時系列文書に対する効率的なエンティティリンキング

長城　沙樹†　　北川　博之††

† 筑波大学システム情報工学研究科　〒305–8573 茨城県つくば市天王台 1–1–1
†† 筑波大学計算科学研究センター　〒305–8573 茨城県つくば市天王台 1–1–1
E-mail: †s.nagaki@kde.cs.tsukuba.ac.jp, ††kitagawa@cs.tsukuba.ac.jp

あらまし　エンティティリンキングは自然言語文中の語句に対して，知識ベース中の対応するエンティティを紐付ける技術である．多くのエンティティリンキング手法は，(1) 文書中に含まれる各語句に対する候補エンティティ集合を抽出し，(2) エンティティ同士の関連度が強い組み合わせを検出することで実現される．しかしながら，作成時刻によって話題となるエンティティが変化する時系列文書では，ある語句に紐づく候補エンティティ集合が多くなり，エンティティリンキングの処理速度に大きな影響を与える．そこで，本研究ではエンティティリンキングの処理速度削減を目的とし，各語句に対する候補エンティティ集合の効率的な抽出手法を提案する．提案手法は，時系列文書における各エンティティの頻出度を考慮することで，精度を落とさずに効率的な候補エンティティの抽出を行う． 本稿では実データを用いた評価実験により，提案手法の有効性を示す．

キーワード　エンティティリンキング, 時系列文書データ，セマンティック Web

1. 序　　論

ウェブ上には大量の時系列文書データが存在し，様々な応用が可能となっている．時系列文書データの例には，ソーシャル・ネットワーキング・サービス（SNS）の投稿，オンラインニュース記事，検索システム内の検索ログがある．そのデータを利用した例として，Twitter (注1) に投稿されたツイートを利用した実世界で発生したイベントの抽出手法や，Twitter 内でのトレンド検出手法が考案されている [1–3].

一方，機械が自然言語文の意味を理解し，様々なアプリケーションに利用する時，いくつかの問題に対処する必要がある．大きな問題は以下の 2 点である．

- **類義語：**異なる語句が同じ意味を持つ．(例えば，2 つの語句「NY」と「the big apple」は，どちらもニューヨークを意味する.)
- **同形異義語：**同じ語句が異なる意味を持つ．(例えば，語句「Apple」は，果物のりんごと，Apple inc. の両方を意味する可能性がある.)

エンティティリンキングは，自然言語文に含まれる語句に対し，Wikipedia (注2) や DBpedia (注3) のような知識ベース中のエンティティを紐付ける技術である．機械は知識ベース中のエンティティを関連知識（例えば,「ニューヨーク」は「アメリカ合衆国」の都市であること）と共に解釈することが可能である．このように，エンティティリンキングは，自然言語文を機械解釈可能な形式に変換できる．エンティティリンキングは，その技術自体の発展と共に，Question Answering [4] や，ソーシャルメディア上のユーザプロファイリング [5,6]，実世界で起きたイベントの検出 [3] など，様々な技術に利用されている．

Shen ら [7] が論じているように，従来のエンティティリンキング手法は，以下の 3 ステップからなる．

（1） **Mention Detection**: 入力された自然言語文からエンティティを割り当てる語句の抽出.

（2） **Candidate Selection**: 各語句に対応づけるエンティティの候補集合を抽出.

（3） **Linking Decision**: 候補エンティティ集合から最適な組み合わせを抽出.

このようにして，与えられた自然言語文に含まれる語句に対して，知識ベース中のエンティティを割り当てる．仮に最適なエンティティが知識ベース中に存在しない場合，エンティティリンキングは unlinkable を意味する [[NIL]] という擬似的なエンティティを割り当てる．(なお，本論文において,「」でくくられた語を語句，[[]] でくくられた語をエンティティとする.)

Candidate Selection ステップで抽出される候補エンティティの数は，エンティティリンキングの処理速度に大きく影響を与える．これは，Linking Decision ステップにおいて，組み合わせ爆発が起きるためである．多くの手法は，各候補エンティティをノードとし，エンティティ間の関連度をエッジの重みに持つグラフを作成し，そのグラフを利用してリンキングを行う．これは,「ひとつの文章のトピックは一貫しており，同一文章内に含まれるエンティティ同士は関連が強い」という仮定にもとづいている．そのため，各語句に対する候補エンティティ数は，Linking Decision ステップでの計算量に影響し，結果的にエンティティリンキング全体の処理速度に大きな影響を与える．

それにもかかわらず，最新の手法も含めて，多くの手法は，候補エンティティの抽出手法についてほとんど議論しておらず，事前に作成した辞書を利用している [7]．この辞書は，Wikipedia などの，語句とエンティティの情報を包含したデータセットを

(注1)：https://twitter.com/
(注2)：https://www.wikipedia.org/
(注3)：http://wiki.dbpedia.org/

利用して作成されており，キーを各語句，各キーの値をキーの候補エンティティ集合のリストとする．また，辞書にはデータセット内の統計情報を基にした，各語句と候補エンティティ間の関連度を示すスコアが付帯している．候補エンティティ数がエンティティリンキングの処理速度に大きな影響を与えるため，多くの手法ではしきい値を利用して候補エンティティの選択を行っている．すなわち，各語句に対して，辞書内に含まれる候補エンティティ集合のうち，スコアが上位のエンティティのみを候補エンティティとして用いている．

しかしながら，ニュースやソーシャルメディアなどの時系列文書データ内の話題は，実世界での出来事に伴い**動的**に変化し，語句とエンティティ間の関連度も**動的**に変化する．例えば，我々が作成した辞書内のスコアによると「sparrow」という語句は，レコード会社の [[Sparrow Records]] や鳥の種である [[Sparrow]]，サッカー選手の [[Matt Sparrow]] を意味することが多い．ところが，[[Pirates of the Caribbean]] の映画が公開された直後では，語句「sparrow」とエンティティ[[Jack Sparrow]]（[[Pirates of the Caribbean]] の主人公）は，辞書内での関連度は低いにもかかわらず，実際の関連度が一時的に非常に強くなる．エンティティリンキングでの精度を向上するためには，このようなエンティティを候補エンティティに含めなければならず，候補エンティティ数が大きくなる．そのため，エンティティリンキングの処理速度が非常に遅くなる．

