アルファベット表記への自動読み付与

市川博通，黒澤義明，目良和也，竹澤寿幸
広島市立大学大学院情報科学研究科

1. はじめに

長年の音声認識の研究開発により，技術レベルは着実に進歩し，音声認識を応用したアプリケーションも広がってきた．自動車運転時でも音声を使ってハンズフリーでのカーナビ操作が可能な音声カーナビ，議場で行われる膨大な音声をテキスト化し，修正・編集を行う音声議会録作成支援システム等，音声認識は様々な場面で用いられている．音声認識の利点には，高齢者をはじめ，ほとんどの人間にとって扱いやすいインタフェースであること，両手がふさがっていても利用可能であること等が挙げられる．

しかし，音声認識は誤認識が生じる場合がある．原因としては，騒音や人の声等の雑音，発話した単語が音声認識辞書に存在しない(未知語)という理由が考えられる．特に未知語の問題は重要である．音声認識システムでは，音声認識辞書に存在しない単語の場合，その単語の発話を認識することができないからである．

人手によらない未知語の解決策として，テキストから未知語を自動獲得し，辞書を拡張する手法がある．しかし，この手法では，未知語を自動で獲得することができても，音声認識などで必要な単語の読みが獲得できない．このため，未知語に読みを付与することが必要になる．平仮名やカタカナで構成される単語については，容易に読みを付与することが可能である．しかし，漢字やアルファベットで構成される単語については，簡単に解決できる問題ではない．読みの曖昧性が存在するからである．アルファベット表記の単語を例に挙げると，Wi-Fiを“ワイファイ”，Wikiを“ウィキ”というような多義性である．このため，読みを自動で付与することは困難である．

そこで，本研究ではアルファベット表記の単語の読みを，インターネット百科事典のWikipedia[1]及びn-gramモデルを用いて，自動で付与する手法を提案する．

2. 関連研究

本章では，未知語自動獲得，及び読み付与の観点から，2つの関連研究について紹介する．

2.1. 未知語自動獲得

未知語自動獲得の研究として，村脇ら[2]の研究がある．村脇らは，形態素解析結果から未知語を検出し，前後の文脈から考えられる語幹と品詞の候補を列挙し，最適な候補を選択する．このうち，列挙は日本語の持つ形態論的制約を利用する．検出された未知語の後続文字列を用い，可能性のある品詞，語幹の列挙を行い用例の蓄積を行う．選択は，複数の用例の比較を行う手続きである．そして，曖昧性が十分に解消できた時点で未知語の獲得を行っている．

2.2. 読み付与の関連研究

未知語の読みを付与する研究としては，笹田ら[3]の研究がある．笹田らは，n-gram モデルを記述するため，単語と読みの組$<c,y>_1^h$を以下に定義した．

$$<c,y>_1^h = <c,y>_1<c,y>_2\cdots<c,y>_h$$

ここで c は 1 文字を指し，y はその 1 文字に対応する読みを指す．なお h は単語の総文字数を意味する．

次に生成確率を以下の式で求める．

$$P_{<c,y>,n}(<c,y>_1^h) = \prod_{i=1}^{h+1} P\,(<c,y>_i \,|<c,y>_{i-n+1}^{i-1})$$

その上で，未知語の読み推定を行った．具体的には，はじめに漢字で構成されている未知語に対し，それぞれの文字について単漢字辞書から得られる読みを列挙する．その後，人手によって読みと単語境界が付与されているコーパスを用い，文字と読みの組を単位とする n-gram モデルから，単語と読みの同時確率を計算し，上位 L 個を発音辞書に追加する．その後，テキストと同じ話題を扱った音声と，音声認識用の音響モデルを用意し，音声認識の信頼度が閾値以上の音素列を抽出し，読みの付与を行っている．

3. 提案手法

本章では，アルファベットに対する読み付与の問題，及びアルファベット表記の単語の読みを自動で付与する手法について述べる．

3.1. アルファベットに対する読み付与の問題

漢字に関しては，読みが付与されているコーパスがあれば，漢字と読みを組とし，n-gramモデルのコーパスに用いることが可能である．しかし，アルファベットから構成される単語の場合は，1単語に対して読みを付与することが一般的であり，「 Wiki (ウィキ) 」を「Wi (ウィ)　ki（キ）」という記述はされていない．そのため，読みがどの部分のアルファベットに対応しているか曖昧である．そこで，アルファベットと音素の対応付けを行う必要がある．

