エンティティ・リンキングにおける知識グラフ埋め込みを利用した候補生成

堀口 博 [1]　　新保 仁 [1,2]　　松本 裕治 [1,2]
[1] 奈良先端科学技術大学院大学 先端科学技術研究科，[2] 理化学研究所 AIP センター
{horiguchi.hiroshi.hf3, shimbo, matsu}@is.naist.jp

1 はじめに

近年, Wikipedia を対象としたリンキングモデル及び, Wikipedia 内に存在するメンションーエンティティ教師対を活用したエンティティ分散表現学習モデルが多く提案されている．知識ベース内の実体 (エンティティ) を指しうる文書内文字列範囲を以降メンションと表記する．既存のリンキングモデルにおいては Alias Table と呼ばれる候補辞書を用いて知識ベース全体のエンティティからリンクされうるエンティティの候補生成を行うことが主流であった．Alias Table は Wikipedia 内に存在するハイパーリンクより構成される．エンティティ・リンキングにおける Alias Table を用いた候補生成はメンション及びエンティティの表層形に大きく依存し，メンションとエンティティの表層形が大きく異る場合については，Alias Table に含まれる場合を除き考慮されていなかった．これを受け，Gillick らは表層形に依存しない候補探索手法として，エンティティ・リンキングに初めて近似近傍探索を導入した [1]. しかし，Alias Table や近似近傍探索を用いた Gillick らの候補生成ではどちらも大量のメンションーエンティティ教師対を必要とする．ドメインが限定された場合 Wikipedia 中のハイパーリンクのように大量のメンションーエンティティ教師対を獲得することは困難である．

本研究では，そのような大量の教師対を前提としない候補探索手法を提案する．実際に生物医学ドメインにおけるエンティティ・リンキングにおいて提案手法を適用し，候補集合内での正解エンティティ生成成功率及びリンキング精度が向上した．

2 リンキングにおける手続き

リンキングの手順として，1. 注目するメンション及び周辺文脈の表現を得る encoder から各メンションに対する表現を獲得し，2. 対象知識ベースからエンティティに対して候補生成を行い，3. 生成された候補内でのリンキングを行うことが通例である．提案モデルにおいてもこの手続を踏襲する．

本研究では対象メンションが含まれる 1 文にのみ注目する (local model)．リンキングの対象となるメンションを $m = w_{start}, ..., w_{end}$, 及びそれを含む 1 文を $s = w_1, w_2, ..., m, ..., w_{N-1}, w_N$ と表記する．それぞれの encoder の入出力について以下に具体的に記す．

(1) メンションのエンコーディング Mention Encoder は Left/Right Context Encoder 及び Mention/Sentence Encoder の 4 つから構成される．

Left/Right Context Encoder Left Context Encoder 及び Right Context Encoder にはどちらも双方向 LSTM

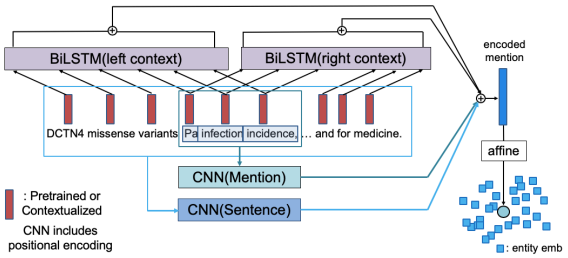

図 1　Mention Encoder の概要

を用い, それぞれ $(w_1, w_2, ..., m), (m, ...w_{N-1}, w_N)$ を入力として受け取る．ここで N は対象メンションを含む 1 文に含まれる語数である．ここで,LSTM は $\boldsymbol{h}_i, \boldsymbol{s}_i = \mathrm{LSTM}(\boldsymbol{u}_i, \boldsymbol{h}_{i-1}, \boldsymbol{s}_{i-1}), \boldsymbol{u}_i \in \mathbb{R}^{d_w}$ であり，$\boldsymbol{u}_i$ は word embedding，d_w は word embedding の次元数を表す．また，$\boldsymbol{h}_{i-1}, \boldsymbol{s}_{i-1} \in \mathbb{R}^l$ は LSTM の一つ前のセルからの出力および，注目するセルの状態をそれぞれ表す．以下では Left Context Encoder について詳細を示す．LSTM は文の左から右への forward 方向及び，右から左への backward 方向をそれぞれ考慮する．forward 方向の LSTM より $\boldsymbol{h}_m$，backward 方向の LSTM より $\boldsymbol{h}_1$ を連結した $\boldsymbol{v}_{left} = [\boldsymbol{h}_1; \boldsymbol{h}_m]$ を Left Context Encoder の出力とする．ここで，; はベクトル同士の連結操作を表す．同様に Right Context Encoder から $\boldsymbol{v}_{right} = [\boldsymbol{h}_m; \boldsymbol{h}_N]$ が出力される．