この問題に対処するため，本研究では時系列文書データに対するエンティティリンキングの処理速度改善を目的とし，効率的なエンティティリンキング手法を提案する．本手法の基本的なアイデアは以下の 2 点である．

- **語句とエンティティの効果的なスコアリング.** 時系列文書データは，実世界のイベントによって各語句に関係が強いエンティティが変化する．そのため，ある時刻における語句とエンティティの関連度（**Entity Recency**）を，直前に各エンティティが使用されている回数を用いて計算する．
- **動的スコアを考慮した効率的な候補エンティティの選択.** 効率的に候補エンティティ集合を抽出するために，**FULL-linking** と **LIGHT-linking** という 2 種類のタスクを組み合わせる．FULL-linking では，処理時間がかかる処理を行う．具体的には，動的なスコアを計算するために多くの候補エンティティを用いて正確にリンキングを行い，また，Entity Recency の管理を行う．LIGHT-linking では，FULL-linking で計算された Entity Recency を用いて候補エンティティ数を制限し，軽量にリンキングを行う．

提案手法の概要を図 1 に示す．提案手法は時系列文書データに対して，データを Producer-Consumer パターンで割り振り，並列に処理を行う．タスク FULL-linking に振り分けられたデータに対しては，多くの候補エンティティを用いて正確にリンキングを行う．また，タスク FULL-linking では，リンキングされた結果と文書の発行時刻を元に Entity Recency の管理を行う．タスク LIGHT-linking では，保存されたエンティティの使用時刻をもとに計算される語句に対するエンティティ

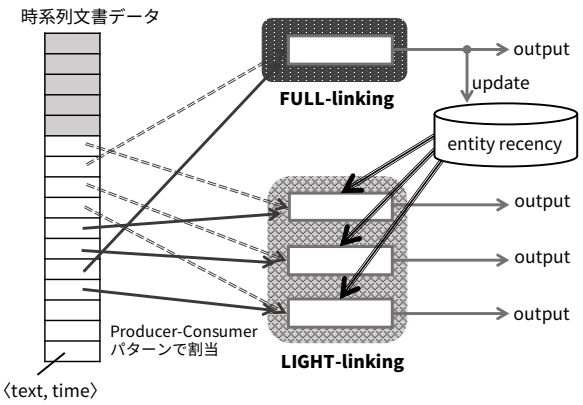


図 1: **提案フレームワーク.** 提案手法は時系列文書データに対して，データを Producer-Consumer パターンで割り振り，並列に処理を行う．タスク FULL-linking に振り分けられたデータに対しては，多くの候補エンティティを用いて正確にリンキングを行う．また，タスク FULL-linking では，リンキングされた結果と文書の発行時刻を元に Entity Recency の管理を行う．タスク LIGHT-linking では，保存されたエンティティの使用時刻をもとに計算される語句に対するエンティティの動的な関連度を用いて候補エンティティ数を制限し，効率的にリンキングを行う．

の動的な関連度を用いて候補エンティティ数を制限し，効率的にリンキングを行う．このようにすることで，精度を保ったまま，時系列文書データに対するエンティティリンキングの処理速度を改善することができる．

本研究の貢献は以下の通りである．

- **効率性.** 提案手法は，時系列文書データ内における語句とエンティティの動的な関連度を考慮し，効率的にある語句に紐づく候補エンティティ集合を抽出する．(3. 章)．
- **柔軟性.** 提案手法は，Linking Decision アルゴリズムによらず，強固に有効である．(4. 章)
- **実験による証明.** 複数の実データを用いた実験により，処理速度，精度の評価を行い，提案手法の有効性を示す．（4. 章）

評価実験での評価項目は，処理速度，精度，正解エンティティが候補エンティティに含まれているか，の 3 点である．

2. 問 題 定 義

本章では，本研究で用いる用語及び本研究で扱う問題を定義する．エンティティリンキングは，自然言語文に含まれる語句に対して，知識ベース中のエンティティを紐付ける技術である．ここで，エンティティ，及び語句を以下のように定義する．

定義 1：エンティティ
知識ベース中の各 URI. □

定義 2：語句
エンティティに紐づくことができる自然言語句. □

また，エンティティリンキングを以下のように定義する．

定義 3：エンティティリンキング

自然言語文 d に対して，d 内に含まれる語句を出現順に並べたリストを，$M(d) := \{m_i\}_{i=1}^{n(d)}$ とする．自然言語文 d と知識ベース KB が与えられた時，エンティティリンキングは，エンティティリスト $E(d) := \{(m_i, e(m_i))|m_i \in M(d)\},\ s.t.\ e(m) \in KB \cup \{NIL\}.$ を出力する．$e(m_i)$ は m_i に対応するエンティティ，もしくは NIL である．また，$n(d)$ は d に含まれる語句の個数を示す． □

本研究では，時系列文書データに対するエンティティリンキングに取り組む．時系列文書データは以下で定義される．

定義 4：時系列文書データ
各文書 d_i にタイムスタンプ t_i が付与されている文書集合であり，$TD := \{(t_i, d_i)\}_{i=0}^{N}$ で定義する． □

従って，本研究で取り組む問題を以下で定義する．

定義 5：時系列文書データに対するエンティティリンキング
時系列文書データ $TD = \{(t_j, d_j)\}_{j=0}^{N}$ に対して，d_j 内に含まれる語句を出現順に並べたリストを，$M(d_i) := \{m_i\}_{i=1}^{n(d)}$ とする．時系列文書データ TD と知識ベース KB が与えられた時，エンティティリンキングは，エンティティリスト $TE(TD)$ を出力する．この時，$TE(TD)$ は $TE(TD) := \{(t_j, E(d_j)\}_{j=1}^{N}, s.t.\ E(d_j) := \{(m_i, e(m_i))|m_i \in M(d_j)\}, s.t.\ e(m) \in KB \cup \{[[NIL]]\}.$ で定義される．$e(m_i)$ は m_i に対応するエンティティ，もしくは [[NIL]] である． □

