Studienprojekt Lehrstuhl für Computerlinguistik Universität Heidelberg JETLAG Teilprojekt LeJa: Ein Lemmatisierer für Japanisch

Iris Vogel (tori_ayame@yahoo.co.jp)

19. August 2003

Inhaltsverzeichnis

1 Die Theorie

1.1 Die Aufgabe

In diesem Teil des Studienprojekts wird ein flektierter String auf seine grammatischen Merkmale untersucht, in seine Lemmaform (Wörterbuchform) umgewandelt und im Wörterbuch nachgeschlagen. Als Hilfsmittel dienen das frei im Internet verfügbare, von Jim Breem entwickelte Japanisch-Englisch/Deutsch/Französisch Wörterbuch JMdict (http://www.csse.monash.edu.au/ jwb/edict.html), im

Unicode-Format (allerdings nicht in UTF-8 Kodierung, sondern umgewandelt in UCS2), die Theorie der Grammatik von Bruno Lewin [1] und natürlich C++.

1.2 Die japanische Schrift

Als eines der schwierigsten Schriftsysteme gilt das Japanische, in welchem regulär drei Schriften gemischt werden: Kanji, die von der chinesischen Schrift übernommenen Idiome und die in Japan davon abgeleiteten Silbenschriftsysteme Hiragana und Katakana[2]. Für flektierende Wörter sind nur zwei Schriftsysteme von Bedeutung (Katakana werden nur für Nomen verwendet) und ihre Kombination ist für die Analyse der Texte wichtig, denn grammatische Endungen können nur in Form von Hiragana-Ketten auftreten, während Kanji grundsätzlich zum Stamm eines flektierendes Wortes gehören müssen. Der Punkt des Übergangs zwischen Kanji und Hiragana ist aber nicht automatisch der Trennpunkt zwischen Stamm und Endung, der Stamm kann auch durch eine Kanji-Hiragana Abfolge dargestellt werden. Ebenfalls ist zu beachten, dass jedes Wort auch in einem reinem Hiragana-String realisiert werden kann (umgekehrt aber nicht alle Worte mit Kanji). So ergeben sich folgende Möglichkeiten:

Stamm	**Endung**
Kanji	Hiragana
yo	manai
読	まない
Kanji + Hiragana	Hiragana
tabe	nai
食べ	ない
Hiragana	Hiragana
yo	manai
よ	まない
tabe	nai
たべ	ない

[1] Bruno Lewin: *Abriss der Japanischen Grammatik*, Harrosowitz, Wiesbaden: 1990

[2] Lateinische Buchstaben finden zwar in Texten auch vereinzelt Verwendung, für die flektierenden Wortarten spielen sie aber überhaupt keine Rolle und werden deshalb auch nicht in die Überlegungen einbezogen.

Es steht also nur fest, dass die Endung aus Hiragana bestehen muss (genauer gesagt die Endungen, denn es gibt eine beträchtliche Anzahl).

Mit Verbstamm ist der Bestandteil gemeint, mit dem durch Anhängen der Finalform der Verbgruppe das Lemma (Wörterbuchform) gebildet werden kann, für *yo/manai* ist die Finalform *-mu*, *yomu* steht im Wörterbuch mit der Bedeutung *lesen*. Obwohl *tabe-nai* die gleiche Negierungsendung hat, gehört *be* zum Stamm, die dazugehörige Finalform lautet *-ru*, was sich an Hand des Wörterbuchs nachprüfen lässt. Theoretisch könnte sich *yomanai* auch auf *yomaru* zurückführen lassen, beziehungsweise tabenai auf *tabu*, diese Möglichkeiten werden durch das Wörterbuch jedoch ausgeschaltet.

1.3 Die Idee

Hinter diesem Programm steckt die Idee, dass japanische Verben (und Adjektive) nicht nur sehr regelmäßig flektieren und grundsätzlich agglutinieren sondern dass es auch eine Reihenfolge gibt, in der die Suffixe angehängt werden müssen. Es gibt Suffixe, die sich gegenseitig ausschliessen und von manchen grammtischen Suffixen gibt es verschiedene Formen, je nach Verbgruppe, an die sie anschliessen. Auch Suffixe können flektieren, also müssen folgende Bedingungen zwischen grammatischer Endung und dem vorherigen Element erfüllt sein:

- Übereinstimmung von Verbart (Suffixart) und der vom Suffix geforderten Verbart
- Übereinstimmung von Verbform (Suffixform) und der Verbform, an die das Suffix anschließt
- Level des Suffixes muss höher sein als das des Stammes (beziehungsweise Suffixes, das dem Stamm näher ist)

