大規模な単語活用辞書を用いた英単語の見出し語化

駒井　雅之† 　進藤　裕之† 　松本　裕治†

† 奈良先端科学技術大学院大学　〒 630–0192 奈良県生駒市高山町 8916-5
E-mail: †{komai.masayuki.jy4,shindo,matsu}@is.naist.jp

あらまし　自然言語処理において, 単語の活用を認識し, 見出し語化 (lemmatization) することは重要なタスクである. しかしながら, 計算機で処理可能な英単語の活用辞書はあまり整備されておらず, 従来研究では様々なドメインのテキストに対する活用辞書の網羅率や, 見出し語化の精度を十分に評価できていない. そこで本研究では, 誰でもフリーで利用可能な言語資源から英単語の活用表を網羅的に収集し, 見出し語化の性能評価を行う. また, 未知語に対しては大規模な Web テキストとニューラルネットワークを用いて, 高精度に見出し語を推定可能なモデルを提案する.

キーワード　自然言語処理, 見出し語化, lemmatization

1. はじめに

見出し語化 (lemmatization) とは, テキスト中に現れる単語の原形を推定する自然言語処理タスクの 1 つであり, 高精度な構文解析や機械翻訳などを実現するために重要なステップである. 英語を例にとると, テキスト中に現れる動詞は, 過去形, 過去分詞形, 現在完了形, 過去完了形など様々な活用を伴うため, これらの活用を正しく認識し, 原形＋活用の種類という形式で正規化する必要がある.

一般に, 名詞や動詞などの単語は, 活用や格変化を伴ってテキストに出現する. これらの単語を見出し語化するためには, まず第一に網羅性の高い大規模な単語活用辞書が必要となる. 単語活用辞書とは, 単語ごとに, 表層形とその原形, 活用の種類や品詞とを対応付けた辞書である. このような活用辞書は自然言語処理において極めて重要な言語資源であることは容易に想像できるが, 誰でもフリーで利用可能, かつ計算機で処理することができる (machine-readable) 網羅的な活用辞書は, 現在でもあまり整備されていないのが現状である. また, たとえ大規模な単語活用辞書が存在したとしても, 表層形に対して複数の見出し語の可能性が存在する場合や, 未知語に対して正しく原形を推定するためには, 辞書引きだけでは不十分で, 統計的な手法を援用する必要がある.

自然言語処理において最も研究が盛んな言語の 1 つである英語では, 単語の活用推定は, 品詞タグ付けタスクの一部として扱われてきた. 例えば, 英語の標準的なテキストコーパスである Penn Treebank [4] では, 原形の動詞は ”VB”, 過去形は”VBD” といった細分化された品詞タグが定義されており, 品詞タグを推定することは, 同時に活用の種類も推定することとなる. そのため, 英語のテキスト解析に関する従来研究の多くは, 見出し語を直接推定することは行わずに, 品詞タグ付けのみで形態的解析を完了させ, 構文解析や意味解析などのより高次な解析へ進むことが一般的である. 一方, アラビア語やヘブライ語などの形態的に豊かな言語 (morphologically-rich language) では, 単語の活用が豊富で語彙数も膨大になることから, 一定規模の単語活用辞書が整備されている場合があり, 英語とは状況が異なる.

以上の背景から, 本研究では, 英単語を高精度に見出し語化する研究に取り組む. 先に述べたように, 高精度な見出し語化のためには, 大規模な単語活用辞書が必要不可欠である. そこで, 本研究では, まずはじめに, 誰もがフリーで利用できる Wiktionary から単語活用情報を網羅的に収集し, machine-readable な見出し語辞書として公開する. また, それらの活用辞書を用いて, いくつかのテキストコーパスにおける見出し語辞書の網羅性や見出し語化の精度を評価する. さらに, 見出し語の曖昧性や未知語の問題に対処するために, 活用情報の付与されていない大規模なテキストとニューラルネットワークを用いて, 見出し語化の精度を向上させる統計的手法を提案する. 本研究の見出し語化ソフトウェアは, オープンソースソフトウェアとして公開する予定である.

2. 関連研究

本研究の関連研究として, 見出し語化を行う手法に関する研究と, 英語の単語活用辞書の構築に関する研究とが挙げられる.

Chrupala ら [1] は, 形態タグ付けを行う最大エントロピーモデルと, 見出し語化を行う最大エントロピーモデルとを別々に構築し, それらを統合して最適な形態タグと見出し語の組み合わせを探索する手法を提案している. ただし, 実験で用いているデータは, ロマンス語, スペイン語, ポーランド語の 3 種類であり, 英語の実験は行われていない. ロマンス語やスペイン語では, 10 万トークン規模の見出し語付きテキストデータが存在するため, これらの言語を用いた評価実験を行っていると考えられる.

