

Finite-State Transducer of Japanese Verb Inflection

1 Motivation

A Finite-State Transducer (FST) is a computational model that represents a finite automaton with both input and output symbols associated with its transitions. It is a type of finite-state machine that can perform transformations on input sequences, producing corresponding output sequences. Creating a Finite-State Transducer (FST) for Japanese verb inflection is important for several reasons, particularly in the field of natural language processing (NLP) and computational linguistics. I found it most interesting is that to develop Language Learning Tools for the Japanese language because FSTs can be used as educational tools for Japanese language learners which provide insights into the conjugation patterns of verbs, helping learners understand and practice the proper usage of verb forms.

To begin with, Japanese is an agglutinative language with complex verb conjugation patterns (Toru, 1994) which include an abundance of information in Japanese expression. An FST for Japanese verb inflection can be used for morphological analysis, breaking down verbs into their constituent morphemes and providing information about tense, mood, politeness level, and other grammatical features and therefore help learners to understand how the conjugation has made and comprehend the information that the verbs revealed. Here is how I put it into practice.

First, incorporate FSTs to create interactive exercises on the server terminal where learners can input a certain verb form, and the FST provides instant feedback, visualizing all inflections of the verb in lexical form. For example, the lexical form of the verb *to read* in past form (読んだ) should show in 読む+V+Cons+Plain+Past which indicates that the input is a consonant-stem verb in its past plain form. Also, learners can interact with these visualizations to explore the relationships between different verb forms and gain a clearer understanding of conjugation patterns by entering the lexical form of the verb.

By integrating FSTs into educational tools, learners can benefit from hands-on practice and immediate feedback which points to a more engaging and effective way of acquiring verb conjugation skills in the context of the Japanese language.

2 Preparation

Initially, I collected the general inflection rules of Japanese verbs from grammar textbooks and online material. The Japanese verbs have 3 groups which are regular, Sahen, and irregular. Because there are only 2 irregular verbs in Japanese which are する(to do) and 来る(to come), and する(to do) is also the base verb to compound with verbal nouns, etc. to form Sahen verb. So, there's no need to explain the irregular verbs in my transducer. Instead, I include consonant-stem verbs, vowel-stem verbs that belong to regular verbs, and Sahen verbs in it.

Move on to draw the networks of verb inflections with some examples, set the multiple characters, and write them into the lexc file. For consonant-stem verb, the verbs end with Hiragana ぶ, ぐ, く, む, ぬ, る, す, つ, う. For vowel-stem verbs, the verbs all end with

る. For Sanhen verb group, the verbs all end with する(to do). There is a question, how can I tell the verbs ending with る are consonant-stem verbs or vowel-stem verbs, or vice versa? Fortunately, there are less than 50 consonant-stem verbs that end with る in daily use which I will define in the lexc file. Also, there are special rules that consonant-stem verb has but vowel verbs and Sahen verbs don't have, the harmony phenomenon. All xfst rules set for consonant-stem verbs can be applied to both vowel-stem and Sahen verbs. For vowel-stem verbs and Sahen verbs, I depart the stem and their affix.

In the lexicon file, I collect 60 consonant-stem verbs, 10 vowel-stem verbs, and 10 Sahen verbs to my transducer from the JapanDict website which is available for checking if the outputs are correct in the Japanese dictionary.

3 Test

At the first test in the Puhti terminal with the hfst-xfst tool set, I didn't set rules in the xfst file to see whether the Japanese characters could be recognized or not. The test has passed. But as I wrote the verb with its stem form (e.g. 読む:読) in lexc file which cannot present the Japanese. As I switched the presentation of verbs into Roma characters (yomu:yo), the test passed. However, Roma characters are not Japanese but are used to represent Japanese phonemes. So, I was wondering if it's possible to set rules in xfst file so that Japanese characters can be transferred to Roma characters. After the verbs have been conjugated in the form of Roma characters, they will be inverted back to Japanese characters. But very soon I found out that I had no ideas how to converse the Japanese input to part of Japanese characters (stem) and part of Roma characters (affix). Therefore, the third test has failed.

