Resolving ambiguity in merged English verb forms

Hurskainen, Arvi

University of Helsinki, Institute for Asian and African Studies

2020

Hurskainen , A 2020 ' Resolving ambiguity in merged English verb forms ' Technical Reports on Language Technology , no. 55 , University of Helsinki, Institute for Asian and African Studies , Helsinki . <
http://www.njas.helsinki.fi/salama/resolving-ambiguity-in-merged-english-verb-forms.pdf >

http://hdl.handle.net/10138/317508

Technical Reports on Language Technology
Report No 55, 2020
http://www.njas.helsinki.fi/salama

Resolving ambiguity in merged English verb forms

Arvi Hurskainen
Department of Languages, Box 59
FIN-00014 University of Helsinki, Finland
arvi.hurskainen@helsinki.fi

Abstract

In English writing, it is customary to merge the pronoun subject and the following auxiliary verb as a single string, pointing the abbreviated part with an apostrophe. Part of these structures can be spelled out unambiguously. However, there are structures, where the verb can be either be or have. In other structures, the verb can be will, shall, would, could, or have. The problem can be solved only on the basis of context. There are also genitive structures, which look very much the same as the verb structures. These must be kept separate and treated accordingly. The report discusses these problems.

Key Words: *ambiguity, morphological analysis, disambiguation.*

1 Introduction

Resolving ambiguity in English text is a major challenge. Inflection of words is limited, whereby a single wordform may have several interpretations. It is common that a wordform can be a verb, a noun, or an adjective. Resolving such ambiguities is normal work in rule-based language technology.

In addition, there are also such structures, where two words are merged into a single string. Many of these are ambiguous, and the ambiguity can be solved only on the basis of context.

Also, such genitive structures, which are formed by adding a suffix 's, must be identified as genitives, and they should not be mixed with other similar structures.

In brief, we discuss here the following three problem types, (a) such merged combinations of a pronoun and an auxiliary verb that are ambiguous, (b) genitive constructions formed with the suffix *'s*, and (c) such merged combinations of a pronoun and an auxiliary verb that are not ambiguous.

2 Ambiguous merged combinations of a pronoun and an auxiliary verb and genitive constructions

English has two methods of forming genitive constructions. The method of using the particle *of* applies to almost all cases. In addition, there is also the method of using the genitive suffix *s*, which in writing is connected to the word using the apostrophe *'*. This diacritic is, however, used also in merged word constructions. These two roles of the

apostrophe must be kept strictly apart. How can this be done, especially when in both types there is the sequence *'s* at the end of the word?

We propose that the problem is handled in two phases. In the first phase, such structures are identified, where the sequence *'s* is part of the merged words. These constructions include the forms such as *that's*, *who's*, *what's*, *where's*, *there's*, *here's*, *it's*, *he's*, *she's*, and *whoever's*.

The method of keeping these structures separate from genitive structures is that we first detach the parts as in (1).

(1)
that's > that 's
who's > who 's
what's > what 's
where's > where 's
there's > there 's
it's > it 's
he's > he 's
she's > she 's
whoever's > whoever 's

Now the sequence 's is interpreted as an ambiguous verb with two values, *be* and *have*. The Perl rules that implement the separation are in (2).

(2)
```
perl -pe "s/(T|t)hat's /\1hat 's /gm" | \
perl -pe "s/(W|w)ho's /\1ho 's /gm" | \
perl -pe "s/(W|w)hat's /\1hat 's /gm" | \
perl -pe "s/(W|w)here's /\1here 's /gm" | \
perl -pe "s/(T|t)here's /\1here 's /gm" | \
perl -pe "s/(H|h)ere's /\1ere 's /gm" | \
perl -pe "s/(I|i)t's /\1t 's /gm" | \
perl -pe "s/(H|h)e's /\1e 's /gm" | \
perl -pe "s/(S|s)he's /\1he 's /gm" | \
perl -pe "s/(W|w)hoever's /\1hoever 's /gm"
```

When we analyse these structures, we get the result as in (3).