Toutanoba ら [9] は, 文字レベルのトランスデューサを用いて, 単語の表層形から品詞および見出し語を同時に推定する手法を提案している. 実験では, CELEX データベース [8] を用いており, これには英語の見出し語情報も含まれるが, CELEX の規模は高々数万程度の語彙数であり, 一般的な英語の語彙数を考えると, 非常に小規模である.

Muller ら [5] は, 対数線形モデルを用いて, 単語の見出し語化と品詞タグ付けを同時に行う手法を提案している. 単語の表層

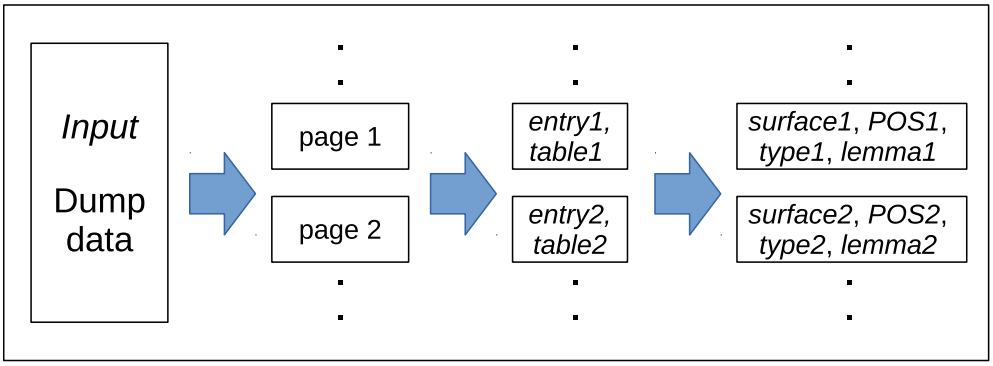

図 1　辞書構築の流れ

形と原形とのアラインメントは編集木という形式で表現され, 表層形をどのように編集すれば原形を生成できるかを学習するモデルを対数線形モデルとして定義している. 実験では, 英語を含む 6 種類の言語の見出し語化の実験を行っており, 英語の評価データとして SANCL コーパス [7] を用いている. しかしながら, 著者らの知る限りでは, 英語の見出し語の正解データは公開されておらず, 評価実験に利用することは困難である.

以上のように, 言語に依存しない汎用的な見出し語化手法はいくつか提案されてきたが, 英語の見出し語化の標準的なコーパスはあまり存在していないのが現状である. そのため, 我々はまずはじめに大規模な単語活用辞書を構築し, それを一般に利用可能な形式で公開する. また, 近年盛んに研究されているニューラルネットワークを用いた見出し語化の手法を提案し, 上記の単語活用辞書を用いてモデルの学習および評価を行う.

3. 単語活用辞書の構築

本研究では, まずはじめに, Wiktionary を用いて大規模な英単語活用辞書を構築する. Wiktionary は, ユーザーが編集可能な辞書およびシソーラスであり, 170 以上の言語に対応している. また, Creative Commons Attribution-ShareAlike 3.0 Unported ライセンスと GNU Free Documentation ライセンスによって公開されており, 基本的には誰でもフリーで利用可能である. ただし, Wiktionary は人間の奉仕活動によって編集されている辞書であるため, 辞書エントリごとに形式が必ずしも統一されているとは限らず, ノイズも多く含まれている. そこで本節では, Wiktionary のダンプデータを処理して形式を統一し, machine-readable な英単語活用辞書を構築する手順について説明する.

図 1 に辞書構築の概要図を示す. 我々の辞書構築法は, 主に次の 3 ステップから成る:

1. ダンプデータの分割
2. 表層語と屈折表の抽出
3. 屈折表の展開

上記の手続きによって最終的に得られる辞書は, 各エントリごとに表層語, 品詞, 活用の種類, 見出し語の 4 つの情報を持つ.

```
  </page>
  <page>
    <title>take</title>
    <ns>0</ns>
 ------- 中略 -------
===Verb===
{{en-verb|takes|taking|took|taken}}

# {{lb|en|transitive}}
```

図 2　Wiktionary のダンプデータの例

3.1 ダンプデータの分割

Wiktionary のダンプデータは XML 形式で公開されている. ダンプデータの例を図 2 に示す. ダンプデータ中で, それぞれの語彙はページという単位で構成されている. そこで, まずはじめに, `<page>`タグに挟まれた領域を抽出する. 抽出された領域は, `<title>` タグに挟まれた語彙の要素を必ず 1 つ含み, それに付随する屈折表などの情報などが含まれている.