In my fourth try, I combined the inflections to specific symbols, for example, the stem will change when it meets with the negative form “ない” but will not change when meets with the forbidden form “な”. Thus I set the negative form to “n い” in the lexc file and wrote the rule “n -> な” to conjugate with the negative stem in the xfst file. I am confident with the rules that are practical to harmony rules, too. So I set the rules of harmony for b/m/n consonant-stem verbs, etc. (introduced in my presentation) in xfst, for instance,

```
define NasalMopheme ぶ -> ん || _ t .o. む -> ん || _ t .o. ぬ -> ん || _ t ;
define NHarmony t -> だ ;
```

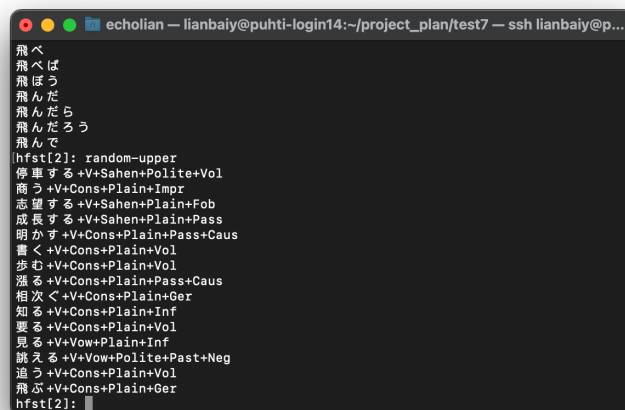
But only the nasal harmony triggered, the other two fail, the morpheme “だ” in i harmony and tone-boosting harmony are stay still. I try to figure it out by composing conditions, for example, the special symbol t in nasal harmony is recognized when is followed by “ん” which has reversed to its harmony form.

```
define NHarmony t -> だ || ん _ ;
```

After all tests have passed, the results are shown in the next section.

4 Result

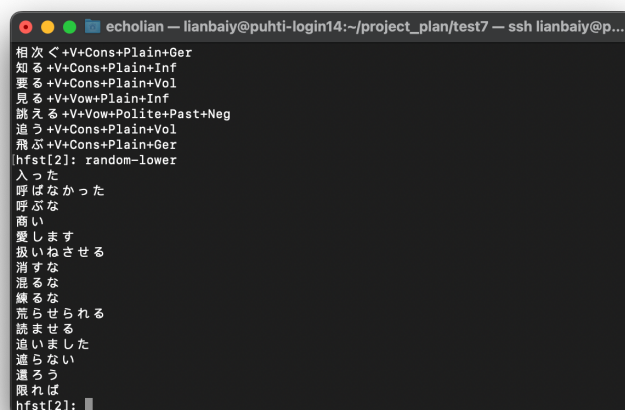
Command `random-upper`, see Figure 4.1



```
echolian — lianbaiy@puhti-login14:~/project_plan/test7 — ssh lianbaiy@p...
飛べ
飛べば
飛ぼう
飛んだ
飛んだら
飛んだろう
飛んで
hfst[2]: random-upper
停車する+V+Sahen+Polite+Vol
商う+V+Cons+Plain+Impr
忘望する+V+Sahen+Plain+Fob
成長する+V+Sahen+Plain+Pass
明かす+V+Cons+Plain+Pass+Caus
書く+V+Cons+Plain+Vol
歩む+V+Cons+Plain+Vol
通る+V+Cons+Plain+Pass+Caus
相次ぐ+V+Cons+Plain+Ger
知る+V+Cons+Plain+Inf
要る+V+Cons+Plain+Vol
見る+V+Vow+Plain+Inf
読める+V+Vow+Polite+Past+Neg
追う+V+Cons+Plain+Vol
飛ぶ+V+Cons+Plain+Ger
hfst[2]:
```