(3)
```
"<that>"
      "that" CONJ REL
      "that" CONJ
      "that" PRON DEM SG
"<'s>"
      "be" V PRES SG3
      "have" V PRES SG3
"<who>"
      "who" PRON REL
      "who" QUEST WH
```

```
"<'s>"
      "be" V PRES SG3
      "have" V PRES SG3
"<what>"
      "what" PRON REL
      "what" QUEST WH
"<'s>"
      "be" V PRES SG3
      "have" V PRES SG3
"<where>"
      "where" PRON REL
      "where" ADV
      "where" QUEST WH
"<'s>"
      "be" V PRES SG3
      "have" V PRES SG3
"<there>"
      "there" ADV
"<'s>"
      "be" V PRES SG3
      "have" V PRES SG3
"<it>"
      "it" PRON SG3
      "it" PRON SG3 ACC
"<'s>"
      "be" V PRES SG3
      "have" V PRES SG3
"<he>"
      "he" PRON MALE SG3
"<'s>"
      "be" V PRES SG3
      "have" V PRES SG3
"<she>"
      "she" PRON FEM SG3
"<'s>"
      "be" V PRES SG3
      "have" V PRES SG3
"<whoever>"
      "whoever" ADV
"<'s>"
      "be" V PRES SG3
      "have" V PRES SG3
```

We see that in each case the sequence *'s* has two readings, *be* and *have*. Both have the tags V, PRES, and SG3.
When this reading is directed to the disambiguator, it is easy to select either of the interpretations on the basis of context.

In the pre-processing phase, only the structures listed above are detached as separate words. All other similar structures are left as such and interpreted as genitive

constructions. In (4) is an example of both types of structures. The sentence is: It's a pity that Tim's and Ali's team didn't win.

(4)

```
"<*it>"
      "it"  CAP PRON SG3
      "it"  CAP PRON SG3 ACC
"<'s>"
      "be"  V PRES SG3
      "have"  V PRES SG3
"<a>"
      "a"  DET INDEF
"<pity>"
      "pity"  N  SG
"<that>"
      "that"  CONJ REL
      "that"  CONJ
      "that"  PRON DEM SG
"<*tim's>"
      "tim"  PROPN  SG GEN
"<and>"
      "and"  CONJ CC
"<*ali's>"
"<team>"
      "team"  V vi INF/IMP
      "team"  V vi PRES SG1
      "team"  V vi PRES SG2/PL2
      "team"  V vi PRES PL1/PL3
      "team"  N  SG
"<did>"
      "do"  V PAST
      "do"  V AUXV PAST
"<not>"
      "not"  NEG
"<win>"
      "win"  V  INF/IMP
      "win"  V  PRES SG1
      "win"  V  PRES SG2/PL2
      "win"  V  PRES PL1/PL3
"<.>"
      "."  **CLB
```

The sequence *It's* was interpreted as two words, and the sequence *Tim's* was interpreted as one word and proper name with genitive suffix. On the other hand, the word *Ali's* was not found in the lexicon, and no analysis was given. We also notice that the sequence *didn't* was interpreted as two words. We are not, however, concerned about it here, because it is not ambiguous. We will deal with such structures later.

When we run the guessing module, which tries to guess the interpretation of the unknown word, we get the result as in (5).

(5)

```
"<*it>"
      "it"  CAP PRON SG3
      "it"  CAP PRON SG3 ACC
"<'s>"
      "be"  V PRES SG3
      "have"  V PRES SG3
"<a>"
      "a"  DET INDEF
"<pity>"
      "pity"  N  SG
"<that>"
      "that"  CONJ REL
      "that"  CONJ
      "that"  PRON DEM SG
"<*tim's>"
      "tim"  PROPN  SG GEN
"<and>"
      "and"  CONJ CC
"<*ali's>"
      "*ali's" Heur N
"<team>"
      "team"  V vi INF/IMP
      "team"  V vi PRES SG1
      "team"  V vi PRES SG2/PL2
      "team"  V vi PRES PL1/PL3
      "team"  N  SG
"<did>"
      "do"  V PAST
      "do"  V AUXV PAST
"<not>"
      "not"  NEG
"<win>"
      "win"  V  INF/IMP
      "win"  V  PRES SG1
      "win"  V  PRES SG2/PL2
      "win"  V  PRES PL1/PL3
"<.>"
      "."  **CLB
```