3.2 表層語と屈折表の抽出

次に, `<page>`タグに挟まれた領域から, 語彙と屈折表の組を抽出する. ここで, *word* は語彙エントリ, *table* は屈折表, *POS* は品詞タグ, $option_{en}$ と $option_{head}$ は不規則活用を表すこととする. 語彙エントリと屈折表の抽出は, 各ページに対して以下の操作を行えばよい.

処理 `a`.　ページから *word* を抽出する
処理 `b`.　後続の領域から全ての可能な *table* を抽出する
処理 `c`.　全ての可能な組 (*word*, *table*) を出力する

処理 `a` において, *word* は`<title>`タグに挟まれているため, 正規表現を用いて抽出できる. また, 処理 `b` についても, 正規表現を用いて *table* を抽出できる. Wiktionary において, 英語の屈折表は, {{`en-`*POS*`|`$option_{en}$}} という形式（テンプレート）で表されている. また, 英語特有の表記として, {{`head|en|`$option_{head}$}} というパターンで表される場合もある. 後者の場合, 屈折形や見出し語の情報が $option_{head}$ に含まれないことがありうるので, 後続の領域から屈折語や見出し

表 1　辞書統計量

	種類数
見出し語	597149
表層語	803694

語の情報を抽出する．この場合の屈折表の後続には，一般に {{inflection of|$lemma$}} の形式で表される．このとき注意すべきこととして，後者の形式の場合，$word$ と $lemma$ を結びつける操作のみを行い，3.3 節で述べる屈折表の展開は行わない．また，個々のページに含まれる語彙の要素は必ず 1 種類であるが，屈折表については 2 種類以上含まれる場合もある．

3.3　屈折表の展開

最後に，抽出された見出し語と屈折表から，ありうる全ての表層語を生成し，見出し語と表層語の組を生成する．このとき，3.2 節でも触れたように，{{en-POS|$option_{en}$}} のパターンで表された屈折表のみを展開する．これは見出し語と屈折表の組に対し，屈折表が持つルールを適用することで表層語へと展開することができる．屈折表を解析する際，我々は python のライブラリである mwparserfromhell を用いた．英語の屈折表が持つ詳細なルールは，Wiktionary のページ (注1) で確認することができる．

ある語彙が規則活用である場合，屈折表は $option_{en}$ を持たず，不規則活用を行う表現は，$option_{en}$ を持つ．例えば，(award, {{en-noun}}) における屈折表には，$option_{en}$ は含まれない．また，en-noun は名詞し，この語彙の屈折表は，award が規則活用を行うことを意味している．従って，この例は単数形の名詞 award と複数系の名詞 awards を表現している．したがって，この場合には，単数形の名詞 award に接尾辞 s を加える操作を行い，複数形 awards を生成すればよい．以上をまとめると，(award, {{en-noun}}) からは以下の辞書情報が生成される．

```
 award: (1) 名詞 (2) 単数形 (3) award
awards: (1) 名詞 (2) 複数形 (3) award
```

また，(take, {{en-verb|takes|taking|took|taken}}) における屈折表は，不規則活用を説明するための $option_{en}$ を含む．また，en-verb は動詞を意味しているので，この屈折表は動詞 take が不規則活用を行うことを意味する．この場合には，動詞 take の全ての活用語が $option_{en}$ に記述されており，次のような辞書情報を出力すればよい．

```
  take: (1) 動詞 (2) 現在形       (3) take
 takes: (1) 動詞 (2) 3 人称単数 (3) take
taking: (1) 動詞 (2) 現在分詞   (3) take
  took: (1) 動詞 (2) 過去形     (3) take
 taken: (1) 動詞 (2) 過去分詞   (3) take
```

上記の手続きによって，Wiktionary のダンプデータから統一的な形式の辞書情報を出力させることができる．我々の構築した辞書の統計量を表 1 に示す．表 1 に示されているように，全部で 597149 種類の見出し語が抽出され，そこから 803694 種類の表層語が生成された．

4.　未知語のための見出し語化モデル

前節では，Wiktionary を用いた単語活用辞書の構築法について述べた．辞書に掲載されている表現に限っては，辞書を参照することによって，表層語と品詞の組から適切な見出し語へと変換することができる．しかしながら，データには必ず未知語が存在し，単純な辞書引きでは正しい見出し語化を行うことができない．