Die Idee ist also beginnend am Ende (denn da muss auf jeden Fall eine Hiragana-Endung vorhanden sein) die möglichen Aufteilungen in Stamm und Endung, beziehungsweise Endungen hypothetisch durchzuspielen und am Stamm angelangt anhand eines Wörterbuchs, in welchem die Flexionsart der Verben verzeichnet ist die Hypothese auf ihre Tauglichkeit zu prüfen. Dabei ist es allerdings in seltenen Fällen möglich, dass es mehr als eine Rückführungsmöglichkeit gibt (oft decken sich beispielsweise die Formen für Passiv und Potentialis), und vor allem bei Verben, die vollständig in Hiragana realisiert wurden, ist es möglich, dass mehrere Einträge im Lexikon für einen String gefunden werden, unter Umständen mit sehr unterschiedlicher Bedeutung.

2 Die Realisierung

2.1 Die String-Klasse UMString

Zunächst wird eine String-Klasse zur Verwaltung von Multibyte und/oder Unicode strings benötigt. Die UMString-Klasse (von Torben Pastuch) relativiert Kodierungsprobleme insofern, als ein in sie eingbauter Konvertierer dafür sorgt, dass Unicode und Multibyte (Shift-JIS, JIS und EUC) gleichermassen behandelt werden. Leider ist die Funktion Windows-spezifisch, das heisst für eine Portierung nach Linux müsste die Konvertierung neu implementiert werden[3].

Ein grosser Vorteil sind die 'sliding indices', mit welchen man gut den String durchlaufen, überprüfen und trennen kann. Für den Lemmatisierer scheint die String-Klasse eher überdimensioniert, dahinter steht aber der Gedanke, dass sie auch für den Segmentierer Funktionen bieten soll.

2.2 Das grammatische Gerüst

Die grammatischen Regeln, nach denen Verbendungen aufgebaut sind, sind in einer XML-Datei (flexuc.xml) festgelegt und werden bei Programmstart eingelesen. Das bedeutet, dass Einträge geändert werden können ohne den Quellcode zu ändern[4]. Der Aufbau lässt sich aus der dtd erschließen, inhaltlich werden folgende Merkmale festgeschrieben:

gramform beinhaltet die Merkmale jeweils einer grammatischen Form

[3] Eine Möglichkeit sind die C-Routinen wctomb() und mbtowc(), die jedoch von der jeweiligen locale abhängig sind.

[4] Die Flexionformen und Verbflexionsarten sind noch von Änderungen ausgenommen, siehe ToDo Liste der Dokumentation.

name Bennung der Form: Verben/Adjektive werden nach ihrer Flexionsart benannt, grammatische Endungen nach ihrer Funktion, gibt es mehrere Varianten zusätzlich nummeriert

Position Level der Form, der die Abfolge der Formen bestimmt; im Programm werdem Formen mit der gleichen Positionsangabe in einem FlexLevel realisiert, 0 schließt nur an die Stammform an 7 Positionen werden unterschieden; Formen im gleichen Level können nicht verbunden werden

reqform im nächsttieferen Level obligatorische Flexionsform

reqflex legt fest, welche Flexionsart das Element in der nächsttieferen Ebene haben muss (vor allem, bei mehreren Realisierungen einer grammatischen Form; diese Anforderung wird gegen das Attribut 'form' von verbflex abgeglichen

verbflex die Flexionsart des Verbes/Adjektivs, hängt mit dem ELEMENT verbflex zusammen, folgende Flexionsformen werden unterschieden:

MZ *mizenkei* Indefinitform

OB *onbinkei* Assimilationsform

SR *suiryōkei* Dubitativform

SS *shūshikei* Finalform

RT *rentaikei* Attributivform

RY *renyōkei* Konjunktionalform

KT *kateikei* Konditionalform

MR *meireikei* Imperativform

Betrachtet man die DTD der Grammatikdatei, wird deutlich, welche Werte zwischen Stamm (dem Stamm näheren Suffix) und Suffix übereinstimmen müssen. Bei 6 Flexionsleveln könnte man also theoretisch an einen Stamm fünf Endungen agglutinieren können, im Sprachgebrauch kommt das aber in der Regel nicht vor.