Figure 4.1

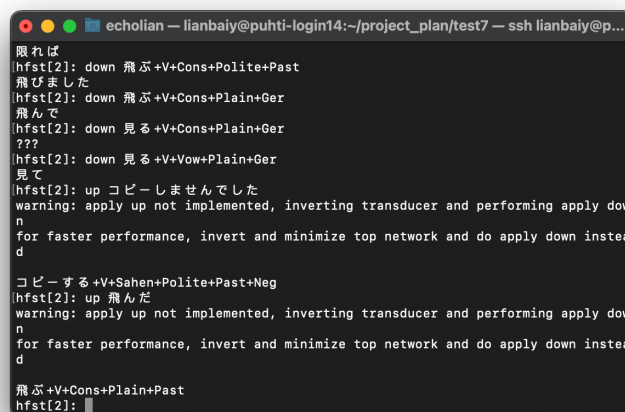
Command `random-lower`, see Figure 4.2



```
echolian — lianbaiy@puhti-login14:~/project_plan/test7 — ssh lianbaiy@p...
相次ぐ+V+Cons+Plain+Ger
知る+V+Cons+Plain+Inf
要る+V+Cons+Plain+Vol
見る+V+Vow+Plain+Inf
読める+V+Vow+Polite+Past+Neg
追う+V+Cons+Plain+Vol
飛ぶ+V+Cons+Plain+Ger
hfst[2]: random-lower
入った
呼ばなかった
呼ぶな
商い
愛します
扱いねさせる
消すな
知るな
驚かされる
読ませる
追いました
追わない
追ろう
限れば
hfst[2]:
```

Figure 4.2

See Figure 4.3, the transducer gives the correct output of the polite past form of consonant-stem, for example, the verb 飛ぶ (to fly). To check the plain gerund form of 飛ぶ (to fly), input down 飛ぶ+V+Cons+Polite+Past, and the transducer produces the correct output. If I accidentally input the lexicon form of the vowel-stem verb 見る (to see), the transducer will not recognize the word. To check the lexicon form of verb コピーしました (didn't copy), input up コピーませんでした with a correct output of コピーする+V+Sahen+Polite+Past+Neg. To check whether the nasal harmony has been applied, input the plain past form of the verb 飛んだ (flew). The output show it's a consonant-stem verb, in its plain past form.



```

echolian — lianbaiy@puhti-login14:~/project_plan/test7 — ssh lianbaiy@p...
限れば
[hfst[2]: down 飛ぶ+V+Cons+Polite+Past
飛びました
[hfst[2]: down 飛ぶ+V+Cons+Plain+Ger
飛んで
[hfst[2]: down 見る+V+Cons+Plain+Ger
???
[hfst[2]: down 見る+V+Vow+Plain+Ger
見て
[hfst[2]: up コピーしませんでした
warning: apply up not implemented, inverting transducer and performing apply down
for faster performance, invert and minimize top network and do apply down instead
コピーする+V+Sahen+Polite+Past+Neg
[hfst[2]: up 飛んだ
warning: apply up not implemented, inverting transducer and performing apply down
for faster performance, invert and minimize top network and do apply down instead
飛ぶ+V+Cons+Plain+Past
[hfst[2]: ]

```

Figure 4.3

The results prove that this FST is ready to apply in Japanese verb inflections analyzers. All the codes of lexicon and xfst files are attached in Attachment.

5 Reference

Toru Hisamitsu and Yoshihiko Nitta. 1994. An efficient treatment of Japanese verb inflection for morphological analysis. In Proceedings of the 15th conference on Computational linguistics - Volume 1 (COLING '94). Association for Computational Linguistics, USA, 194–200. doi.org/10.3115/991886.991919

Kuniya Nasukawa. No consonant-final stems in Japanese verb morphology. *Lingua*. Volume 120, Issue 10, 2010, Pages 2336-2352, ISSN 0024-3841. doi.org/10.1016/j.lingua.2010.03.026.

<https://www.japandict.com/>

Attachments

lexc file

Multichar_Symbols

+V	! Verbs tag
+Cons	! Consonant-stem verb
+Vow	! Vowel-stem verb
+Sahen	! Sahen verb
+Plain	! Dictionary form of verbs
	! 简体
+Polite	! Polite form is used to improve politeness tone in conversations
	! 礼貌体
+Past	! Past tense

