コンピュータ科学科
知能システム系
卒業論文

文の意味的類似度評価における グラフ構造の利用

2022 年 2 月

101830073　加藤 辰弥

要 旨

文の類似度を求めることは, 翻訳の性能評価のほか, 迷惑メールやスミッシングの検出などに有用である．文の類似度を求める手法としては, 編集距離や共通する単語数, さらには構文情報を用いた手法などが利用されてきたが, いずれも表層的な類似度を計算するだけで, 意味的な類似度を考慮していない．そこで, 文の意味的な類似度を評価する STS タスクが提案されている．STS タスクでは, 文 “Kids in red shirts are playing in the leaves” と文 “Children in red shirts are playing in the leaves” は意味的類似度が高く, 文 “A woman is riding a horse” と文 “A man is opening a small package that contains headphones” は意味的類似度が低いと評価する．

ニューラルネットワークの発展に伴い, STS タスクでは LSTM や RNN などの自然言語処理モデルを用いた手法が提案された．LSTM を用いた手法では, Elvys らがピアソンの相関係数 0.8549 という結果を示している．LSTM や RNN は文の意味情報だけを利用し, 文の依存関係のような構文情報を利用していない．意味情報に加えて構文情報も扱うことにより, 文を表す特徴量を増やすことが期待できる．構文情報はグラフで表現できる．

そこで, 構文情報からのアプローチとして, グラフ構造と, それを処理するためのグラフニューラルネットワーク (GNN) の利用を検討した．代表的な GNN として, Graph Convolutional Network (GCN) や Graph Attention Network (GAT), GraphSAGE といった手法がある．また, グラフ編集距離を近似的に求める SimGNN や, GCN を用いたグラフプーリング手法である Self-Attention Graph Pooling (SAGPool) も提案されている．

本研究の目的は, STS タスクにおいて, グラフ構造を用いたアプローチの有効性を明らかにすることである．

実験は, 文からのグラフの作成, ノードの畳み込み, グラフを表現する埋め込み生成, グラフ間の類似度の評価という流れで行い, それぞれの過程で 2 通りずつの方法を採用し, 計 16 通りの方法を試した．実験の評価にはピアソンの相関係数を利用した．データセットには, 短文データセットである Sentences Involving Compositional Knowledge (SICK) と混在長データセットである STS Benchmark (STS-b) の二つのデータセットを用いた．

実験の結果, 16 通りの方法のうち, グラフの作成には単語の依存関係に基づく構築法を, ノードの畳み込みには GCN を, グラフを表現する埋め込みの生成にはアテンションモジュールを, グラフ間の類似度評価にはニューラルテンソルネットワークをそれぞれ利用した方法において最も優れた結果が得られた．この方法では SICK で, ピアソンの相関係数 0.848 ± 0.014 という結果が得られた．これは Elvys らの結果と同等であり, このことは STS タスクにおけるグラフ構造の利用の有効性を示唆している．しかし, STS-b では十分な結果を得られなかった．この理由として 次の二つが考えられる．第一に, 作成した

グラフには構文情報は含まれていたが, 意味情報が十分に含まれていなかったことが挙げられる. これは, 単語の埋め込みの生成に文脈を考慮しないモデルを利用したことに起因していると推測される. 第二に, グラフサイズに差が生まれることが挙げられる. これは, SICK は短文データセットであるのに対し STS-b は混在長データセットであることに起因していると推測される.

将来の展望としては, 単語の埋め込み生成に文脈を考慮したモデルを用いることで, 今回のモデルの欠点を補うことが期待できる.

目次

第 1 章　はじめに 1

第 2 章　関連研究 3

2.1　fastText . . . 3

2.1.1　Skip-gram モデル . . . 3

2.1.2　サブワードモデル . . . 4

2.2　グラフニューラルネットワーク . . . 5

2.2.1　GNN の歴史 . . . 5

2.2.2　Graph Convolutional Network . . . 6

2.2.3　GraphSAGE . . . 6

2.2.4　SimGNN . . . 9

2.2.5　Self-Attention Graph Pooling (SAGPool) . . . 10

第 3 章　提案手法 11

3.1　文からのグラフ作成 . . . 12

3.1.1　ノード生成 . . . 12

3.1.2　エッジ生成 . . . 12

3.2　ノードの特徴量畳込み . . . 13

3.2.1　Graph Convolutional Network (GCN) . . . 13

3.3　グラフを表現する埋め込み生成 . . . 14

3.3.1　アテンションモジュール . . . 14

3.3.2　SAGPool . . . 14

3.4　グラフ間の類似度計算 . . . 14

3.4.1 ニューラルテンソルネットワーク 15
3.4.2 コサイン類似度 15

第 4 章 実験 17
4.1 実験環境 17
4.2 データセット 17
4.3 グラフの作成 18
4.3.1 ノード生成 18
4.3.2 エッジ生成 18
4.4 ハイパーパラメータ 19
4.5 評価指標 19
4.6 ベースライン 19
4.7 実験結果 19

第 5 章 考察 21
5.1 グラフ構築法 21
5.2 GNN モデル 21
5.3 データセット 22
5.3.1 単語の意味情報の差 22
5.3.2 文の長さの差 23

第 6 章 おわりに 25

謝辞 27

表目次

4.1 データセットの詳細 18
4.2 実験結果 20
4.3 ベースラインの実験結果 20

5.1 データセットに含まれる文の例 23
5.2 データセットに含まれる最長文と最短文 (学習用データ) 24

図目次

2.1 GraphSAGE の前方伝播 ([1] より引用) 7
2.2 トランスダクティブ学習 8
2.3 帰納学習 8

3.1 モデルの概要 11

第 1 章

はじめに

文の類似度を求めることは, 翻訳の性能評価のほか, 迷惑メールやスミッシングの検出などに有用である．文の類似度を求める手法としては, 編集距離や共通する単語数, さらには構文情報を用いた手法などが利用されてきたが, いずれも表層的な類似度を計算するだけで, 意味的な類似度を考慮していない．そこで, 文の意味的な類似度を評価する STS タスクが提案されている．STS タスクでは, 文 “Kids in red shirts are playing in the leaves” と文 “Children in red shirts are playing in the leaves” は意味的類似度が高く, 文 “A woman is riding a horse” と文 “A man is opening a small package that contains headphones” は意味的類似度が低いと評価する.

ニューラルネットワークの発展に伴い, STS タスクでは LSTM や RNN などの自然言語処理モデルを用いた手法が提案された．LSTM を用いた手法では, Elvys ら [2] が Sentences Involving Compositional Knowledge (SICK) [3] データセットでピアソンの相関係数 0.8549 という結果を示している．LSTM や RNN は文の意味情報だけを利用し, 文の依存関係のような構文情報を利用していない．意味情報に加えて構文情報も扱うことにより, 文を表す特徴量を増やすことが期待できる．構文情報はグラフで表現できる.