We see that the word *Ali's* is guessed and glossed as Heur N. It is not a bad guess but not perfect. We can continue processing by using the available information. Because the word has a capital initial letter, it is likely to be a proper name. However, this is not sure, because in the beginning of the sentence, every word starts with capital initial. We should check whether the word is inside the sentence, before we can make the decision. This cannot be done until we continue to disambiguation. Therefore, we must give the word two readings, one for proper name and another one for ordinary noun. This can be done using a Perl rule (6).

(6)
```
perl -pe "s/^(\t(.)\*(\S+))\'s. Heur N\n/\1\2 PROPN SG
GEN\n\t\2\3\2 CAP N SG GEN\n/gm"
```

When we apply this rule, we get the result as in (7).

(7)
```
"<*it>"
      "it"  CAP PRON SG3
      "it"  CAP PRON SG3 ACC
"<'s>"
      "be"  V PRES SG3
      "have"  V PRES SG3
"<a>"
      "a"  DET INDEF
"<pity>"
      "pity"  N  SG
"<that>"
      "that"  CONJ REL
      "that"  CONJ
      "that"  PRON DEM SG
"<*tim's>"
      "tim"  PROPN  SG GEN
"<and>"
      "and"  CONJ CC
"<*ali's>"
      "*ali" PROPN SG GEN
      "ali" CAP N SG GEN
"<team>"
      "team"  V vi INF/IMP
      "team"  V vi PRES SG1
      "team"  V vi PRES SG2/PL2
      "team"  V vi PRES PL1/PL3
      "team"  N  SG
"<did>"
      "do"  V PAST
      "do"  V AUXV PAST
"<not>"
      "not"  NEG
"<win>"
      "win"  V  INF/IMP
      "win"  V  PRES SG1
      "win"  V  PRES SG2/PL2
      "win"  V  PRES PL1/PL3
"<.>"
      "."  **CLB
```

We see that now the string *Ali's* has two interpretations, one for the proper name and another one for the ordinary noun. This ambiguous reading can then be disambiguated in the normal disambiguation process.

We saw above that the genitive constructions and the similar merged verb constructions were kept strictly apart in the pre-processing phase and analysis. The morphological analyser allows the genitive suffix *'s* to be attached only to nouns and proper names.

3 Merged combinations of a pronoun and an auxiliary verb that are ambiguous but dissimilar with genitive constructions

There are also other types of merged constructions that are likely unambiguous. Consider such structures as *he'll*, *we'll*, *I'll*, *you'll*, *they'll*, *that'll*, *there'll*, *it'll*. The verb in these can be *will* or *shall*.

Another group of ambiguous structures includes combinations such as *he'd*, *she'd*, *how'd*, *it'd*, *they'd*, *we'd*, *who'd*, and *you'd*. These constructions may hide such verbs as *would* or.

The verbs that are merged with the pronoun are actually will and shall, and their conditional forms would and should. Each of these four forms have a separate meaning, which is important to know in translation.

We handle these merged forms in the similar way as we did with the suffix 's. First, we separate them into two words, using Perl rules as in (8).

(8)
```
perl -pe "s/(I)'d /\1 'd /gm" | \
perl -pe "s/(H|h)ow'd /\1ow 'd /gm" | \
perl -pe "s/(H|h)e'd /\1e 'd /gm" | \
perl -pe "s/(I|i)t'd /\1t 'd /gm" | \
perl -pe "s/(S|s)he'd /\1he 'd /gm" | \
perl -pe "s/(T|t)hey'd /\1hey 'd /gm" | \
perl -pe "s/(W|w)e'd /\1e 'd /gm" | \
perl -pe "s/(W|w)ho'd /\1ho 'd /gm" | \
perl -pe "s/(H|h)e'll /\1e 'll /gm" | \
perl -pe "s/(I|i)t'll /\1t 'll /gm" | \
perl -pe "s/(S|s)he'll /\1he 'll /gm" | \
perl -pe "s/(T|t)hat'll /\1hat 'll /gm" | \
perl -pe "s/(T|t)here'll /\1here 'll /gm" | \
perl -pe "s/(T|t)hey'll /\1hey 'll /gm" | \
perl -pe "s/(W|w)e'll /\1e 'll /gm" | \
perl -pe "s/(Y|y)ou'll /\1ou 'll /gm"
```

