Joint Honours Dissertation Celtic and Gaelic

"Implementing a morphological analyser for Scottish Gaelic verbs using a Finite State Transducer"

2106416P

Submitted in partial fulfilment of the requirements for the degree of Master of Arts in Celtic and Gaelic, School of Humanities, University of Glasgow, 2014

Table of Contents

Acknowledgements

There are many people I would like to thank for their contribution to this dissertation. Firstly I would like to thank Dr Mark McConville for his guidance, understanding and patience through the development of my dissertation. I would like to thank Dr Sheila Kidd for teaching me all I know about Scottish Gaelic, and Prof Roibeard O Maolalaigh for helping set up this dissertation and helping me with modules at the start of the academic year.

From my home University, Trinity College, I would like to thank Dr Carl Vogel, Dr Martin Emms and Eoin Mac Cárthaigh for setting up my exchange to Glasgow University. I would also like to thank Elaine Uí Dhonnchada, without whom I would never have known what Finite State Transducers were or how to implement them with Celtic languages.

Finally I would like to thank my great-uncle Oilibhéar de Paor (Brother de Paor) for teaching, nurturing and developing my knowledge and love of the Irish language since I was eight years old, without whom I would not be in the course or position that I am in today, and to whom I am forever grateful.

1. Introduction

For this dissertation, I intend to design, implement and evaluate a morphological transducer for verbs in the Scottish Gaelic language. Initially I shall describe the morphological phenomena to do with verbs in Scottish Gaelic, giving a brief overview of the linguistic status and history of the language as a whole. I also intend to investigate the resources already present for Scottish Gaelic to do with morphological analysis and computational linguistics as a whole.

When building a morphological transducer for a language, both computational and linguistic resources are necessary. One must have the software capable of implementing the transducer and one must have prior knowledge of the grammatical rules and word formation in the language. There are many finite state softwares out there that are language independent, the software that I intend to use for this dissertation is XFST or Xerox Finite State Technology, information on which can be found on this site - http://www.stanford.edu/~laurik/fsmbook/home.html , where it can also be downloaded for free. The main resource I shall be using for all of the grammar rules and morphological operations that occur with verbs in Scottish Gaelic shall be my “Oilthigh Ghlaschu Gàidhlig 1B” coursework book Glasgow University (2013), along with Scottish Gaelic dictionaries.

2. Language Background

It is commonly accepted that the Gaelic language was brought into Western Scotland by Irish settlers around the year 500 AD (Gilles, 2012). It is therefore safe to assume that there are many similarities between the two languages Modern Irish and Scottish Gaelic, both having descended from Old Irish. Both languages belong to the Celtic branch of Indo-European languages, and like the majority of Indo-European languages, Scottish Gaelic is an inflectional language. This means that it displays grammatical relationships morphologically. For example the lenition of a verb to show tense - “Ghlan e an doras” - “He cleaned the door”, or the lenition of a noun to show gender or case - “Chan eil a’ bhò an seo” - “The cow is not here”.

The phenomena of initial mutation are a typifying feature of Celtic languages, as the languages changed over time “many of the phonological conditions causing the mutations disappeared, but the mutations remain and have become grammaticalised” (Ó Cuív, 1987: 395 - 400; Russell, 1995: 237; Campbell, 200, 324; cited Ui Dhonnchadha, C. Nic Phaidin and J. Van Genabith). Scottish Gaelic “uses inflectional distinctions to mark number, gender and case in nouns, adjectives and the definite article” (Gilles, 2012, p 171) which may be done through suffixing, a qualitative change in a final consonant or a combination of the different strategies.

Scottish Gaelic uses lenition, nasalization (which is also referred to as eclipsis) and non mutation as morphological operations. Lenition, often called aspiration in grammar books, is orthographically when a ‘h’ is inserted as the second character of a word, assuming that the word is lenitable, lenition can phonologically be described as a weakening of a consonant, a change in its sound, making it more sonorous, for example:

(1) ‘balach’ -> ‘bhalach’ - boy

In (1) the bilabial plosive [b] is altered into the labiodental fricative [v]

Orthographically, nasalization is when the last letter of a word is altered for linguistic purposes, for example, ‘an’ always goes to ‘am’ before ‘b’, ‘p’, ‘f’ or ‘m’. Phonetically nasalisation can be defined as the production of sound while the velum is lowered so that the air escapes out through the nose instead of the mouth, while the sound is being produced in the mouth, for example:

(2) ‘an cat’ - the cat , ‘am fiaclair’ - the dentist

In (2) ‘an’ has changed to ‘am’ before [f] as it is easier to move from a bilabial to a labiodental, than it is to move from a post alveolar to a labiodental, ‘an’ was not changed before [k] in ‘cat’ as it is not easier to move from a bilabial to a velar, than from a post alveolar to a velar.