そこで, 構文情報からのアプローチとして, グラフ構造と, それを処理するためのグラフニューラルネットワーク (GNN) の利用を検討した．代表的な GNN として, Graph Convolutional Network (GCN) [4] や Graph Attention Network (GAT) [5], GraphSAGE [1] といった手法がある．これらの手法を用いたタスクとしてはノード分類やリンク予測, グラフ分類などが挙げられる．また, グラフ編集距離を近似的に求める SimGNN や, GCN を用いたグラフプーリング手法である Self-Attention Graph Pooling

(SAGPool) も提案されている. 現実世界では, Apple 社ではスマートアシスタントのトリガーフレーズ検出性能向上 [6] や, Amazon 社での AutoKnow[7] を用いた製品カテゴリの識別, Uber 社の UberEats でのレコメンドシステム [8][9] など, 様々な場面で GNN が用いられている.

本研究の目的は, STS タスクにおいて, グラフ構造を用いたアプローチの有効性を明らかにすることである. 文中の単語間の依存関係や隣接関係を元に文をグラフ化し利用することで, 文を表す特徴量を増やすことを試みた. グラフ構造を利用するにあたって, 学習には GNN を用いた.

本論文の構成を示す. 第 2 章で本研究の関連研究について述べる. 第 3 章で文からのグラフ構築や, 文間の類似度計算などの提案手法について述べる. 第 4 章で実験, 第 5 章で実験結果に対する考察について述べる. 第 6 章で, 本論文を通して得られた知見や今後の展望についてのまとめを述べる.

第 2 章

関連研究

本章では, 本研究で取り扱っている基礎知識について説明する. 本章の構成は次の通りである.2.1 節では fastText について述べる. 2.2 節ではグラフニューラルネットワークについて述べる.

2.1 fastText

fastText[10] は Facebook AI Research によって開発された自然言語処理モデルである. このモデルを使用することで, 単語埋め込みの生成やテキストの分類ができる. 本節では, Word2Vec で用いられている Skip-gram モデルについて説明し, その後, fastText で提案されたサブワードモデルについて説明する.

2.1.1 Skip-gram モデル

Skip-gram モデルでは, サイズ W の単語の語彙が与えられ, 単語はそのインデックス $w \in \{1, ..., W\}$ によって識別される場合, 各単語 w のベクトル表現を学習することが目標となる. 分布仮説 (周囲の単語によって単語の意味が形成されるという仮説) に基づき, 単語のベクトル表現はその文脈に現れる単語を予測して学習される.Skip-gram モデルでは式 2.1 の対数尤度が最大になるように学習される.

$$\sum_{t=1}^{T} \sum_{c \in C_t} \log p(w_c | w_t) \tag{2.1}$$

ここで, T は与えられた文脈の長さ, C_t は注目単語 w_t の周辺単語のインデックスの集合である. 注目単語と周辺単語のペアを実数にスコア付けする関数として $s(w_t, w_c) = \mathbf{u}_{w_t}^{\mathsf{T}} \mathbf{v}_{w_c}$(ここで,$\mathbf{u}_{w_t}, \mathbf{v}_{w_c}$ は w_t, w_c に対応する単語ベクトル) が与えられるとする. 周辺単語が出現する確率 $p(w_c|w_t)$ の 1 つの選択肢として, 式 2.2 で示すソフトマックス関数が挙げられる.

$$p(w_c|w_t) = \frac{e^{s(w_t,w_c)}}{\sum_{j=1}^{W} e^{s(w_t,j)}} \tag{2.2}$$

しかし, ソフトマックス関数を用いると計算量が大きくなる (W は 10^5〜10^7 程度) という問題が存在する. そのため, 周辺単語の予測を, 独立した二値分類問題の集合として構成する (Negative Sampling). 注目単語 w_t に対して周辺単語 w_c を正の例とし, 周辺単語以外の語彙からランダムに抽出した例を負の例とみなす. その後, ロジスティック損失を用いて、式 2.3 に示す対数尤度を得る.

$$\log(1 + e^{-s(w_t,w_c)}) + \sum_{n \in \mathcal{N}_{t,c}} \log(1 + e^{s(w_t,n)}) \tag{2.3}$$

ここで, $\mathcal{N}_{t,c}$ が語彙からランダムに抽出した負の例の集合である. 最終的に, ロジスティック損失 $l : x \mapsto \log(1 + e^{-x})$ とすると, Skip-gram モデルの目的は式 2.4 を最小にすることである.

$$\sum_{t=1}^{T} \left[\sum_{c \in C_t} l(s(w_t, w_c)) + \sum_{n \in \mathcal{N}_{t,c}} l(-s(w_t, n)) \right] \tag{2.4}$$

Skip-gram モデルでは各単語に個別のベクトル表現を用いており, 単語の内部構造は考慮していない.

2.1.2 サブワードモデル

サブワードモデルでは, Skip-gram モデルで考慮されていなかった単語の内部構造を考慮するために, スコア付け関数を変更した. このモデルでは, 各単語の先頭と末尾に < と > を追加した, 文字単位での n-gram を用いる. 例えば, *where* という単語に対して, $n = 3$ の場合の n-gram の集合は以下のようになる.

{<*wh*, *whe*, *her*, *ere*, *re*>}

このような, 文字単位での n-gram を利用したスコア付け関数は式 2.5 で表される.

$$s(w,c) = \sum_{g \in \mathcal{G}_w} \mathbf{z}_g^{\mathsf{T}} \mathbf{v}_c \tag{2.5}$$

ここで, $\mathcal{G}_w$ は単語 w の n-gram(実際には $n = 3, 4, 5, 6$ のすべての n-gram) と w 自身 (*where* の例では <*where*>) の集合であり, $\mathbf{z}_g$ は n-gram g のベクトル表現である.

サブワードモデルでは, スコア付け関数を変えたことによって単語の内部構造を考慮できるようになり, 出現頻度が低い語彙に対して, 信頼性の高い単語のベクトル表現を学習することができるようになった. また, 単語の内部構造を考慮することにより, 未知語に対しても単語のベクトル表現を推論できるようになった.

2.2 グラフニューラルネットワーク

本節では, グラフニューラルネットワーク (GNN) に関する歴史や関連研究について述べる.

2.2.1 GNN の歴史

1997 年に Sperduti ら [11] により, ニューラルネットワーク上で初めてグラフ構造が扱われた. この研究は初期の GNN の先駆的な研究となった. その後, Goriet ら [12], Scarselli ら [13] , Gallicchio ら [14] により GNN に関する研究が行われてきた. これらの研究は RNN のアーキテクチャを用いてグラフのノード埋め込みを学習することを目的としている (RecGNNs). RecGNNs でのノードの更新は以下の 2 つの過程を繰り返す.

1. 隣接ノード (+ ノード自身) の状態を集約・解析 (メッセージパッシング)
2. ノード自身の状態を更新

周辺ノードの状態の集約はすべての層で同じ関数 (かつ同じ重み) を用いて再帰的に行い, すべてのノードの状態が平衡状態になるまでノードの更新を続ける.