+Neg ! Negative form
 +Presum ! Presumptive form of verbs
 ! 推断型
 +Vol ! Volitional form of verbs which means "let us~, shall we~"
 ! 意志型
 +Impr ! Imperative form of verbs
 ! 命令型
 ! This is rather impolite way to give command though.
 +Prov ! Provisional form of Hypothetical form
 ! 假定型之一, 暂时的, 接ば
 +Cond ! Conditional form of Hypothetical form
 ! 假定型之二, 条件的, 接たら
 ! has three other forms
 +Inf ! Infinitive form of Partical form which is use to conjugate with other words or the pause in sentence
 ! 接续之一, 复合名词时使用
 +Ger ! Gerund form of Partical form
 ! 接续之二, 变成て型
 +Alt ! Alinative form of Partical form
 ! 接续之三, 变成た型
 +Pass ! Passive form
 ! 被动式
 +Caus ! Causative form
 ! 使役型
 +Fob ! Fobiddent form
 ! 禁止型
 ! Imperative, Provisional, Conditional and Fobiddent form have neither tense nor polite form.

LEXICON Root

Verbs ; ! No input, no output

!

! Verbs start here

!

LEXICON Verbs

飛ぶ ConsV ; ! Consonant-stem verbs
 呼ぶ ConsV ;
 遊ぶ ConsV ;
 読む ConsV ;
 怪しむ ConsV ;
 赤らむ ConsV ;

歩む ConsV ;
上がり込む ConsV ;
死ぬ ConsV ;
書く ConsV ;
欺く ConsV ;
婀娜めく ConsV ;
暴く ConsV ;
相次ぐ ConsV ;
泳ぐ ConsV ;
嘲る ConsV ;
焦る ConsV ;
要る ConsV ;
居る ConsV ;
煎る ConsV ;
帰る ConsV ;
還る ConsV ;
限る ConsV ;
切る ConsV ;
覆る ConsV ;
蹴る ConsV ;
遮る ConsV ;
茂る ConsV ;
湿る ConsV ;
知る ConsV ;
滑る ConsV ;
散る ConsV ;
照る ConsV ;
握る ConsV ;
煉る ConsV ;
練る ConsV ;
罵る ConsV ;
煽る ConsV ;
入る ConsV ;
走る ConsV ;
減る ConsV ;
参る ConsV ;
混る ConsV ;
漲る ConsV ;
肖る ConsV ;
操る ConsV ;
立つ ConsV ;
開け放つ ConsV ;
追う ConsV ;
商う ConsV ;

嘲笑う ConsV ;
 扱う ConsV ;
 宛がう ConsV ;
 消す ConsV ;
 明かす ConsV ;
 暴き出す ConsV ;
 荒らす ConsV ;
 追い出す ConsV ;

見る:見 VowV ; ! Vowel-stem verbs
 着る:着 VowV ;
 崇める:崇め VowV ;
 開ける:開け VowV ;
 憧れる:憧れ VowV ;
 扱いねる:扱いね VowV ;
 逃える:逃え VowV ;
 当て付ける:当て付け VowV ;
 充てる:充て VowV ;
 暴れる:暴れ VowV ;
 有り触れる:有り触れ VowV ;

勉強する:勉強 SahenV ; ! San-hen verbs, a type of verbs
 販売する:販売 SahenV ;
 運動する:運動 SahenV ;
 コピーする:コピー SahenV ;
 愛する:愛 SahenV ;
 チェックする:チェック SahenV ;
 停車する:停車 SahenV ;
 志望する:志望 SahenV ;
 移動する:移動 SahenV ;
 イライラする:イライラ SahenV ;
 成長する:成長 SahenV ;

! Japanese verbs have three main groups
 ! Consonant-stem verbs and Vowel-stem verbs belong to first group
 ! They are the regular verbs
 ! The second group is Sahen verbs and irregular verbs
 ! There are two irregular verbs generally

LEXICON ConsV

+V:0 IndicateConsV ; ! can have tense and politeness inflection

LEXICON VowV

+V:0 IndicateVowV ;

LEXICON SahenV

+V:0 IndicateSahenV ;

LEXICON IndicateConsV

+Cons+Plain:0 PlainConsV ;

+Cons+Polite:ます PoliteConsV ;

LEXICON IndicateVowV

+Vow+Plain:0 PlainVowV ;

+Vow+Polite:ます PoliteVowV ;