2.3 Die Funktion zur Bestimmung eines Lemmas

Zunächst wird zwischen Hiragana- und Kanjistring unterschieden, denn der Kanjistring ist immer Teil des Stammes, nur Hiragana

Zeichenketten können flektieren.[5]. Was den Hiragana-String betrifft, wird für jeden möglichen Trennpunkt ein Objekt vom Typ 'Hypo-Lemma' angelegt. Zum Beispiel im Fall von '*yomanai*' ist '*yo*' ein Kanji, gehört also zum Stamm, für die Endung '*manai*' ergeben sich (bedingt durch die Silbenschrift) die Möglichkeiten in der Form 'Teil1/Teil2': '*manai/*', '*ma/nai*','*mana/i*' und '*/manai*'. Diese drei Hypothesen werden dann angefangen bei dem höchsten FlexLevel versucht zu verifizieren. In diesem Beispiel wird '*nai*' in Level 4 gefunden. Das bedeutet nun werden zwei neue Hypothesen angelegt, die davon ausgehen, dass '*nai*' abgearbeitet ist, es nun also nur noch um '*ma*' geht (welches Teil des Stammes oder Teil der Endung sein kann). Nun wird erneut eine Suche durch die verschiedenen FlexLevels gestartet, aber mit 6 Hypothesen und nur ab Level 3, da die Reihenfolge der grammatischen Endungen fix ist.

Die zwei hinzugefügten Hypothesen haben auch eine 'Geschichte', dass heisst es ist im HypoLemma-Objekt vermerkt, welche grammatische Form schon identifiziert wurde. Das ist insofern wichtig, als nur verschiedene Formen zum Teil in unterschiedlicher Weise angeschlossen werden. In dem XML Grammatikspezifikationen ist diese Eigenschaft in der 'reqform'-Element beziehungsweise in dem 'verbflex'-Attribut 'form=xy' realisiert.

Sind alle Hypothesen gegen die FlexLevel geprueft worden, werden sie durch die ihnen entsprechenden Finalform-Suffixe ergänzt (die Form SS in Level 0) und im Wörtertbuch nachgeschlagen. Auch hier wird darauf geachtet, dass nicht nur der Lemmastring sonder auch die Verflexionsart der Finalformendung mit dem Wörterbucheintrag übereinstimmt.

3 Das Ergebnis

3.1 Anwendungsbeispiel

Um möglichst unabhängig von der japanischen Locale zu sein, wurde als Ein- und Ausgabe Dateien im UNICODE (UCS2) Format angelegt. Werden keine Parameter übergeben, liest LeJa von ei-

[5]Hier wird von den Ausnahmefällen kuru (kommen) und suru (tun) abgesehen: da nur in diesen beiden Verben der Stamm sich verändert, können sie nicht nach den Regeln kodiert werden sondern müssen als Sonderfall hardcodiert werden, sind also für die Lemmatisierung nicht interessant.

ner Datei 'input.txt' Zeilenweise Vollformen von japanischer Verben und schreibt ihre Ausgabe in 'LeJa.txt'. Als erster Parameter kann aber auch ein Dateiname für das Einlesen einer ähnlichen Datei und optional als zweiter Parameter ein Dateiname für die Ausgabe angegeben werden. Der Lemmatisierungsprozess wird in der Datei 'LeJa.log' festgehalten. Zu Beginn der Datei wird vor allem auf die verschiedenen grammatischen Formen anhand eines Verbes getestet, das Ergebnis ist durchaus zufriedenstellend.

input.txt

```
読む
読まない
読まれる
読まれない
読ませる
読ませられる
読ませられない
読んだ
読んで
読まなかった
読まれなかった
読ませられなかった
読ませなかった
読まれた
読まれません
読まれました
読みたい
読みたくない
読ませたくない
読めば
考えられません
思い出せば
買いました
買いたかった
買いたくなかった
買わなかった
```

LeJa.txt

```
読む
Result: 読む
Reading: よむ
English: to read
German: lesen
Flexion: v5m
Form:
読まない
Result: 読む
Reading: よむ
English: to read
German: lesen
```

Flexion: v5m
Form: + Negation1
読まれる
Result: 読む
Reading: よむ
English: to read
German: lesen
Flexion: v5m
Form: + Passiv2
読まれない
Result: 読む
Reading: よむ
English: to read
German: lesen
Flexion: v5m
Form: + Negation1 + Passiv2
読ませる
Result: 読む
Reading: よむ
English: to read
German: lesen
Flexion: v5m
Form: + Faktitiv2
読ませられる
Result: 読む
Reading: よむ
English: to read
German: lesen
Flexion: v5m
Form: + Passiv1 + Faktitiv2
読ませられない
Result: 読む
Reading: よむ
English: to read
German: lesen
Flexion: v5m
Form: + Negation1 + Passiv1 + Faktitiv2
読んだ
Result: 読む
Reading: よむ
English: to read
German: lesen
Flexion: v5m
Form: + Past2
読んで
Result: 読む
Reading: よむ
English: to read
German: lesen
Flexion: v5m
Form: + TE-Form2
読まなかった
Result: 読む
Reading: よむ
English: to read
German: lesen
Flexion: v5m
Form: + Past1 + Negation1
読まれなかった