RecGNNs での第 l 層におけるノード v の出力 $\mathbf{h}_v^l$ は式 2.6 で表される.

$$\mathbf{h}_v^l = \mathrm{Rec}(\mathbf{h}^{l-1}, \sum_{u \in \mathcal{N}(v)} W\mathbf{h}_u^l) \tag{2.6}$$

ここで, Rec は RNN, LSTM などの再帰型ネットワーク, $\mathcal{N}(v)$ はノード v の隣接ノードの集合である. RecGNNs には計算コストが非常に高いという問題がある. RecGNNs の

メッセージパッシングの概念は, GCN に受け継がれている.

2.2.2 Graph Convolutional Network

Graph Convolutional Network (GCN) [4] は,CNN のような畳み込み演算を GNN に適用したモデルである.1 層の GCN を用いるとそれぞれのノードが隣接するノードの情報を畳み込むという特徴がある. GCN は RecGNNs とは異なり, 層ごとに異なる重みを用いる. 隣接行列 A と特徴行列 X を GCN に与える入力とすると,GCN の第 l 層における出力 $\mathbf{h}^l$ は式 2.7 で表される.

$$\mathbf{h}^l = f(\tilde{D}^{-\frac{1}{2}}\tilde{A}\tilde{D}^{-\frac{1}{2}}\mathbf{h}^{l-1}W_1^{(l-1)}) \tag{2.7}$$

ここで,$\tilde{A} = A + I_N$ はノード自身へのエッジを追加した無向グラフの隣接行列であり,I_N は単位行列である. $\tilde{D}_{ii} = \sum_j \tilde{A}_{ij}$ であり,$W_1^{(l)}$ は層に応じた学習可能な重み行列,$f(\cdot)$ は $ReLU(\cdot) = \max(0, \cdot)$ のような活性化関数である. また,$H^{(0)} = X$ となる.

GCN は層の数を増やしすぎると, 全てのノードの埋め込みが同じ値に収束してしまい, 性能が低下することがわかっている.

2.2.3 GraphSAGE

GraphSAGE は GCN を拡張したものであり, 帰納学習 (本項末尾の補足で説明) を行うことができるモデルである. GCN との違いとして, GraphSAGE では近傍ノードを抽出し, それを集約するための aggregate 関数を学習を学習する.GraphSAGE での前方伝播では, まずノードの近傍ノードを抽出し, サブグラフを作成する. その後, 近傍ごとに異なる aggregate 関数を用いて, サブグラフ内のノードから特徴量を集約する. 最終的に集約した特徴量を元にラベルの推論を行う. 前方伝播の過程を図 2.1 に示す.

aggregate 関数のアーキテクチャには 3 種類が存在する.

2.2.3.1 Mean aggregator

Mean aggregator を用いた場合, ノード v に対する GraphSAGE の第 l 層の出力 $\mathbf{h}_v^l$ は式 2.8 で表される.

$$\mathbf{h}_v^l \leftarrow \sigma(\mathbf{W_2} \cdot \mathrm{MEAN}(\{\mathbf{h}_v^{l-1}\} \cup \{\mathbf{h}_u^{l-1}, \forall u \in \mathcal{N}(v)\})) \tag{2.8}$$

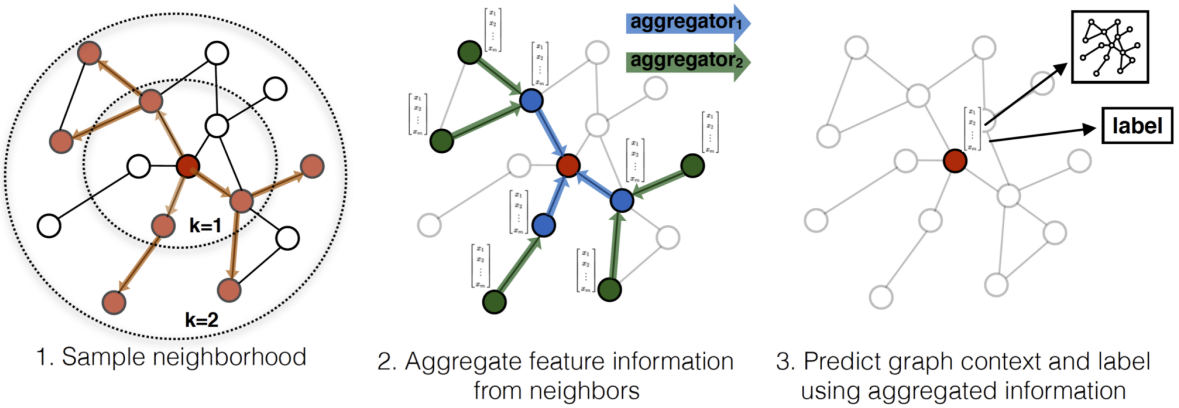

図 2.1　GraphSAGE の前方伝播 ([1] より引用)

ここで, $\mathbf{h}_v^l$ は第 l 層でのノード v の埋め込みであり, $\mathbf{W_2}$ は学習可能な重み行列, $\mathcal{N}(v)$ はノード v の近傍ノード郡である. このアーキテクチャは GCN と同様の処理を 帰納学習に対応させたものである.

2.2.3.2　LSTM aggregator

LSTM aggregator は, LSTM アーキテクチャに基づくより複雑な aggregator である. Mean aggregator に比べて, LSTM aggregator は表現力に優れている. しかし, LSTM は入力を逐次的に処理するため, ノードの入力順に応じて結果が変化する. そのため, ノードの近傍を不規則に並び替えることで, 非順序集合で LSTM を動作させた.

2.2.3.3　Pooling aggregator

Pooling aggregator は, 近傍ノードを全結合層に与え, その結果の中で最大となるものを選択する. これは, 式 2.9 で表される.

$$\mathrm{AGGREGATE}_k^{\mathrm{pool}} = \max(\{\sigma(\mathbf{W_2}_{pool}\mathbf{h}_{u_i}^k + \mathbf{b}), \forall u_i \in \mathcal{N}(v)\}) \tag{2.9}$$

ここで, σ は非線形の活性化関数であり, $\mathbf{b}$ はバイアスである.

補足:トランスダクティブ学習, 帰納学習

補足として, トランスダクティブ (Transductive) 学習, 帰納 (Inductive) 学習の 2 つの学習について説明をする. トランスダクティブ学習とは, 学習時にデータは存在するもののラベルが割り振られていないデータに対して, 推論を行う学習である. その様子を図