In the analysis system, we give two interpretations for the abbreviated verb. We test this with the sentences *How'd you like to meet?* and *How'd I decide in this case?* The analysis is in (9).

(9)
```
"<*how>"
      "how" PRON CAP REL
      "how" QUEST CAP
"<'d>"
      "would" AUXV COND
```

```
      "should" AUXV COND
      "have" V PAST
"<you>"
      "you" PRON SG2/PL2
      "you" PRON SG2/PL2 ACC
"<like>"
      "like" V vt INF
      "like" V vt IMP
      "like" V vt PRES SG1
      "like" V vt PRES SG2/PL2
      "like" V vt PRES PL1
      "like" V vt PRES PL3
      "like" ADV PREFR
"<to>"
      "to" PREP
      "to" INFMARK
"<meet>"
      "meet" V INF
      "meet" V IMP
      "meet" V PRES SG1
      "meet" V PRES SG2/PL2
      "meet" V PRES PL1
      "meet" V PRES PL3
      "meet" N SG
      "meet" A
"<?>"
      "?" QUESTION-MARK
"<*how>"
      "how" PRON CAP REL
      "how" QUEST CAP
"<'d>"
      "would" AUXV COND
      "should" AUXV COND
      "have" V PAST
"<*i>"
      "i" PRON CAP SG1
      "*i" NUM-ROM
"<decide>"
      "decide" V vt vi INF
      "decide" V vt vi IMP
      "decide" V vt vi PRES SG1
      "decide" V vt vi PRES SG2/PL2
      "decide" V vt vi PRES PL1
      "decide" V vt vi PRES PL3
"<in>"
      "in" PREP
      "in" ADV
"<this>"
      "this" PRON DEM SG
      "this" DET
```

```
"<case>"
      "case" N SG
"<?>"
      "?" QUESTION-MARK
```

The string *'d* has now three interpretations, *would*, *should*, and *have*, and we should disambiguate between them. The problem is not easy to solve, because the form of the main verb in the first two cases is infinitive. The solution requires careful consideration of the context, and it is a difficult task. There are cases, where it is impossible to know which of the two verbs is more appropriate. It is likely that the verb *would* be the more correct interpretation. The third case, *have*, is easier, because the main verb is in participial form.

An example of the base form of the verb is in (10).

(10)

```
"<*it>"
      "it" PRON CAP SG3
      "it" PRON CAP SG3 ACC
"<'ll>"
      "will" AUXV FUT
      "shall" AUXV
"<be>"
      "be" AUXV INF
      "be" AUXV PRES PL
"<a>"
      "a" DET INDEF
"<long>"
      "long" V INF
      "long" V IMP
      "long" V PRES SG1
      "long" V PRES SG2/PL2
      "long" V PRES PL1
      "long" V PRES PL3
      "long" A
"<and>"
      "and" CONJ CC
"<hard>"
      "hard" A
"<path>"
      "path" N SG
"<.>"
      "." **CLB
```

In this sentence, the more obvious choice is *will*, but the other choice cannot be excluded categorically.

4 Merged combinations that are not ambiguous

There is a mixed group of merged constructions, which have no ambiguity. These structures are described in (11) in the form of Perl rules.