第 1. 章で述べた通り，従来のエンティティリンキング手法は，(1)Mention Detection, (2)Candidate Selection, (3)Linking Decision の 3 ステップから構成される．自然言語文 d と知識ベース KB がエンティティリンキングの入力として与えられた時，各ステップの入力と出力を以下に述べる．

定義 6：Mention Detection
入力文 d に対して，エンティティが割り当てられる語句のリスト $M(d) = \{m_i\}_{i=0}^{n(d)}$ を出力する． □

定義 7：Candidate Selection
本ステップは，Mention Detection の出力 $M(d)$ に対して $CSL(M(d)) = \{CS(m) \mid m \in M(d)\}$ を出力する．ここで，$CS(m)$ は，各語句 $m \in M(d)$ に対応するエンティティの候補リストであり，$\forall e \in CS(m)\ s.t.\ e \in KB.$ を満たす． □

定義 8：Linking Decision
本ステップは，Candidate Selection で出力された CSL を用いて，入力文 d に対して，$E(d) := \{(m_i, e(m_i))|m_i \in M(d)\},\ s.t.\ e(m) \in KB \cup \{[[NIL]]\}.$ を出力する．$e(m_i)$ は m_i に紐づくエンティティ，もしくは [[NIL]] である． □

従来手法は，Candidate Selection ステップにおいて，事前に作成した辞書を利用した手法を採用している [7]．この辞書は，Wikipedia などの語句とエンティティの情報が含まれるデータセットを利用して作成されており，キーを各語句，各キーの値をキーの候補となるエンティティリストとする．また，辞書にはデータセット内の統計情報を基に，各語句と候補エンティティ間の関連度を示すスコアが付帯している．

定義 9：辞書
辞書 D のキーを語句 m_i とするとき，その値を $D(m_i) := \{(e_{ij}, p(e_{ij}|m_i))\}_{j=0}^{n_i}$ で定義する．ここで，e_{ij} は語句 m_i の候補エンティティである．$p(e_{ij}|m_i)$ は，辞書内の m に対する e のスコアである． □

本論文では，従来手法に従い，辞書が与えられた時の Candidate Selection を行う．この問題を以下で定義する．

定義 10：辞書が与えられたときの時系列文書データに対する Candidate Selection
Mention Detection ステップで出力された語句リスト $M(d)$ 及び辞書 D が与えられた時，$CSL'(M(d)|D) := \{CS(m) \mid m \in M(d)\},\ s.t.\ \forall e' \in CS(m),\ e' \in D(m).$ を出力する． □

本論文では，時系列文書データに対するエンティティリンキング (定義 5) における，辞書 が与えられたときの Candidate Selection(定義 10) において，各語句 m に対する候補エンティティ $CS(m)$ の大きさがなるべく小さくなるような，候補エンティティ選択手法を考案する．

3. 提 案 手 法

本章では，本研究の提案手法について述べる．第 3.1 節にて，提案手法の概要を述べる．第 3.2 節にて，語句とエンティティとのスコアリングについて述べ，第 3.3 節にて，提案フレームワークについて記す．

3.1 概　　要

本研究は，時系列文書データに対するエンティティリンキングの処理速度改善を目的として，データを Producer-Consumer パターンで割り振り，並列に処理を行う．また，割り振られたタスクによって，以下のように処理を行う．

（1） **FULL-linking:** 動的なスコアを計算するために多くの候補エンティティを用いて正確にリンキングを行い，また，Entity Recency の管理を行う．

（2） **LIGHT-linking:** FULL-linking で計算された Entity Recency を用いて候補エンティティ数を制限し，軽量にリンキングを行う．

3.2 語句とエンティティ間の動的な関連度

時系列文書データでは，各語句と関連度の強いエンティティが動的に変化する．本論文では，辞書内のスコアと Entity Recency [8] を組み合わせることによって，語句 m とエンティティ e の関連度を計算し，候補エンティティ集合の選択に用いる．

Entity Recency は，ある時刻における各語句と各エンティティの関連度を示し，滑り窓を用いて計算される．本論文では，滑り窓，及び Entity Recency を以下で定義する．

定義 11：滑り窓
滑り窓の時間幅 τ が与えられた時，時刻 $t - \tau$ から時刻 t までの区間を時刻 t における滑り窓とする． □

定義 12：Entity Recency

時刻 t における滑り窓を $\mathbb{T}$ とする．辞書 D が与えられた時，時刻 t における文章 d 内の語句 m に対するエンティティ e の Entity Recency は以下で計算される．

$$ER(e|m,\mathbb{T}) = \begin{cases} \frac{|\mathbb{D}_e^{\mathbb{T}}|}{\sum_{e_i \in D(m)} |\mathbb{D}_{e_i}^{\mathbb{T}}|} & (\sum_{e_i \in D(m)} |\mathbb{D}_{e_i}^{\mathbb{T}}| \neq 0) \\ p(e|m) & (otherwise) \end{cases},$$

ここで，$\mathbb{D}_e^{\mathbb{T}}$ は，エンティティ e に紐付けられた語句が存在する文章のうち，滑り窓 $\mathbb{T}$ 内に含まれる文章集合を示す．(この時，$\mathbb{D}_e^{\mathbb{T}}$ には，文章 d 自体は含まない．) $p(e|m)$ は辞書 D における語句 m とエンティティ e のスコアを示す． □

滑り窓 $\mathbb{T}$ 内の文章に対して，エンティティ e が紐付けられている回数が多いほど，ある語句に対する e の Entity Recency は 1 に近づく．