そこで我々は，大規模な教師なしデータを有効に活用して，屈折を伴う未知語から見出し語へと変換する統計モデルを提案する．説明のために，辞書の単位項目である表層語と見出し語の組を（s，l）と表記し，ある表層語 s の見出し語の候補となる語を c_s と表記する．

本節では，以下の 3 つのモデルについて説明する．

1. ルールベースの手法
2. Support Vector Machine (SVM) を用いた手法
3. ニューラルネットワークを用いた手法

4.1　ルールベースの手法

まずはじめに，ルールベースの手法を説明する．この手法は大規模な教師なしデータ（平文）を必要とする．ある見出し語対象の表層語 s に対し，平文に出現する全ての語との間のレーベンシュタイン距離を計算する．そして，最も距離が近い表現を s の見出し語として選択する．ただし，レーベンシュタイン距離が 0 の語は，見出し語の候補から取り除くこととする．これは，表層語 s と全く同じ文字列が選ばれることを防ぐためである．もしも同じ距離の候補が複数種類存在した場合は，その中からランダムで見出し語を選択する．

4.2　Support Vector Machine (SVM) を用いた手法

次に，機械学習を用いた統計的見出し語化の手法について説明する．1 つ目は，Support Vector Machine (SVM) を用いる手法である．SVM に基づく手法では，学習時に，表層語と，正解の見出し語の組（s，l）を必要とする．評価時は，学習されたモデルに加えて，K 個の見出し語の候補を必要とする．K 個の見出し語の候補については 5.2 節にて説明する．提案するモデルは，事前に用意した K 個の見出し語候補の中から，SVM を用いて最適な候補を決定する．

(注1)：https://en.wiktionary.org/wiki/Category:English_headword-line_templates

表 2　見出し語を選択をするための素性リスト．a_p は，s と c_s との間の部分アラインメント列中において，p 番目の部分アラインメントを指す．levenshtein 関数は (s, c_s) は s と c_s のレーベンシュタイン距離を返す．

部分文字アラインメント	(a_p)
	(a_{p-1}, a_p)
	(a_{p-2}, a_{p-1}, a_p)
距離	levenshtein (s, c_s)

SVM の学習を行う際，学習データとして正例と負例の事例を用意する必要がある．　正例と負例については，事前に構築した Wiktionary 辞書を用いて，以下のように用意する．　ある辞書の組 (s, l) に対し，s の正例を l とする．　次に，Wiktionary の見出し語集合から l を除いた集合のあらゆる語と，s との間のレーベンシュタイン距離を計算し，そこから上位 N ベストを導出する．　この上位 N 個を s の負例とする．　以上の操作を行い，辞書中の全ての表層語に対し，正例と負例を生成する．

適切な見出し語を選択するために，(s, c_s) に対して 4 種類の素性を用いた．　素性を表 2 に示す．　我々が用いた素性は，表層語と見出し語の文字アラインメントに基づいている．　文字アラインメントは，レーベンシュタイン距離の計算によって得ることができる．　1 つ目の素性は，s と c_s 間の部分文字アラインメントである．　ここで部分文字アラインメントとは，2 つの文字列間において，レーベンシュタイン距離を最小にするような部分文字列のアラインメントを指す．　例えば図 4 に示すように，(play, played) の組に対しては，[('p', 'p'), ('l', 'l'), ('a', 'a'), ('y', 'yed')] という 4 つの部分文字アラインメントが獲得され，これらが素性として抽出される．　2 つ目，3 つ目の素性は，部分アラインメントのバイグラム，トライグラムである．　最後に 4 つ目の素性として，s と c_s 間のレーベンシュタイン距離を用いる．

SVM の実装は，python ライブラリの sklearn に含まれている liblinear を用いた．　カーネル関数は線形カーネルを用いた．

4.3　ニューラルネットワークを用いた手法

3 つ目の手法として，ニューラルネットワークを用いた見出し語化の手法を提案する．　モデルのアーキテクチャを図 3 に示す．　我々のモデルでは，表層語の各文字から，見出し語の文字または文字列を順番に生成することによって見出し語を予測する．　この手法では，前述のルールベースや SVM の手法とは異なり，学習データには含まれていない見出し語を生成可能であるという特徴がある．　学習データには，Wiktionary 辞書 $D = \{(s^1,\ l^1), ..., (s^{|D|},\ l^{|D|})\}$ を用いる．

