
        else
            H ← H ∪ HYPERNYM_IN_DATASET(t)        ▷ なければ上位語義を追加
        end if
    end for
    return H
end function
```

語義グラフの作成例として，animal.n.01 を根とする語義グラフを図 3.1 に示す．各ノードは synset（語義）を意味しており，エッジは直接の上位概念（上位語義），下位概念（下位語義）の関係を意味している．また，この語義グラフ中のすべての語義は，上位語義を辿る操作を繰り返すことで，最終的に animal.n.01 に行き着く．この図では，(i) 複数の下位語義をもつ語義，(ii) 複数の上位語義をもつ語義，(iii)1 つしか上位語義と下位語義をもたない語義があることがわかる．(i) は多くの語義に該当するが，特に mammal.n.01 や bird.n.01 は他の語義より多くの下位語義をもつ．(ii) は puppy.n.01 や chick.n.01 などのごく一部の語義が該当する．例えば，puppy.n.01 の意味（子犬）が pup.n.01 の意味（イヌ科動物の子供）と dog.n.01 の意味（犬）の両方に包含されていることを意味する．(iii) も多くの語義に該当している．WordNet では，上位語義が 1 つであることはよくあるが，下位語義が 1 つであることはあまりない．しかし，本研究では，訓練データに出現する語義のみで語義グラフを構築したため，下位語義を 1 つしかもたない語義も多い．

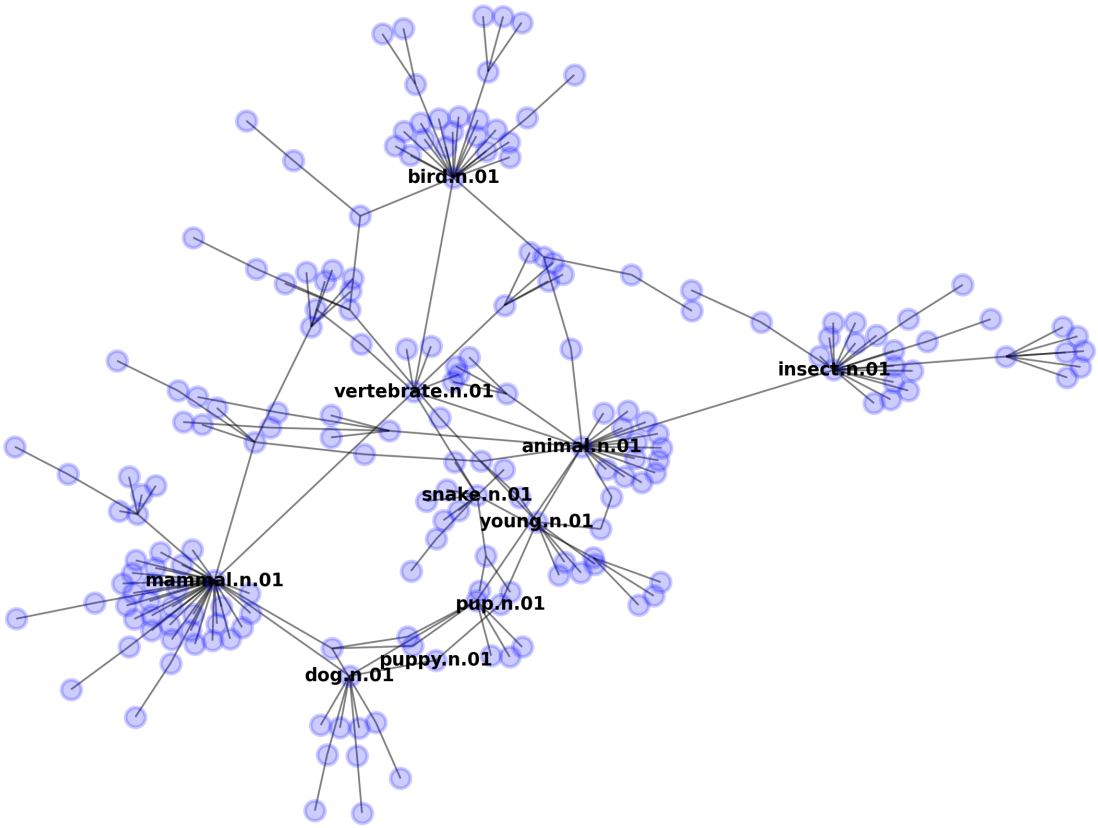


図 3.1: animal.n.01 を根とする語義グラフ

3.2 箱埋め込み

箱埋め込みは，アイテムをベクトル空間上で超直方体として領域的に表現する埋め込みであり，非対称的な関係を自然に表現できる．本研究では，先行研究 [32, 20] に倣い，箱埋め込みを $\mathbf{b} = (\mathbf{c}, \mathbf{o})$ と表現する．$\mathbf{c}$ は箱の中心，$\mathbf{o}$ は箱の辺の長さを表すベクトルであり，両者の次元数は同一である．図 3.2 に中心と辺の長さで表された箱埋め込みの概念図を示す．

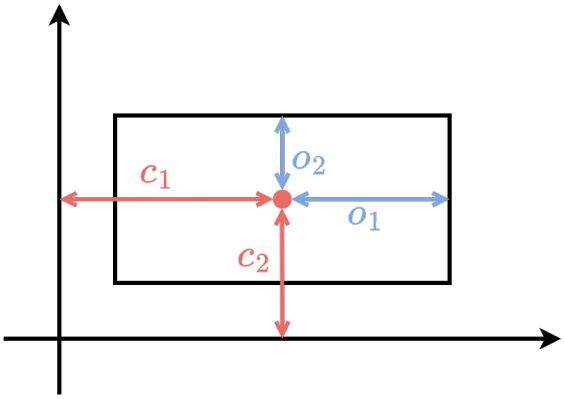
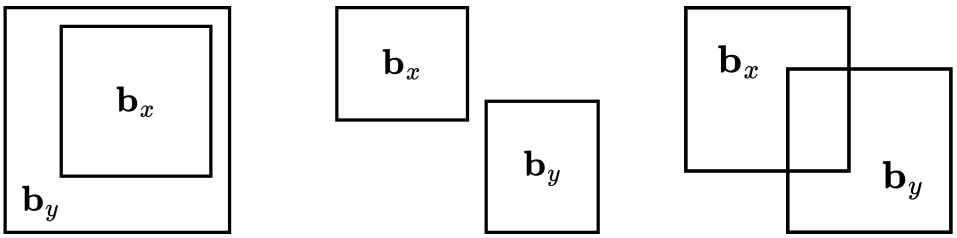


図 3.2: 箱埋め込みの概念図

箱埋め込みでは，単一のベクトルと異なり，2 つのアイテム間に図 3.3 のような 3 つの状況が存在する．図 3.3a の enclose は一方の箱がもう一方の箱を完全に包含している状況であり，大きい箱のアイテムが小さい箱のアイテムを概念的・意味的に包含していることを意味する．図 3.3b の disjoint は 2 つの箱がまったく重なっていない状況であり，2 つのアイテム間に概念的・意味的な関係がないことを意味する．図 3.3c の intersect は 2 つの箱が部分的に重なっている状況であり，2 つのアイテムの一部が概念的・意味的に共通していることを意味する．2 つのアイテムの一部が概念的・意味的に共通している状況の例として，3.1 節で述べた pup.n.01 と dog.n.01 のように，2 つの語義が共通の下位語義（この場合は puppy.n.01）をもつ場合がある．この場合，pup.n.01 の箱と dog.n.01 の箱が部分的に重なり，その重なった領域が puppy.n.01 の箱に該当する．



(a) enclose (b) disjoint (c) intersect

図 3.3: アイテム間の 3 つの関係

アイテム x の箱埋め込みを $\mathbf{b}_x$，アイテム y の箱埋め込みを $\mathbf{b}_y$ とすると，アイテム x がアイテム y を包含する確率は式 (3.1) で計算される．

$$P(\mathbf{b}_y|\mathbf{b}_x) = \frac{\mathrm{Vol}(\mathbf{b}_x \cap \mathbf{b}_y)}{\mathrm{Vol}(\mathbf{b}_x)} \tag{3.1}$$

Vol は箱の体積を計算する関数であり，$\mathrm{Vol}(\mathbf{b}_x \cap \mathbf{b}_y)$ は箱 $\mathbf{b}_x$ と $\mathbf{b}_y$ が重なっている部分の体積を意味する．ただし，Vol のハードな定義では 2 つの箱が重なっていない場合に値が 0 となり，これを損失関数としてモデルを学習しようとすると，勾配がなくなるために学習できないという問題が生じる．そのため，Vol としてソフトな定義を用いる必要がある．本研究では，先行研究 [32, 20] と同じく Gumbel Box[12] によって関数 Vol を定義する．Gumbel Box では，箱の体積は式 (3.2) の Gumbel 分布に基づき計算される．

$$f(x;\mu,\beta) = \frac{1}{\beta}\exp\left(-\frac{x-\mu}{\beta} - e^{-\frac{x-\mu}{\beta}}\right) \tag{3.2}$$

μ と β は Gumbel 分布のパラメータである．2 つの箱の重なりを表すベクトル t は式 (3.3) で計算される．

$$t = -\beta\mathrm{LogSumExp}\left(-\frac{\mu^\wedge}{\beta}\right) - \beta\mathrm{LogSumExp}\left(\frac{\mu^\vee}{\beta}\right) \tag{3.3}$$

$\mu^\wedge$ と $\mu^\vee$ は，それぞれ 2 つの箱の辺の開始位置を連結したものと終了位置を連結したものを表す．つまり，箱の次元数を d とすると，$\mu^\wedge$ と $\mu^\vee$ の次元数は $2 \times d$，t の次元数は d となる．また，箱の体積は式 (3.4) で計算される．

$$m(t) = 2\beta K_0(2e^{-\frac{t}{2\beta}}) \tag{3.4}$$

K_0 はオーダが 0 の第 2 種ベッセル関数である．式 (3.4) は softplus 関数を用いて式 (3.5) のように近似できる．

$$m(t) \approx \beta\log(1+\exp(t/\beta - 2\gamma)) \tag{3.5}$$

γ はオイラーの定数（$= 0.5772...$）である．

3.3 語義の箱埋め込みの学習

本節では語義の箱埋め込みを学習する手法について述べる．提案手法は，2.2.5 項で述べた MetricWSD[10] を語義の箱埋め込みを学習するように拡張したものである．

本研究で学習するモデルは，入力文の各単語に対し，その単語が文中でどのように用いられているかを表す箱埋め込みを出力する．以下，これを「文脈箱埋め

込み」と呼ぶ．図 3.4 の例を用いて文脈箱埋め込みと語義の箱埋め込みの関係を説明する．語義 plant.n.01 を含む 1 から 3 の用例と語義 plant.n.02 を含む 4 から 6 の用例があり，それらの用例中の対象単語 plant（下線部）の文脈箱埋め込みは右の 1 から 6 の箱埋め込みとなる．一方，語義の箱埋め込みは，各語義が付与された単語の文脈箱埋め込みの平均により得られる．箱埋め込みの平均とは，$\mathbf{c}$ と $\mathbf{o}$ のそれぞれについて平均ベクトルを計算することで求められる箱埋め込みである．図 3.4 において，plant.n.01 と表示された赤い箱と plant.n.02 と表示された青い箱は，それぞれ語義 plant.n.01 と plant.n.02 の箱埋め込みを表す．MetricWSD の基盤となるメタ学習手法 Prototypical Networks[38] においては，1 から 6 がサポート表現，plant.n.01 と plant.n.02 がプロトタイプ表現に該当する．

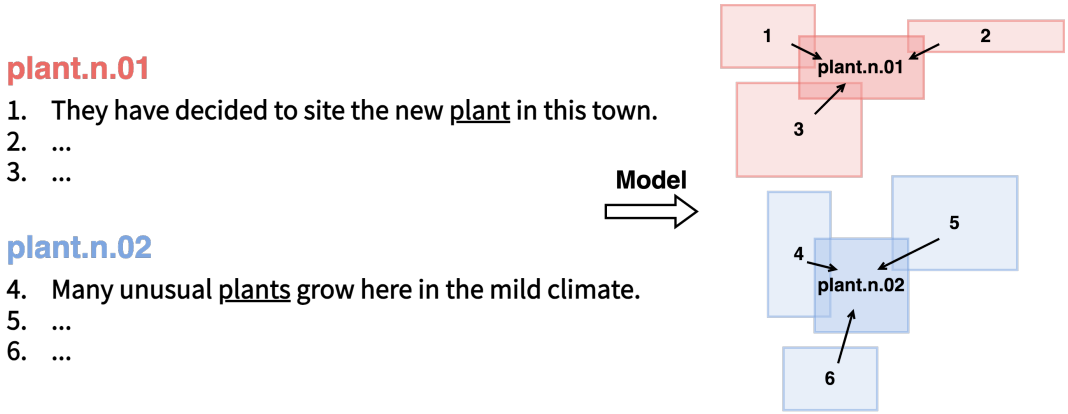


図 3.4: 文脈箱埋め込みと語義の箱埋め込み

MetricWSD[10] では，対象単語の文脈埋め込みを得るためのモデル f_θ として BERT を用いていた．本研究では，対象単語の文脈箱埋め込みを得るために，図 3.5 の点線の枠内に示すように，BERT の最終層の後に全結合層（Fully Connected Layer; FCL）をつなげたモデルを用いる．この層の入力は BERT の対象単語の最終層のベクトル表現であり，出力はベクトル $\mathbf{b}$ である．先行研究 [32] と同様に，$\mathbf{b}$ は半分に分割され，それぞれ $\mathbf{c}$ と $\mathbf{o}$ となる．すなわち，この全結合層により，単一のベクトルを箱埋め込みに変換する．