(11)
```
perl -pe "s/I'm /I am /gm" | \
perl -pe "s/(A|a)in't /\1re not /gm" | \
perl -pe "s/(A|a)ren't /\1re not /gm" | \
perl -pe "s/(C|c)an't /\1an not /gm" | \
perl -pe "s/(D|d)idn't /\1id not /gm" | \
perl -pe "s/(D|d)id'nt /\1id not /gm" | \
perl -pe "s/(D|d)on't /\1o not /gm" | \
perl -pe "s/(D|d)oesn't /\1oes not /gm" | \
perl -pe "s/(H|h)adn't /\1ad not /gm" | \
perl -pe "s/(H|h)asn't /\1as not /gm" | \
perl -pe "s/(H|h)aven't /\1ave not /gm" | \
perl -pe "s/(I|i)sn't /\1s not /gm" | \
perl -pe "s/(L|l)et's /\1et us /gm" | \
perl -pe "s/(S|s)houldn't /\1hould not /gm" | \
perl -pe "s/(T|t)hey're /\1hey are /gm" | \
perl -pe "s/(I)'ve /\1 have /gm" | \
perl -pe "s/(T|t)hey've /\1hey have /gm" | \
perl -pe "s/(W|w)asn't /\1as not /gm" | \
perl -pe "s/(W|w)e're /\1e were /gm" | \
perl -pe "s/(W|w)e've /\1e have /gm" | \
perl -pe "s/(W|w)eren't /\1ere not /gm" | \
perl -pe "s/(W|w)ho've /\1ho have /gm" | \
perl -pe "s/(W|w)on't /\1ill not /gm" | \
perl -pe "s/(W|w)ould've /\1ould have /gm" | \
perl -pe "s/(W|w)ouldn't /\1ould not /gm" | \
perl -pe "s/(C|c)ouldn't /\1ould not /gm" | \
perl -pe "s/(Y|y)ou'd /\1ou had /gm" | \
perl -pe "s/(Y|y)ou're /\1ou are /gm" | \
perl -pe "s/(Y|y)ou've /\1ou have /gm"
```

We only need to rewrite the merged structure into proper words in the pre-processing phase. The words can then be handled in normal way.

5 Disambiguating merged verb forms

The detailed inspection of wordforms that include an apostrophe reveals four different types of cases.
(a) There are genitives formed with the suffix *'s*. These structures are kept as such, without detaching the suffix.
(b) There are structures, which contain a pronoun and the auxiliary verb *be* or *have*. These can be disambiguated on the basis of the form of the main verb.

(c) There are structures, where a pronoun or adverb is merged with one of the verb forms *will*, *shall*, *would*, *should,* or *have*. The disambiguation of these forms is challenging, because no morphological information can be used as criteria.
(d) There is a mixed group of structures, which can be written out directly without disambiguation.

We can conclude that the structures of group (b) and (c) need disambiguation. The disambiguation process itself is carried out as part of the normal morphological disambiguation. In (12) there are examples of the type (c) sentences, where the merged structure ends in *'d*.

(12)

```
"<*he>"
      "he" PRON CAP MALE SG3
"<'d>"
      "would" AUXV COND
      "should" AUXV COND
      "have" V PAST
"<survived>"
      "survive" V vt vi PAST
      "survive" V vt vi EN
      "survived" A
"<.>"
      "." **CLB
"<*he>"
      "he" PRON CAP MALE SG3
"<'d>"
      "would" AUXV COND
      "should" AUXV COND
      "have" V PAST
"<always>"
      "always" ADV
"<be>"
      "be" AUXV INF
      "be" AUXV PRES PL
"<there>"
      "there" ADV
"<.>"
      "." **CLB
"<*they>"
      "they" PRON CAP PL3
"<'d>"
      "would" AUXV COND
      "should" AUXV COND
      "have" V PAST
"<had>"
      "have" AUXV PAST
      "have" AUXV EN
"<problems>"
      "problem" N PL
"<.>"
```

```
      "." **CLB
"<*she>"
      "she" PRON CAP FEM SG3
"<'d>"
      "would" AUXV COND
      "should" AUXV COND
      "have" V PAST
"<be>"
      "be" AUXV INF
      "be" AUXV PRES PL
"<interested>"
      "interest" V PAST
      "interest" V EN
      "interested" A
"<in>"
      "in" PREP
      "in" ADV
"<having>"
      "have" AUXV ING
      "having" V N
"<the>"
      "the" DET DEF
"<job>"
      "job" N SG
"<.>"
      "." **CLB
```

We see that the detached string *'d* has three interpretations, *would*, *should,* and *have*. The last one is easy to disambiguate, because it requires a participial form of the main verb. The first two are more problematic, because the sentence structure in both is the same. The practical solution is to keep the form *would* as a default and write rules only for such cases, which require that the form *should* should be selected. After disambiguation, the result is as in (13).