Mention/Sentence Encoder Mention Encoder 及び Sentence Encoder にはどちらも CNN を使用する．それぞれの出力を $\boldsymbol{v}_{ment}, \boldsymbol{v}_{sent}$ と表記する．

知識グラフ埋め込みを用いたエンティティの候補探索器の学習には $\boldsymbol{v}_{left}, \boldsymbol{v}_{right}, \boldsymbol{v}_{ment}, \boldsymbol{v}_{sent}$ を，リンキングモデルの学習時には $\boldsymbol{v}_{left}, \boldsymbol{v}_{right}$ を連結したものを，メンション周辺を表現するベクトル v とする．

$$\boldsymbol{v} = [\boldsymbol{v}_{left}; \boldsymbol{v}_{right}; \boldsymbol{v}_{ment}; \boldsymbol{v}_{sent}] \text{ or } [\boldsymbol{v}_{left}; \boldsymbol{v}_{right}] \quad (1)$$

(2) 候補エンティティの生成　既存の Wikipedia へのリンキングを行うモデルでは，Alias Table を構成するために Wikipedia 内に存在するハイパーリンクを用いている．ハイパーリンクにはそれぞれ Wikipedia が保有する entity の URL がアノテーションされており，このアノテーションを用いて文字列ーエンティティ対を大量に獲得し Alias Table を充実させることが可能である．本研究で行う生物医学ドメインにおいて，対象知識ベースはハイパーリンク機構を保有しない．また，既存の辞書から同名辞書を人手で構成してはいるものの，Wikipedia の Alias Table と比較した場合小規模である．このような場合，メンションやエンティティ自身の文字列がこれまでエンティティの候補生成に使用

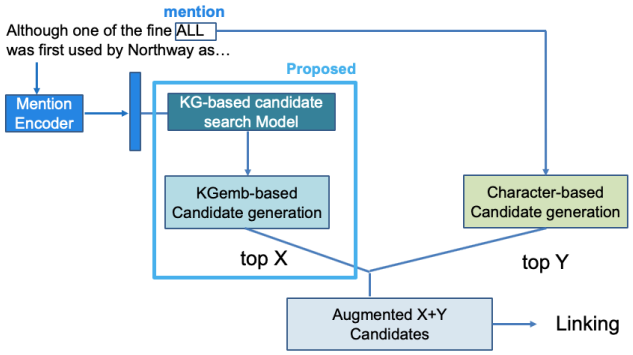

図 2　提案手法

Mention in document: "**ALL**"	Abbreviation
Generated Candidates: "All Sites", "All of the Time", "Alleviation"	
Gold entity: "**Acute lymphocytic leukemia**"	
Mention in document: "**rs988712**"	Broader concept annotation
Generated Candidates: "RB 88712", "MK 8712", "R-268712"	
Gold entity: "**Gene Mutant**"	
Mention in document: "**Giα**"	Common name(通称) mention
Generated Candidates: "Gin", "Gibraltar", "Gill structure"	
Gold entity: "**GTP-Binding Protein alpha Subunit, Gi**"	

図 3　正解エンティティの候補生成に失敗するメンションの例

されている [2, 3]．本研究では Murty らの研究に倣いメンション及びエンティティの n-gram 特徴量をエンティティ候補生成のベースラインとし，提案手法では知識グラフ埋め込みを用いた候補生成により，表層形ベースの候補生成に対する補完を行う．

提案手法により生成されるメンション m に対するトップ X 個のエンティティ候補集合を U，ベースラインとなるメンション及びエンティティの表層形を用いて生成されるトップ Y 個の entity 候補集合を V とする．ここで X, Y は各集合においてトップ何個を考慮するかを表すパラメータである．$X + Y$ を固定しつつ，$X = 0$ の場合をベースライン，X を増加させた場合を提案手法とし比較を行う．