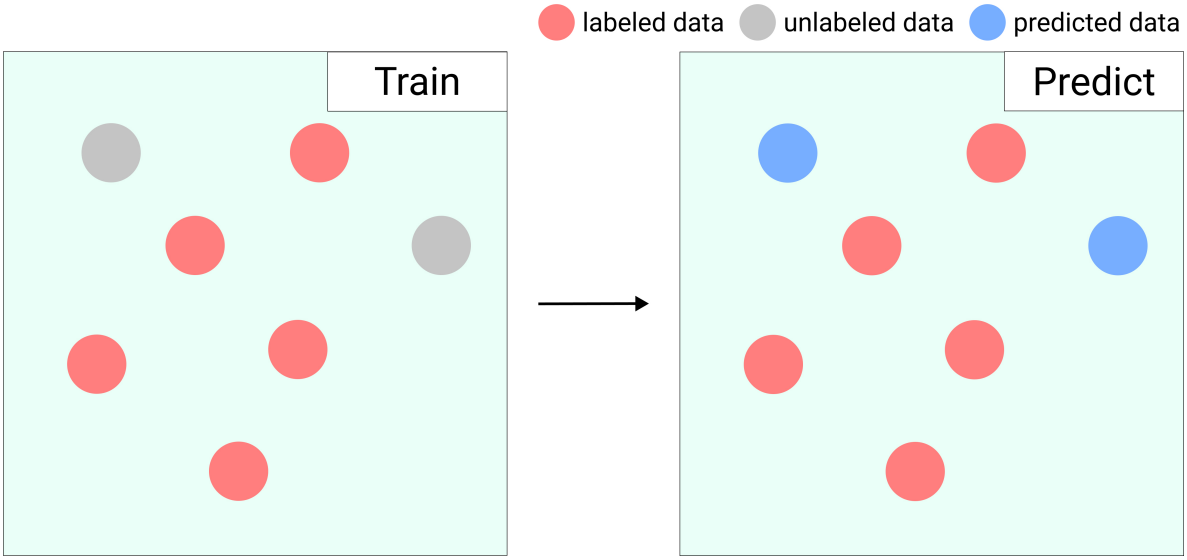

図 2.2　トランスダクティブ学習

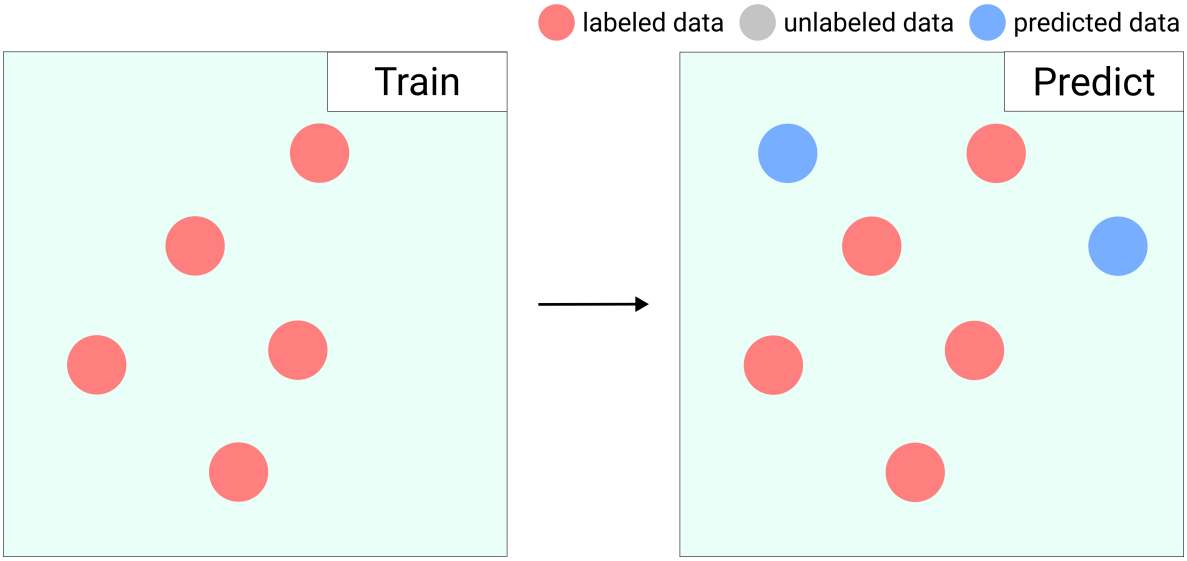

図 2.3　帰納学習

2.2 に示す.

それに比べて帰納学習とは, 学習時に存在しなかったデータに対しての推論を行う学習である. その様子を図 2.3 に示す.

2.2.4 SimGNN

SimGNN[15] は GNN を用いて近似的に Graph Edit Distance(GED) を求めるモデルである. グラフの入力ノードにはラベルが割り振られており, ノードの特徴量はラベルの One-Hot ベクトルとなる. この手法では, 入力グラフを複数層の GCN に与えることでノードレベルの埋め込みを生成した後, 以下に示すアテンションモジュールを元にグラフレベルの埋め込みを生成する. それらの非線形の変換をし, global graph context $c \in \mathbb{R}^D$ を得る.

$$c = \tanh(\frac{1}{N} W_3 \sum_{n=1}^{N} u_n) \tag{2.10}$$

ここで,$u_n \in \mathbb{R}^D$ は入力ノード n の埋め込み,D はノードの埋め込みの次元,$W_3 \in \mathbb{R}^{D \times D}$ は学習可能な重み行列である. c は重み行列を学習することで, グラフの全体的な構造及び特徴量を表すことができ, これに基づいて各ノードの重要度を計算することが可能となる.

ノード n に対しての重要度 a_n の計算は, ノード n の埋め込み u_n と c の内積を計算した後, シグモイド関数 $\sigma(x) = \frac{1}{1+exp(-x)}$ を用いることで, 重要度 a_n を $(0, 1)$ の範囲に収める. ここではグラフサイズを埋め込みに反映するため, 全体の重要度を正規化しない. 重要度を計算した後, グラフレベルの埋め込み $h \in \mathbb{R}^D$ を重み付きのノードの埋め込みの総和 $h = \sum_{n=1}^{N} a_n u_n$ で計算する. アテンションモジュールについてまとめると式 2.11 になる.

$$h = \sum_{n=1}^{N} \sigma(u_n^{\mathsf{T}} c) u_n = \sum_{n=1}^{N} \sigma(u_n^{\mathsf{T}} \tanh(\frac{1}{N} W_3 \sum_{m=1}^{N} u_m)) u_n \tag{2.11}$$

AttentionModule から 2 つのグラフの埋め込み $h_i \in \mathbb{R}^D, h_j \in \mathbb{R}^D$ が与えられた後, それらの関係をモデル化するために, 式 3.5 で表されるニューラルテンソルネットワークを用いる.

$$g(h_i, h_j) = f(h_i^{\mathsf{T}} W_4^{[1:K]} h_j + V \begin{bmatrix} h_i \\ h_j \end{bmatrix} + b) \tag{2.12}$$

ここで,$W_4^{[1:K]} \in R^{D \times D \times K}$ は重み行列,$\left[\ \right]$ は結合操作,$V \in \mathbb{R}^{K \times 2D}$ は重みベクトル. $b \in \mathbb{R}^K$ はバイアスベクトル,$f(\cdot)$ は $ReLU(\cdot) = \max(0, \cdot)$ のような活性化関数である. K は K は各グラフ埋め込みペアに対してモデルが生成する類似度の数を制御するハイパーパラメータである.