具体的な計算手順は以下の通りである．　まず，ある単語内の文字位置 t に対し，$t-3$ から $t+3$ までの表層語の文字 $s_{t-3:t+3}$ と，それに対応する見出し語の文字 $l_{t-3:t-1}$ をベクトル x_t^s，x_t^l へ変換する．　そして，この 2 つのベクトルを連結し，文字位置 t における入力ベクトル x_t を得る．

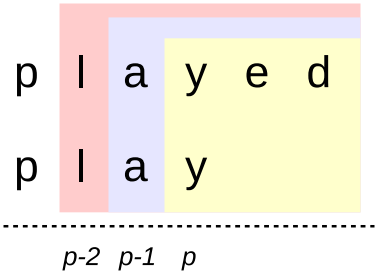

図 4　部分文字アラインメントの例

$$x_t^s = \text{concat}\ \ (c(s_{t-3}), ..., c(s_{t+3}))$$

$$x_t^l = \text{concat}\ \ (c(l_{t-3}), ..., c(l_{t-1}))$$

$$x_t = \text{concat}\ \ (x_t^s, x_t^l)$$

ここで関数 c は，引数の文字と対応する成分のみが 1 となり，それ以外の成分が 0 であるようなベクトルを返す．　関数 concat は，引数として受け取った全てのベクトルを連結し，新しい 1 つのベクトルを返す．

このベクトル x_t を，ニューラルネットワークの入力とする．そして，ベクトル x_t に対してアフィン変換を行い，活性化関数 tanh を適用し，隠れ層の出力ベクトル h を得る．　すなわち，

$$h_t = \tanh\ \ (W_h x_t + b_h)$$

ここで，$W_h \in \mathbb{R}^{d_h \times 10|C|}$ はアフィン変換の重み行列，$b_h \in \mathbb{R}^{d_h}$ はバイアスベクトルである．　また，$|C|$ は文字の種類数を指す．　また，d_h は隠れ層の次元であり，モデルのハイパーパラメータである．

次に，隠れ層の出力ベクトル h_t に対してさらにアフィン変換を行い，活性化関数 softmax を適用し，出力ベクトル o_t を得る．

$$o_t = \text{softmax}\ \ (W_o x + b_o)$$

ただし，$W_o \in \mathbb{R}^{|C| \times d_h}$ は重み行列，$b_o \in \mathbb{R}^{|C|}$ はバイアスベクトルである．　この出力ベクトル o_t は $|C|$ と同じ大きさのベクトルであり，各次元は文字と対応し，各文字が生成される確率値を持つ．　実際に見出し語の文字をデコードする際は，出力ベクトル o において最大の成分と対応する文字を生成する．

$$l_t^* = c'(\arg\max_{1 \leqq j \leqq |C|}\ [o_t]_j)$$

ここで，$[o_t]_j$ はベクトル o_t の j 次元目の成分を指し，関数 c' は引数として受けとった文字 ID を，対応付けられた文字へと変換する．　以上の計算を経て，見出し語の文字を生成することができる．　我々のニューラルネットワークのモデルを $y(\theta)$ と表記する．　ただし，θ はモデルのパラメータであり，θ = $\{W_h, b_h, W_o, b_o\}$ である．　学習時は，以下の誤差関数 $L(\theta)$ を最適化する．

$$L(\theta) = -\sum_{i=0}^{|D|} \sum_{t=0}^{|s^i|} c(l_t^i) \cdot \log(y(\theta))$$

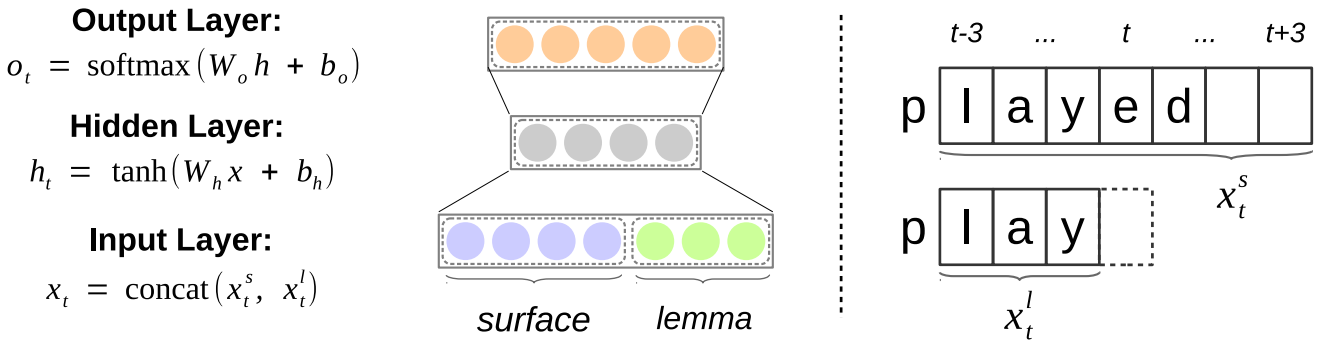

図 3　ニューラルネットワークのアーキテクチャ (左) と文字生成のイメージ図 (右).
右図において, 実線で描かれた長方形中の文字から破線部の文字を予測する.