図 3.5 の ϕ はクエリ表現とプロトタイプ表現との類似度を計算する関数である．MetricWSD では，対象単語の語義のプロトタイプ表現とクエリ表現との類似度を式 (2.11) で計算していた．本研究では，モデルを学習する際には，クエリ表現と各プロトタイプ表現のペア $(\mathbf{b}_x, \mathbf{b}_y)$ ごとに，類似度を式 (3.1) で計算し，損失を式 (3.6) の binary cross-entropy で計算する．

$$L = -\delta \cdot \log P(\mathbf{b}_y|\mathbf{b}_x) - (1-\delta) \cdot \log(1 - P(\mathbf{b}_y|\mathbf{b}_x)) \tag{3.6}$$

δ は正例の場合に 1，負例の場合に 0 の値をとる変数である．ここでの正例とは，図 3.6 に示すように，語義 y が語義 x と同じもしくは語義 x の上位概念（祖先）で

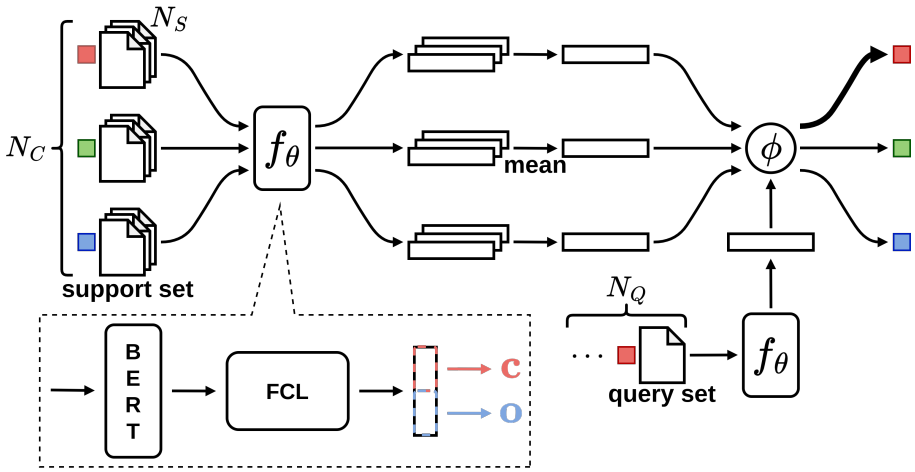


図 3.5: 文脈箱埋め込みを得るためのモデル f_θ

ある場合，負例とはそれ以外の場合を指す．直観的には，ある語義の箱埋め込みは，自身もしくは上位概念の語義との箱埋め込みと重なるように，それ以外とは重ならないように学習される．

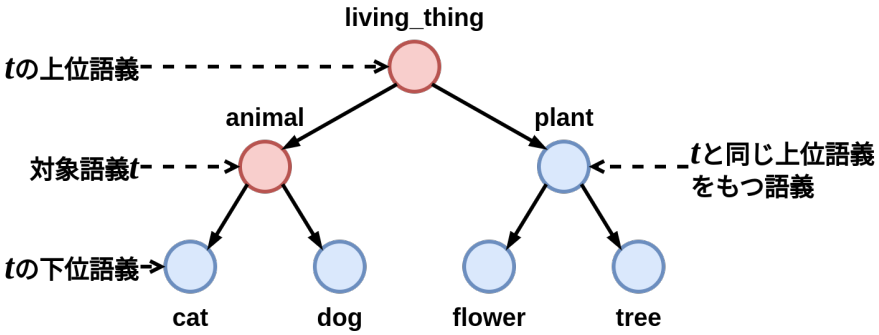


図 3.6: 損失計算における正例と負例（赤が正例，青が負例）

3.4 エピソードの作成

一般に損失を計算するために分割された少数のサンプルからなるデータセットはバッチとよばれるが，Prototypical Networks のようなメタ学習ではエピソードとよばれる．2.2.5 項で述べたように，MetircWSD では，1 つの単語に対して 1 つのエピソードを用意し，対象単語を含む用例によってエピソードを作成する．本研究では，語義間の関係を学習するため，1 つの語義に対して 1 つのエピソードを用意する．語義 t に対するエピソードの作成手順を以下に示す．

1. 訓練データに出現する語義の集合の中から N_C 個の語義を選択する．

2. それぞれの語義について，その語義の用例をランダムに N_S 個選択し，サポートセットとする．

3. 対象語義 t について，サポートセットに含まれていない用例の中からランダムに N_Q 個の用例を選択し，クエリセットとする．

上記の手続きのステップ 1 において，N_C 個の語義を選択する 2 つの戦略を提案する．

戦略 S_r t と t の直近の上位概念（上位語義）を正例として選び，残りの語義をランダムにサンプリングする．3.3 節で述べたように，t の祖先に該当する語義が選ばれたときは正例，それ以外は負例となる．

戦略 S_n t と t の上位語義を正例，t の直近の下位概念（下位語義）ならびに t と同じ上位語義をもつ語義（兄弟関係にある語義）を負例として選ぶ．S_r と同様に残りの語義はランダムにサンプリングする．

戦略 S_n では，ある語義の箱埋め込みが，特に下位語義や兄弟関係にある語義の箱埋め込みと重ならないことを重視している．対象語義とその下位語義や兄弟関係にある語義は概念的に近いため，それらの箱埋め込みは対象語義の箱埋め込みと似やすい．語義の数が多い場合，戦略 S_r では，ランダムに選ばれた語義の中に対象語義の下位や兄弟の概念が含まれないことが起こりうる．その場合，対象語義とそれらの語義との関係が学習されない（これらの語義の箱埋め込みが重ならないように学習されない）といった問題が生じることが予想される．

ステップ 2 とステップ 3 において，候補となる用例の数がそれぞれ N_S と N_Q より少ない場合，候補となる用例をすべてサポートセットもしくはクエリセットとする．クエリセットのサイズが 0 である場合，対象語義 t のサポートセットから 1 つをランダムに抜き出しクエリセットとする．これにより，対象語義 t のサポートセットのサイズが 0 になった場合，t のプロトタイプ表現は作成されない．

3.5 語義の箱埋め込みの応用

前節までは，語義の箱埋め込みを学習する手法，正確には語義の箱埋め込みや文脈箱埋め込みを生成するモデルを学習する手法について述べた．本節では，学習されたモデルを適用できる 3 つの応用タスクと，そのタスクに対してモデルを適用する具体的な手法を説明する．

3.5.1 WSD

WSD は，ある用例に出現する対象単語 w の語義として適切なものを，あらかじめ定義された対象単語 w の語義の集合 $\mathcal{S}_w = \{s_1, s_2, ..., s_k\}$ の中から，1 つ選ぶタ

スクである．$\mathcal{S}_w$ の各語義に対し，訓練データからプロトタイプ表現を作成して得られる語義の箱埋め込みの集合を $\mathcal{B}_w = \{\mathbf{b}^p_{s_1}, \mathbf{b}^p_{s_2}, ..., \mathbf{b}^p_{s_k}\}$ とし，対象単語 w を含むテストデータの用例 x における w の文脈箱埋め込みを $\mathbf{b}^q$ とすると，x における w の語義は，式 (3.7) の類似度が最大となる s_i と予測される．

$$\mathrm{sim}(\mathbf{b}^p_{s_i}, \mathbf{b}^q) = 2 \times \frac{P(\mathbf{b}^p_{s_i}|\mathbf{b}^q)P(\mathbf{b}^q|\mathbf{b}^p_{s_i})}{P(\mathbf{b}^p_{s_i}|\mathbf{b}^q) + P(\mathbf{b}^q|\mathbf{b}^p_{s_i})} \tag{3.7}$$

式 (3.7) は $P(\mathbf{b}^p_{s_i}|\mathbf{b}^q)$ と $P(\mathbf{b}^q|\mathbf{b}^p_{s_i})$ の調和平均である．$P(\mathbf{b}^p_{s_i}|\mathbf{b}^q)$ のみを類似度とすると，$\mathbf{b}^p_{s_i}$ が $\mathbf{b}^q$ を包含している場合に，箱の大きさに関係なく類似度が 1 となるため，箱の大きさが大きく異なる場合でも s_i が w の語義として誤って選択される可能性がある．$P(\mathbf{b}^q|\mathbf{b}^p_{s_i})$ のみを類似度とする場合にも同様の問題が発生する．類似度を調和平均とすることで，2 つの箱埋め込みが包含関係にありかつ箱の大きさがほぼ同じときに類似度が高くなるようにすることができる．

3.5.2 新語義の判定

新語義の判定は，ある用例に出現する対象単語 w の語義が新語義かどうかを判定するタスクである．すべての語義について，$\mathbf{b}^p_{s_i}$ と $\mathbf{b}^q$ との類似度 $\mathrm{sim}(\mathbf{b}^p_i, \mathbf{b}^q)$ が閾値 α より小さい場合は新語義，それ以外は新語義ではないと判定する．

閾値 α は，開発データを用いて対象単語ごとに設定する．開発データの用例の語義を決定する際，式 (3.7) が最大となる語義を選ぶが，その類似度の平均値を α とし，最大類似度がこの平均値以下か以上かで新語義か否かを判定する．具体的な計算方法は以下の通りである．対象単語 w が含まれた開発データの用例の集合を $\mathcal{Q}_w = \{q_1, q_2, ..., q_m\}$，$q_j$ における w の文脈箱埋め込みをクエリ表現 $\mathbf{b}^q_j$，対象単語の正解の語義 s_i のプロトタイプ表現を $\mathbf{b}^p_{s_i}$ とする．$\mathbf{b}^p_{s_i}$ は訓練データから作成されたものである．α は式 (3.8) に示すように $\mathbf{b}^p_{s_i}$ と $\mathbf{b}^q_j$ の類似度 $\mathrm{sim}(\mathbf{b}^p_{s_i}, \mathbf{b}^q_i)$ の平均とする．

$$\alpha = \frac{1}{m}\sum_{j=1}^{m} \mathrm{sim}(\mathbf{b}^p_{s_i}, \mathbf{b}^q_j) \tag{3.8}$$

開発データに出現しない単語に対しては，すべての対象単語の類似度の平均を閾値とする．

3.5.3 新語義の上位語義の推定