また，出現する並びによって，異なる音素が用いられる．例えば「c」を例に挙げると，「script」では“ku”，「CD」では“shi:”，「carbon」では“k”，「document」では“ky”，「zilch」では“c”など，多くの異なる音素をもつ．そのため，アルファベットに対応する可能性のある音素を網羅しておく必要がある．

さらに，本研究では200,000件のTweetに対し，2.1節の手法を用いて，自動で未知語の獲得を行った．その結果，1,675個の未知語を獲得することができた．獲得した未知語を調べると，漢字を含む未知語が128個に対し，アルファベットを含む未知語は383個存在した．したがって，アルファベットから構成される単語についても，読みの付与を行い，辞書に登録する必要がある．

これらの理由により，本研究では，アルファベットで構成される未知語に対し，n-gramモデルを用いて，自動で読みの付与を行う．

3.2. アルファベットと音素の対応付け

本研究の基本的な考え方としては，2.2節の笹田らが行ったn-gramモデルを構築し，自動で読みの付与を行う枠組みを用いる．しかし，アルファベットの場合，読みがどの部分に対応しているか曖昧なため，本研究ではアルファベット1文字と音素を1つの組としてn-gramモデルの構築を行う．

n-gramモデル構築のコーパスとしては，Wikipediaを用いる．まず，読みが付与されているアルファベット表記のみから構成されるタイトルを5,300個収集した．そして，読みを音素に変換した．収集したアルファベット表記の単語と音素から，アルファベット1文字ごとに分割し，それぞれのアルファベットと音素の対応付けを行う．しかし，アルファベット表記の場合，アルファベットの音素がどこに対応しているか分からない．そこでアルファベット1文字に対する，可能性のある音素の候補リストを作成する．

初めに考えられる音素の候補として，ローマ字読み，アルファベット読みを想定し，図1のような候補リストを人手で作成する．

アルファベット	対応する音素
a	a，e:
b	b，bi:
c	k，shi:
⋮	⋮

図1　音素の候補リスト

作成した音素の候補リストを用い，アルファベット表記の単語に対する音素列の候補を全通り推定する．例えば単語 usb の場合，図2のように候補を生成する．

u|yu:	s|esu	b|bi:
u|yu:	s|esu	b|b
u|yu:	s|s	b|bi:
u|yu:	s|s	b|b
	⋮	

図2　単語usbに対する音素列の候補

音素の候補リストを用いて生成した音素列が，正しい音素列と完全に一致した場合，図3のように，アルファベットに対応する音素とし，n-gramモデルのコーパスとして用いる．

u|yu:	s|esu	b|bi:

図3　単語usbに対するアルファベットの音素

初めに生成した音素の候補リストでは，5,300個の単語に対し，1,261個の単語の正しい音素列の生成を確認した．しかし，残りの4,039個については，正しい音素列の生成ができず，アルファベット1文字と音素の対応付けが行えなかった．

3.3. 音素の網羅

初めに作成した図 1 の候補リストでは，単純な音素列しか生成できないため，考えられる音素を追加する必要がある．追加する方法としては，正しい音素列を生成できなかった単語について調べ，そのアルファベットに対応する音素が 2 つ以上の単語で使われていた場合，音素の候補リストに追加する．図 4 に対応が必要な例を挙げる．このようにして，アルファベットに対する音素の候補を追加した結果，5,300 個の単語に対し，5,056 個の単語の正しい音素列を生成することができた．したがって，本研究が提案する音素の候補リストが重要であることがわかる．

単語	アルファベット	追加する音素
“post|posuto” “smart|suma:to”	t	to
“globus|guro:basu” “linux|linaqkusu”	u	a
“xperia|ekusuperia” “xfy|ekusufai”	x	ekusu
“interix|iNterikusu” “lynx|riNkusu”	x	kusu