2.2.5 Self-Attention Graph Pooling (SAGPool)

Self-Attention Graph Pooling (SAGPool) [16] はグラフプーリング手法の 1 つである. SAGPool を用いることで Self-Attention に基づいたノードの削減を行うことができる. この手法では, ノードの重要度は式 2.13 で示される.

$$Z = \sigma(\tilde{D}^{-\frac{1}{2}}\tilde{A}\tilde{D}^{-\frac{1}{2}}X\Theta_{att}) \tag{2.13}$$

ここで, $\tilde{A} = A + I_N$ はノード自身へのエッジを追加した無向グラフの隣接行列であり,I_N は単位行列である. $\tilde{D}_{ii} = \sum_j \tilde{A}_{ij}$ であり Θ_{att} は SAGPool 層のパラメータである. GCN を用いることで, グラフの特徴と構造情報の両方に基づいた重要度を得ることができる. ノードの重要度を元に, Gao ら [17] によるノード選択方法を行うことで, 上位 $\lceil kN \rceil$ 個のノードを保持する. ここで,$k \in (0, 1)$ はプーリングの割合である. 式 2.14 はノード選択法を表す.

$$\text{idx} = \text{top-rank}(Z, \lceil kN \rceil), Z_{mask} = Z_{\text{idx}} \tag{2.14}$$

ここで, top-rank は上位 $\lceil kN \rceil$ 個のノードを返し, $\cdot_{\text{idx}}$ は idx が示すノードのみの重要度である. 最終的に, 式 2.15 に表すように, 上位 $\lceil kN \rceil$ 個のノードのみのグラフを作成する.

$$X' = X_{\text{idx},:}, X_{out} = X' \odot Z_{mask}, A_{out} = A_{\text{idx,idx}} \tag{2.15}$$

ここで, $X_{\text{idx},:}$ は idx が示すノードの特徴量行列であり, $\odot$ はアダマール積 (要素ごとの積) であり, $A_{\text{idx,idx}}$ は idx が示すノードの隣接行列である.

第 3 章

提案手法

本章では, 提案手法について説明する. 本研究は大きく分けて以下の 4 つに分類される.

1. 文からのグラフ作成
2. ノードの特徴量畳込み
3. グラフを表現する埋め込み生成
4. グラフ間の類似度の評価

それぞれの方法について 2 種類の手法を採用し, 計 16 種類の手法で比較を行った.

モデルの概要を図 3.1 に示す.

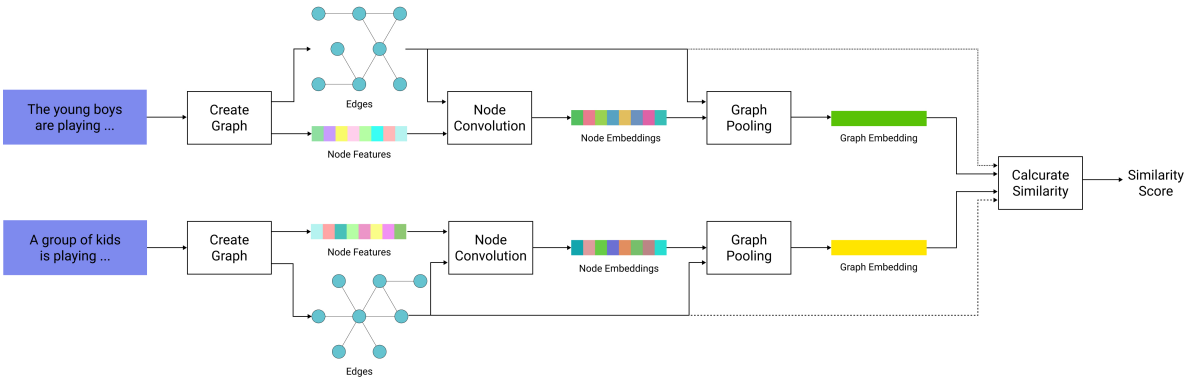

図 3.1　モデルの概要

3.1 節では文からのグラフ作成,3.2 節ではノード畳込み 3.3 節ではグラフを表現する埋め込み生成, 3.4 節ではグラフ間の類似度計算について述べる.

3.1　文からのグラフ作成

本節では, 文からのグラフ生成手法について説明する. グラフの生成について, 以下の 2 つを行う.

1. ノード生成
2. エッジ生成

また, 本研究では以下の 2 通りのエッジ生成方法を元に, 2 種類のグラフを生成し実験した.

1. 依存関係に基づくエッジ生成
2. 依存関係と隣接関係に基づくエッジ生成

3.1.1　ノード生成

文からのグラフの生成において, 各ノードはそれぞれの単語とし, その単語のレンマの埋め込みを取得し, ノードの特徴量とした.

3.1.2　エッジ生成

構文情報には単語間の依存情報や隣接関係などが存在する. そこで本研究では, 依存関係に基づくエッジの生成方法と, それに加えて隣接関係も追加したエッジの生成方法の 2 つを採用した. 隣接関係のみに基づくエッジの生成を行わなかった理由としては, 隣接関係のみだとグラフ構造が鎖状になってしまうためである.

依存関係に基づくエッジ生成

各エッジの生成には Universal Dependencies に基づく構文解析を行い, 単語間に依存関係があれば双方向にエッジを作成し, 無向グラフを生成した.

依存関係と隣接関係に基づくエッジ生成

依存関係に基づくエッジ生成に加えて, 単語間に隣接関係があれば双方向エッジを作成し, 無向グラフを生成した.

3.2 ノードの特徴量畳込み

本節では, ノードの特徴量畳込み手法について説明する. ノードの特徴量畳み込みでは 2 通りの方法を採用した.

3.2.1 Graph Convolutional Network (GCN)

この方法では, ノードの畳み込みに Kipf ら [4] と同様の Graph Convolutional Network (GCN) を用いた. この方法は SimGNN[15] でも用いられている. GCN の第 l 層における出力は式 3.1 で表される.

$$H^{(l)} = f(\tilde{D}^{-\frac{1}{2}}\tilde{A}\tilde{D}^{-\frac{1}{2}}H^{(l-1)}W_1^{(l-1)}) \tag{3.1}$$

ここで,$\tilde{A} = A + I_N$ はノード自身へのエッジを追加した無向グラフの隣接行列であり,I_N は単位行列である. $\tilde{D}_{ii} = \sum_j \tilde{A}_{ij}$ であり,$W_1^{(l)}$ は層に応じた学習可能な重み行列,$f(\cdot)$ は $ReLU(\cdot) = \max(0, \cdot)$ のような活性化関数である. また,$H^{(0)} = X$ となる.

3.2.1.1 GraphSAGE

この方法では, ノードの畳み込みに GraphSAGE[1] (SAGE) を用いた. 本手法を採用した背景は, 近傍ノードをサンプリングして学習する GraphSAGE を用いることで, グラフサイズの影響を減らして効果的に特徴量の畳み込みができるだろうと言う直感に基づいている. aggregator には, MeanAggregator を用いる. Mean aggregator を用いた場合, ノード v に対する GraphSAGE の第 l 層の出力は式 3.2 で表される.

$$h_v^l \leftarrow \sigma(\mathbf{W_2} \cdot \mathrm{MEAN}(\{\mathbf{h}_v^{l-1}\} \cup \{\mathbf{h}_u^{l-1}, \forall u \in \mathcal{N}(v)\})) \tag{3.2}$$

ここで, $\mathbf{h}_v^l$ は第 l 層でのノード v の埋め込みであり, $\mathbf{W_2}$ は学習可能な重み行列, $\mathcal{N}(v)$ はノード v の近傍ノード郡である.

