未整備の歴史的文献への濁点の自動付与アプリケーション

岡 照晃 † 小町 守 † 小木曽 智信 †‡ 松本 裕治 †
† 奈良先端科学技術大学院大学 ‡ 人間文化研究機構国立国語研究所

生の歴史的文献の中には，濁点が期待されるのに濁点の付いていない，濁点無表記の文字が多く含まれている．濁点無表記文字は可読性・検索性を下げるため，歴史コーパス整備の際には濁点付与が行われる．しかし，濁点付与は専門家にしか行えないため，作業人員の確保が大きな課題となっている．また，作業対象が膨大であるため，作業を完了するまでにも時間がかかる．そこで，我々は統計的機械学習を使った濁点自動付与アプリケーションを開発した．このアプリケーションは太陽コーパスにおける濁点の統計データに基づき，濁点無表記の文字を含んだ近代文語論説文へ自動で濁点付与を行うことができる．本アプリケーションを用い，近代の雑誌「国民之友」に適合率約 96%，再現率約 98%の濁点付与を達成した．本論文では，アプリケーションに実装した手法と，アプリケーションの仕様について概説する．

Application of Automatic Labeling of *Dakuten* for Raw Historical Text

Teruaki Oka† Mamoru Komachi† Toshinobu Ogiso†‡ Yuji Matsumoto†
†Nara Institute of Science and Technology
‡National Institute for Japanese Language and Linguistics

Raw historical texts often include mark-lacking characters, which lack compulsory *dakuten*. Since mark-lacking characters degrade readability and retrievability, dakutens are annotated when creating a historical corpus. However, since only experts can perform the labeling procedure for historical texts, getting annotators is a large challenge. Also, it is time-consuming to conduct annotation for large-scale historical materials. Therefore, we developed an application of automatic labeling of dakuten for mark-lacking characters by using a machine learning approach. Our application labels dakuten automatically for raw texts written in near-modern literary style of Japanese based on the statistics of dakuten in Taiyo corpus. We used this application, and achieved about 96% precision and 98% recall on a near-modern Japanese magazine, Kokumin-no-Tomo. In this paper, we abstract our implemented method and specification of the application.

1 はじめに

日本語で初めての大規模均衡コーパスとなる**現代日本語書き言葉均衡コーパス（BCCWJ）**[1] が昨年公開された．これにより今，コーパスを利用した日本語研究が急速に増えつつある．しかし，平安時代や明治時代といった古い時代の文献資料（**歴史的文献**）のコーパスは，現代語のコーパスほど整備が進んでいない．そのため，日本語研究の大きな位置を占める歴史的研究に，コーパスを用いることは未だ難しい．

歴史コーパスの整備が進まない原因の一つとして，歴史コーパスの整備に必要な**表記整理**の作業コストが高いことがある．表記整理とは，歴史的文献の記述中にある**現代の正書法に一致しない表記**をコーパスユーザにとって扱い易い形に修正し，可読性・検索性を上げることである．その一例として例えば，**濁点付与**の作業が挙げられる．濁点付与は，図 1 に示したような**濁点無表記**

今や廣島は其名大に内外國に顯はれ苟も時事を談するものは同地の形勢如何を知らんと欲せさるはあらす是れ征清の大師一たひ海に航せしより　大元帥陛下大纛を此に駐め大本營となし軍務を親裁し玉ふに因てなり先つ其大勢より叙述して次第に細事に及はんとす
（『太陽』 1925 年 2 号 p.64 より抜粋）

図 1: 濁点無表記の例．このテキストは太陽コーパスの原文から抜き出したものである．下線を引いた文字には，通常なら濁点が期待されるが，ここでは付けられていない．

表 1: 濁点と濁音の統計．**明六雑誌**[1]第 1 号（2011 年 12 月段階のデータ，総文字数:8,423）中に含まれる濁点文字と濁点を付けることが可能な文字の中から，実際の発音が清音の文字数と濁音の文字数の内訳を調査した．

表記＼発音	濁音（ガ，ザ，ダ，…）	清音（カ，サ，タ，…）	計
濁点文字（が，ざ，だ，…）	78	0	78
濁点を付けることが可能な文字（か，さ，た，…）	368 （濁点無表記文字）	1,273	1,641
計	446	1,273	