本論文では，Entity Recency を辞書のスコアに対して組み込むことによって，滑り窓 $\mathbb{T}$ における語句 m とエンティティ e の関連度を以下のように計算する．

$$\begin{aligned} score(e,m,\mathbb{T}) =& (1 - \frac{rank(e,m)}{size(D(m))}) \cdot (1-\lambda) \cdot p(e|m) \\ &+ \frac{rank(e,m)}{size(D(m))} \cdot \lambda \cdot ER(e|m,\mathbb{T}). \end{aligned} \quad (1)$$

ここで，$size(D(m))$ は辞書内における語句 m の候補エンティティ数を示し，$p(e|m)$ は辞書内における語句 m に対するエンティティ e のスコアを表す．また，$rank(e,m)$ は辞書内での語句 m の候補エンティティを，スコア $p(e|m)$ 降順に並び替えた時，エンティティ e の順位を示す．$rank(e,m)$ を考慮するのは，辞書内においてある程度上位にあるエンティティについては，候補エンティティとして妥当であると考えられるためである．λ は，$\sum_{e \in D(m)} |\mathbb{D}_e^{\mathbb{T}}|$ の値が小さい時に，$ER(e|m)$ の値が不当に大きくなるため，$ER(e|m)$ の重みを制御するパラメータである．λ は以下で計算される．

$$\lambda = \frac{\sum_{e \in D(m)} |\mathbb{D}_e^{\mathbb{T}}|}{|\mathbb{D}^{\mathbb{T}}| \cdot \alpha + \sum_{e \in D(m)} |\mathbb{D}_e^{\mathbb{T}}|} \quad (2)$$

ここで，α はハイパーパラメータである．このパラメータは，$|\mathbb{D}^{\mathbb{T}}|$ に対して，$\mathbb{D}_e^{\mathbb{T}}$ の数が多いほど大きくなる．

3.3 時系列文書に対するエンティティリンキング

本節では，時系列文書に対するエンティティリンキング手法について述べる．図 1 に示した通り，提案手法は入力データを Producer-Consumer パターンで割り振り，タスク並列で処理を行う．割り当てられたタスクによって，処理を変更し，効率的にリンキングを行う．

FULL-linking の処理を，アルゴリズム 1 に示す．まず，入力された文書 d_j 内に含まれる語句を抽出する (1 行目)．次に，抽出した語句リスト M_j 内に含まれる語句 m_i に対して，辞書内の候補エンティティ集合に対して，辞書のスコアが上位 K_1 件のエンティティを，リンキングに用いる候補エンティティとして抽出する（3–6 行目）．選択された候補エンティティ集合を用いて，各語句に対応するエンティティ集合を確定する（7 行目）．最後に，対応づけられたエンティティ集合と文書の作成時刻 t_j を用いて，Entity Recency データベースを更新する (8 行目)．Entity Recency データベースには，Entity Recency を計算するために，その時刻における滑り窓 $\mathbb{T}$ 内で各エンティティ e が使用された回数 $|\mathbb{D}_e^{\mathbb{T}}|$ が格納されている．

Algorithm 1: FULL-linking

Input: 文書データ (t_j, d_j), 辞書 D, パラメータ K_1, 滑り窓幅 τ
Output: エンティティリスト $E(dj) := (mi, e(mi))|mi \in M(dj)$
1 $M(d_j) \leftarrow MentionDetection(d_j)$
2 $CSL(M(d_j)) \leftarrow [\,]$
3 **for** $m_i \in M(d_j)$ **do**
4 $\quad CS(m_i) \leftarrow CandidatesSelection(m_i, t_j, D, K_1)$
5 $\quad CSL(M(d_j)).push(CS(m_i))$
6 **end**
7 $E(d_j) \leftarrow LinkingDecision(M_j, CSL(M(d_j)))$
8 $\mathbb{D} \leftarrow UpdateDB(E(d_j), t_j, \tau, \mathbb{D})$
9 **return** $E(d_j)$

Algorithm 2: LIGHT-linking

Input: 文書データ (t_j, d_j), 辞書 D, パラメータ K_2
Output: エンティティリスト $E(dj) := (mi, e(mi))|mi \in M(dj)$
1 $M(d_j) \leftarrow MentionDetection(d_j)$
2 $CSL(M(d_j)) \leftarrow [\,]$
3 **for** $m_i \in M(d_j)$ **do**
4 $\quad CS(m_i) \leftarrow CandidatesSelectionWithER(m_i, t_j, D, tau, \mathbb{D}, K_2)$
5 $\quad CSL(M(d_j)).push(CS(m_i))$
6 **end**
7 $E(d_j) \leftarrow LinkingDecision(M_j, CSL(M(d_j)))$
8 **return** $E(d_j)$

LIGHT-linking の処理を，アルゴリズム 2 に示す．アルゴリズム 1 と異なる点は，抽出した語句リスト M_j 内に含まれる語句 m_i に対して，候補エンティティ集合を抽出する（3–6 行目）手法，及びデータベースの更新が存在しない点である．本処理では，語句に対する候補エンティティ集合を，Entity Recency データベースを利用して抽出する．具体的には，ある語句 m_j に対する時刻 t_j でのタイムウィンドウ $\mathbb{T}_j$ について，辞書内の候補エンティティ集合内のエンティティ e に対して，式 1 を用いて $score(e,m,\mathbb{T}_j)$ を計算する．計算したスコアが上位 K_2 件のエンティティを，リンキングに用いる候補エンティティとして抽出する．

4. 評価実験

本章では，提案手法の有効性を示すために行った評価実験について記す．本実験の評価項目を以下に示す．

- **処理時間:** 提案手法で候補エンティティを選択した場合に処理時間はどのように変化するか．
- **精度:** 提案手法で候補エンティティを選択した場合に精度はどのように変化するか．
- **柔軟性:** 複数の Linking Decision に対して，提案手法は同様に有効性を示すか．

実験設定を 第 4.1 節に示す．また，実験結果を 4.2 節に示し，第 4.3 節に本実験の考察を述べる．

4.1 設　　定

本実験では，提案手法の有効性を示すために，複数の Linking Decision アルゴリズムと提案手法を組み合わせることで，処理時間及び精度がどのように変化するかを評価する．実験は全て Ubuntu 14.04LTS, Intel® Xeon ® E5-1620 v2 CPU, 128GB