Result: 読む
Reading: よむ
English: to read
German: lesen
Flexion: v5m
Form: + Past1 + Negation1 + Passiv2
読ませられなかった
Result: 読む
Reading: よむ
English: to read
German: lesen
Flexion: v5m
Form: + Past1 + Negation1 + Passiv1 + Faktitiv2
読ませなかった
Result: 読む
Reading: よむ
English: to read
German: lesen
Flexion: v5m
Form: + Past1 + Negation1 + Faktitiv2
読まれた
Result: 読む
Reading: よむ
English: to read
German: lesen
Flexion: v5m
Form: + Past1 + Passiv2
読まれません
Result: 読む
Reading: よむ
English: to read
German: lesen
Flexion: v5m
Form: + Negation2 + Hoeflichkeit1 + Passiv2
読まれました
Result: 読む
Reading: よむ
English: to read
German: lesen
Flexion: v5m
Form: + Past1 + Hoeflichkeit1 + Passiv2
読みたい
Result: 読む
Reading: よむ
English: to read
German: lesen
Flexion: v5m
Form: + Intentionalis1
読みたくない
Result: 読む
Reading: よむ
English: to read
German: lesen
Flexion: v5m
Form: + Negation1 + Intentionalis1
読ませたくない
Result: 読む
Reading: よむ
English: to read

German: lesen
Flexion: v5m
Form: + Negation1 + Intentionalis1 + Faktitiv2
読めば
Result: 読む
Reading: よむ
English: to read
German: lesen
Flexion: v5m
Form: + Konditionalis1
考えられません
Result: 考える
Reading: かんがえる
English: to consider
German: denken, meinen, glauben, penser, considerer, reflechir, croire que
Flexion: v1
Form: + Negation2 + Hoeflichkeit1 + Passiv1
買いました
Result: 買う
Reading: かう
English: to buy
German: kaufen
Flexion: v5u
Form: + Past1 + Hoeflichkeit1
買いたかった
Result: 買う
Reading: かう
English: to buy
German: kaufen
Flexion: v5u
Form: + Past1 + Intentionalis1
買いたくなかった
Result: 買う
Reading: かう
English: to buy
German: kaufen
Flexion: v5u
Form: + Past1 + Negation1 + Intentionalis1
買わなかった
Result: 買う
Reading: かう
English: to buy
German: kaufen
Flexion: v5u
Form: + Past1 + Negation1

Weitere Beispiele befinden sich in dem Verzeichnis 'sample'.

3.2 Ausblick

Womit es vor prinzipiell Probleme gibt, sind zusammengesetzte Verben, wie zum Beispiel das in obigen Beispiel nicht gefundene *omoidasu*, welches sich aus einer Kombination von Kanji+Hiragana+Kanji-+Hiragana zusammensetzt ist, semantisch aber eine Einheit ergibt.

Da auf die gleiche Weise auch Verbkomposita gebildet werden können, die nicht lexikalisiert sind, muss man sich entweder auf das Input verlassen und davon ausgehen, der Stamm geht mindestens bis zum letzten Kanji, oder man bildet zwei Hypothesen, eine als zwei separate Verbformen und eine als zusammengesetztes Verb. Findet sich die zusammengesetzte Form nicht im Wörterbuch, kann man zumindest auf die Einzelbedeutung der beiden Komponenten zugreifen.

Auch ist offen, wie genau die Verben mit Sonderflexion, vor allem *kuru* (kommen) und *suru* (tun) sowie die als Kopula gehandelten Formen von *desu* zu behandeln sind. Aufgrund ihrer unregelmässigen Bildung und Häufigkeit würde es sich wahrscheinlich lohnen, sie nicht über Regeln abzuleiten, sondern sie in bei Bedarf in einer Tabelle nachzuschlagen.

Beim Wörterbuch-Lookup gibt es ausserdem Probleme an Stellen, an welchen Einträge für die deutsche Übersetzung nicht klar von denen für die französische Übersetzung abgegrenzt sind.