文字（濁音が期待されるのに，濁点の付いていない仮名文字）を**濁点文字**（濁点付きの文字）に置き換えるという作業であり，現在はすべて人手で実施されている．表 1 を見ると，時代が比較的新しい明治初期の資料であっても，濁音の仮名文字の内，約 83%は濁点無表記で書かれている．そのため濁点付与を行なっておかないと，歴史コーパスのユーザにとっては読み辛く，検索にも不便である．

ただし，濁点付与のような表記整理の作業を人手で行うのは非常にコストが高い．特に作業者が専門家に限られてしまうことが問題である．表記整理の作業を行うためには，まず対象となる資料を読んで理解しなくてはならない．しかし，歴史的文献の中では現代とは語彙が異なる上，表記や語法の面でも多様であり，読解には専門的な知識が必要である．作業の信頼性を保証するためにも，作業者は歴史的文献の専門家に限定される．だが，そういった専門家を集めることは難しく，作業人員の確保が大きな課題となっている．

作業対象となる資料の量が膨大であることも問題である．例えば，国立国語研究所が構築した近代語のコーパス**太陽コーパス** [8] は総文字数約 1,450 万文字の規模[2]である．これに対し，熟練した作業者でも 23 ページ分の資料（約 1 万 6,000 文字）の濁点付与に 1 日掛かりで取り組む必要がある[3]．上述の通り，作業者を大量に確保することは困難であり，人海戦術を取ることができない．そのため，大量の資料を少数人で少しずつ整備していくしかなく，整備を終えるまでに多大な時間が必要となっている．

また濁点付与を人手で行う場合，熟練した作業者であっても単純なミスをおかすことがよくある．特に，濁点を付与すべき文字を見逃してしまうことが多い．実際，2 人の作業者がそれぞれに行なった濁点付与の結果を比較してみたところ，濁点を付けた個所の一致率は 84%であり，一致しなかった原因のほとんどが濁点の付け逃しであった．また，コーパス整備の初期段階で 1 人の作業者が実施した濁点付与の結果を完成後のコーパスと見比べた．すると作業者が単独で濁点付与を行なった結果では，濁点無表記文字のおよそ 7%を見逃していたことが分かった．

[1] 1874 年（明治 7 年）～1875 年（明治 8 年）の間に明六社から発行された啓蒙雑誌．全 43 号．
[2] 2005 年 11 月段階のデータ．
[3] 1 日の総作業時間を 5～6 時間とした場合．

図 2: AYTC〜にごり ONLY〜（キーボード入力モード）

国立国語研究所では太陽コーパスに引き続き，更なる近代語コーパスの整備が進められている．しかし，表記整理の作業の負担から，現場ではその自動化を望む声が上がっている．

本研究では，統計的機械学習を用い，表記整理作業の支援技術を開発することを目的としている．これは従来行われてこなかった新しい試みである．歴史的文献の表記整理はこれまですべて人手で行われてきたが，作業者不足と作業対象の膨大さから実施に高いコストを必要とした．また人手の作業にミスはつきものである．しかしその作業を計算機に手伝わせることで，大規模な表記整理も低コストで実現でき，濁点無表記文字の見逃しといった単純なミスも減らすことができると考えられる．

表記整理支援技術開発の第一段階として，濁点付与の作業を取り上げ，濁点付与の自動化に取り組んだ [3, 4, 5]．最初の目標として濁点付与を取り扱ったのは，濁点付与のタスクが「濁点が抜け落ちた文字に濁点を補う」と明確で取り組みやすかったためである．またタスクが単純な割に，表記整理の中での必要性は高い．これはコーパスの可読性や検索性向上の観点から，濁点の抜け落ちている問題が無視できないためである．

本論文では，文献 [3, 4, 5] で提案した濁点の自動付与手法を実装した濁点自動付与アプリケーション **AYTC〜にごり ONLY〜**[4]（図 2）について述べる．2 章で，提案手法について簡潔な説明を行い，3 章では，アプリケーションの仕様と濁点付与の性能について概説する．最後に，4 章でまとめを述べる．

2 提案手法: 点予測による濁点の自動付与

文献 [3, 4, 5] では，濁点の自動付与のタスクを文字単位のクラス分類問題として定式化した．具体的には，資料中に存在する「濁点の付く可能性のある文字（**清濁曖昧文字**）」を濁点文字に置き換えるべきか否かを分類器を用いて判定する．ただし本論文では，清濁曖昧文字として平仮名と