新語義の上位語義の推定は，ある用例に出現する新語義に対し，その上位語義の候補をランキングするタスクである．訓練データに出現するすべての単語のすべての語義のプロトタイプ表現を $\mathcal{B} = \{\mathbf{b}_{s_1}, \mathbf{b}_{s_2}, ..., \mathbf{b}_{s_m}\}$ とする．まず，$\mathcal{B}$ の中から $P(\mathbf{b}^p_{s_i}|\mathbf{b}^q)$ が閾値 β より大きい語義を選択し，上位語義の候補の集合を得る．次

に，これらの候補を $\mathrm{Vol}(\mathbf{b}^p_{s_i})$ と $\mathrm{Vol}(\mathbf{b}^q)$ の差が小さい順にソートする．しきい値 β は開発データを用いて設定することが望ましいが，4 章で述べる評価実験では開発データの数が不足していることから，β はあらかじめ決めておくものとする．箱の体積 $\mathrm{Vol}(\mathbf{b})$ は，式 (3.9) に示すように，各次元の辺の長さの積で計算される．

$$\mathrm{Vol}(\mathbf{b}) = \prod_{i=1}^{d} 2o_i \tag{3.9}$$

第4章　評価

本章では，提案手法の評価実験について述べる．本実験では，語義の箱埋め込みを学習し，これを3.5節で述べた3つのタスク，WSD，新語義の判定，新語義の上位語義の推定，に適用する．これらのタスクについて，提案手法である語義の箱埋め込みを用いる手法とベースライン手法を比較する．4.1節では実験に用いるデータセットについて述べる．4.2節では提案手法と比較するためのベースラインについて述べる．4.3節では語義の箱埋め込みモデルの学習について述べる．4.4節では語義の箱埋め込みを応用タスクに適用した結果を報告し，その考察を述べる．

4.1　データセット

実験には既存の WSD のデータセットを用いる．Raganato らの提案した枠組み [34] に従い，SemCor 3.0[27, 2] を訓練データ，SemEval-2007[33] を開発データ，Senseval-2[15]，Senseval-3[39]，SemEval-2013[30]，SemEval-2015[28] をテストデータとする．それぞれのデータセットは正しい語義が付与された用例の集合である．また，いずれのデータセットでも，語義は WordNet によって定義され，WordNet における synset が語義として付与されている．

上記のデータセット群から，$\mathcal{D}_{\text{living_thing.n.01}}$，$\mathcal{D}_{\text{artifact.01}}$，$\mathcal{D}_{\text{entity.n.01}}$ という3つのデータセットを作成する．living_thing.n.01，artifact.n.01，entity.n.01 は WordNet における synset であり，各データセットはこれらの synset より下位の概念をもつ用例から構成される．$\mathcal{D}_{\text{entity.n.01}}$ は名詞全体であり，$\mathcal{D}_{\text{living_thing.n.01}}$ と $\mathcal{D}_{\text{artifact.01}}$ は名詞のサブセットである．

訓練データ，開発データ，テストデータの統計をそれぞれ表4.1，表4.2，表4.3に示す．表4.1より，$\mathcal{D}_{\text{living_thing.n.01}}$ と $\mathcal{D}_{\text{artifact.n.01}}$ の語義の数は同程度，$\mathcal{D}_{\text{entity.n.01}}$ の語義の数は他の2つの約7倍であることがわかる．また，$\mathcal{D}_{\text{living_thing.n.01}}$ と $\mathcal{D}_{\text{artifact.n.01}}$ では，単語の数が語義の数よりも多い．これは，3.1節で述べたように，語義の数は synset の数と等しく，1つの synset には複数の単語が属することがあるためである．例えば，car.n.01 は car, automobile, motorcar などの複数の単語から構成される synset である．表4.2に示すように，開発データの語義の数は訓練データの約100分の1であり，表4.3に示すように，テストデータの語義の数は訓練データの約10分の1である．

表 4.1: 訓練データの統計

	$\mathcal{D}_{\text{living_thing.n.01}}$	$\mathcal{D}_{\text{artifact.n.01}}$	$\mathcal{D}_{\text{entity.n.01}}$
語義数	1,713	1,938	12,757
単語数	1,809	1,993	11,026
用例数	15,835	8,704	84,944

表 4.2: 開発データの統計

	$\mathcal{D}_{\text{living_thing.n.01}}$	$\mathcal{D}_{\text{artifact.n.01}}$	$\mathcal{D}_{\text{entity.n.01}}$
語義数	17	12	125
単語数	17	12	127
用例数	25	20	159

表 4.3: テストデータの統計

	$\mathcal{D}_{\text{living_thing.n.01}}$	$\mathcal{D}_{\text{artifact.n.01}}$	$\mathcal{D}_{\text{entity.n.01}}$
語義数	194	132	1,704
単語数	202	131	1,515
用例数	628	273	4,186

次に，表 4.3 に示したテストデータから用例を抜粋することで，3.5 節で述べた 3 つのタスクのテストデータを作成する．その作成方法を以下に示す．

- **WSD タスクのテストデータ**
  表 4.3 のテストデータから，訓練データに出現する語義を 2 種類以上もつ単語 (多義語) で，かつその正解の語義が訓練データに出現する用例の集合を抜粋して作成する．その統計を表 4.4 に示す.「ALL」はテストデータすべて，「≤10」はテストデータのうち訓練データにおける出現頻度が 10 回以下の語義の用例である．後者は低頻度の語義に対する WSD の性能を測るために用いる．

表 4.4: WSD タスクのテストデータの用例数

	$\mathcal{D}_{\text{living_thing.n.01}}$		$\mathcal{D}_{\text{artifact.n.01}}$		$\mathcal{D}_{\text{entity.n.01}}$	
	ALL	≤ 10	ALL	≤ 10	ALL	≤ 10
用例数	190	66	78	40	2,514	992

- **新語義判定タスクのテストデータ**
  表 4.3 のテストデータから，対象単語が訓練データに出現する用例を抜粋して作成する．それらの用例に対し，正解の語義が訓練データに出現していな

ければ「新語義」，そうでない場合は「新語義ではない」としてラベル付けする．このテストデータの統計を表 4.5 に示す．

表 4.5: 新語義判定タスクのテストデータの用例数

	$\mathcal{D}_{\text{living_thing.n.01}}$	$\mathcal{D}_{\text{artifact.n.01}}$	$\mathcal{D}_{\text{entity.n.01}}$
新語義の用例数	20	12	295
新語義ではない用例数	500	221	3,379

- **新語義の上位語義推定タスクのテストデータ**
  表 4.3 のテストデータから，訓練データに出現しない語義を正解とする用例を抜粋して作成する．表 4.5 に示す新語義判定タスクにおける「新語義」の用例とは異なり，対象単語自体が訓練データに出現しない語義の用例も含まれる．このテストデータの統計を表 4.6 に示す．

表 4.6: 新語義の上位語義推定タスクのテストデータの用例数

	$\mathcal{D}_{\text{living_thing.n.01}}$	$\mathcal{D}_{\text{artifact.n.01}}$	$\mathcal{D}_{\text{entity.n.01}}$
用例数	107	32	658

4.2 ベースライン

単一のベクトルで語義を表現する手法をベースラインとし，箱埋め込みを学習する提案手法と比較する．具体的には，以下の 2 つの手法をベースラインとする．

BERT-NN
ファインチューニングを行わず，事前学習済みの BERT を用いて語義の文脈埋め込みを得る手法．訓練データにおけるすべての語義の用例について，その文脈埋め込みを BERT で生成し，その平均ベクトルを語義の埋め込み表現とする．