3.3　グラフを表現する埋め込み生成

本節では, グラフを表現する埋め込み生成手法について説明する. グラフを表現する埋め込み生成では 2 通りの方法を採用した. これらの方法では, 前節の方法で得られたノードレベルの埋め込みを元にグラフを表現する埋め込みを生成する.

3.3.1　アテンションモジュール

この方法は SimGNN で用いられているアテンションモジュール (Att) と同様のものである. アテンションモジュールから出力される, グラフを表現する埋め込み h は以下の式で表される.

$$h = \sum_{n=1}^{N} \sigma(u_n^{\mathsf{T}} c) u_n = \sum_{n=1}^{N} \sigma(u_n^{\mathsf{T}} \tanh(\frac{1}{N} W_2 \sum_{m=1}^{N} u_m)) u_n \tag{3.3}$$

ここで, u_n はノード n の埋め込み, σ はシグモイド関数, N はノード数, W_2 は学習可能な重み行列である.

3.3.2　SAGPool

この方法では, グラフを表現する埋め込み生成に Self-Attention Graph Pooling (SAGPool) [16] を用いた. 本手法を採用した背景は, 畳み込みを行った後に最も周辺ノードの情報を集めることができたノードはグラフ全体を表しているのではないかという直感に基づいている. SAGPool では, 各ノードの重要度を式 3.4 で算出し, 重要度が最も高いノードの表現からグラフを表現する埋め込みを取得する.

$$Z = \sigma(\tilde{D}^{-\frac{1}{2}} \tilde{A} \tilde{D}^{-\frac{1}{2}} X \Theta_{att}) \tag{3.4}$$

ここで, $\tilde{A} = A + I_N$ はノード自身へのエッジを追加した無向グラフの隣接行列であり,I_N は単位行列である. $\tilde{D}_{ii} = \sum_j \tilde{A}_{ij}$ であり Θ_{att} は SAGPool 層のパラメータである.

3.4　グラフ間の類似度計算

本節では, グラフ間の類似度計算手法について説明する. グラフ間の類似度計算では 2 通りの方法を採用した. これらの方法では, 前節の方法で得られたグラフを表現する埋め

込みから類似度を計算する.

3.4.1 ニューラルテンソルネットワーク

この方法は SimGNN で用いられている ニューラルテンソルネットワーク (NTN) と同様のものである. 2 つのグラフの埋め込み h_i, h_j を ニューラルテンソルネットワークに与えた出力は式 3.5 である.

$$g(h_i, h_j) = f(h_i^{\mathsf{T}} W_3^{[1:K]} h_j + V \begin{bmatrix} h_i \\ h_j \end{bmatrix} + b) \tag{3.5}$$

を用いる. ここで,$W_3^{[1:K]} \in R^{D \times D \times K}$ は重み行列,$\begin{bmatrix} \end{bmatrix}$ は結合操作,$V \in \mathbb{R}^{K \times 2D}$ は重みベクトル. $b \in \mathbb{R}^K$ はバイアスベクトル, $f(\cdot)$ は $ReLU(\cdot) = \max(0, \cdot)$ のような活性化関数である. 得られた出力を 2 層の全結合層に与えることで, 最終的な類似度を得る.

3.4.2 コサイン類似度

この方法はコサイン類似度 (cos) を用いてグラフ間の類似度計算を行うものである. 本手法を採用した理由は, ベクトル間の類似度を求めるのに一般的に用いられているコサイン類似度を利用することで, ニューラルテンソルネットワークの有効性を検証するためである. 2 つのグラフの埋め込み h_i, h_j のコサイン類似度は, 式 3.6 で表される.

$$\cos(h_i, h_j) = \frac{h_i \cdot h_j}{|h_i||h_j|} \tag{3.6}$$

第 4 章

実験

4.1 実験環境

実験環境は以下のとおりである.

ハードウェア

- CPU: AMD Ryzen9 5900X(12 コア 24 スレッド)
- メモリ: 32GB
- GPU: RTX 3090
- GPU メモリ: 24GB

ソフトウェア

- OS: Ubuntu 20.04
- CUDA: 11.2
- cuDNN: 8.1
- PyTorch: 1.10.1

4.2 データセット

実験には Sentences Involving Compositional Knowledge (SICK) [3] データセットと Semantic Textual Similarity Benchmark (STS-b) [18] データセットを用いた. それぞれ

表 4.1　データセットの詳細

	train	test	validation	total
SICK	4,439	4,906	495	9,840
STS-b	5,749	1,379	1,500	8,628

のデータセットには文のペアとその類似度が 0-5 の範囲で含まれている. それぞれのデータセットの詳細は以下の表 4.1 の通りである.

4.3　グラフの作成

この節では, グラフの作成手法について述べる.

4.3.1　ノード生成

各ノードはそれぞれの単語とし, ノードを生成した. 各ノードの特徴量は, 対応する単語のレンマをもとに, 事前学習済み fastText[10] モデルから得られるものとした. ここで, 事前学習済みモデルには Common Crawl データセットでの事前学習済みモデルを利用した. 本研究では, 未知語に対応するために Word2Vec ではなく fastText を利用した.

4.3.2　エッジ生成

エッジの生成方法では,2 種類の方法について以下のように実験を行った.

4.3.2.1　依存関係に基づくエッジ生成

依存関係に基づくエッジ生成には, Stanza[19] ライブラリを利用した Universal Dependencies に基づく係り受け解析を用いた. 係り受け解析の結果から, 単語間に依存関係が存在した場合に双方向エッジを作成した.

4.3.2.2　依存関係と隣接関係に基づくエッジ生成

依存関係と隣接関係に基づくエッジ生成には, 依存関係に基づくエッジに加えて, 単語間に隣接関係があれば双方向エッジを作成した.

4.4 ハイパーパラメータ

この節ではモデルのハイパーパラメータについて説明する．Graph Convolutional Network (GCN) [4] , または GraphSAGE [1] の層数は 3 層とし, GCN の第 1 層, 第 2 層, 第 3 層の出力次元はそれぞれ 128, 64, 32 とした．また, NTN 層では $K = 16$ とした．

4.5 評価指標

モデルの性能評価にはピアソンの相関係数を用いた．

4.6 ベースライン

ベースラインには以下の 3 つを用いた．1 つ目が Elvys らによる, 長さ 5 の局所的文脈を考慮した Siamese LSTM (Elvys's Siamese LSTM), 2 つ目が TextSimGNN[20] でもベースラインとして用いられている, 畳み込み層, Max プーリング層, Flatten 層からなるモデル (SemEval-CNN) , 3 つ目が TextSIMGNN のグラフ構築法 2a (STGCM2a) , 4 つ目が TextSIMGNN のグラフ構築法 3 (STGCM3) である．