[4] http://cl-lab.naist.jp/~teruaki-o/AYTC

か，き，く，け，こ，　さ，し，す，せ，そ，　た，ち，つ，て，と， は，ひ，ふ，へ，ほ，　ゝ，／＼

図 3: 清濁曖昧文字一覧．この 22 文字が濁点付与の対象となる文字である．

踊字である**くの字点**（／＼，／″＼）だけを扱い，片仮名は扱わないこととした[5]．そのため，濁点付与の対象となる文字は図 3 に挙げた 22 種類である．

提案手法では，分類は点予測によって行う．点予測では，資料中に存在する清濁曖昧文字に対し，それぞれ独立に分類を実施する．当該文字の分類において，周囲の文字の分類結果を見ないため，分類の誤りが伝播することがない．そのため，表記の整っていない未整備の資料に対しても頑健に動作することができる．

また提案手法では，分類の**素性**[6]として**分類対象文字**の周辺文字列の表層的な情報のみを使い，周囲の単語境界の情報や品詞の情報は使用しない．そのため，学習用コーパスに形態素解析済みコーパスを必要とせず，形態素解析辞書も必要ない．

分類器には 2 値分類器を用い，濁点を付ける（分類対象文字が濁点無表記文字である）事例を正例，濁点を付けない（分類対象文字が単なる清音仮名文字である）事例を負例とした．また，分類器の学習には LIBLINEAR[7]（バージョン 1.8）の L1 正則化ロジスティック回帰を使用した [2].

2.1 分類に使用する素性

分類の素性には図 4 のように，分類対象文字と，その左右 3 文字の範囲内の文字 n-gram の 2 つ組みを使用する．ここでは文字 1〜3-gram までを素性として利用した．各 n-gram には出現位置（分類対象文字からの相対位置）を添え字として設けており，各素性は,「その位置にその n-gram が現れたか（1）否か（0)」を表す 2 値素性となっている．

また，歴史的文献は表 1 で示しているように，完全に無濁点になっているとは限らない．所々には濁点が付いた状態の資料もある．濁点の使い方は書き手によって一定でない．そのため，あらゆる濁点の表記状態に対応するためには，こういった濁点を外して素性に使うべきである．しかしながら，予め施されていた濁点は分類の際の証拠として有効な場合もあると考えられる．そこで分類時には，文字 n-gram 内に含まれる濁点文字の一部〜すべてより濁点を外した n-gram も一緒に発火させることにした．この素性は，図 4 において，括弧付きで示してある．

2.2 学習用の事例の作成手順

分類器の学習に用いる事例は，学習用コーパス中にある学習文字（図 5 参照）から作成する．学習用コーパスには表記整理済みで濁点無表記文字を含まない歴史的文献を使用する．学習用事例作成の手順は以下の通り．

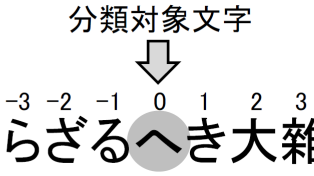
Step 1: 学習用コーパスから学習文字を 1 つ取り出す（e.g.,「が」）.

Step 2: 取り出した文字とその左右 3 文字を合わせて 1 つの事例とみなす．この際，取り出した学習文字が濁点文字であれば，濁点を外してから事例を作る（「が」→「か」）.

[5]漢字片仮名交じり文を除いて，基本的に片仮名は外来語や固有名詞などの限られた表記にしか用いられていない．またそういった語に関しては，濁点を付けるか否かの判定が人間でも困難な場合が多い．そのため，本論文では片仮名は対象外とした．

[6]分類の手がかりとなる情報.