MetricWSD
MetricWSD[10] によって語義の埋め込み表現（プロトタイプ表現）を学習する手法．ベースモデルとして BERT が使われているため，MetricWSD は BERT-NN をファインチューニングした手法といえる．

各タスクについて，単一のベクトルで表された語義の埋め込み表現を用いたベースライン手法は以下のように実装する．

- **WSD タスクのベースライン手法**
  テストデータの用例における対象単語の語義は，対象単語の語義の候補のうち，その埋め込み表現と対象単語の文脈埋め込みとの距離が最も近い語義と予測する．

- **新語義判定タスクのベースライン手法**
  対象単語のすべての語義の候補について，語義の埋め込み表現と対象単語の文脈埋め込み表現との内積が閾値 α を下回っている場合は新語義，それ以外は新語義ではないと判定する．閾値 α の設定方法は，3.5.2 項で述べた方法と同じである．ただし，語義と用例の類似度の計算を 2 つの埋め込み表現の類似度から 2 つのベクトルの内積に変更する．

- **新語義の上位語義推定タスクのベースライン手法**
  訓練データに出現するすべての単語のすべての語義の埋め込み表現を，対象単語の文脈埋め込みとの距離が近い順にソートし，上位 10 個の語義を対象単語の上位語義の候補とする．

4.3 モデルの学習

BERT-NN，MetricWSD，提案手法のいずれも，事前学習済み BERT モデルとして bert-base-uncased[1] を使用した．提案手法の設定として，箱埋め込みを出力するための全結合層の次元数は 256（$\mathbf{c}$ と $\mathbf{o}$ の次元数は 128），サポートサイズ N_S は 5，クエリサイズ N_Q は 20，各エピソードの語義の数 N_C は 128 とした．サポートサイズ N_S とクエリサイズ N_Q は MetricWSD の文献 [10] と同一である．また，MetricWSD の各エピソードの語義の数 N_C は対象単語の語義の数である．

語義の箱埋め込みを生成するモデルのエポック数を開発データを用いて最適化する．5 エポックから 100 エポックまで 5 エポックの間隔で学習したモデルを保存し，その中から最良のモデルを開発データを基に決定する．開発データに出現する語義の集合を $\mathcal{S} = \{s_1, s_2, ..., s_m\}$，$\mathcal{S}$ から 2 つの語義を選んで並べる順列の集合を $\mathcal{P}$ とする (式 (4.1))．

$$\begin{aligned}\mathcal{P} = \{&(s_1, s_2), (s_1, s_3), ..., (s_1, s_m), (s_2, s_1), (s_2, s_3), ..., (s_2, s_m),\\ &..., (s_m, s_1), (s_m, s_2), ..., (s_m, s_{m-1})\}\end{aligned} \tag{4.1}$$

さらに，この中から s_j が s_i の上位概念となっているペア (s_i, s_j) を抽出し，$\mathcal{P}_{\text{hyper}}$ とする．s_j が s_i の直接の上位概念である場合だけでなく，語義グラフにおける階層構造を上に辿って s_i から s_j に到達する場合も含む．$\mathcal{P}_{\text{hyper}}$ 内の各ペア (s_i, s_j) について，開発データからそれらのプロトタイプ表現 $\mathbf{b}_{s_i}$ と $\mathbf{b}_{s_j}$ を作成し，$P(\mathbf{b}_{s_j}|\mathbf{b}_{s_i})$ を計算する．そして，それらの平均を，モデルに対する上位下位関係にある語義間の学習度スコアとする．この学習度スコアは，モデルによって学習された語義

の箱埋め込みが語義の上位下位関係をどれだけ適切に表現できているかを評価している．この処理を保存されたすべてのモデルについて行い，スコアが最大のモデルを選択し，これをテストに用いる．

$\mathcal{D}_{\text{living_thing.n.01}}$，$\mathcal{D}_{\text{artifact.n.01}}$，$\mathcal{D}_{\text{entity.n.01}}$ のそれぞれについて，エポックの増加に伴う学習度スコアの変動を図 4.1a，図 4.1b，図 4.1c に示す．各データセットの $\mathcal{P}_{\text{hyper}}$ のサイズは 15，2，52 となった．$\mathcal{D}_{\text{living_thing.n.01}}$ では学習が進むに従ってスコアが下がる傾向にあること，$\mathcal{D}_{\text{artifact.n.01}}$ では学習が進むに従ってスコアが上がる傾向にあること，$\mathcal{D}_{\text{entity.n.01}}$ では学習が進んでもスコアに大きな変化が生じないことがわかる．$\mathcal{D}_{\text{entity.n.01}}$ について，学習度スコアが向上しないのは，このデータセットにおける語義の数は他のデータセットと比べて多く，1 エポックが長いことから，エポックの最初に学習した情報がエポックの終わりごろまで保持できていないためと考えられる．$\mathcal{D}_{\text{living_thing.n.01}}$ と $\mathcal{D}_{\text{artifact.n.01}}$ の違いについては，語義の数が同程度であることから，エポックの長さが原因ではなく，語義グラフの構造や開発データの質の違いにより生じると考えられる．また，$\mathcal{D}_{\text{artifact.n.01}}$ の $\mathcal{P}_{\text{hyper}}$ のサイズは 2 であり，かなり小さいことから，適切な学習度スコアが算出できていない可能性がある．また，戦略 S_r と S_n を比較すると，$\mathcal{D}_{\text{living_thing.n.01}}$ と $\mathcal{D}_{\text{entity.n.01}}$ では S_n より S_r の方が全体的にスコアが高い．一方，$\mathcal{D}_{\text{artifact.n.01}}$ では 2 つの戦略でスコアは同程度である．最適化の結果，テストに用いるモデルは，$\mathcal{D}_{\text{living_thing.n.01}}$ では S_r がエポック 15，S_n が 5，$\mathcal{D}_{\text{artifact.n.01}}$ では S_r が 85，S_n が 60，$\mathcal{D}_{\text{entity.n.01}}$ では S_r が 5，S_n が 5 となった．

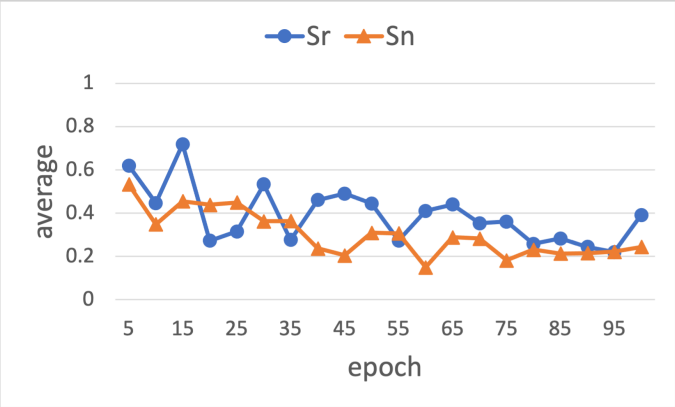
4.4 結果と考察

4.4.1 WSD の実験結果

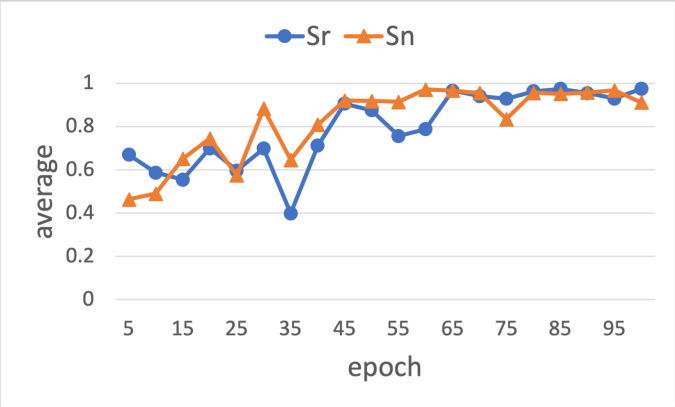
各手法の WSD の正解率を表 4.7 に示す．ProtoBox S_r と ProtoBox S_n は，それぞれ S_r, S_n をサンプリング戦略としたときの提案手法を表す.「ALL」はテストデータすべて,「≤10」はテストデータのうち訓練データにおける出現頻度が 10 回以下の語義のみを対象としたときの正解率である．

「ALL」の結果を見ると，提案手法は小規模なデータセットである $\mathcal{D}_{\text{living_thing.n.01}}$ と $\mathcal{D}_{\text{artifact.n.01}}$ でおおむねベースラインを上回ったが，名詞全体のデータセットである $\mathcal{D}_{\text{entity.n.01}}$ ではベースラインを下回った．小規模なデータセットでは，語義間の上位下位関係の情報が WSD の性能の向上に寄与することがわかる．一方，名詞全体のデータセットでそのような効果が見られない原因としては，4.3 節で述べたように，$\mathcal{D}_{\text{entity.n.01}}$ は語義の数が他の 2 つのデータセットと比べて非常に多いため，語義の箱埋め込みを適切に学習できていないことが考えられる．

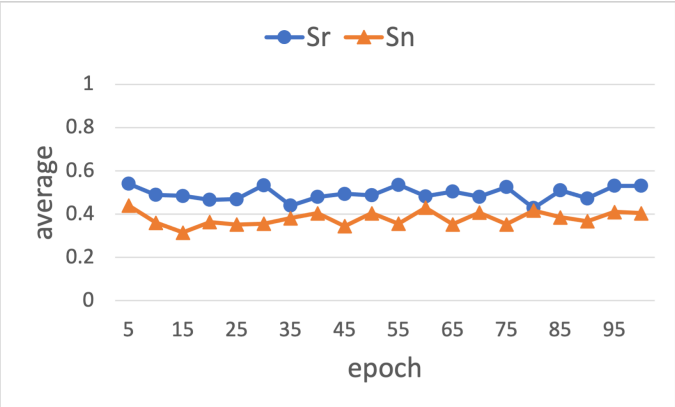
「≤10」の結果を見ると，$\mathcal{D}_{\text{living_thing.n.01}}$ と $\mathcal{D}_{\text{entity.n.01}}$ で提案手法が MetricWSD を上回った．このことから，低頻度語義についても，語義間の上位下位関係の情報が