(13)

```
"<*he>"
      "he" PRON CAP MALE SG3
"<'d>"
      "have" V PAST
"<survived>"
      "survive" V vt vi EN
"<.>"
      "." **CLB
"<*he>"
      "he" PRON CAP MALE SG3
"<'d>"
      "would" AUXV COND
      "should" AUXV COND
      "have" V PAST
```

```
"<always>"
      "always" ADV
"<be>"
      "be" AUXV INF
      "be" AUXV PRES PL
"<there>"
      "there" ADV
"<.>"
      "." **CLB
"<*they>"
      "they" PRON CAP PL3
"<'d>"
      "have" V PAST
"<had>"
      "have" AUXV EN
"<problems>"
      "problem" N PL
"<.>"
      "." **CLB
"<*she>"
      "she" PRON CAP FEM SG3
"<'d>"
      "would" AUXV COND
      "should" AUXV COND
"<be>"
      "be" AUXV INF
"<interested>"
      "interest" V EN
"<in>"
      "in" PREP
"<having>"
      "have" AUXV ING
      "having" V N
"<the>"
      "the" DET DEF
"<job>"
      "job" N SG
"<.>"
      "." **CLB
```

The disambiguation has now been performed. In cohorts with more than one reading, the first one is the default choice. The clean disambiguated result is in (14).

(14)

```
"<*he>"
      "he" PRON CAP MALE SG3
"<'d>"
      "have" V PAST
"<survived>"
      "survive" V vt vi EN
"<.>"
```

```
      "." **CLB
"<*he>"
      "he" PRON CAP MALE SG3
"<'d>"
      "would" AUXV COND
"<always>"
      "always" ADV
"<be>"
      "be" AUXV INF
"<there>"
      "there" ADV
"<.>"
      "." **CLB
"<*they>"
      "they" PRON CAP PL3
"<'d>"
      "have" V PAST
"<had>"
      "have" AUXV EN
"<problems>"
      "problem" N PL
"<.>"
      "." **CLB
"<*she>"
      "she" PRON CAP FEM SG3
"<'d>"
      "would" AUXV COND
"<be>"
      "be" AUXV INF
"<interested>"
      "interest" V EN
"<in>"
      "in" PREP
"<having>"
      "have" AUXV ING
"<the>"
      "the" DET DEF
"<job>"
      "job" N SG
"<.>"
      "." **CLB
```

The case with the string *'ll* has only two readings, *will* and *shall*. They require an identical sentence structure, which makes disambiguation difficult. In (15) are sentences with this structure.