学習を行う前に，-1 から+1 の範囲の一様分布で重みパラメータの初期化を行った． また，モデルのハイパーパラメータ d_h は 50 に設定した． 学習時は確率的勾配法の一種である ADAM [3] を用いてパラメータの最適化を行った． ADAM のハイパーパラメータは，$\alpha = 0.001, \beta_1 = 0.9, \beta_2 = 0.999, \epsilon = 10^{-8}$ に設定した．

5. 実　　験

5.1 辞書の網羅率の検証

構築した辞書の妥当性を示すために，実コーパスに出現した表層語に対する英単語活用辞書の網羅率を検証した． 実験に用いたコーパスは，新聞記事を基にした Penn Treebank と，Wikipedia のテキストデータである． Wikipedia のデキストは，Muller ら [6](注2) によって公開されている，文分割とトークナイズの前処理がされたものを用いる．

Penn Treebank は新聞記事のコーパスであり，約 122 万トークンで構成されている． 実験では，このデータ全てを評価対象とする． 一方，Wikipedia のテキストデータは，Web 上の記事コーパスであるため，Penn Treebank とは異なった語彙集合で構成されていると考えられる． Wikipedia のテキストデータは，約 19 億トークンから成り，この内 1000 万トークンを網羅率の検証の対象とする．

ここで，我々が意味する網羅率とは，コーパスに出現した語が辞書に含まれる割合を指す． この割合の算出において，コーパスに出現した語の頻度を基にした割合と，種類を基にした割合の 2 つが考えられる． 本論文では前者をトークンベースの網羅率と呼び，後者をタイプベースの網羅率と呼ぶこととし，実験では 2 つの網羅率を算出する． 網羅率を算出する際は，コーパスに出現した語に対して事前に小文字化の操作を行っている．

しかし，2 つのコーパスは固有名詞や数字列といった辞書への登録が困難な語を多数含んでいる． さらに複数の語をハイフンで結合した連語も多数含んでおり，辞書にこのような表現を登録することは困難であるため，素直に網羅率を算出することはできない． 加えて，Wikipedia のテキストデータは Web 上のテキストであるため，顔文字や崩れた表現，スペルミスなどの非形式的な表現も数多く含むため，このような表現の登録も現実的ではない． そこで，辞書登録が困難な表現の存在を考慮し，評価対象の表層語の範囲にいくつかの制約を与え，網羅率を算出する． したがって，以下の 3 つの設定で網羅率の評価を行った:

制約 1． 全トークン
制約 2． 固有名詞以外の活用可能なトークン
制約 3． 連語を除いたトークン

品詞を用いた制約は，コーパスに割り当てられた品詞タグを用いて制約を与える． ここで，Penn Treebank は正解の品詞タグを有しているが，Wikipedia のテキストデータには品詞タグが割り当てられていないため，前処理として Stanford Log-linear Part-Of-Speech Tagger(注3) を用いて事前に品詞タグを割り当てる． 連語かどうかについては，表層語が結合記号-(ハイフン) を含むか否かで判断する．

5.2 見出し語化モデルの評価

4. 節で述べた 3 つの見出し語化のモデルの評価を行う． このとき，モデルの予測結果を評価するために，見出し語の正解データを人手で構築する必要がある． 評価用の正解データ構築において，5.1 節においても利用した Wikipedia のテキストデータ全てと，我々の構築した Wiktinoary 辞書を用いた．具体的な手順は以下の通りである．

まず，トークナイズされたテキストデータの中から，我々の Wiktionary 辞書に含まれない語（未知語）をリストアップする． このリストアップされた候補語の中から，何らかの屈折現象を有する語を人手で選定し，正解の見出し語を注釈する．実際には，100 種類の未知語に対して，正解の見出し語を注釈した．

また，ルールベースと SVM による見出し語手法のために，Wikipedia に現れた全ての語と未知語との間のレーベンシュタイン距離を計算する． この距離を基に上位 K 個の単語をリストアップする． 以上の操作を全ての未知語に対して行い，リ