[7]http://www.csie.ntu.edu.tw/~cjlin/liblinear/



位置-3 の文字 1-gram =	ら	+ 分類対象文字 = へ	位置-1 の文字 2-gram =	るへ	+ 分類対象文字 = へ
位置-2 の文字 1-gram =	ざ	+ 分類対象文字 = へ	位置 0 の文字 2-gram =	へき	+ 分類対象文字 = へ
(位置-2 の文字 1-gram =	さ	+ 分類対象文字 = へ)	位置 1 の文字 2-gram =	き大	+ 分類対象文字 = へ
位置-1 の文字 1-gram =	る	+ 分類対象文字 = へ	位置 2 の文字 2-gram =	大雑	+ 分類対象文字 = へ
位置 0 の文字 1-gram =	へ	+ 分類対象文字 = へ	位置-3 の文字 3-gram =	らざる	+ 分類対象文字 = へ
位置 1 の文字 1-gram =	き	+ 分類対象文字 = へ	(位置-3 の文字 3-gram =	らさる	+ 分類対象文字 = へ)
位置 2 の文字 1-gram =	大	+ 分類対象文字 = へ	位置-2 の文字 3-gram =	ざるへ	+ 分類対象文字 = へ
位置 3 の文字 1-gram =	雑	+ 分類対象文字 = へ	(位置-2 の文字 3-gram =	さるへ	+ 分類対象文字 = へ)
位置-3 の文字 2-gram =	らざ	+ 分類対象文字 = へ	位置-1 の文字 3-gram =	るへき	+ 分類対象文字 = へ
(位置-3 の文字 2-gram =	らさ	+ 分類対象文字 = へ)	位置 0 の文字 3-gram =	へき大	+ 分類対象文字 = へ
位置-2 の文字 2-gram =	ざる	+ 分類対象文字 = へ	位置 1 の文字 3-gram =	き大雑	+ 分類対象文字 = へ
(位置-2 の文字 2-gram =	さる	+ 分類対象文字 = へ)			

図 4: 提案手法で使用する素性．ただし，上記の事例において発火する素性のみを示している．

か，き，く，け，こ， さ，し，す，せ，そ， た，ち，つ，て，と，は，ひ，ふ，へ，ほ， ゝ，／＼
が，ぎ，ぐ，げ，ご， ざ，じ，ず，ぜ，ぞ， だ，ぢ，づ，で，ど，ば，び，ぶ，べ，ぼ， ゞ，／〵＼

図 5: 学習文字一覧．

Step 3: Step 1 で取り出した学習文字（「が」）が濁点文字であれば当該事例を正例とし，濁点が付いていなければ負例とする．

3 濁点自動付与アプリケーション「AYTC〜にごり ONLY〜」

AYTC は，現在開発中の歴史的文献用表記整理支援ツールの総称である．本論文で述べるにごり **ONLY** バージョンでは，2 章で述べた手法を濁点自動付与モジュールにごりとして先行実装している．今回の実装では，近代文語論説文のみを対象とし，分類器の学習用コーパスとして以下の表記整理済みコーパスを使用した．

- **UniDicMLJ-TRAIN:** 近代文語 UniDic のコスト算出に用いられたコーパス．
- **SUN-TRAIN:** 太陽コーパスの文語記事[8]から 9 割を学習用コーパスとして使用．実際には，太陽コーパスの 1895 年 5 号，1901 年 5 号，1909 年 5 号，1917 年 5 号，1925 年 5 号を評価用コーパスとして別に分け (SUN-TEST)，残りを学習に利用している (SUN-TRAIN).

学習用コーパスから抽出した事例の内訳を表 2 に示す．2 章で述べたように濁点付与の対象となる文字は平仮名とくの字点に限っている（図 3）．ただし AYTC では以下の処理をアプリケーション内部で行うことで，明六雑誌のような漢字片仮名交じり文への濁点付与にも対応している．

Step 1: 入力文章中の片仮名文字をすべて平仮名文字に置換．

[8]2011 年 1 月段階のデータ

表 2: 学習用事例の内訳.

学習用コーパス	総文字数	正例	負例	合計
UniDicMLJ-TRAIN	604,966	26,123	110,974	137,097
SUN-TRAIN	6,380,398	208,099	962,580	1,170,679

Step 2: 濁点の自動付与を実行.

Step 3: 濁点を付与した文章の中の平仮名文字を再度すべて片仮名文字に置換し，出力する.

AYTC〜にごり ONLY〜（以下，単に AYTC）の開発に当たって，多くの文系研究者が本アプリケーションを利用可能なよう，以下の要件を満たすものを目指した.

要件 1: 幅広い PC 環境で動作可能.

要件 2: 利用者が自分の PC にインストールせずとも利用可能．もしくはインストールが簡単.

要件 3: 操作方法が単純明快.