4.7 実験結果

実験結果を表 4.2, ベースラインの実験結果を表 4.3 に示す．表 4.2 の 2 行目では, 依存関係のみに基づくグラフ構築法を UD, 依存関係と隣接関係に基づくグラフ構築法を UD-adjacent と表記している．また, 表 4.2 の 1 列目の略称は, 第 3 章で示したそれぞれの手法の略称と対応している．

表 4.2　実験結果

	SICK		STS-b	
	UD	UD-adjacent	UD	UD-adjacent
GCN-Att-NTN	$\mathbf{0.848 \pm 0.014}$	$\mathbf{0.829 \pm 0.019}$	$\mathbf{0.419 \pm 0.052}$	$\mathbf{0.406 \pm 0.004}$
SAGE-Att-NTN	0.830 ± 0.020	0.829 ± 0.019	0.389 ± 0.042	0.387 ± 0.054
GCN-SAGPool-NTN	0.767 ± 0.022	0.732 ± 0.037	0.312 ± 0.067	0.351 ± 0.095
SAGE-SAGPool-NTN	0.700 ± 0.041	0.723 ± 0.047	0.219 ± 0.044	0.270 ± 0.046
GCN-Att-cos	0.754 ± 0.008	0.754 ± 0.011	0.400 ± 0.013	0.398 ± 0.011
SAGE-Att-cos	0.751 ± 0.010	0.744 ± 0.010	0.347 ± 0.020	0.343 ± 0.021
GCN-SAGPool-cos	0.462 ± 0.045	0.581 ± 0.040	0.315 ± 0.025	0.245 ± 0.021
SAGE-SAGPool-cos	0.586 ± 0.041	0.498 ± 0.025	0.321 ± 0.017	0.309 ± 0.017

表 4.3　ベースラインの実験結果

	SICK	STS-b
Elvys's Siamese LSTM	0.855	0.758
SemEval-CNN	0.855 ± 0.015	0.776 ± 0.015
TextSimGNN (STGCM2a)	0.754 ± 0.010	0.352 ± 0.010
TextSIMGNN (STGCM3)	0.428 ± 0.010	0.786 ± 0.010

第 5 章

考察

本章ではグラフ構築法, GNN モデル, データセットについての考察を述べる.

5.1 グラフ構築法

実験の結果, 2 通りのグラフ構築法のうち, 依存関係のみに基づくグラフ構築法がわずかに依存関係と隣接関係を用いたグラフ構築法よりも相関係数が高くなった. 情報量が増えたのにもかかわらずこのような結果となった理由として, 本実験のようにエッジの重みが同等である場合に, 隣接関係を含めたことによって依存関係の情報が埋もれてしまったことに起因していると推測される. Graph Attention Network (GAT) などのエッジの重みを考慮するモデルを利用することで, この問題は解決できることが期待される.

また, 依存関係に基づくグラフ構築法が Sentence Involving Compositional Knowledge (SICK) [3] データセットでは TextSimGNN [20] のどのグラフ構築法よりも優れていた. このことは, 文の意味的類似度評価タスクにおいて, 依存関係に基づくグラフ構築法の有効性を示唆している.

5.2 GNN モデル

実験の結果, 8 通りのモデルのうち, ノードの畳み込みには GCN[4] を, グラフを表現する埋め込みの生成にはアテンションモジュール [15] を, グラフ間の類似度評価にはニューラルテンソルネットワーク [21] をそれぞれ利用した方法において SICK データセットでは 0.848 ± 0.014 , STS Benckmark (STS-b) [18] データセットでは 0.419 ± 0.052 と最も

優れた結果が得られた. この方法はグラフの作成を除き Bai ら [15] や [20] と同様のモデルを利用している. また, ノードの畳み込みに GraphSAGE [1] を用いた場合でも, SICK データセットでは 0.830 ± 0.020 , STS-b データセットで 0.389 ± 0.042 という同等の結果を得られた. SICK データセットに関しての結果は, Elvys ら [2] と同等の結果を示した. このことは, 文の意味的類似度評価タスクにおけるグラフニューラルネットワークの有効性を示唆している.

しかし, SAGPool を用いた手法では相関係数が大きく下がった. これは, SAGPool 内で GCN を用いたことにより, GCN の層数が増えてしまった結果, SAGPool 内で扱う重要度が近い値に収束してしまったことに起因していると推測される. また, SAGPool と GraphSAGE を組み合わせた方法について, 更に相関係数が下がってしまった結果としては SAGPool での重要度の算出には GCN を用いているが, ノードの畳み込みには GraphSAGE を用いていることにより整合性がない状態になってしまっていることに起因していると推測される.

グラフ間の類似度評価にコサイン類似度を用いた場合にも相関係数が下がった. コサイン類似度では, [21] で述べられているように, 2 つのデータの表現の相互作用が不十分になることが多いことが知られている.

5.3 データセット

本実験で用いた 2 つのデータセットのうち, SICK データセットではベースラインと同等の結果を示したが, STS-b データセットでは TextSimGNN (STGCM2a) を除くすべてのベースラインを下回る結果となった. このことについては 2 つのことに起因していると考えられる. 本節第 1 項ではデータセット内の単語の意味情報の差について, 第 2 項ではデータセット内の文の長さの差について考察する.

5.3.1 単語の意味情報の差

表 5.1 に示すように, SICK データセットでは含まれる単語が類似している場合に文の類似度は高い傾向があるが, STS-b データセットでは含まれる単語が類似していても文の類似度が低いことが多々ある.

STS-b データセットでは類似度が高いものは文の言い換えであるものが多く, モデルは

表 5.1 データセットに含まれる文の例

	Sentence 1	Sentence 2	Similarity
SICK	A woman is boiling noodles in water	The person is boiling noodles	4.5
SICK	There are no children playing and waiting	Three Asian kids are dancing and a man is looking	1.6
STS-b	The man is using a sledghammer to break the concrete block that is on the other man	A man breaks a slab of concrete that is lying on a prone man with a sledge hammer	5.0
STS-b	A man is playing the piano	A man is playing the trumpet	1.6

より文の意味情報を学習する必要がある．しかし今回のグラフ構築法では, 単語の埋め込み生成で文脈を考慮しなかった．このことによる意味情報の不足が実験結果に影響を与えたと推測される．単語の埋め込み生成で, BERT などの文脈を考慮したモデルを利用することでこの問題が解決できることが期待できる．

5.3.2 文の長さの差

SICK データセットでは主に短文が含まれているのに対して, STS-b データセットでは文の長さが様々である．学習用データにおいて, 表 5.2 に示すように, 最長文と最短文の間の単語数の差は SICK データセットでは 29 であるのに対して, STS-b データセットではその倍程度の 53 となっている．このことにより生じるグラフサイズの差により, STS-b データセットでは学習が適切に行えなかったのではないかと推測される．

表 5.2　データセットに含まれる最長文と最短文 (学習用データ)

	Longest Sentence	Shortest Sentence	Word Count Gap
SICK	A piece of bread, which is big, is having butter spread upon it by a man OR A piece of bread, which is big, is being spread with butter by a man	Men are sawing	29
STS-b	iraq has been lobbying for the security council to stop using the country's oil revenue to pay compensation to victims of the 1991 gulf war and the salaries of the united nations monitoring, verification and inspection commission inspectors and to have all money remaining in the united nation's oil-for-food accounts transferred to the government's development fund.	Two robots kiss.	53