(15)

```
"<*i>"
      "i" PRON CAP SG1
      "*i" NUM-ROM
"<'ll>"
```

```
      "will" AUXV FUT
      "shall" AUXV
"<do>"
      "do" V INF
      "do" V IMP
      "do" V PRES SG1
      "do" V PRES SG2/PL2
      "do" V PRES PL1
      "do" V PRES PL3
      "do" AUXV INF
      "do" AUXV IMP
      "do" AUXV PRES SG1
      "do" AUXV PRES SG2/PL2
      "do" AUXV PRES PL1
      "do" AUXV PRES PL3
"<it>"
      "it" PRON SG3
      "it" PRON SG3 ACC
"<on>"
      "on" PREP
      "on" ADV
"<my>"
      "i" PRON SG1 GEN
"<own>"
      "own" V vt vi INF
      "own" V vt vi IMP
      "own" V vt vi PRES SG1
      "own" V vt vi PRES SG2/PL2
      "own" V vt vi PRES PL1
      "own" V vt vi PRES PL3
      "own" A
"<time>"
      "time" V vt INF
      "time" V vt IMP
      "time" V vt PRES SG1
      "time" V vt PRES SG2/PL2
      "time" V vt PRES PL1
      "time" V vt PRES PL3
      "time" N PREFR SG
"<.>"
      "." **CLB
"<*it>"
      "it" PRON CAP SG3
      "it" PRON CAP SG3 ACC
"<'ll>"
      "will" AUXV FUT
      "shall" AUXV
"<be>"
      "be" AUXV INF
      "be" AUXV PRES PL
```

```
"<a>"
      "a" DET INDEF
"<long>"
      "long" V INF
      "long" V IMP
      "long" V PRES SG1
      "long" V PRES SG2/PL2
      "long" V PRES PL1
      "long" V PRES PL3
      "long" A
"<path>"
      "path" N SG
"<.>"
      "." **CLB
"<*we>"
      "we" PRON CAP PL1
"<'ll>"
      "will" AUXV FUT
      "shall" AUXV
"<see>"
      "see" V INF
      "see" V IMP
      "see" V PRES SG1
      "see" V PRES SG2/PL2
      "see" V PRES PL1
      "see" V PRES PL3
      "see" N SG
"<how>"
      "how" PRON REL
      "how" QUEST
"<this>"
      "this" PRON DEM SG
      "this" DET
"<works>"
      "work" V vt vi PRES SG3
      "works" N SG
      "work" N PL
"<.>"
      "." **CLB
```

The practical solution here is to keep the form *will* as default, because in most cases it is the correct one. Only when necessary, will the form *shall* be selected. The last example would fit to that category. The fully disambiguated result is in (16).

(16)

```
"<*i>"
      "i" PRON CAP SG1
"<'ll>"
      "will" AUXV FUT
"<do>"
      "do" V INF
```

```
"<it>"
      "it" PRON SG3 ACC
"<on>"
      "on" PREP
"<my>"
      "i" PRON SG1 GEN
"<own>"
      "own" A
"<time>"
      "time" N PREFR SG
"<.>"
      "." **CLB
"<*it>"
      "it" PRON CAP SG3
"<'ll>"
      "will" AUXV FUT
"<be>"
      "be" AUXV INF
"<a>"
      "a" DET INDEF
"<long>"
      "long" A
"<path>"
      "path" N SG
"<.>"
      "." **CLB
"<*we>"
      "we" PRON CAP PL1
"<'ll>"
      "will" AUXV FUT
"<see>"
      "see" V INF
"<how>"
      "how" PRON REL
"<this>"
      "this" PRON DEM SG
"<works>"
      "work" V vt vi PRES SG3
"<.>"
      "." **CLB
```

The structures with *'s* are ambiguous, because this string may represent the genitive suffix, the verb *be*, and the verb *have*. The disambiguation of these structures is carried out in two phases. The genitive structures are separated from the merged structures already in the pre-processing phase, as was shown above.

The disambiguation between the two verbs is carried out along with morphological disambiguation. Examples in (17) display the merged structures.

(17)

```
"<*he>"
      "he" PRON CAP MALE SG3
"<'s>"
      "be" V PRES SG3
      "have" V PRES SG3
"<been>"
      "be" AUXV EN
"<sleeping>"
      "sleep" V ING
      "sleeping" N  SG
      "sleeping" A
"<with>"
      "with" PREP
      "with" ADV
"<his>"
      "he" PRON MALE SG3 GEN
"<wife>"
      "wife" N SG
"<.>"
      "." **CLB
"<*it>"
      "it" PRON CAP SG3
      "it" PRON CAP SG3 ACC
"<'s>"
      "be" V PRES SG3
      "have" V PRES SG3
"<been>"
      "be" AUXV EN
"<cold>"
      "cold" A
"<and>"
      "and" CONJ CC
"<windy>"
      "windy" A
      "windy" A
"<.>"
      "." **CLB
"<*that>"
      "that" CONJ CAP REL
      "that" CONJ CAP
      "that" PRON CAP DEM SG
"<'s>"
      "be" V PRES SG3
      "have" V PRES SG3
"<not>"
      "not" NEG
"<necessary>"
      "necessary" A
"<.>"
      "." **CLB
```

```
"<*there>"
      "there" ADV CAP
"<'s>"
      "be" V PRES SG3
      "have" V PRES SG3
"<always>"
      "always" ADV
"<something>"
      "something" N SG
"<new>"
      "new" N SG
      "new" A
"<.>"
      "." **CLB
```

In the first two sentences, the correct verb is *have*. In the two last sentences the correct verb is *be*. We see this when we disambiguate the sentences (18).