表 1: 実験データセット

種類	データ数	期間	本文の単語数の平均値	語句数の平均値
ニュース	63,071 件	1 年間	- (注10)	4.62 語/記事
ツイート	997,079 件	20 日間	19.02 単語/ツイート	4.53 語/ツイート

表 2: ツイートデータに対する精度評価のために抽出した語句

washington	black	trump	moonlight
dakota	epn	bolt	patrick kennedy

RAM で構成された PC 上で行った．事前処理には，Python を使用し，ライブラリは gensim (注4), TextBlob (注5) を用いた．実験用プログラムは C++で記述し，gcc 4.9 を用いてコンパイルを行った．また，並列化には OpenMP (注6) を用いた．

4.1.1 データセット

本実験では，2 種類のデータセットを用いて実験を行った．それぞれの詳細を以下に示す．また，概要を表 1 に記す．

ニュース： New York Times (注7) が 2017 年に発行したニュース記事と，Wikipedia 内の Portal:Current events (注8) における 2017 年の出来事情報を組み合わせて作成したデータセットである．Portal:Current events については，各出来事の説明文を 1 つの文書とした．これらを日時で並び替えて，使用するデータセットとした．精度評価は Portal:Current events 内の，Wikipedia 記事へのリンクを利用した．

ツイート： Twitter API (GET statuses/sample) (注9) を用いて，2017 年 1 月 20 日から 2017 年 2 月 9 日に投稿された英語ツイートから，語句が 4 語以上となるツイートを抽出して作成したデータセットである．これは本手法は，各エンティティ間の最適な組み合わせを抽出することでエンティティリンキングが実現できると仮定しているためである．精度評価に用いる正解データは，データセット内に含まれる語句集合のうち，多く使用された語句上位 100 件から，ツイートが投稿された期間に起きたイベントに強く関連する 8 語句を抽出し，各語句が含まれるツイートをランダムに 30 ずつ抽出した．抽出されたツイートに対して，人手で関連するエンティティを対応づけたものを精度評価に用いた．抽出された語句一覧を表 2 に示す．

また，語句に対応づけるエンティティは，Wikipedia の記事とした．この記事データは，2016 年 10 月時点での Wikipedia ダンプデータ (注11) を利用した．本データセットに含まれている記事数は 4,923,385 件であった．

(注4)：https://radimrehurek.com/gensim/
(注5)：http://textblob.readthedocs.io/en/dev/
(注6)：www.openmp.org/
(注7)：https://www.nytimes.com/
(注8)：https://en.wikipedia.org/wiki/Portal:Current_events
(注9)：https://developer.twitter.com/en/docs/tweets/sample-realtime/overview/GET_statuse_sample
(注10)：NYTimes Archive API では，本文の情報が収集できないため，不明
(注11)：http://downloads.dbpedia.org/2016-10/core-i18n/en/pages_articles_en.xml.bz2

4.1.2 Mention Detection

本実験では，候補エンティティ集合の選択，及びそれによるリンキングの処理速度，精度の評価を目的とするため，語句抽出については単純な手法を利用した．

ニュースデータについては，データセット内の情報を基に語句抽出をした．具体的には，New York Times については API (注12) で記事を取得した時に，得られる各記事のキーワードを語句とした．また，Portal:Current events については，エンティティが紐付いているアンカーテキストを語句とした．

ツイートデータについては，Python ライブラリ Text Blob を用いて名詞句を抽出し，作成した辞書に存在する名詞句をエンティティを紐付ける語句とした．

4.1.3 Linking Decision

本実験では，異なる 3 種類の手法に提案手法を組み込み，提案手法の柔軟性を評価する．各手法の詳細を以下に示す．

Voting [9] ある語句と候補エンティティとのスコアを，他の語句に対応する各候補エンティティとの関連度を用いて計算する．各語句に対してその候補エンティティ集合内で最もスコアが高いエンティティを，その語句に紐づくエンティティとする．

Iterative Substitution (Itr-Sub) [10] 各語句と紐づくエンティティの組み合わせに対するスコアを定義し，各語句に対してひとつずつ組み合わせるエンティティを変化させていき，エンティティの組み合わせを評価する．これを繰り返すことにより，エンティティの組み合わせに対するスコアが最大になる組み合わせを求め，各語句に紐づくエンティティ集合とする．

Pair-Linking [11] ある語句があるエンティティに紐づく可能性を，他の語句とエンティティとの対応を基に推定する．候補エンティティ数が少ない語句から逐次的に紐づくエンティティを推定することで，精度を保ったまま効率的にリンキングを行う．

4.1.4 事 前 処 理

実験に際して必要となる事前処理として，エンティティ間の関連度計算，及び辞書の作成がある．

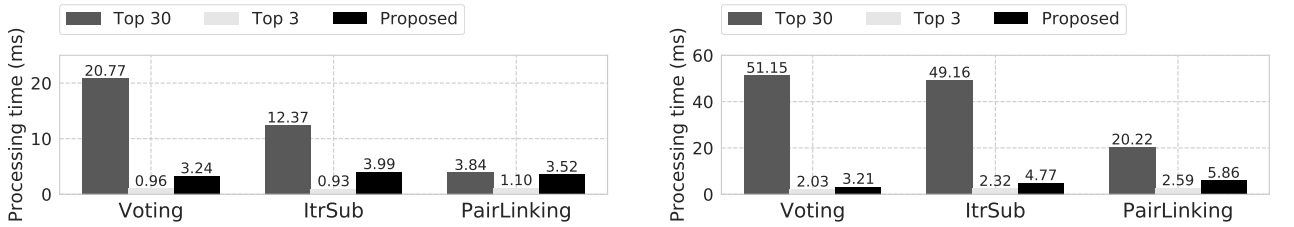
a） エンティティ間の関連度計算

本研究では近年多くの手法で採用されている Word embedding を利用した手法を用いる．本実験では，Phan ら [11] の手法によって，Wikipedia に現れる語句とエンティティをベクトル空間上に埋め込み，その空間上での cos 類似度をもって，エンティティ間の類似度とした．Wikipedia ダンプデータの情報抽出には，Annotated-WikiExtractor (注13) を利用した．また，本手法で使用される Word2vec [12] には，Python ライブラリ gensim を利用した．