LEXICON IndicateSahenV

+Sahen+Plain:0 PlainSahenV ;

+Sahen+Polite:します PoliteSahenV ;

LEXICON PlainConsV

+Past:t # ;

+Neg:n い # ;

+Past+Neg:n かった # ;

+Presum:t ろう # ;

+Vol:m う # ;

+Impr:e # ;

+Prov:r ば # ; ! Hypothetical

+Cond:t ら # ; ! Hypothetical

+Inf:I # ; ! Partical

+Ger:G # ; ! Partical

+Alt:t ら # ; ! Partical

+Pass:s る # ;

+Caus:せる # ;

+Pass+Caus:せられる # ; ! Passive-Cauasive form

+Pass+Neg:s ない # ;

+Fob:な # ;

LEXICON PoliteConsV

+Past:した # ;

+Neg:せん # ;

+Past+Neg:せんでした # ;

+Vol:しょう # ;

LEXICON PlainVowV

+Past:た # ;

+Neg:ない # ;

+Past+Neg:なかった # ;

+Presum:たろう # ;
+Vol:よう # ;
+Impr:ろ # ;
+Prov:れば # ;
+Cond:たら # ;
+Inf:0 # ;
+Ger:て # ;
+Alt:たら # ;
+Pass:られる # ;
+Caus:させる # ;
+Pass+Caus:させられる # ;
+Pass+Neg:られない # ;
+Fob:な # ;

LEXICON PoliteVowV

+Past:した # ;
+Neg:せん # ;
+Past+Neg:せんでした # ;
+Vol:しょう # ;

LEXICON PlainSahenV

+Past:した # ;
+Neg:しない # ;
+Past+Neg:しなかった # ;
+Presum:せよ # ;
+Vol:しょう # ;
+Impr:しろ # ;
+Prov:すれば # ;
+Cond:したら # ;
+Inf:し # ;
+Ger:して # ;
+Alt:したら # ;
+Pass:される # ;
+Caus:させる # ;
+Pass+Caus:させられる # ;
+Pass+Neg:されない # ;
+Fob:するな # ;