(a) $\mathcal{D}_{\text{living_thing.n.01}}$



(b) $\mathcal{D}_{\text{artifact.n.01}}$



(c) $\mathcal{D}_{\text{entity.n.01}}$

図 4.1: 上位下位関係にある語義間の学習度スコア

WSD の性能の向上に寄与することがわかる．一方，$\mathcal{D}_{\text{entity.n.01}}$ ではファインチューニングを行わない BERT モデルである BERT-NN が最も高い正解率となった．

サンプリング戦略 S_r と S_n を比較すると，データセットや語義の出現頻度によっ

て，2つの戦略の優劣は変わることがわかる．しかし，$\mathcal{D}_{\text{entity.n.01}}$ に限っては,「ALL」と「≤10」の両方で戦略 S_n が S_r を上回っている．3.4 節で述べたように，S_n では下位の語義や兄弟関係にある語義など，語義グラフ上で近い位置にあり意味も似ている語義の箱同士が重ならないように学習しているが，その戦略が語義の数が多い場合に有効であるといえる．

表 4.7: WSD の結果 (正解率)

Model	$\mathcal{D}_{\text{living_thing.n.01}}$		$\mathcal{D}_{\text{artifact.n.01}}$		$\mathcal{D}_{\text{entity.n.01}}$	
	ALL	≤ 10	ALL	≤ 10	ALL	≤ 10
BERT-NN	81.6	72.7	74.4	77.5	57.9	**60.2**
MetricWSD	82.1	77.3	85.9	**92.5**	**73.3**	57.9
ProtoBox S_r	**83.2**	80.3	**87.2**	87.5	62.8	57.2
ProtoBox S_n	78.4	**81.8**	84.6	87.5	63.2	58.3

4.4.2 新語義判定の実験結果

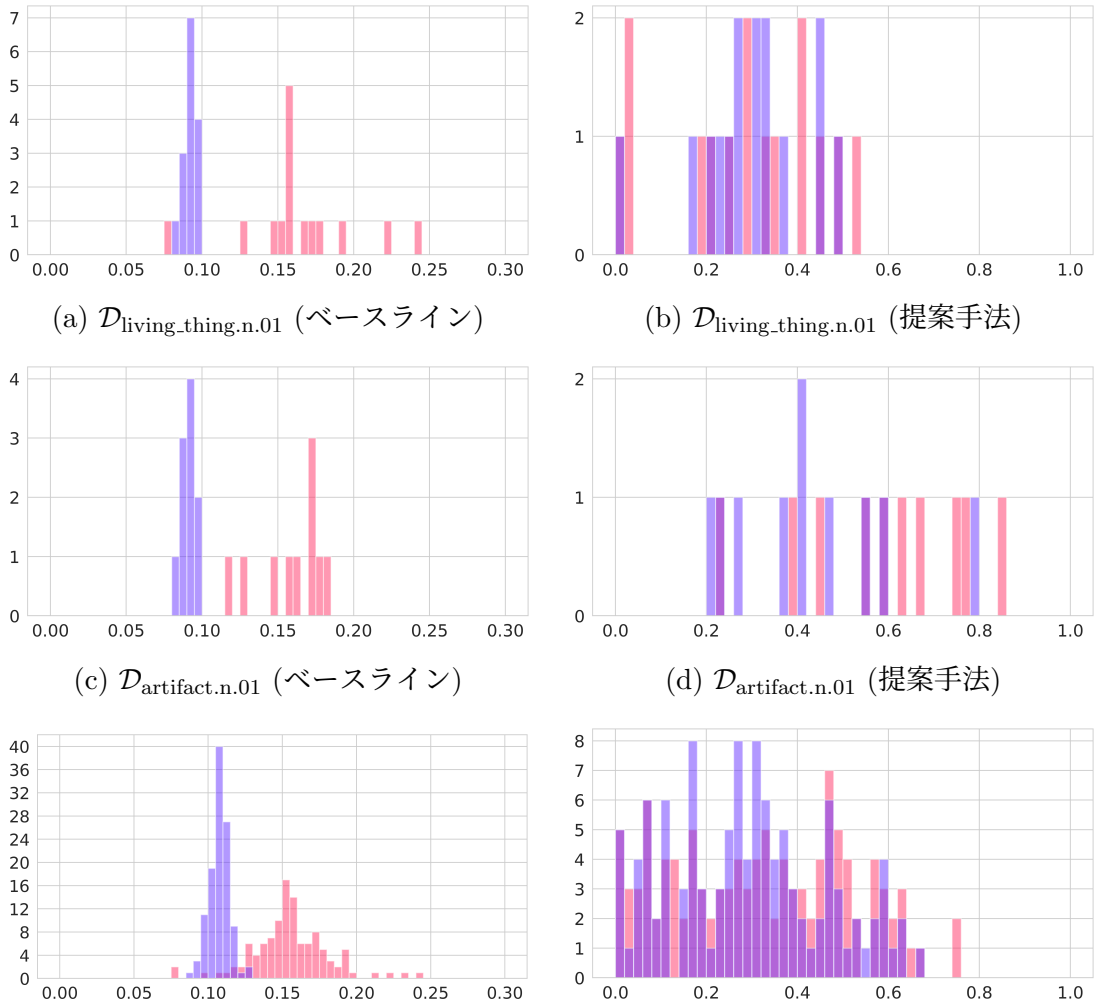
各手法の新語義の判定の F1 スコアを表 4.8 に示す．表 4.7 に示した WSD の結果とは異なり，$\mathcal{D}_{\text{entity.n.01}}$ では提案手法がベースラインを上回り，残りの 2 つのデータセットでは下回った．2 つのサンプリング戦略を比較すると，全体的には S_r の方が S_n よりも高い F1 スコアが得られた．

表 4.8: 新語義の判定の結果 (F1 スコア)

Model	$\mathcal{D}_{\text{living_thing.n.01}}$	$\mathcal{D}_{\text{artifact.n.01}}$	$\mathcal{D}_{\text{entity.n.01}}$
BERT-NN	**16.4**	10.2	19.5
MetricWSD	15.2	**17.7**	21.2
ProtoBox S_r	14.5	14.1	**25.1**
ProtoBox S_n	10.9	15.5	22.5

新語義の判定はすべての語義の埋め込み表現とテストデータの用例の文脈埋め込みの類似度が閾値 α より大きいか（新語彙ではないと判定）小さいか（新語義と判定）で判定し，その閾値は開発データを用いて 3.5.2 項と 4.2 節で述べた方法で決める．ここでは実際に設定された閾値 α の結果について述べ，それについて考察する．

閾値 α は対象単語ごとに個別に設定するが，開発データに対象単語の用例がないときはすべての対象単語についての平均値とする．開発データに用例があり，個別に α を設定した単語の数は，$\mathcal{D}_{\text{living_thing.n.01}}$ では 15 個，$\mathcal{D}_{\text{artifact.n.01}}$ では 10 個，$\mathcal{D}_{\text{entity.n.01}}$ では 113 個であった．対象単語ごとに設定した閾値の分布を図 4.2 に示



(a) $\mathcal{D}_{\text{living_thing.n.01}}$ (ベースライン)　(b) $\mathcal{D}_{\text{living_thing.n.01}}$ (提案手法)

(c) $\mathcal{D}_{\text{artifact.n.01}}$ (ベースライン)　(d) $\mathcal{D}_{\text{artifact.n.01}}$ (提案手法)

(e) $\mathcal{D}_{\text{entity.n.01}}$ (ベースライン)　(f) $\mathcal{D}_{\text{entity.n.01}}$ (提案手法)

(a),(c),(e): 赤が BERT-NN，青が MetricWSD　(b),(d),(f): 赤が戦略 S_r，青が S_n

図 4.2: 閾値 α の分布

す．同図は 6 つのグラフに分かれている．上段は $\mathcal{D}_{\text{living_thing.n.01}}$，中段は $\mathcal{D}_{\text{artifact.n.01}}$，下段は $\mathcal{D}_{\text{entity.n.01}}$ の結果である．また，左はベースライン（BERT-NN）と MetircWSD，右は提案手法（戦略 S_r と S_n）の結果である．図 4.2a，4.2c，4.2e より，すべてのデータセットで，MetricWSD の α は BERT-NN の α より小さいことがわかる．WSD の結果では MetricWSD の方が BERT-NN より良いことから，MetricWSD がクエリ表現（テストデータの用例の埋め込み表現）と正解語義以外の語義のプロトタイプ表現の類似度が低くなるように語義のプロトタイプ表現を学習し，これにより閾値 α も低く設定されたからだと考えられる．

図 4.2b，4.2d，4.2f に示した提案手法の α の分布を見ると，すべてのデータセットで，提案手法の α は 2 つの戦略ともベースラインよりも広く分布していること

がわかる．箱埋め込みでは，単一のベクトルと異なり，箱の大きさの違いやずれによって類似度が大きく変化するためと考えられる．

先に述べたように，開発データに出現しない単語については，すべての対象単語についての平均を α と設定している．その値を表 4.9 に示す．この表からも，提案手法 ProtoBox S_r と ProtoBox S_n の閾値は高く設定される傾向が確認できる．

表 4.9: 開発データに出現しない単語に対する α

Model	$\mathcal{D}_{\text{living_thing.n.01}}$	$\mathcal{D}_{\text{artifact.n.01}}$	$\mathcal{D}_{\text{entity.n.01}}$
BERT-NN	0.159	0.167	0.156
MetricWSD	0.091	0.092	0.107
ProtoBox S_r	0.254	0.580	0.322
ProtoBox S_n	0.256	0.423	0.289

4.4.3 新語義の上位語義推定の実験結果

新語義の上位語義の推定の結果を表 4.10 に示す．3.5.3 項で述べた閾値 β は 0.5, 0.7，0.9 のいずれかと設定した．ACC はランクが最上位の上位語義の正解率（Accuracy），MRR は正解の上位語義のランクの逆数の平均（Mean Reciprocal Rank），WP はランクが最上位の上位語義と正解の語義との Wu-Palmer 類似度 [41] である．

Wu-Palmer 類似度は，語義グラフ上での深さから計算される 2 つの語義の類似度である．語義 s_1 と s_2 の Wu-Palmer 類似度は式 (4.2) で計算される．

$$\mathrm{WP} = 2 \times \frac{d(\mathrm{lcs}(s_1, s_2))}{d(s_1) + d(s_2)} \tag{4.2}$$

ここで，$d(s)$ は根ノードから語義 s までの最短距離，$\mathrm{lcs}(s_1, s_2)$ は語義 s_1 と s_2 の最も近い共通の上位概念（Least Common Subsumer）である．