図 4　アルファベットと音素の対応付けが必要な例

図5のように，アルファベットと音素を1つの組とした5,056個の単語をn-gramモデルのコーパスに用いる．また，<s>は文頭，</s>は文末を表す記号である．

```
<s>  i|ai  m|m  a|a  c|qku  </s>
<s>  u|a  n|N  d|d  o|u  </s>
<s>  a|a  d|do  r|r  e|e  s|su  </s>
<s>  o|o  k|kish  v|i:  </s>
                ⋮
```

図5　n-gramモデルのコーパス

本研究では，図5に示したアルファベットと音素の対応付けを行ったコーパスからn-gramモデルの構築を行い，読みの付与されていない未知語に対し，自動で読みの付与を行う．

3.4. 未知語に対する音素列の候補

未知語に対しての読み付与は，アルファベットを1文字ごとに分割し，それぞれについて，構築したn-gramモデル中の1-gramから得られる音素を列挙する．そして，各音素を組み合わせ，可能性のある単語の音素列を全通り生成する．その後，アルファベットと音素の組を単位とするn-gramモデルにより，単語表記からの音素列の生成確率を計算し，確率の高い順に音素列を並び変える．また，図6のように，生成した音素列が一致する場合は，最終的に音素列から読みに変換すると，同じ読みになるため，確率の高い値を優先する．

```
p|p  e|*  r|a:ru  l|*
p|p  e|*  r|a:  l|ru
```

図6　生成する音素列が同じ場合の例

3.5. 生成する音素列の選択

1-gramを用いて，可能性のある音素列を全通り生成した場合，非常に多くの音素列の候補が生成され，処理に多大な時間を費やしてしまう．そこで，「末尾は必ず母音になる」，「qの音素は文頭に出現しない」という規則を簡易に実装するため，今回は2-gramに存在しない並びの候補は採用しないこととする．

4. 実験

実験はアルファベットと音素の対応付けが行えたWikipediaの単語5,056個に対しての読み付与の実験(4.1)と，Twitterのコーパスに対し，自動で獲得したアルファベット表記の未知語100個に対する読み付与の実験(4.2)を行った．n-gramは2-gram，3-gram，4-gramを用い，評価は正しい音素列が，生成した音素列の上位1件，3件，5件に含まれる場合，正解とする．

4.1. Wikipedia の単語に対しての読み付与実験

まず，5,056個の単語に対し，Leave-one-out 法を用いた時の実験結果を表1に示す．

表1　Wikipediaの単語に対しての実験結果

	2-gram	3-gram	4-gram
上位1件	39.9%	63.2%	49.9%
上位3件	57.5%	77.7%	67.3%
上位5件	64.5%	79.9%	72.2%

3-gramを用いた場合が上位1件，3件，5件ともに一番高い値となり，正解率はそれぞれ63.2%，77.7%，79.9%となった．上位5件までに正しい音素列を生成できた例としては「Alex|areqkusu」，「Sylpheed|shirufi:do」,「RADWIMPS|raqdowiNpusu」等が挙げられる．

次に誤り例を述べる．「tobaccojuice|tabakoju:su」，「LAZYgunsBRISKY|reiji:gaNzuburisuki:」等，比較的多くのアルファベットで構成されている単語に誤りが多い傾向があった．また，上位5件ではほとんどの場合，正しい音素列を生成できた．一方，上位1件で正しい音素列を生成できなかった単語としては，「rolly」の正しい音素列“ro:ri:”に対して“rori:”と音素列を生成した．加えて，「humanizer」の正しい音素列“hyu:manaiza:”に対して“hyu:maniza:”と音素列を生成する場合があった．

4.2. 自動獲得した単語に対しての読み付与実験

2.1節の手法を用いて，200,000件のTweetから自動獲得したアルファベット表記の単語100個に対して実験を行った．また，HN(ホームネーム)やAAA(トリプルエー)，GK(ゴールキーパー)等は本研究では正しい音素列を作りだすことができないため，除外している．さらに，自動獲得した単語がn-gramモデル構築時のコーパスに存在する場合は，その単語をコーパスから除いてn-gramモデルの構築を行った．実験結果を表2に示す．

表2 自動獲得した単語に対しての実験結果

	2-gram	3-gram	4-gram
上位1件	50.0%	72.0%	69.0%
上位3件	77.0%	93.0%	91.0%
上位5件	88.0%	94.0%	95.0%