要件 1 と 2 を満たすため，AYTC は Silverlight アプリケーション[9]として開発を行なった．Silverlight アプリケーションはオンライン環境のブラウザ上で動作するアプリケーションであり，アプリケーション自体をユーザの PC にインストールすることなく利用できる．また Silverlight アプリケーションの動作には，Microsoft が提供する Web ブラウザ用プラグイン Silverlight[10]をインストールする必要があるが，Windows OS の PC には購入時からプリインストールされていることもあり，PC の操作に不慣れな者にも敷居は低いと考えられる．プリインストールされていない場合でも，下記のサイトから簡単にインストールすることができる．ブラウザには Microsoft の Internet Explorer だけでなく，Firefox や Chrome，Safari が対応している．また Silverlight アプリケーションは Windows OS だけでなく，Mac OS 上でも動作可能である.

Silverlight アプリケーションは基本的にオンライン環境のブラウザ上で動作する（ブラウザ内実行）．しかし，アプリケーションを各ユーザの PC へインストールすることで，通常のデスクトップアプリケーションと同様，ブラウザを介さずオフライン環境での動作が可能になる（ブラウザ外実行）．AYTC もブラウザ外実行を念頭に入れて開発を行なっている．インストールには単純な 1 ステップの操作しか必要とせず，面倒な設定等は一切必要ない.

要件 3 を満たすために，AYTC では単純かつ直観的な操作方法を実現している．複雑な操作は一切必要ない．以下の 3.1 節では，AYTC の機能について概説するが，詳細な操作方法などについては文献 [6] を参照してほしい.

3.1 AYTC による濁点の自動付与

AYTC では，**キーボード入力モード**（図 2）と**ファイル入出力モード**の 2 種類のモードで濁点の自動付与を行うことができる.

キーボード入力モードでは，ユーザがキーボードやクリップボードから入力した文章 (図 2 左) に対し，AYTC がリアルタイムで濁点自動付与を行なっていく (図 2 右).

[9]Silverlight のバージョンは 5.
[10]http://www.microsoft.com/ja-jp/silverlight/

ファイル入出力モードでは，指定された入力テキストファイルに対して濁点自動付与が一括で行われ，その結果は指定された出力先ファイルへと書き出される．入力ファイルにはテキストファイルの他にも，太陽コーパスと同形式の XML ファイルを指定することができる．XML ファイルはファイル識別子.xml で自動認識される[11]．AYTC には，ファイルのコンバート機能は搭載しておらず，出力ファイルの形式は入力ファイルの形式に準じたものとなる．そのため例えば，テキストファイルでの入力を XML ファイルとして出力することはできない．

ファイルの形式が XML の場合，AYTC は本文中に存在するすべての清濁曖昧文字に対し，以下の AYTC タグを付与する．

・**濁点付与を行なった場合：** 本文は濁点文字に置き換え，濁点の付いていない元の文字をタグ内の属性「原文」に残す．

〈**AYTC 原文=”か” 確信度=”0.9”**〉が〈**/AYTC**〉

・**濁点を付与しなかった場合：** AYTC タグを付けるだけで，本文への変更は行わない．また，タグ内に属性「原文」を含まない．

〈**AYTC 確信度=”0.4”**〉か〈**/AYTC**〉

AYTC では，**濁点付与の確信度**を使って各文字に濁点付与を行なっている．この値には分類器の分類スコアを利用する．分類器の学習にロジスティック回帰を使用しているため，確信度は 0〜1 の値を取り，0 に近いほど濁点を付与することへの確信が低く，反対に 1 に近いほど確信が高い事を表す．AYTC では，確信度 0.5 以上で本文中の文字を濁点文字と置き換える処理を行なっている．各文字に対する確信度は，AYTC タグ内に保持してあるので，タグを見れば各文字がどのくらいの確信をもって濁点を付けたり付けなかったりしたかを確認できる．

3.2 AYTC による濁点自動付与性能の評価

AYTC の濁点付与の性能評価実験を行なった．表記整理が行われていない近代文語論説文を対象とし，濁点付与の適合率と再現率を調べた．アプリケーションの評価には以下のコーパスの文語記事を利用する．

・**SUN-TEST:** 太陽コーパス 1895 年 5 号，1901 年 5 号，1909 年 5 号，1917 年 5 号，1925 年 5 号．

・**NF-TEST:** 現在コーパス化の作業が進められている明治期の雑誌**国民之友**．ここでは，2011 年 3 月の段階で濁点付与が実施されていた 1887 年 10 号，1888 年 20 号，1888 年 30 号，1888 年 36 号を使用した．