(3) リンキング　(2) の手続きを経て得られた，各 m に対する候補集合 $E_m = U \cup V$ から 1 つのエンティティへのリンキングを行う．

3　提案手法

知識グラフ埋め込みを用いた，リンキングタスクにおける候補エンティティの生成及び表層形候補の補完について説明する．図 2 に提案手法の概要を示す．

ドメインが限定されたり，十分量のメンション－エンティティペアの存在が保証されない場合，表層形を用いた候補生成が行われることを 2 章で述べた．しかし，表層形特徴量のみを用いてメンションに対する候補生成を行う場合，正解エンティティが候補エンティティ集合内に含まれないケースが存在する．その一例を図 3 に示す．代表的なエラーケースとしてメンションが略語や出現文書内における通称であったり，メンションに対して上位概念がアノテーションされている場合が存在する．これらのケースに対処するためには，表層形に依存しない候補生成手法が必要である．

本研究では知識グラフ埋め込みをエンティティの特徴量として用いることで，表層形に依存しないエンティティの候補生成及び表層形を用いた候補の補完を目的とする．従来のリンキングモデルは大量のメンション－エンティティ教師対の存在を前提とする [1, 4] が，ドメインが限定された場合に大量のメンション－エンティティ教師対を獲得することは困難である．ドメインが限定された場合でも，構造化データのみ存在すればエンティティに特徴量を付与できる点が，提案手法のメリットである．

本研究では ComplEx[5] を用いて事前にエンティティに対して埋め込みを付与する．エンティティ e に付与された埋め込みを以降 $\boldsymbol{g}_e$ と表記する．これを用いてまず各 m に対する候補生成を行う．

図 1 に概要を示す．(1) 式により得られたメンションベクトル $\boldsymbol{v}$ を知識グラフ埋め込み空間に写像する．

$$\boldsymbol{v}_{proj} = \boldsymbol{W}\boldsymbol{v} + \boldsymbol{b} \tag{2}$$

ここで写像 $\boldsymbol{W}$ 及びバイアス $\boldsymbol{b}$ は，訓練時に学習される．$\boldsymbol{v}_{proj}$ を用いて，知識ベース内に存在する全エンティティを対象に近傍探索を行う．知識ベース内のエンティティ数のオーダーは一般に $10^7 \sim 10^8$ であり，探索時間の短縮が必要とされる．本論文では，探索時間の短縮に近似近傍探索ライブラリ faiss[6] を使用する．

各メンションに対して，近似近傍探索を用いメンションに対する候補生成を行う．$\boldsymbol{v}_{proj}$ を用いて知識ベースに含まれる e の集合 $\mathcal{E}$ 内の探索を行う．コサイン距離

$$c_{e_i} = \frac{\boldsymbol{v}_{proj} \cdot \boldsymbol{g}_{e_i}}{||\boldsymbol{v}_{proj}||\,||\boldsymbol{g}_{e_i}||} \tag{3}$$

の小さい順に候補生成を行いトップ 10,000 エンティティを候補集合とする．その後，ユークリッド距離

$$l_{e_i} = ||\boldsymbol{v}_{proj} - \boldsymbol{g}_{e_i}|| \tag{4}$$

を用いて 10,000 エンティティをリランキングしソートしたものを U とし，これを用いて V に含まれない候補の生成を狙う．

4　実験

4.1　リンキング用データセット

リンキングを行うデータセットは生物医学ドメインの MedMentions st21pv[7] を使用し，リンク先知識ベースには UMLS[8] 2017AA を用いる．

	train	dev	test
メンション数	122,241	40,884	40,157
メンションの固有エンティティ集合	18,250	8,643	8,457
訓練データと重複するエンティティ数		4,984	4,867

表 1　MedMentions st21pv データセットの概要

前処理の結果，UMLS からは 2,578,551 エンティティ及び 42,509,331 triplet が得られた．この前処理によって得られたエンティティの集合を $\mathcal{E}$ とする．得られた triplet を 90%，5%，5% に分割したものをそれぞれ知識グラフ埋め込み訓練用 train，dev，test データとし train データを用いて ComplEx による知識グラフ埋め込みの学習を埋め込み次元を 300 として行った．得られた知識グラフ埋め込みを用いて test データで Link Prediction を行った結果，MRR は 0.76 であった．

4.2 複素埋め込みを用いたエンティティ候補生成

4.2.1 学習

Mention Encoder と $\boldsymbol{W}$ が学習される過程で，1 つのバッチに対してその都度学習が行われる．本論文における実験では，1 つのバッチに対して学習時に負例サンプリングも併せて行う．バッチ内の 1 メンション に対して，Mention Encoder 及び $\boldsymbol{W}$ により知識グラフ埋め込み空間にメンションは写像され，写像されたメンションのコサイン近傍 (式 3) から負例をサンプリングする．