(注2)：http://cistern.cis.lmu.de/marmot/naacl2015/

(注3)：http://nlp.stanford.edu/software/tagger.shtml

表 3　網羅率の検証結果

コーパス	評価対象のトークン	網羅率		誤り例
		トークンベース	タイプベース	
Penn Treebank	全トークン	78.53	57.88	45.2, 877, Cairenes
	固有名詞以外の活用可能なトークン	96.05	81.80	mid-week, on-board, CDs
	ハイフンが含まれないトークン	97.86	96.53	p.m, goosey, Lilly
Wikipedia のテキストデータ	全トークン	75.45	31.35	HCFC, 0-0A, well.They
	固有名詞以外の活用可能なトークン	96.83	63.48	e., long-liners, identitification
	ハイフンが含まれないトークン	97.96	79.13	manager/producer, cuttin, ***.com

表 4　モデルごとの実験結果. 各セルの数値は正解率を意味する.

	正解が K ベストに含まれる割合	モデル				
		ルールベース	SVM			ニューラルネットワーク
			$N=3$	$N=5$	$N=10$	
$K=50$	0.73	0.10	0.69	0.70	0.69	0.94
100	0.78		0.74	0.75	0.74	
200	0.82		0.76	0.79	0.78	

ストアップされた K 個の語を，それぞれの未知語の見出し語の候補とする.

6. 結　　果

6.1 辞書の網羅率の評価結果

網羅率の評価結果を表 3 に示す． 全トークンの場合の評価では，コーパスが固有名詞等を多く含んでいるため低い網羅率となった． しかし，評価の範囲を狭めることで，十分な網羅率を示すことがわかった． Penn Treebank では，トークンベース·タイプベースで約 98%と，非常に高い網羅率となった. 一方で，Wikipedia 上での評価は，トークンベースではそれなりに高い結果であるが，タイプベースの評価は非常に低い網羅率となった． これは，Wikipedia のテキストデータが多くの非形式的な表記を含むことや，トークナイズのミスや書き手のスペルミスなどにより，語彙が巨大となってしまったことに起因すると考えられる． これらの結果から，我々が構築した単語活用辞書は，全体的に十分高い網羅率であるといえる.

次に，辞書に含まれなかった単語について検証する． Penn Treebank では，45.2 や Cairenes をはじめとする数字や固有名詞といった表現が数多く見られ，mid-week，on-board などのハイフンで結合された連語も数多く見られた． その他，p.m のようなコーパスにおける単語分割の誤りに由来する例や，副詞 (RB) とタグ付けされてしまった Lilly のような品詞タグのエラーに由来する例も見られた． 次に，Wikipedia データでは，manager/producer や cuttin といった連語やスペルミスなどの非慣習的な表現が数多く見られた． また，e. や ***.com のようなトークナイズのエラーに起因する例も数多く見られた． Penn Treebank と同様に，0-0A や long-liners のような数字や連語の例も多く見られた.

6.2 見出し語化の実験結果

3 つの統計的見出し語化手法の実験結果を表 4 に示す.

ルールベース手法の正解率は，0.10 と極めて低い結果となった． 適切な見出し語に置き換えられた事例も，表層語と見出し語の文字列の差異が接尾辞 s の有無に依存しているような，比較的簡単な事例に限られた． 一方，SVM を用いた統計的な手法は，ルールベースの手法を大きく上回る結果となった． さらに，辞書から導出される負例数 N を増加させるにつれて，正解率が上昇する傾向にあることを確認した． また，見出し語候補数 K を増加させるとモデルの正解率も上昇した． これは，見出し語の候補が増えるにつれて，その中に正解の見出し語が含まれる可能性が高くなるためだと考えられる． 我々が行った実験設定の中では，$N=200$，$K=5$ のモデルが最も良い正解率を示した.

ニューラルネットワークを用いた見出し語化のモデルは，最も高い正解率であった． SVM のモデルと異なり，ニューラルネットワークのモデルは，学習データに含まれていない見出し語も生成することができるため，未知語の見出し語推定に適していると考えられる.

次に，モデルごとの出力結果を比較する． ここでは，SVM の実験において最も高い正解率を示した $K=200$，$N=5$ のモデルと，ニューラルネットワークのモデルに焦点を当てる. 2 つのモデルの出力結果の比較を表 5 に示す.

SVM のモデルは，表 4 にも示した通り，正解が K ベストに含まれている場合に限っては十分高い正解率であった． しかし，正しい見出し語が見出し語の候補 K ベスト中に含まれない限り，モデルは誤った予測を行ってしまうため，モデルの正解率は K ベストの導出精度に大きく依存してしまう． そのため，更なる精度の向上のためには，より洗練された素性設計や，見出し語の候補生成を高精度にする必要がある． 今回の実験では，レーベンシュタイン距離を用いて K 種類の見出し語の候補を導出したが，異なる距離尺度や，K を増加させるといった可能性が考えられる.