b） 辞書の作成

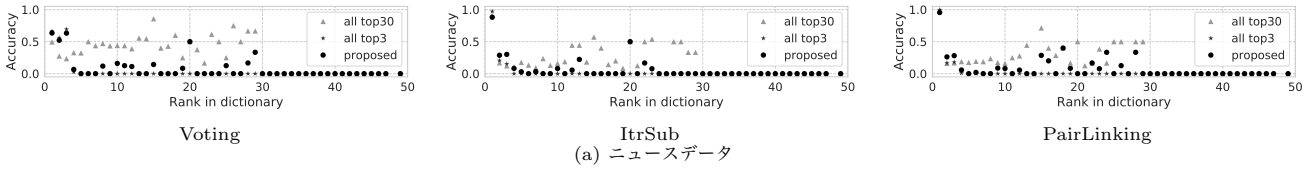
第 5.1 節で述べた通り，多くの手法では予め作成した辞書に基くスコアが利用されている．本実験では，Wikipedia 記事を利用した辞書を構築し，そのスコアをもって語句とエンティティ間の関連度とした．まず，Wikipedia 記事からタイトル，アンカーテキスト，曖昧さ回避ページ，リダイレクトの情報を

(注12)：https://developer.nytimes.com/archive_api.json#/README
(注13)：https://github.com/jodaiber/Annotated-WikiExtractor



(a) ニュースデータ　　(b) ツイートデータ

図 2: **各文書あたりの処理速度の平均値**．x 軸に各手法，y 軸に各文書あたりの処理速度の平均値を示す．提案手法は，全ての候補エンティティ数を 30 としたときよりも，大幅に処理速度が改善できることを示している．



(a) ニュースデータ

(b) ツイートデータ

図 3: **精度**．x 軸は辞書内の順位，y 軸は精度の平均値を示す．提案手法は，特にニュースデータに対して，全ての候補エンティティ数を 3 とした時よりも精度が改善できていることを示す．

抽出し，Shen ら [7] の手法によって，語句とエンティティのペア集合を作成する．次に，第 a) 節でエンティティをベクトル空間上に埋め込む際に，使用頻度が極端に低いために除いたエンティティを含むペアを除去する．作成されたペア集合を語句について集約することで，各語句とエンティティ間の関連度を計算する．作成された辞書内のキー数（語句数）は 8,805,246 語，エンティティ数は 2,126,042 エンティティであった．

4.1.5 パラメータ設定

提案手法では，複数のパラメータ（滑り窓の大きさ τ，式 2 内のパラメータ α，候補エンティティの数，使用するスレッド数）が存在する．滑り窓の大きさ τ については，Hua ら [8] と同様の 3 日間を採用した．パラメータ α については，0.005 を採用した．候補エンティティの数については，FULL-linking での候補エンティティ数 K_1 を 30 に，LIGHT-linking での候補エンティティ数 K_2 を 3 とした．また，FULL-linking と LIGHT-linking のいずれを行うかは，データが割り当てられたスレッドによって変更した．使用するスレッド数については，FULL-linking を行うスレッド数を 24，LIGHT-linking を行うスレッド数を 40 とした．

4.1.6 評 価 指 標

本評価実験では，エンティティリンキングの処理速度，精度の評価を行う．本実験で使用する評価指標は処理速度，精度，候補エンティティに正解が入っているか否か，の 3 点である．この時，文書 d に対するリンキング精度は以下の式を用いる．

$$acc_{doc}(d) := \frac{|\{m_i \in M(d) | e(m_i) == trueEntity\}|}{|M(d)|}$$

また，精度評価については，時系列文書における語句とエンティティの関連度の動的な変化を評価するため，辞書における正解データの順位に対して，どの程度の精度でリンキングできるかを評価する．

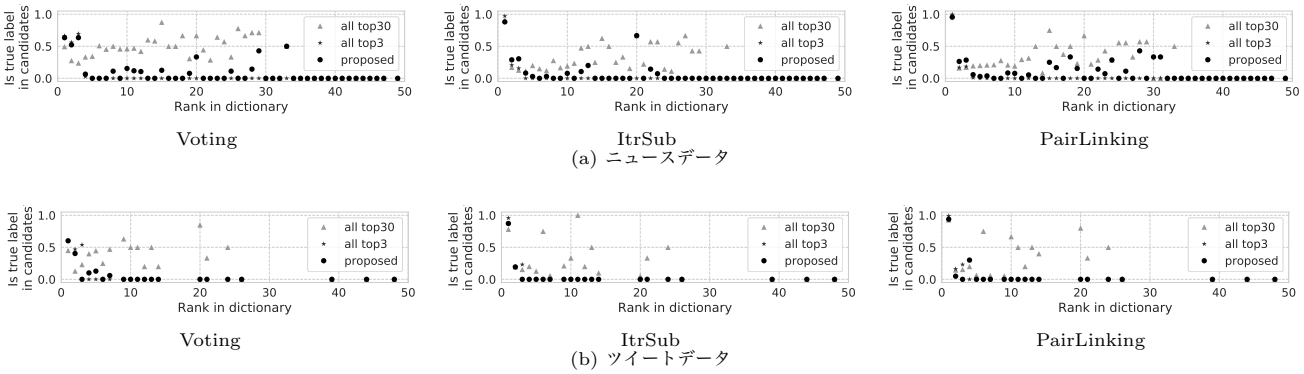
4.1.7 ベースライン

ベースラインとしては，辞書内のスコアをもちいた手法を用いる．具体的には，各語句の辞書における候補エンティティのうち，辞書内のスコアが上位 $K_1(=30)$ 件，及び $K_2(=3)$ のエンティティを候補とする手法をベースラインとする．