損失関数にはクロスエントロピー関数と，メンション－正解エンティティ間距離の二乗誤差を足し合わせたものを使用する．あるエンティティが正解である確率 $p(e_i|m)$ 及び損失関数 $\mathcal{L}$ はそれぞれ

$$p(e_i|m) = \frac{\exp(-l_{e_i})}{\sum_i^N \exp(-l_{e_i})} \tag{5}$$

$$\mathcal{L} = -\sum_i^{X_{train}} y_i \log p(e_i|m) + l_{e_{gold}}^2 \tag{6}$$

となる．ただし、y_i は正解エンティティに対して 1，それ以外には -1 を割り当てる．また X_{train} は訓練時に獲得する負例と正解エンティティの総数を表す．

学習の最適化関数には AdamW[9] を用いた．バッチサイズを 64，エポックを 30，学習率を 0.001 とし，訓練時の負例サンプリング数 X_{train} は 500 とした．

4.2.2 実験結果

知識グラフ埋め込みのみを用いたエンティティ候補生成の結果を表 2 に示す．評価は，リランキングの結果正解エンティティを何位より上位に位置づけたかを表す Hits で行った．

Hits@1	@10	@100	@1000	@10000
33.83	35.04	36.39	49.71	71.97

表 2　知識グラフ埋め込みを用いた正解エンティティの Hits 評価

ここで学習された，知識グラフ埋め込みを用いたエンティティの候補生成器を test データ内メンションに対して適用し，式 4 によりソートされた候補を保存する．

4.3 リンキング

4.3.1 学習

知識グラフ埋め込みを用いた候補生成器とは別に新しくリンキング用の Mention Encoder を学習させる．学習は，$\boldsymbol{v}$ と候補エンティティが持つテキスト情報の類似度 q_i をベースに行われる．ここで $q_i \in [0, 1]$ であり，損失関数は q_i を用いて

$$\mathcal{L} = -\sum_i^{Y_{train}} z_i \log q_i \tag{7}$$

と表される．ここで Y_{train} はリンキングモデル学習時における，後述するメンション及びエンティティの表層形ベースで生成された負例の数であり，z_i は正解エンティティに対して 1 を，そうでないならば -1 を割り当てる．

以下，q_i の算出法について説明する．エンティティは自身を指す名称 (Canonical Name) 及びエンティティの定義・説明文 (Definition) を持ち，これを Canonical Names，Definition の順に繋げた系列を以下では t_{e_i} と表記する．t_{e_i} を入力とする LSTM を用意する．2 章 (1) での Left/Right Context Encoder で説明した場合と同様の手続きで $\boldsymbol{v}_{left}, \boldsymbol{v}_{right}$ を得，これを連結した $\boldsymbol{v}_{e_i} = [\boldsymbol{v}_{left}; \boldsymbol{v}_{right}]$ をエンティティが持つテキストを表すベクトルとする．線形写像 $\boldsymbol{J}$ を用いてメンションとエンティティが持つコサイン類似度を算出する．メンションから得られた $\boldsymbol{v}$ を線形写像したベクトルを $\hat{\boldsymbol{v}}$ として

$$\hat{\boldsymbol{v}} = \boldsymbol{J}\boldsymbol{v} + \hat{\boldsymbol{b}} \tag{8}$$

$$q_i = \frac{\hat{\boldsymbol{v}} \cdot \boldsymbol{v}_{e_i}}{||\hat{\boldsymbol{v}}||\,||\boldsymbol{v}_{e_i}||} \tag{9}$$

が、候補エンティティに対するテキスト類似度スコアとなる．ここで $\hat{\boldsymbol{b}}$ はバイアスであり，$\boldsymbol{J}$ と共に訓練時に学習される．学習の Optimizer には AdamW を用いた．バッチサイズを 64，エポックを 10，学習率を 0.001 とし，訓練時における負例サンプリング数 Y_{train} は 13 とした．