ニューラルネットワークのモデルは SVM のモデルと異なり，見出し語の候補生成を必要としないという利点がある． そのた

表 5　$K = 200, N = 5$ 時の SVM モデルとニューラルネットワークのモデルの出力結果の比較.
太字かつ下線付きの出力結果は正解した事例を意味する.

	表層語	SVM ($K = 200, N = 5$)	ニューラルネットワーク	正しい見出し語
正解が K ベストに含まれる	free-running	**free-run**	**free-run**	free-run
	snoRNAs	**snoRNA**	**snoRNA**	snoRNA
	powerbombed	**powerbomb**	**powerbomb**	powerbomb
	mega-cities	**mega-city**	**mega-city**	mega-city
	populares	popularis	populare	popularis
正解が K ベストに含まれない	redocumented	redocumented	**redocument**	redocument
	enemy-occupied	enemy-occupied	**enemy-occupy**	enemy-occupy
	tonaries	tonare	**tonary**	tonary
	hellenize	hellenized	**hellenize**	hellenize
	match-winning	match-winning	match-winn	match-win

め，表 4 に示されているように，未知語に対しても正しい見出し語が予測できていることが多いことがわかった． ニューラルネットワークモデルの場合，本研究で用いたフィードフォーワード型だけでなく，近年盛んに研究されている Long Short-Term Memory (LSTM) [2] のようなリカレント型のアーキテクチャを取り入れることによって更なる精度向上が達成できる可能性がある.

7. 結　　論

本論文では，フリーで利用可能な Wiktionary を用いて，大規模な英単語の活用辞書を構築する手法について述べた． 既存のコーパスを用いて辞書の網羅性を検証したところ，構築した活用辞書は，新聞記事や Web のテキストに対して高い網羅率であることがわかった． さらに，未知語の見出し語を統計的に予測する手法を提案した． ニューラルネットワークを用いた見出し語化のモデルは，見出し語の候補生成が不要で，学習データに含まれていない見出し語も生成可能であり，他の手法と比較して最も高い精度を達成した． 我々が構築した単語活用辞書およびソフトウェアは一般に公開する予定である.

今後は，今回構築した辞書を他の自然言語処理のタスクに適用することを検討している.

文　　献

[1] Georgiana Dinu Grzegorz Chrupala and Josef van Genabith. Learning morphology with morfette. In Bente Maegaard Joseph Mariani Jan Odijk Stelios Piperidis Daniel Tapias Nicoletta Calzolari (Conference Chair), Khalid Choukri, editor, *Proceedings of the Sixth International Conference on Language Resources and Evaluation (LREC'08)*, Marrakech, Morocco, may 2008. European Language Resources Association (ELRA).

[2] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Comput.*, Vol. 9, No. 8, pp. 1735--1780, November 1997.

[3] D. Kingma and J. Ba. Adam: A Method for Stochastic Optimization. *ArXiv e-prints*, December 2014.

[4] Mitchell Marcus, Grace Kim, Mary Ann Marcinkiewicz, Robert MacIntyre, Ann Bies, Mark Ferguson, Karen Katz, and Britta Schasberger. The penn treebank: Annotating predicate argument structure. In *Proceedings of the Workshop on Human Language Technology*, HLT '94, pp. 114--119, Stroudsburg, PA, USA, 1994. Association for Computational Linguistics.

[5] Thomas Müller, Ryan Cotterell, Alexander Fraser, and Hinrich Schütze. Joint lemmatization and morphological tagging with lemming. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pp. 2268--2274, Lisbon, Portugal, September 2015. Association for Computational Linguistics.

[6] Thomas Müller and Hinrich Schuetze. Robust morphological tagging with word representations. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 526--536, Denver, Colorado, May--June 2015. Association for Computational Linguistics.

[7] Slav Petrov and Ryan McDonald. Overview of the 2012 shared task on parsing the web. In *First Workshop on Syntactic Analysis of Non-Canonical Language*, 2012.

[8] R. Piepenbrock R. H. Baayen and L. Gulikers. *The CELEX lexical database*. 1995.

[9] Kristina Toutanova and Colin Cherry. A global model for joint lemmatization and part-of-speech prediction. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, pp. 486--494, Suntec, Singapore, August 2009. Association for Computational Linguistics.