実験の結果，ACC と WP はすべてのデータセットで提案手法がベースラインを上回った．このことは，提案手法による語義の箱埋め込み学習が上位語義の推定（タクソノミ拡張）に有効的であることを示唆する．$\mathcal{D}_{\text{entity.n.01}}$ においては，提案手法の MRR がベースラインを下回った．この原因は，ランキングの対象とする上位語義の候補の選定方法の違いによるものと考えられる．3.5.3 項で述べたように，提案手法では $P(\mathbf{b}^p_{s_i}|\mathbf{b}^q) > \beta$ という条件を満たす語義を上位語義の候補とするが，ベースラインではすべての語義を上位語義の候補としている．より多くの語義を評価対象とする $\mathcal{D}_{\text{entity.n.01}}$ では，真の上位語義が $P(\mathbf{b}^p_i|\mathbf{b}^q) > \beta$ の条件を満たさずに候補から除外され，その結果 MRR の値が低下したと推測される．したがって，上位語義の候補を絞り込む条件を洗練する必要がある．また，$\mathcal{D}_{\text{entity.n.01}}$ におけるベースラインと提案手法の ACC と WP の差は，他の 2 つのデータセットよりも小さくなっている．これは，WSD の場合と同じく，$\mathcal{D}_{\text{entity.n.01}}$ の語義の数が他の 2 つ

のデータセットと比べて非常に多いことから，適切な語義の箱埋め込みが必ずしも学習できていないためと考えられる．

表 4.10: 新語義の上位語義推定の結果

Model	β	$\mathcal{D}_{\text{living_thing.n.01}}$			$\mathcal{D}_{\text{artifact.n.01}}$			$\mathcal{D}_{\text{entity.n.01}}$		
		ACC	MRR	WP	ACC	MRR	WP	ACC	MRR	WP
BERT-NN	–	15.0	25.9	75.4	9.4	15.0	56.7	6.8	11.3	46.0
MetricWSD		10.3	22.3	76.5	3.1	15.1	47.5	7.9	**13.2**	50.1
ProtoBox S_r	0.5	37.4	51.5	86.3	12.5	15.6	62.4	5.9	12.5	52.6
ProtoBox S_n		**53.3**	**59.3**	86.9	9.4	10.9	62.6	6.8	12.2	50.8
ProtoBox S_r	0.7	51.4	58.9	**88.7**	**18.8**	**18.8**	64.9	6.1	11.9	**54.2**
ProtoBox S_n		**53.3**	55.0	87.1	3.1	4.7	59.1	6.7	11.4	50.8
ProtoBox S_r	0.9	**53.3**	54.7	86.6	9.4	9.4	**65.4**	**10.0**	12.7	52.5
ProtoBox S_n		34.6	34.6	76.2	3.1	3.1	60.3	4.4	6.0	47.8

4.5 語義の箱埋め込みの可視化

いくつかの語義について，学習された語義の箱埋め込み（訓練データから作成されたプロトタイプ表現）を可視化することで，語義間の関係を正しく学習できているかを調査する．可視化に用いるモデルの学習時のエポック数は，$\mathcal{D}_{\text{living_thing.n.01}}$，$\mathcal{D}_{\text{artifact.n.01}}$，$\mathcal{D}_{\text{entity.n.01}}$ のすべてで 15 とした．

まず，語義の上位下位関係を正しく学習できているかを調査する．$\mathcal{D}_{\text{living_thing.n.01}}$ のデータセットから戦略 S_r の提案手法によって得られた語義 animal.n.01 と dog.n.01 の箱埋め込みを図 4.3 に，$\mathcal{D}_{\text{artifact.n.01}}$ のデータセットから戦略 S_r の提案手法によって得られた語義 structure.n.01 と house.n.01 の箱埋め込みを図 4.4 に示す．図の横軸は箱埋め込みの次元，縦軸は各次元の辺の区間を表す．各次元の辺の区間はシグモイド関数により 0 から 1 の範囲の値に変換されている．紫と緑の部分は，赤と青，黄と青それぞれの 2 つの箱埋め込みが重なっている領域を表す．これらの図から，animal.n.01 の箱が dog.n.01 の箱を包含し，structure.n.01 の箱が house.n.01 の箱を包含していることが読み取れる．WordNet では，animal.n.01 は dog.n.01 の上位概念，structure.n.01 は house.n.01 の上位概念と定義されているため，モデルはこれらの語義間の上位下位関係を正しく学習できていることがわかる．

$\mathcal{D}_{\text{entity.n.01}}$ のデータセットから戦略 S_r の提案手法によって得られた語義 animal.n.01 と dog.n.01 の箱埋め込みを図 4.5 に，$\mathcal{D}_{\text{entity.n.01}}$ のデータセットから戦略 S_r の提案手法によって得られた語義 structure.n.01 と house.n.01 の箱埋め込みを図 4.6 に示す．先ほどと同様に，animal.n.01 の箱は dog.n.01 の箱を包含し，structure.n.01 の箱は house.n.01 の箱を包含している．しかし，図 4.6 は図 4.4 に比べて下位の語

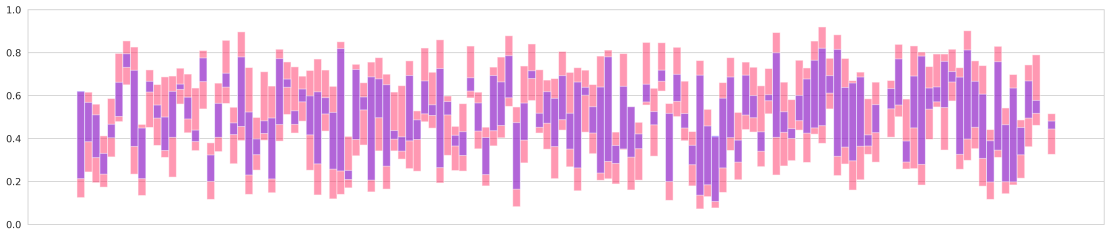


図 4.3: 語義 animal.n.01（赤）と dog.n.01（青）の箱埋め込み（$\mathcal{D}_{\text{living_thing.n.01}}$，ProtoBox S_r）

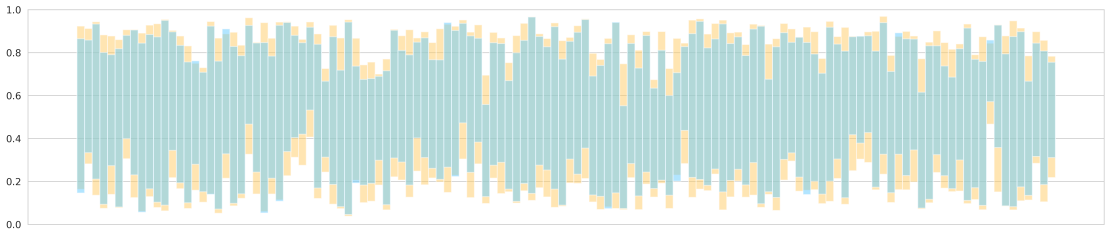


図 4.4: 語義 structure.n.01（黄）と house.n.01（青）の箱埋め込み（$\mathcal{D}_{\text{artifact.n.01}}$，ProtoBox S_r）

義 house.n.01 の青い領域も目立っていることから，$\mathcal{D}_{\text{artifact.n.01}}$ のときよりは正確に上位下位関係を学習できていないといえる．4.3 節において，$\mathcal{D}_{\text{entity.n.01}}$ では語義の数が多いために語義の箱埋め込みを適切に学習できていないと述べたが，ここでの可視化でも同様のことがいえる．また，図 4.3，図 4.4 の箱埋め込みと比較すると，同じエポック数で学習しても，$\mathcal{D}_{\text{entity.n.01}}$ の方が他の 2 つのデータセットより箱埋め込みのサイズが大きくなっている．提案手法は式 (3.6) の損失を最小化するために箱埋め込みのサイズを大きくする傾向があり，また語義の数が多い大規模なデータセットではエピソードの数が増え，1 つの語義がサンプリングされる回数が増えるためと考えられる．さらに，4 つの語義の箱埋め込みに共通する特徴として，0 から 1 を辺の区間とする次元が多い．こういった次元は語義間の関係を表現することには貢献しない．また，次元数が必要以上に多く設定され，このことが適切な語義の箱埋め込みの学習を難しくしている要因となっている可能性もある．Jiang らの研究 [20] でも，箱埋め込みの次元数は 4 次元や 6 次元のような小さい値に設定した方が性能が良いことが報告されている．

次に，上位下位ではない関係を学習できているかを調査する．$\mathcal{D}_{\text{living_thing.n.01}}$ のデータセットから戦略 S_r の提案手法によって得られた語義 tree.n.01 と dog.n.01 の箱埋め込みを図 4.7 に，$\mathcal{D}_{\text{artifact.n.01}}$ のデータセットから戦略 S_r の提案手法によって得られた語義 art.n.01 と house.n.01 の箱埋め込みを図 4.8 に示す．これらの図から，tree.n.01 と dog.n.01，art.n.01 と house.n.01 の両方の組で，一方の箱がもう一方の箱を包含していないことがわかる．WordNet では，tree.n.01 と dog.n.01，art.n.01 と house.n.01 の間には上位下位関係があるとは定義されていないため，モデルは

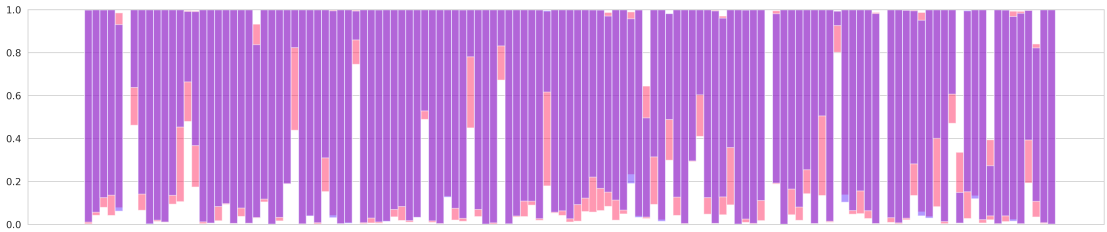


図 4.5: 語義 animal.n.01（赤）と dog.n.01（青）の箱埋め込み（$\mathcal{D}_{\text{entity.n.01}}$，ProtoBox S_r）

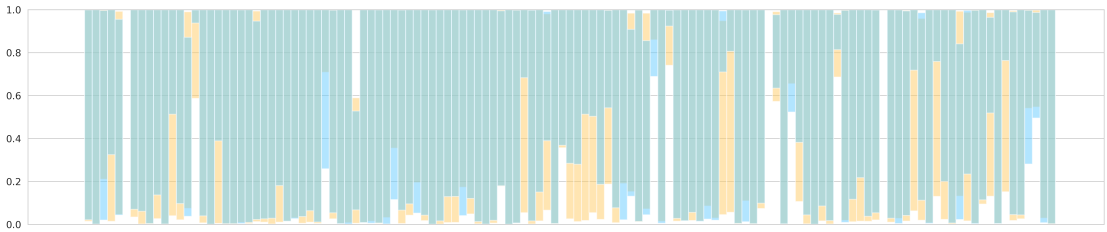


図 4.6: 語義 structure.n.01（黄）と house.n.01（青）の箱埋め込み（$\mathcal{D}_{\text{entity.n.01}}$，ProtoBox S_r）