4.3.2 表層形を用いた候補生成

ドメインが限定され充実した Alias Table の作成が困難な場合，メンションおよびエンティティの表層形を用いた候補辞書作成が行われる．本研究では Murty ら [2] の方法を元に，メンション及びエンティティの文字 n-gram($n = 3$) 特徴量を使用する．UMLS に含まれるエンティティが持つ Canonical Name から文字 n-gram 特徴の集合・出現頻度の統計及び tf-idf 値を得る．3-character-gram 特徴ごとの tf-idf 値を用いて，各メンション及びエンティティを bag-of-3-character-gram によりベクトルとし正規化した後，コサイン類似度の高い順にエンティティの候補を獲得する．またこの時得られたコサイン類似度を [0,1] 区間に線形写像したものを p_i とする．UMLS は同じエンティティを指しうる文字列情報も保有している．以下ではこの情報を同名辞書と呼ぶ．同名辞書が候補検索対象であるメンションの文字列を含んでいる場合，文字列の表層形コサイン類似度を無視し p_i を 1.0 に設定する．

4.4 実験結果

4.4.1 $X+Y$ 固定時における正解エンティティ生成率

$X + Y = 50$ と固定させたまま，知識グラフ埋め込みの候補数 X を増加させた場合の，test データにおける正解エンティティ生成に成功したメンションの割合を確かめた．結果を図 4 に示す．

図 4 から，知識グラフ埋め込みを用いた候補数を増加させた場合，最初の $X = 5, Y = 45$ においてのみ正解エンティティ生成率の増加が見られるが，それ以降は減少傾向が見られる．実際には $X = 1, Y = 49$ とした場合に最も高い正解エンティティ生成成功率が観測された．この傾向は総候補数 $X + Y$ を変化させた場合も変わらなかった．以降では $X + Y$ を種々の値に固定した場合について，$X = 0$ をベースライン，$X = 1$ を提案手法とし比較を行う．dev データを用いた実験の結果，$p_i \geq 0.90$ なるエンティティ e_i を候補集合 E_m に含む場合，知識グラフ埋め込みを用いた候補を追加

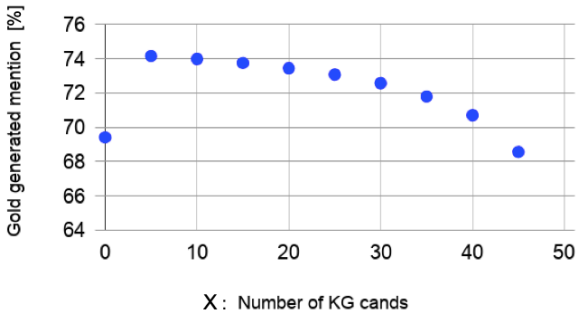

図 4　X+Y=50(固定) において X を変化させた場合の, 正解エンティティの生成に成功したメンションの割合

しない場合の方がリンキング正解率が高かった．この結果を受け，test データに対しても，E_m に含まれる候補エンティティ e_i の表層形類似度 p_i が全て 0.9 より小さい場合にのみ，知識グラフ埋め込みを用いたエンティティ候補の追加を行った．

4.4.2　リンキングの結果

候補エンティティに対する評価スコアを $p_i + q_i$ とし test データ内メンションについて $X+Y = 15, 30, 60$ の場合に評価を行った．Gold Recall は, 指定された X, Y を用いた場合に正解エンティティの候補生成に成功したテストデータ内メンションの割合を示す. Hits@1 の場合はリンキング正解率を表している．p_i のみを用いた Hits@1 は 45.74% であった．

	Hits@1	@5	@10	Gold Recall
Baseline ($X = 0$)	52.53	64.99	66.53	67.40
Proposed($X = 1$)	**54.14**	68.15	69.81	70.74

表 3　$X + Y = 15$ におけるリンキングの結果

	Hits@1	@5	@10	Gold Recall
Baseline ($X = 0$)	52.44	65.57	67.41	69.48
Proposed($X = 1$)	53.68	**68.44**	70.44	72.61

表 4　$X + Y = 30$ におけるリンキングの結果

	Hits@1	@5	@10	Gold Recall
Baseline ($X = 0$)	52.54	65.97	68.15	71.38
Proposed($X = 1$)	53.19	68.40	**70.89**	**74.31**