4.2 結　　果

処理時間の結果を図 2 に示す．x 軸に各手法，y 軸に各ツイートのリンキングにかかった時間を示す．提案手法を適用することで，全ての候補エンティティ数を 30 としたときよりも，処理速度が改善できることを示している．特にツイートデータに対しては，全ての文書内の語句が 4 語以上であり，組合せ爆発による処理速度の影響が強い．そのようなデータに対して，大幅に処理速度が改善できている．

精度の結果を図 3 に示す．x 軸に各手法，y 軸に精度を示す．提案手法を適用することで，精度が向上することがわかる．提案手法は，全ての候補エンティティ数を 3 としたときよりも，特に辞書の順位が低いエンティティについて，精度が改善できることを示している．ツイートデータについては，改善がほぼ見られなかったが，これは全文書に対して Entity Recency



(a) ニュースデータ

(b) ツイートデータ

図 4: **各文書あたりの正解データが入っているかの平均値**．x 軸に辞書内の順位，y 軸に各文書あたりの正解データが入っているかの平均値を示す．特にニュースデータに対して，提案手法は，全ての候補エンティティ数を 3 としたときよりも，候補エンティティ内に正解データを含めることができていることを示している．

を計算したデータ数が少なく，かつ，大量のツイートに対して Entity Recency を寄与させる割合（パラメータ α）が低すぎたのではないかと考えられる．

各文書あたりの正解データが入っているかの平均値を図 4 示す．x 軸に辞書内の順位，y 軸に各文書あたりの正解データが入っているかの平均値を示す．提案手法は，特にニュースデータに対して，全ての候補エンティティ数を 3 としたときよりも，特に辞書の順位が低いエンティティについて，候補エンティティ内に正解データを含めることができていることを示している．ツイートデータについての改善がほぼ見られなかったのは，精度と同様の理由と考えられる．

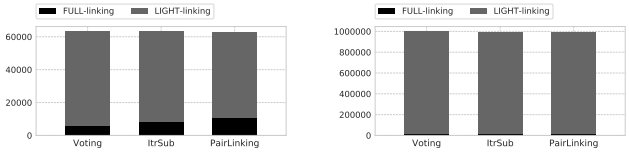
4.3 考　　察

提案手法は複数の Linking Decision アルゴリズムに対して，辞書内のスコアが低いエンティティに対しての精度をある程度保ちつつ，処理時間を削減することができた．ニュースデータとツイートデータの結果の違いより，提案手法は特に語句が多く組合せ爆発が処理時間に影響するデータセットに対して，特に強く効果的であると考えられる．

ツイートデータに対しては，静的に上位 3 件のデータを用いた時との精度向上が余り見られなかった．これは全文書に対して Entity Recency を計算した（FULL-linking に割り振られ r た）データ数が少なく，かつ，大量のツイートに対して Entity Recency を寄与させる割合（パラメータ α）が低すぎたのではないかと考えられる．FULL-linking と LIGHT-linking に割り当てられた文書数を図 5 に示す．ニュースデータに対して，ツイートデータは全文書データに対して FULL-linking に割り当てられた文書数の割合が低いことがわかる．それにより，Entity Recency を計算できずに，精度に影響を与えることができなかったためと考えられる．今後の研究では，FULL-linking と LIGHT-linking を行うスレッド数を調整し，データ，及び Linking Decision アルゴリズムとの関連を分析したい．また，このパラメータの最適値を自動推定する手法を考案したい．

5. 関 連 研 究

本節は本研究の関連研究を述べる．第 5.1 節で既存の候補エンティティ集合作成手法について述べ，第 5.2 節で時系列情報を用いたエンティティリンキングについて述べる．



(a) ニュースデータ　　(b) ツイートデータ

図 5: **各タスクに割り当てられた文書数**．x 軸に各手法，y 軸に各タスクに割り当てられた文書数を示す．ニュースデータに対して，ツイートデータは全文書データに対して FULL-linking に割り当てられた文書数の割合が低いことがわかる．

5.1 候補エンティティ集合作成

第 2. 章に述べたとおり，多くの手法では，Candidate Selection において事前に作成した辞書を用いている．典型的な手法は Wikipedia の各記事をエンティティとし，記事の種類（エンティティに関する記事，リダイレクトページ，曖昧さ解消ページ）及びアンカーテキストを利用して語句とエンティティのペアを抽出する [7,13]．抽出したペア集合を語句について集約することで，各語句 m に対するエンティティ e のスコア $p(e|m)$ を求める．また，Spitkovsky ら [14] は，様々な Web サイトの情報を利用することで，エンティティリンキングだけでなく，機械翻訳など応用技術に使える，多言語辞書を開発した (注14)．

エンティティリンキングの従来手法の多くは上記で作成された辞書を用いて，各語句に対してスコアが上位 K 件のエンティティを候補エンティティとして選択する [15,16]．また，語句とエンティティを同一空間上に埋め込むことで，語句と各エンティティの類似度を計算し，各語句に対するスコア上位のエンティティを候補とする手法も考案されている [17]．スコア上位のエンティティを抽出するのではなく，各語句に対するエンティティのスコアがある閾値より高いエンティティを候補エンティティにする手法も採用されている [9,18]．

本研究は，語句とエンティティの時間局所的な関連度を利用

(注14)：https://nlp.stanford.edu/data/crosswikis-data.tar.bz2/

した動的なスコアを定義し，与えられた時刻と語句に対して，効率的な候補エンティティ集合の選択を行う．我々の知る限り，本研究は，動的なスコアを用いて効率的な候補エンティティ集合を選択することで，エンティティリンキングの処理速度向上を達成した初めての研究である．