これらの語義が上位下位関係にないことを正しく学習できていることがわかる．

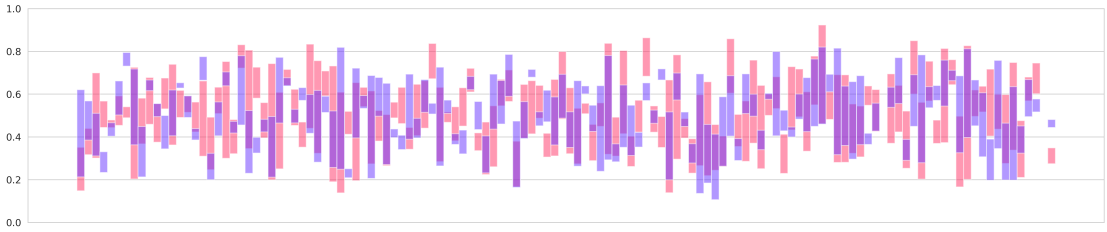


図 4.7: 語義 tree.n.01（赤）と dog.n.01（青）の箱埋め込み（$\mathcal{D}_{\text{living_thing.n.01}}$，ProtoBox S_r）

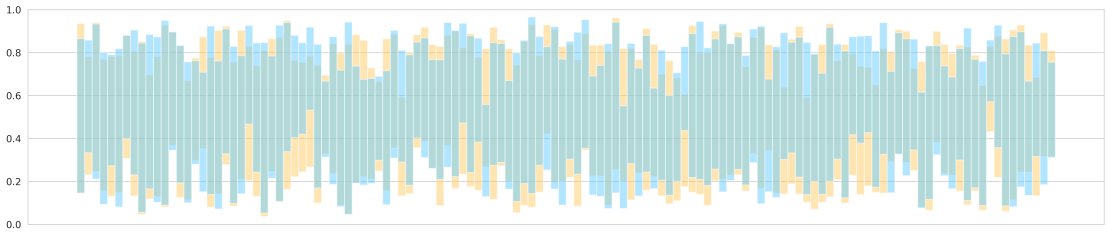


図 4.8: 語義 art.n.01（黄）と house.n.01（青）の箱埋め込み（$\mathcal{D}_{\text{artifact.n.01}}$，ProtoBox S_r）

次に，意味的に近いが上位下位関係にない語義の組について，学習された語義の箱

埋め込みの例を示す．また，戦略 S_r と S_n の違いについても考察する．$\mathcal{D}_{\text{living_thing.n.01}}$ のデータセットから戦略 S_r の提案手法によって得られた語義 cat.n.01 と dog.n.01 の箱埋め込みを図 4.9 に，$\mathcal{D}_{\text{artifact.n.01}}$ のデータセットから戦略 S_r の提案手法によって得られた語義 hotel.n.01 と house.n.01 の箱埋め込みを図 4.10 に示す．まず，cat.n.01 と dog.n.01 の箱は，図 4.7 に示した tree.n.01 と dog.n.01 の箱よりも重なりが大きいことがわかる．両者の重なりを定量的に評価すると，$P(\text{cat.n.01}|\text{dog.n.01}) = 0.17$，$P(\text{dog.n.01}|\text{cat.n.01}) = 0.29$ であり，部分的に重なっているといえる．この理由として，cat.n.01 と dog.n.01 の概念的な距離（語義グラフ上での距離）は tree.n.01 と dog.n.01 よりも近いため，3.4 節で述べたように，戦略 S_r ではこれらの関係を適切に学習できなかったことが考えられる．hotel.n.01 と house.n.01 の箱についても，$P(\text{hotel.n.01}|\text{house.n.01}) = 0.01$，$P(\text{house.n.01}|\text{hotel.n.01}) = 0.19$ であり，両者は部分的に重なっている．

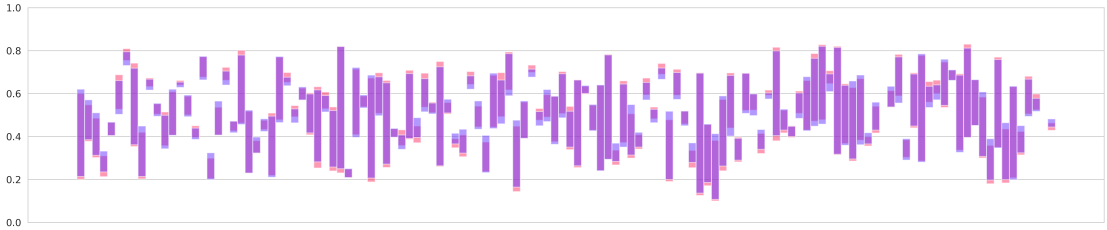


図 4.9: 語義 cat.n.01（赤）と dog.n.01（青）の箱埋め込み（$\mathcal{D}_{\text{living_thing.n.01}}$，ProtoBox S_r）

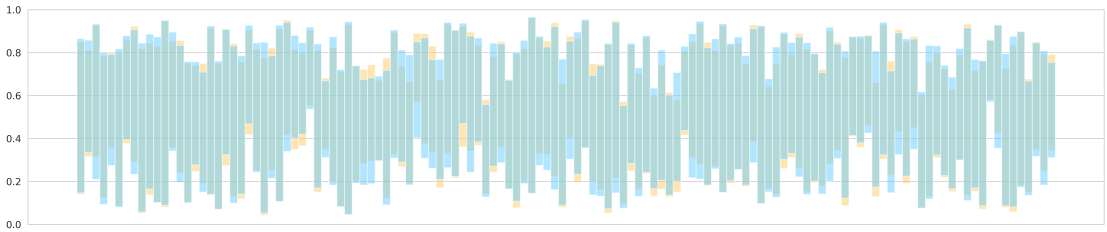


図 4.10: 語義 hotel.n.01（黄）と house.n.01（青）の箱埋め込み（$\mathcal{D}_{\text{artifact.n.01}}$，ProtoBox S_r）

$\mathcal{D}_{\text{living_thing.n.01}}$ のデータセットから戦略 S_n の提案手法によって学習された語義 cat.n.01 と dog.n.01 の箱埋め込みを図 4.11 に，$\mathcal{D}_{\text{artifact.n.01}}$ のデータセットから戦略 S_n の提案手法によって学習された語義 hotel.n.01 と house.n.01 の箱埋め込みを図 4.12 に示す．これらの図と図 4.9，図 4.10 を比較すると，すなわち戦略 S_r と S_n を比較すると，前者の方が箱の重なりが小さいことがわかる．両者の関係を定量的に評価しても，$P(\text{cat.n.01}|\text{dog.n.01}) = 0.06$，$P(\text{dog.n.01}|\text{cat.n.01}) = 0.04$，$P(\text{hotel.n.01}|\text{house.n.01}) = 0.01$，$P(\text{house.n.01}|\text{hotel.n.01}) = 0.05$ であり，戦略 S_r

と比べて小さくなっている．つまり，3.4 節で述べたように，戦略 S_n では兄弟関係にあるような概念的に近い語義間の関係を適切に学習できている．

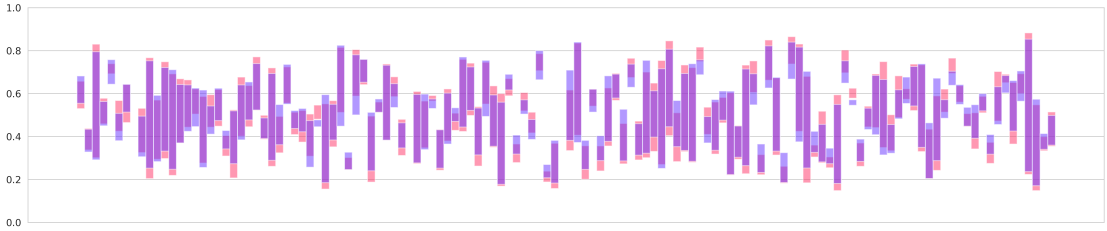


図 4.11: 語義 cat.n.01（赤）と dog.n.01（青）の箱埋め込み（$\mathcal{D}_{\text{living_thing.n.01}}$，ProtoBox S_n）

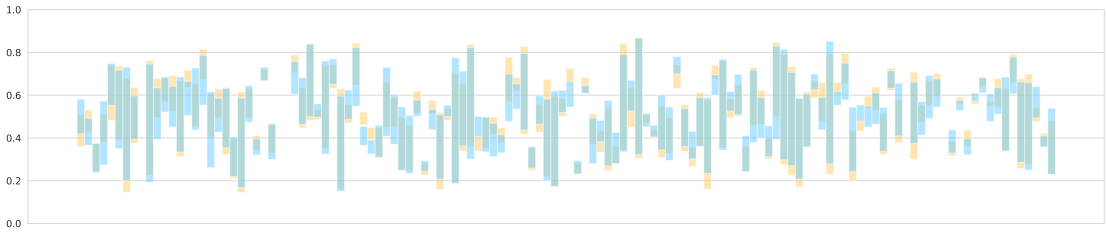


図 4.12: 語義 hotel.n.01（黄）と house.n.01（青）の箱埋め込み（$\mathcal{D}_{\text{artifact.n.01}}$，ProtoBox S_n）

第5章　おわりに

5.1　まとめ

本研究では，MetricWSD を拡張し，Prototypical Networks を用いて語義の箱埋め込みを学習する手法を提案した．さらに，学習時の損失を計算するために用いる小さいデータセット（エピソード）を作成する 2 通りの方法を提案した．学習したモデルを WSD，新語義の判定，新語義の上位語義の推定の 3 つのタスクに適用し，語義を単一のベクトルで表現する従来手法と実験的に比較した．

エピソードを作成する戦略として S_r と S_n の 2 つを提案した．戦略 S_r では，まず対象語義とその上位語義を選び，残りをランダムに選んだ．戦略 S_n では，対象語義とその上位語義に加えて下位語義と兄弟関係にある語義を選び，その後 S_r と同じく残りをランダムに選んだ．戦略 S_n では，ある語義の箱埋め込みが，特に下位語義や兄弟関係にある語義の箱埋め込みと重ならないことを重視した．

WSD の実験では，一部の名詞から構成されデータセットに出現する語義の数が少ないデータセットでは，提案手法はベースラインを上回った．しかし，名詞全体から構成され語義の数が多いデータセットでは，提案手法はベースラインを下回った．これは，語義の数が多くなると，語義の箱埋め込みが必ずしも適切に学習されないことを示唆するものであった．また，一部のデータセットでは，提案手法により低頻度語義に対する WSD の性能が向上することも確認できた．

新語義判定の実験では，WSD の結果とは異なり，名詞全体から構成され語義の数が多いデータセットでは，提案手法はベースラインを上回った．また，新語義かどうかを判定する際に用いる閾値の分布を見ると，先行研究の MetricWSD では狭い範囲に分布していたのに対し，提案手法では広い範囲に分布していることがわかった．

新語義の上位語義推定の実験では，提案手法は全体的にベースラインを上回った．上位語義の候補を絞り込む際に用いる閾値は 0.5，0.7，0.9 のいずれかと設定したが，データセットや戦略によって最良の結果が得られる閾値の設定は異なることがわかった．

5.2　今後の課題

本研究の今後の課題を以下に挙げる．

- 実験の結果や語義の箱埋め込みの可視化による考察から，提案手法では，語義の数が多くなると，語義の箱埋め込みが必ずしも適切に学習されないという問題があることがわかった．原因として，4.5 節で述べた箱埋め込みの次元数が不必要に大きいという問題以外に，現在のエピソードを作成する方法が語義の箱埋め込みの学習に最適ではないことが考えられる．このことから，箱の次元数を小さくして箱埋め込みを学習する実験を行ったり，エピソードの作成方法を改善したい．

- 本研究では，名詞のみを対象に語義の箱埋め込みを学習した．名詞は他の品詞に比べて単語や語義の数が多く，したがって上位下位関係にある語義も多いため，名詞を対象とした実験が語義の箱埋め込みによって語義間の上位下位関係をどれだけ正確に表現できるかを検証するのに適していると考えたためである．しかし，名詞だけでなく動詞にも上位下位関係はある．動詞の場合，WordNet 上では，名詞のように 1 つの根があるわけではなく，複数の根がある．また，それぞれの根の下位概念から構成される語義グラフの深さは浅い．こういった名詞とは異なる特徴をもつ動詞に対する提案手法の有効性を検証したい．

- 2.3 節で述べた双極埋め込みも，箱埋め込みと同じく，上位下位関係を表現できる．そのため，双極埋め込みなどの箱埋め込み以外の埋め込み表現を Prototypical Networks によって学習することにも取り組みたい．

謝辞

本研究を進めるにあたり多大なるご指導をいただきました白井清昭准教授に深厚なる謝意を表します．また，研究・生活全般にわたりお世話になりました白井研究室の皆様に感謝の意を示します.