表 5　$X + Y = 60$ におけるリンキングの結果

5　考察

知識グラフ埋め込みのみを用いた近似近傍探索による候補生成では，エンティティは知識グラフ埋め込み以外の情報を保持していないにも関わらず約 3 割のメンションがリンキングに成功した．このことから，知識グラフ埋め込みはドメイン知識の一部を保有し，リンキングにも有用であることが示唆される．一方で，表 2 において Hits@10 ∼ Hits@100 に注目した場合，正解エンティティの生成成功率に伸びは見られない．かつ 2 からも分かる様に，上位 1 位近傍以外のエンティティはリンキングにおいてノイズとなってしまっている．このことから, 特定の type に属するエンティティや知識ベースにおいて特定の階層に属するエンティティのみのリンキングが行われている，あるいはエンティティ埋め込みの分布に問題が存在すると想定し，テストデータ内メンションを知識グラフ埋め込み空間に写像した場合のトップ 100 の近傍エンティティについて統計を取った．そのヒストグラムを図 5 に示す．ヒストグラムは裾野が長く，特定のエンティティの近傍にメンションが写像されるというような事態は発生していないと考えられる．また同時に，メンションの写像及び知識グラフ埋め込みの分布についてもこの分析からは問題を発見することは出来なかった．より詳細な知識グラフ埋め込みの分布及びエラーの分析が必要とされる．

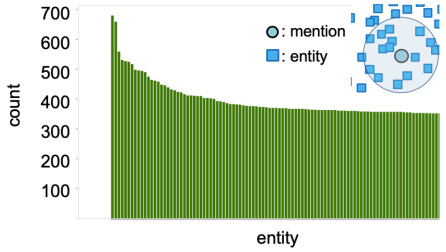

図 5　メンション近傍におけるエンティティ分布

6　結論

エンティティ・リンキングにおける候補生成手法として知識グラフ埋め込みを用いた，表層形エンティティ候補の補完を提案し効果を確かめた．知識グラフ埋め込みを用いたエンティティの近似近傍探索を行うことで，表層形候補の補完およびリンキング精度の向上に成功した．一方，知識グラフ埋め込みを用いた場合でも，候補生成に失敗するメンションも依然として多く観測された．今後の展開としては，大きく 2 つ挙げられる．UMLS や Wikidata をはじめとして，triplet を保有するがメンション－エンティティ教師対が少ない状況で，テキスト情報を活かしつつ知識ベース内で学習が完了するようなエンティティ分散表現の獲得が課題の一つである．2 つ目は本モデルの文書全体への拡張が挙げられる．本論文のモデルでは各メンションごとに候補生成を行うが，注目するメンションが含まれる文書が保有するメンション集合を用いた知識グラフ埋め込み候補生成を用いることで，正解エンティティの候補生成率の改善が期待される．

参考文献

[1] Daniel Gillick, Sayali Kulkarni, Larry Lansing, Alessandro Presta, Jason Baldridge, Eugene Ie, and Diego Garcia-Olano. Learning dense representations for entity retrieval. *arXiv preprint arXiv:1909.10506*, 2019.

[2] Shikhar Murty, Patrick Verga, Luke Vilnis, Irena Radovanovic, and Andrew McCallum. Hierarchical losses and new resources for fine-grained entity typing and linking. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 97–109, 2018.

[3] Lajanugen Logeswaran, Ming-Wei Chang, Kenton Lee, Kristina Toutanova, Jacob Devlin, and Honglak Lee. Zero-shot entity linking by reading entity descriptions. *arXiv preprint arXiv:1906.07348*, 2019.

[4] Matthew E Peters, Mark Neumann, IV Logan, L Robert, Roy Schwartz, Vidur Joshi, Sameer Singh, and Noah A Smith. Knowledge enhanced contextual word representations. *arXiv preprint arXiv:1909.04164*, 2019.

[5] Théo Trouillon, Johannes Welbl, Sebastian Riedel, Éric Gaussier, and Guillaume Bouchard. Complex embeddings for simple link prediction. In *International Conference on Machine Learning*, pages 2071–2080, 2016.

[6] Jeff Johnson, Matthijs Douze, and Hervé Jégou. Billion-scale similarity search with gpus. *IEEE Transactions on Big Data*, 2019.

[7] Sunil Mohan and Donghui Li. Medmentions: A large biomedical corpus annotated with umls concepts. *arXiv preprint arXiv:1902.09476*, 2019.

[8] Peri L Schuyler, William T Hole, Mark S Tuttle, and David D Sherertz. The umls metathesaurus: representing different views of biomedical concepts. *Bulletin of the Medical Library Association*, 81(2):217, 1993.

[9] Ilya Loshchilov and Frank Hutter. Fixing weight decay regularization in adam. *CoRR*, abs/1711.05101, 2017.