5.2 時系列情報を用いたエンティティリンキング

特に Twitter に投稿されたツイートに対して，時系列情報を利用することでエンティティリンキングの精度を改善する手法は複数考案されている．Hua ら [8] は，(1) 各ユーザの興味，(2) エンティティの直近に使用された頻度，そして (3) 知識ベース内におけるエンティティの使用頻度を利用することで，ツイートに対して高い精度でのエンティティリンキングを実現した．また，Fang ら [19] は，各ツイートに対して，時空間情報を含めた学習器を作成し，リンキングを行った．

本研究は，精度向上ではなく時系列情報を各語句の候補エンティティ抽出に使用し，エンティティリンキングの処理速度向上を目指す．この点において，本研究は上記の研究と異なる．

6. 結　　論

本研究は，時系列文書データを対象としたエンティティリンキングの処理速度削減を目的とし，各スレッドに文書を割り振り (1) 多くの候補エンティティを用いて正確にリンキングを行うスレッド，と (2)(1) で得た情報を用いて，候補エンティティ数を制限しリンキングを行うスレッド，を組み合わせることで，効率的にリンキングを行う手法を提案した（第 3. 節）．また，時系列文書データでは，語句とエンティティ間の関連度が動的に変化するが，そのような関連度を表現する指標を提案した（第 3.2 節）．実データを用いた評価実験において，提案手法が特に辞書内のスコアが低い時にリンキングでき，処理速度を改善できることを示した．また，提案手法を複数の Linking Decision と組み合わせることによって，提案手法が Linking Decision 手法によらず強固に有効であることを示した．

今後の課題としては，以下が挙げられる．

- **パラメータとデータセットの関連性分析：**本論文における実験では，滑り窓幅，Entity Recency を影響度合いの調整パラメータ α，及びスレッド数は固定して実験を行った．これらのパラメータはリンキング対象となるデータセットと関係が強いと考えられる．そのため，パラメータを変化させて実験を行い，データセットの特性と併せて分析することによって，さらなる高精度なスコアリング手法を提案できると考えられる．
- **新たなエンティティの抽出手法の検討：**Entity Recency を計算するために滑り窓をもとにした手法を採用したが，本手法はあらたに語句とエンティティの関連度が突然強くなるようなバーストに対応できない．しかしながら，このようなバーストに対する正確にリンキングを行うことは実世界での応用手法に必要である．そのため，バースト抽出などをもとにしたより高精度な動的関連度を提案したい．

謝　　辞

本研究の一部は，NICT 高度通信・放送研究開発委託研究「欧州との連携による公共ビッグデータの利活用基盤に関する研究開発」による．

文　　献

[1] M. Mathioudakis and N. Koudas. Twittermonitor: Trend detection over the twitter stream. In *SIGMOD '10*, pp. 1155–1158, 2010.

[2] C. Zhang, G. Zhou, Q. Yuan, H. Zhuang, Y. Zheng, L. Kaplan, S. Wang, and J. Han. Geoburst: Real-time local event detection in geo-tagged tweet streams. In *SIGIR '16*, pp. 513–522, 2016.

[3] A. Schulz, B. Schmidt, and T. Strufe. Small-scale incident detection based on microposts. In *HT '15*, pp. 3–12, 2015.

[4] F. Hasibi, K. Balog, and S. E. Bratsberg. Exploiting entity linking in queries for entity retrieval. In *ICTIR '16*, pp. 209–218, 2016.

[5] P. Kapanipathi, P. Jain, C. Venkatramani, and A. P. Sheth. User interests identification on twitter using a hierarchical knowledge base. In *ESWC' 14*, pp. 99–113, 2014.

[6] G. Li, J. Hu, J. Feng, and K. L. Tan. Effective location identification from microblogs. In *ICDE '14*, pp. 880–891, 2014.

[7] W. Shen, J. Wang, and J. Han. Entity linking with a knowledge base: Issues, techniques, and solutions. *IEEE Transactions on Knowledge and Data Engineering*, Vol. 27, No. 2, pp. 443–460, 2015.

[8] W. Hua, K. Zheng, and X. Zhou. Microblog entity linking with social temporal context. In *Proc. the 2015 ACM SIGMOD International Conference on Management of Data*, pp. 1761–1775, 2015.

[9] P. Ferragina and U. Scaiella. Fast and accurate annotation of short texts with wikipedia pages. *IEEE Software*, Vol. 1, No. 29, pp. 70–75, 2012.

[10] W. Shen, J. Wang, P. Luo, and M. Wang. LIEGE: : link entities in web lists with knowledge base. In *KDD' 12*, pp. 1424–1432, 2012.

[11] M. C. Phan, A. Sun, Y. Tay, J. Han, and C. Li. Neupl: Attention-based semantic matching and pair-linking for entity disambiguation. In *CIKM'17*, pp. 1667–1676, 2017.

[12] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems 26*, pp. 3111–3119, 2013.

[13] S. Cucerzan. Large-scale named entity disambiguation based on wikipedia data. In *EMNLP-CoNLL '07*, pp. 708–716, 2007.

[14] V. I Spitkovsky and A. X Chang. A cross-lingual dictionary for english wikipedia concepts. In *LREC '12*, pp. 3168–3175, 2012.

[15] Z. Guo and D. Barbosa. Robust entity linking via random walks. In *CIKM '14'*, pp. 499–508, 2014.

[16] E. O. Ganea, M. Ganea, A. Lucchi, C. Eickhoff, and T. Hofmann. Probabilistic bag-of-hyperlinks model for entity linking. In *WWW '16*, pp. 927–938, 2016.

[17] A. Pappu, R. Blanco, Y. Mehdad, A. Stent, and K. Thadani. Lightweight multilingual entity extraction and linking. In *WSDM '17*, pp. 365–374, 2017.

[18] F. Hasibi, K. Balog, and S. E. Bratsberg. Entity linking in queries: Tasks and evaluation. In *ICTIR '15*, pp. 171–180, 2015.

[19] Y. Fang and M. Chang. Entity linking on microblogs with spatial and temporal signals. *Transactions of the Association for Computational Linguistics*, Vol. 2, pp. 259–272, 2014.