第 6 章

おわりに

本研究では, 文の意味的類似度評価におけるグラフ構造の利用手法を提案した. 実験は, 文からのグラフの作成, ノードの畳み込み, グラフを表現する埋め込み生成, グラフ間の類似度の評価という流れで行い, それぞれの過程で 2 通りずつの方法を採用し, 計 16 通りの方法を試した. 16 通りの方法のうち, グラフの作成には単語の依存関係に基づく構築法を, ノードの畳み込みには GCN を, グラフを表現する埋め込みの生成にはアテンションモジュールを, グラフ間の類似度評価にはニューラルテンソルネットワークをそれぞれ利用した方法において最も優れた結果が得られた. この方法では Sentence Involving Compositional Knowledge データセット で, ピアソンの相関係数 0.848 ± 0.014 という結果が得られた. これは Elvys らの結果と同等であり, このことは STS タスクにおけるグラフ構造の利用の有効性を示唆している.

将来の展望としては, 単語の埋め込み生成に文脈を考慮したモデルを用いることで, 今回のモデルの欠点を補うことが期待できる.

謝辞

本研究を進めるにあたり、多くの方々にご指導ご鞭撻を賜りました。

本論文の作成にあたり, 名古屋大学大学院情報科学研究科 外山勝彦教授，小川泰弘准教授には研究に必要な知識や，本論文を執筆する上でのアドバイスを頂きました．また, 研究内容に関する相談にも乗っていただくなど, 終始熱心なご指導を頂きました. ここに深謝の意を表します.

最後に、本研究ならびに学業全般にわたって経済的・心身的に支援して下さる家族に深く感謝し、お礼を申し上げます。

参考文献

[1] William L. Hamilton, Rex Ying, and Jure Leskovec. Inductive representation learning on large graphs. In *Proceedings of the 31st Neural Information Processing Systems (NeurIPS)*, pp. 1025–1035, 2017.

[2] L. Pontes Elvys, Huet Stéphane, C. Linhares Andréa, and Torres-Moreno Juan-Manuel. Predicting the semantic textual similarity with siamese cnn and lstm. In *Actes de la 25e conférence Traitement Automatique des Langues Naturelles (TALN)*, pp. 311–319, 2018.

[3] Islam Beltagy, Stephen Roller, Pengxiang Cheng, Katrin Erk, and Raymond J. Mooney. Representing meaning with a combination of logical and distributional models. *Computational Linguistics*, pp. 763–808, 2016.

[4] Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *Proceedings of the 5th International Conference on Learning Representations (ICLR)*, pp. 1–14, 2017.

[5] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph attention networks. In *Proceedings of the 6th International Conference on Learning Representations (ICLR)*, pp. 1–12, 2018.

[6] Dighe Pranay, Adya Saurabh, Li Nuoyu, Vishnubhotla Srikanth, Naik Devang, Sagar Adithya, Ma Ying, Pulman Stephen, and D. Williams Jason. Lattice-based improvements for voice triggering using graph neural networks. In *Proceedings of 2020 IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pp. 7459–7463, 2020.

[7] L. Dong Xin, He Xiang, Kan Andrey, Li Xian, Liang Yan, Ma Jun, E. Xu Yifan, Zhang Chenwei, Zhao Tong, B. Saldana Gabriel, Deshpande Saurabh, M. Man-

duca Alexandre, Ren Jay, P. Singh Surender, Xiao Fan, Chang Haw-Shiuan, Karamanolakis Giannis, Mao Yuning, Wang Yaqing, Faloutsos Christos, McCallum Andrew, and Jiawei Han. Autoknow: self-driving knowledge collection for products of thousands of types. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (KDD)*, pp. 2724–2734, 2020.

[8] Ankit Jain, Isaac Liu, Ankur Sarda, and Piero Molino. Food discovery with uber eats: Using graph learning to power recommendations. https://eng.uber.com/uber-eats-graph-learning/, 2019. Accessed: 2022-02-07.

[9] Avishek J. Bose, Ankit Jain, Piero Molino, and William L. Hamilton. Meta-graph: Few shot link prediction via meta learning. In *Proceedings of the 8th International Conference on Learning Representations (ICLR)*, pp. 1–15, 2020.

[10] Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics (TACL)*, pp. 135–146, 2017.

[11] Alessandro Sperduti and Antonina Starita. *Supervised neural networks for the classification of structures*, pp. 714–735. IEEE Transactions on Neural Networks, 1997.

[12] Marco Gori, Gabriele Monfardini, and Franco Scarselli. A new model for learning in graph domains. In *Proceedings of 2005 IEEE International Joint Conference on Neural Networks (IJCNN)*, pp. 729–734, 2005.

[13] Franco Scarselli, Marco Gori, C. Ah Tsoi, Markus Hagenbuchner, and Gabriele Monfardini. *The Graph Neural Network Model*, pp. 61–80. IEEE Transactions on Neural Networks, 2009.

[14] Claudio Gallicchio and Alessio Micheli. Graph echo state networks. In *Proceedings of The 2010 International Joint Conference on Neural Networks (IJCNN)*, pp. 1–8, 2010.

[15] Yunsheng Bai, Hao Ding, Song Bian, Ting Chen, Yizhou Sun, and Wei Wang. Simgnn: A neural network approach to fast graph similarity computation. In *Proceedings of the Twelfth ACM International Conference on Web Search and*

Data Mining (WSDM), pp. 384–392, 2019.

[16] Junhyun Lee, Lee Inyeop, and Kang Jaewoo. Self-attention graph pooling. In *Proceedings of 36th International Conference on Machine Learning (IMCL)*, pp. 6661–6670, 2019.

[17] Hongyang Gao and Shuiwang Ji. Graph u-nets. In *Proceedings of 36th International Conference on Machine Learning (IMCL)*, pp. 2083–2092, 2019.

[18] Daniel Cer, Mona Diab, Eneko Agirre, Iñigo Lopez-Gazpio, and Lucia Specia. SemEval-2017 task 1: Semantic textual similarity multilingual and crosslingual focused evaluation. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval)*, pp. 1–14, 2017.

[19] Peng Qi, Yuhao Zhang, Yuhui Zhang, Jason Bolton, and Christopher D. Manning. Stanza: A Python natural language processing toolkit for many human languages. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics: System Demonstrations (ACL)*, pp. 101–108, 2020.

[20] Ke Zhou, Ke Xu, Tanfeng Sun, and Yueguo Zhang. Sentence modeling via graph construction and graph neural networks for semantic textual similarity. In *Proceedings of 13th International Congress on Image and Signal Processing, BioMedical Engineering and Informatics (CISP-BMEI)*, pp. 413–418, 2020.

[21] Richard Socher, Danqi Chen, Christopher D Manning, and Andrew Ng. *Reasoning With Neural Tensor Networks for Knowledge Base Completion*, pp. 926–934. 2013.