LEXICON PoliteSahenV

+Past:した # ;
+Neg:せん # ;
+Past+Neg:せんでした # ;
+Vol:しょう # ;

```
# ;
```

```
END
```

```
xfst file
```

```
! Clear away any possible old data from the system
```

```
clear stack
```

```
! Read lexicon and make a regex of it
```

```
read lexc jp_verb.lexc
```

```
define Lexicon ;
```

```
regex Lexicon ;
```

```
! stem of consonant-stem verb
```

```
define Bcons ぶ ;
```

```
define Mcons む ;
```

```
define Ncons ぬ ;
```

```
define Kcons く ;
```

```
define Gcons ぐ ;
```

```
define Rcons る ;
```

```
define Tcons つ ;
```

```
define Wcons う ;
```

```
define Sconc す ;
```

```
! reverse plain form to polite form
```

```
define PlainToPolite ぶ->び || _ま.ο. む->み || _ま.ο. ぬ->に || _ま.ο.
```

```
く->き || _ま.ο. ぐ->ぎ || _ま.ο. る->り || _ま.ο. つ->ち || _ま.ο.
```

```
う->い || _ま.ο. す->し || _ま ;
```

```
define PolitePast す->0 || _[せ | し] ;
```

```
! negative form of plain form
```

```
define Neg ぶ->ば || _n.ο. む->ま || _n.ο. ぬ->な || _n.ο. く->か ||
```

```
_n.ο. ぐ->が || _n.ο. る->ら || _n.ο. つ->た || _n.ο. う->わ || _
```

```
n.ο. す->さ || _n ;
```

```
define MorphemeNeg n->な ;
```

```
! volitional form, but the verb is the only verb ends with ぬ, 死ぬ (to die)
which has no volitional form
```

```
define Vol ぶ->ぼ || _m.ο. む->も || _m.ο. く->こ || _m.ο. ぐ->ご ||
```

```
_m.ο. る->ろ || _m.ο. つ->と || _m.ο. う->お || _m.ο. す->そ || _
```

```
m ;
```

```
define MorphemeVol m->0 ;
```

```

! imperative form
define Impr ぶ->べ || _ e .o. む->め || _ e .o. ぬ->ね || _ e .o. く->け
|| _ e .o. ぐ->げ || _ e .o. る->れ || _ e .o. つ->て || _ e .o. う->え ||
_e .o. す->せ || _ e ;
define MorphemeImpr e -> 0 ;

! provisional form
define Prov ぶ->べ || _ r .o. む->め || _ r .o. ぬ->ね || _ r .o. く->け
|| _ r .o. ぐ->げ || _ r .o. る->れ || _ r .o. つ->て || _ r .o. う->え ||
_r .o. す->せ || _ r ;
define MorphemeProv r -> 0 ;

! infinitive form
define Inf ぶ->び || _ I .o. む->み || _ I .o. ぬ->に || _ I .o. く->き ||
_I .o. ぐ->ぎ || _ I .o. る->り || _ I .o. つ->ち || _ I .o. う->い || _
I .o. す->し || _ I ;
define MorphemeInf I -> 0 ;

! passive form
define Pass ぶ->ば || _ s .o. む->ま || _ s .o. ぬ->な || _ s .o. く->か
|| _ s .o. ぐ->が || _ s .o. る->ら || _ s .o. つ->た || _ s .o. う->わ ||
_s .o. す->さ || _ s ;
define MorphemePass s -> れ ;

! causative and passive-causative form
define PassCaus ぶ->ば || _ se .o. む->ま || _ se .o. ぬ->な || _ se .o. く
->か || _ se .o. ぐ->が || _ se .o. る->ら || _ se .o. つ->た || _ se .o. う->
わ || _ se .o. す->さ || _ se ;

! harmony rules
! nasal harmony
define NasalMopheme ぶ->ん || _ t .o. む->ん || _ t .o. ぬ->ん || _ t ;
define NHarmony t -> だ || ん _ ;
define NasalGer ぶ->ん || _ G .o. む->ん || _ G .o. ぬ->ん || _ G ;
define MorphemeNasalGer G -> で || ん _ ;
define NasalHarmony NasalMopheme .o. NHarmony .o. NasalGer .o.
MorphemeNasalGer ;

! i harmony for k-stem verb
define kiMorpheme く->い || _ t ;
define kHarmony t -> た || い _ ;
define kGer く->い || _ G ;
define MorphemeKGer G -> て || い _ ;

```

```

define kiHarmony kiMorpheme .o. kHarmony .o. kGer .o. MorphemekGer ;

! i harmony for g-stem verb
define giMorpheme <` -> い || _ t ;
define gHarmony t -> で || い _ ;
define gGer <` -> い || _ G ;
define MorphemegGer G -> で || い _ ;
define giHarmony giMorpheme .o. gHarmony .o. gGer .o. MorphemegGer ;

! tone-boosting harmony
define ToneMorpheme る -> っ || _ t .o. っ -> っ || _ t .o. う -> っ || _ t ;
define THarmony t -> た || っ _ ;
define ToneGer る -> っ || _ G .o. っ -> っ || _ G .o. う -> っ || _ G ;
define MorphemeToneGer G -> て || っ _ ;
define ToneBoHarmony ToneMorpheme .o. THarmony .o. ToneGer .o. MorphemeToneGer ;

! tMorpheme and Gerund form for s-stem verb
define STVerb す -> し || _ [ t | G ] ;
define MorphemeScons t -> た || し _ .o. G -> て || し _ ;
define SconsVerb STVerb .o. MorphemeScons ;

define ConsVerbHarmony ToneBoHarmony .o. NasalHarmony .o. kiHarmony .o.
giHarmony .o. SconsVerb ;

define Rules Lexicon .o. PlainToPolite .o. PolitePast .o. ConsVerbHarmony .o.
Neg .o. MorphemeNeg .o. Vol .o. MorphemeVol .o. Impr .o. MorphemeImpr .o.
Prov .o. MorphemeProv .o. Inf .o. MorphemeInf .o. Pass .o. MorphemegPass .o.
PassCaus ;

regex Rules ;

! Output the upper form
upper-words

! Output the lower form
lower-words

```