参考文献

[1] BERT base model (uncased), (2023 年 12 月閲覧). https://huggingface.co/bert-base-uncased.

[2] SemCor 3.0, (2024 年 1 月閲覧). https://web.eecs.umich.edu/~mihalcea/downloads.html#semcor.

[3] Rami Aly, Shantanu Acharya, Alexander Ossa, Arne Köhn, Chris Biemann, and Alexander Panchenko. Every child should have parents: A taxonomy refinement algorithm based on hyperbolic term embeddings. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pp. 4811–4817, 2019.

[4] Edoardo Barba, Tommaso Pasini, and Roberto Navigli. ESC: Redesigning WSD with extractive sense comprehension. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 4661–4672, 2021.

[5] Michele Bevilacqua, Marco Maru, and Roberto Navigli. Generationary or "how we went beyond word sense inventories and learned to gloss". In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 7207–7221, 2020.

[6] Michele Bevilacqua and Roberto Navigli. Breaking through the 80% glass ceiling: Raising the state of the art in word sense disambiguation by incorporating knowledge graph information. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pp. 2854–2864, 2020.

[7] Terra Blevins and Luke Zettlemoyer. Moving down the long tail of word sense disambiguation with gloss informed bi-encoders. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pp. 1006–1017, 2020.

[8] Georgeta Bordea, Els Lefever, and Paul Buitelaar. SemEval-2016 task 13: Taxonomy extraction evaluation (TExEval-2). In *Proceedings of the 10th*

International Workshop on Semantic Evaluation (SemEval-2016), pp. 1081–1091, 2016.

[9] Niccolò Campolungo, Tommaso Pasini, Denis Emelin, and Roberto Navigli. Reducing disambiguation biases in NMT by leveraging explicit word sense information. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 4824–4838, 2022.

[10] Howard Chen, Mengzhou Xia, and Danqi Chen. Non-parametric few-shot learning for word sense disambiguation. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 1774–1781, 2021.

[11] Gabriella Chronis and Katrin Erk. When is a bishop not like a rook? when it's like a rabbi! multi-prototype BERT embeddings for estimating semantic relationships. In *Proceedings of the 24th Conference on Computational Natural Language Learning*, pp. 227–244, 2020.

[12] Shib Dasgupta, Michael Boratko, Dongxu Zhang, Luke Vilnis, Xiang Li, and Andrew McCallum. Improving local identifiability in probabilistic box embeddings. In *Advances in Neural Information Processing Systems*, Vol. 33, pp. 182–192, 2020.

[13] Tim Dettmers, Pasquale Minervini, Pontus Stenetorp, and Sebastian Riedel. Convolutional 2d knowledge graph embeddings. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence and Thirtieth Innovative Applications of Artificial Intelligence Conference and Eighth AAAI Symposium on Educational Advances in Artificial Intelligence*, 2018.

[14] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pp. 4171–4186, 2019.

[15] Philip Edmonds and Scott Cotton. SENSEVAL-2: Overview. In *Proceedings of SENSEVAL-2 Second International Workshop on Evaluating Word Sense Disambiguation Systems*, pp. 1–5, 2001.

[16] Hristea Florentina and Colhon Mihaela. The long road from performing word sense disambiguation to successfully using it in information retrieval: An

overview of the unsupervised approach. *Comput. Intell.*, Vol. 36, pp. 1026–1062, 2020.

[17] Tianyu Gao, Xingcheng Yao, and Danqi Chen. SimCSE: Simple contrastive learning of sentence embeddings. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pp. 6894–6910, 2021.

[18] Jin Huang, Zhaochun Ren, Wayne Xin Zhao, Gaole He, Ji-Rong Wen, and Daxiang Dong. Taxonomy-aware multi-hop reasoning networks for sequential recommendation. In *Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining*, p. 573–581, 2019.

[19] Luyao Huang, Chi Sun, Xipeng Qiu, and Xuanjing Huang. GlossBERT: BERT for word sense disambiguation with gloss knowledge. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pp. 3509–3514, 2019.

[20] Song Jiang, Qiyue Yao, Qifan Wang, and Yizhou Sun. A single vector is not enough: Taxonomy expansion via box embeddings. In *Proceedings of the ACM Web Conference 2023*, pp. 2467–2476, 2023.

[21] Sawan Kumar, Sharmistha Jat, Karan Saxena, and Partha Talukdar. Zero-shot word sense disambiguation using sense definition embeddings. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pp. 5670–5681, 2019.

[22] Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pp. 7871–7880, 2020.

[23] Bang Liu, Weidong Guo, Di Niu, Chaoyue Wang, Shunnan Xu, Jinghong Lin, Kunfeng Lai, and Yu Xu. A user-centered concept mining system for query and document understanding at tencent. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, p. 1831–1841, 2019.

[24] Mingyu Derek Ma, Muhao Chen, Te-Lin Wu, and Nanyun Peng. HyperExpan: Taxonomy expansion with hyperbolic representation learning. In *Findings of the Association for Computational Linguistics: EMNLP 2021*, pp. 4182–4194, 2021.

[25] Marco Maru, Simone Conia, Michele Bevilacqua, and Roberto Navigli. Nibbling at the hard core of Word Sense Disambiguation. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 4724–4737, 2022.

[26] George A. Miller. Wordnet: A lexical database for english. *Commun. ACM*, Vol. 38, No. 11, p. 39–41, 1995.

[27] George A. Miller, Martin Chodorow, Shari Landes, Claudia Leacock, and Robert G. Thomas. Using a semantic concordance for sense identification. In *Human Language Technology: Proceedings of a Workshop held at Plainsboro, New Jersey, March 8-11, 1994*, 1994.

[28] Andrea Moro and Roberto Navigli. SemEval-2015 task 13: Multilingual all-words sense disambiguation and entity linking. In *Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015)*, pp. 288–297, 2015.

[29] Roberto Navigli. Word sense disambiguation: A survey. *ACM Comput. Surv.*, Vol. 41, No. 2, 2009.

[30] Roberto Navigli, David Jurgens, and Daniele Vannella. SemEval-2013 task 12: Multilingual word sense disambiguation. In *Second Joint Conference on Lexical and Computational Semantics (*SEM), Volume 2: Proceedings of the Seventh International Workshop on Semantic Evaluation (SemEval 2013)*, pp. 222–231, 2013.

[31] Maximillian Nickel and Douwe Kiela. Poincaré embeddings for learning hierarchical representations. In *Advances in Neural Information Processing Systems*, Vol. 30, 2017.

[32] Yasumasa Onoe, Michael Boratko, Andrew McCallum, and Greg Durrett. Modeling fine-grained entity types with box embeddings. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pp. 2051–2064, 2021.

[33] Sameer Pradhan, Edward Loper, Dmitriy Dligach, and Martha Palmer. SemEval-2007 task-17: English lexical sample, SRL and all words. In *Proceedings of the Fourth International Workshop on Semantic Evaluations (SemEval-2007)*, pp. 87–92, 2007.

[34] Alessandro Raganato, Jose Camacho-Collados, and Roberto Navigli. Word sense disambiguation: A unified evaluation framework and empirical comparison. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, pp. 99–110, 2017.

[35] Nils Reimers and Iryna Gurevych. Sentence-BERT: Sentence embeddings using Siamese BERT-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pp. 3982–3992, 2019.

[36] Seifollahi Saeed and Shajari Mehdi. Word sense disambiguation application in sentiment analysis of news headlines: an applied approach to forex market prediction. *Journal of Intelligent Information Systems*, Vol. 52, pp. 57–83, 2019.

[37] Jiaming Shen, Zhihong Shen, Chenyan Xiong, Chi Wang, Kuansan Wang, and Jiawei Han. TaxoExpan: Self-supervised taxonomy expansion with position-enhanced graph neural network. In *Proceedings of The Web Conference 2020*, pp. 486–497, 2020.

[38] Jake Snell, Kevin Swersky, and Richard Zemel. Prototypical networks for few-shot learning. In *Advances in Neural Information Processing Systems*, Vol. 30, 2017.

[39] Benjamin Snyder and Martha Palmer. The English all-words task. In *Proceedings of SENSEVAL-3, the Third International Workshop on the Evaluation of Systems for the Semantic Analysis of Text*, pp. 41–43, 2004.

[40] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, Vol. 30, 2017.

[41] Zhibiao Wu and Martha Palmer. Verb semantics and lexical selection. In *32nd Annual Meeting of the Association for Computational Linguistics*, pp. 133–138, 1994.

[42] Yue Yu, Yinghao Li, Jiaming Shen, Hao Feng, Jimeng Sun, and Chao Zhang. STEAM: Self-supervised taxonomy expansion with mini-paths. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pp. 1026–1035, 2020.