(18)

```
"<*he>"
      "he" PRON CAP MALE SG3
"<'s>"
      "have" V PRES SG3
"<been>"
      "be" AUXV EN
"<sleeping>"
      "sleep" V ING
      "sleeping" N  SG
      "sleeping" A
"<with>"
      "with" PREP
      "with" ADV
"<his>"
      "he" PRON MALE SG3 GEN
"<wife>"
      "wife" N SG
"<.>"
      "." **CLB
"<*it>"
      "it" PRON CAP SG3
"<'s>"
      "have" V PRES SG3
"<been>"
      "be" AUXV EN
"<cold>"
      "cold" A
"<and>"
      "and" CONJ CC
"<windy>"
      "windy" A
"<.>"
```

```
      "." **CLB
"<*that>"
      "that" PRON CAP DEM SG
"<'s>"
      "be" V PRES SG3
      "have" V PRES SG3
"<not>"
      "not" NEG
"<necessary>"
      "necessary" A
"<.>"
      "." **CLB
"<*there>"
      "there" ADV CAP
"<'s>"
      "be" V PRES SG3
      "have" V PRES SG3
"<always>"
      "always" ADV
"<something>"
      "something" N SG
"<new>"
      "new" N SG
      "new" A
"<.>"
      "." **CLB
```

In cohorts with more than one reading, the default choice is the first one. We remove the other readings and get the clean result (19).

(19)

```
"<*he>"
      "he" PRON CAP MALE SG3
"<'s>"
      "have" V PRES SG3
"<been>"
      "be" AUXV EN
"<sleeping>"
      "sleep" V ING
"<with>"
      "with" PREP
"<his>"
      "he" PRON MALE SG3 GEN
"<wife>"
      "wife" N SG
"<.>"
      "." **CLB
"<*it>"
      "it" PRON CAP SG3
"<'s>"
      "have" V PRES SG3
```

```
"<been>"
      "be" AUXV EN
"<cold>"
      "cold" A
"<and>"
      "and" CONJ CC
"<windy>"
      "windy" A
"<.>"
      "." **CLB
"<*that>"
      "that" PRON CAP DEM SG
"<'s>"
      "be" V PRES SG3
"<not>"
      "not" NEG
"<necessary>"
      "necessary" A
"<.>"
      "." **CLB
"<*there>"
      "there" ADV CAP
"<'s>"
      "be" V PRES SG3
"<always>"
      "always" ADV
"<something>"
      "something" N SG
"<new>"
      "new" N SG
"<.>"
      "." **CLB
```

Finally, we test whether genitive structures are kept separate from the merged structures. This is revealed already after pre-processing. In order to save space, we show examples below without analysis (20).

(20)
(a) I think he viewed Paddington's success sort of in that way.
(b) That 's the raised platform and there 's the new terminal.
(c) It 's very amusing knowing now what we know about Paddington's huge success.
(d) I just think it 's a striking expression of love's ability to persevere within the cracks and cogs of inhuman systems.
(e) Sarah Palin's Eldest Son Arrested on Domestic Violence Charges
(f) The case is currently before Alaska's Veteran's Court.

We see that genitive constructions in above sentences are kept intact, and merged structures are detached into two parts. Now the sentence can be safely analysed.

6 Conclusion

In this report we have discussed such English structures, where the apostrophe marks either the genitive structure or the merge of two words. We have gone through each type in detail and shown how they can be analysed so that the result is correct.

We do not claim that we have caught all example types. Writers are sometimes inventive and construct their own abbreviations. We can, however, conclude that whatever the constructions are, they can be handled in an elegant way.