上記のコーパスはいずれもすでに表記整理済みである．しかし，タグを使って原文が保持されているため，評価にはタグから再現した原文を使用する．評価用コーパスから抽出した事例の内訳を表 3 に示す．SUN-TEST における正例数が NF-TEST に比べて少なくなっているが，これは太陽コーパスの基となった近代の雑誌「太陽」の刊行時期において，濁点の使用が普及しつつあり，濁点無表記文字の数が減少しつつあったためである．実際，太陽コーパス内の文語記事全体における濁点使用率は 96.7%であると報告されている [7]．

結果を表 4 示す．SUN-TEST における適合率が NF-TEST よりも低くなっているが，これは定常的に生じてしまう濁点の付け間違いの数に対して，SUN-TEST 中の正例の数（濁点を付けて正解の

[11] 文献 [6] では，「AYTC は XML ファイルの記事・引用タグの属性「文体」で文語文と口語文を区別し，文語文にのみ濁点付与を実行する」と述べたが，現在のバージョンではこの区別は行なっていない．

表 3: 評価用事例の内訳.

評価用コーパス	総文字数	正例	負例	合計
SUN-TEST	619,357	899	92,803	93,702
NF-TEST	172,780	3,842	25,418	29,260

表 4: 濁点付与の性能評価.

評価用コーパス	適合率.[%]	再現率.[%]	F 値
SUN-TEST	70.6	96.0	81.4
NF-TEST	95.8	98.0	96.9

箇所）が極端に少ないためである．さらに詳細な結果の考察やエラー分析については文献 [3, 4, 5] を参照してほしい．

4 おわりに

本論文では，近代文語論説文を対象とした濁点の自動付与アプリケーション AYTC の紹介を行なった．AYTC は太陽コーパスを学習に利用し，点予測により近代の文語資料に濁点の自動付与を行うことができる．濁点自動付与の性能評価実験を行なった結果，国民之友に適合率，再現率共に 95%以上の性能で濁点付与が行えることが分かった．

今後の課題として，近世の資料や中古和文といった近代文語論説文以外の文体への対応が挙げられる．また，濁点付与以外の表記整理作業（e.g., 送り仮名の正規化や文分割）の自動化にも今後取り組んでいく予定である．

謝辞

本研究は，国立国語研究所の共同研究プロジェクト「統計と機械学習による日本語史研究」による研究成果の一部である．

参考文献

[1] Kikuo Maekawa: Balanced Corpus of Contemporary Written Japanese, In *Proceedings of The 6th Workshop on Asian Language Resources* (ALR 2008), pp.101-102, 2008.

[2] Rong-En Fan, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-Rui Wang and Chih-Jen Lin: LIBLINEAR: A Library for Large Linear Classification, *Journal of Machine Learning Research*, 9, pp. 1871-1874, 2008.

[3] Teruaki Oka, Mamoru Komachi, Toshinobu Ogiso and Yuji Matsumoto: Automatic Labeling of Voiced Consonants for Morphological Analysis of Modern Japanese Literature, In *Proceedings of the 5th International Joint Conference of Natural Language Processing* (IJCNLP 2011), pp. 292-300, 2011.

[4] 岡照晃, 小町守, 小木曽智信, 松本裕治: 機械学習による近代文語文への濁点の自動付与, 情報処理学会研究報告 自然言語処理研究会報告, 2011-NL-201:6, pp. 1-8, 2011.

[5] 岡照晃: 統計的機械学習による歴史的資料への濁点の自動付与, 第 1 回コーパス日本語学ワークショップ予稿集, pp. 13-22, 2012.

[6] 岡照晃: 近代文語論説文を対象とした濁点の自動付与アプリケーション, 第 2 回コーパス日本語学ワークショップ予稿集, pp. 305-314, 2012.

[7] 近藤明日子: 濁点文字使用率から見る濁音表記, 雑誌「太陽」による確立期現代語の研究—「太陽コーパス」研究論文集—, 国立国語研究所報告 122, pp. 331-350, 博文館新社, 2002.

[8] 国立国語研究所編: 太陽コーパス, 国立国語研究所資料集 15, 博文館新社, 2005.