上位1件，3件では3-gramを用いた場合，それぞれ正解率72.0%，93.0%となり一番高い数値となった．上位5件では4-gramを用いた場合，正解率95.0%となった．

3-gramで上位5件までに正しい音素列を生成出来なかった単語として，「Twitter」，「Excel」，「LED」，「AI」，「hyde」，「Xbox」があった．「Twitter」について調べると，正しい音素列“tsuiqta:”に対し，“towiqta:”，“tsuwiqta:”，“towita:”，“tsuwita:”，“tsuita:”という音素列を生成していた．「Excel」の場合は「x」に対応する音素“ku”が作成した音素の候補リストに存在しないため，正しい音素列を生成することが出来なかった．「LED」，「AI」に対しては，正しい音素列を生成することは出来たが，それぞれ上位20番目，6番目に生成していた．「Xbox」の場合は，n-gramモデル構築時のコーパスに「x」と「b」の並びが存在しなかったため，正しい音素列を生成することが出来なかった．

上位5件ではほとんどの場合，正しい音素列を生成できた．その一方で，上位1件で正しい音素列を生成できなかった例としては，「Tokyo」の正しい音素列“to:kyo:”に対して，“to:kyou”という音素列を生成していた．また，「firefox」の正しい音素列“faiafoqkusu”に対し“faiya:foqkusu”としている例があった．今回は正しい音素列に対し，生成した音素列が完全に一致した場合のみを正解とした．しかし，Wikipediaのコーパスを調べると，「fire」という表記に対し，「firebox」，「firebeat」の単語では“faiya:”，「firebird」では“faia”，「firewall」では“faia:”というそれぞれ異なる音素列であった．したがって，正解とする基準も正確に定める必要がある．

4.3. 構成されるアルファベットの平均数の分析

また，構成されるアルファベットの平均数を調べたところ，Wikipediaから獲得した単語が6.07に対し，Twitterで自動獲得した単語は4.47であった．よって，Wikipediaに登録されているアルファベットの方が，音素列の候補を多く生成するため，Twitterから自動獲得した単語の方が，正解率が高かったと考えられる．そこで，Wikipediaに登録されている単語に対し，構成されるアルファベットの平均数を自動獲得した単語の平均数と同程度に調整して，実験を行った(表3)．

表 3　平均 4.6 の Wikipedia の単語の実験結果

	2-gram	3-gram	4-gram
上位1件	53.0%	69.8%	58.2%
上位3件	76.4%	87.2%	80.9%
上位5件	85.7%	90.0%	87.8%

表 1 と比べると全体的に正解率が上昇している結果となった．しかし，表 2 と比べると全体的に正解率が低い．原因としては，Twitter で自動獲得した単語は日本語読みに基づいた音素列が多いのに対し，Wikipedia に存在する単語には，様々な言語固有の読み方に基づいた音素列が存在したため，正しい音素列を生成することが難しかったと考えられる．

5. 終わりに

本研究では，自動獲得したアルファベット表記の単語に対し，アルファベット1文字と音素の対応付けを行い，n-gramモデルを用いて読みの付与を行った．その結果，Twitterから自動で獲得したアルファベット表記の未知語に対し，3-gramを用いた場合，上位1件では72.0%，上位3件では93.0%，上位5件では94.0%という正解率を得た．したがって，アルファベット表記の単語に読みを自動で付与する場合，本研究が提案したアルファベット1文字と音素の対応付けが有効であるといえる．

謝辞

この研究の一部は，平成 23 年度 広島市立大学特定研究費(一般研究)の補助を得ている．関係各位に感謝申し上げる．

参考文献

[1] 百科事典 Wikipedia, http://ja.wikipedia.org/wiki/メインページ, (2011 年 11 月アクセス).

[2] 村脇有吾，黒橋禎夫：形態論的制約を用いたオンライン未知語獲得，自然言語処理, Vol. 17, No.1,pp.55-75 (2010)

[3] 笹田哲郎，森信介，河原達也：自動獲得した未知語の読み・文脈情報による仮名漢字変換，自然言語処理, Vol. 17, No.4,pp.131-153 (2010)