In Scottish Gaelic Consonant harmony is very important, leading to the rule that stems and suffixes must agree in terms of broadness or slenderness, which leads to the grammar rule “Caol ri caol agus leathann ri leathann”, to maintain this harmony either the stem or the suffix must be broadened or slenderised. A stem or a suffix is considered to be broad if it’s last vowel is broad (a, à, o, ò, u or ù) and it is considered slender if it’s last vowel is slender (i, ì, e or è).

<u>3. Regular Verbs rules</u>

In order to create a morphological transducer, I must first document and list all the ways in which a verb in Scottish Gaelic can be changed, please find below my attempt to do so:

3.1 Verbal Nouns

In Scottish Gaelic, almost every verb has two parts, the root of the verb and its verbal noun. In English all verbal nouns end the same, simply by adding the suffix “ing” to the end of a verb’s root. Gaelic is very different, there are many different endings and sometimes no ending at all.

Table 1

Verb	**Verbal Noun**
ith - eat	ithe - eating
glan - clean	glanadh - cleaning
freagair - answer	freagairt - answering
beir - catch	breith - catching
obraich - work	obair - working

As a result for my finite state transducer, I may have to include the verbal noun with each verb in the lexicon, so that my FST will recognise it, as there are no rules on how to form the verbal noun.

3.2 Imperatives / Commands

In Scottish Gaelic, the root of the verb is used to form the imperative, the single or informal imperative is therefore just the root of the verb, however if you wish to form the plural or formal imperative, you must add (a)ibh to the end of the root of the verb :-

Table 2

Root	**Singular Imperative (Spoken to one person)**	**Plural Imperative (Spoken to more than one person, or a stranger/superior)**
Sgrìobh - Write	Sgrìobh! - Write!	Sgrìobhaibh! - Write!
Ith - Eat	Ith! - Eat!	Ithibh! - Eat!
Glan - Clean	Glan! - Clean!	Glanaibh! - Clean!

3.3 Past Tense

When forming the past tense of a verb in Scottish Gaelic, you must first check the first letter of the verb, if the word starts with an l, n, r, or if the first two letters are ‘sg’, ‘st’, ‘sm’ or ‘sp’ then nothing is altered when put into the past tense :-

Table 3

Root	**Past Tense**	**Sentence**
leugh - read	leugh - read	Leugh e an leabhar - He read the book

nigh - wash	nigh - washed	Nigh e a chòta - He cleaned his coat
reic - sell	reic - sold	Reic iad brot an-dè - They sold soup yesterday
sgrìobh - write	sgrìobh - wrote	Sgrìobh mi aiste Diluain - I wrote an essay on Monday

If the verb starts with any consonant other than the one mentioned above, the verb is lenited when put into the past tense :-

Table 4

Root	**Past Tense**	**Sentence**
cuir - put	chuir - put	Chuir e an leabhar ar ais sa leabharlann - he put the book back in the library
glan - clean	ghlan - cleaned	Ghlan i a rùm - she cleaned her room
coisich - walk	choisich - walked	Choisich sinn dhachaigh Disathairne - We walked home on Saturday
dùin - close	dhùin - closed	Dhùin iad an doras - They closed the door

If the verb starts with an f, place “dh’fh” in place of the f :-

Table 5

Root	Past Tense	Sentence
fàg - leave	dh’fhàg - left	Dh’fhàg i an taigh anns a’ mhadainn - She left the house in the morning
fan - stay	dh’fhan - stayed	Dh’fhan e ann an rùm - He stayed in the room
foghlaim - learn	dh’fhoghlaim - learned	Dh’fhoghlaim mi Gàidhlig an-uiridh - I learned Gaelic las year
freagair - answer	dh’fhreagair - answered	Dh’fhreagair an tidsear a’ cheist - The teacher answered the question

If the verb starts with a vowel, you simply add “dh’ ” to the root of the verb :-

Table 6

Root	Past Tense	Sentence
iarr - ask for	dh’iarr - asked for	Dh’iarr e mise rudeigin - he asked me something
aithris - report	dh’aithris - reported	Dh’aithris e air a’ thubaist - He reported on the accident
òl - drink	dh’òl - drank	Dh’òl e uisge-beatha - He drank whiskey

èist - listen	dh'èist - listened	Dh'èist e ri ceòl - He listened to music

3. 4 Future Tense

- Independent :
  - Add (a)idh to the end of the root. e.g) glanaidh, òlaidh, fàgaidh
- Dependent :
  - Root by itself for dependent. e.g) glan, òl, fàg
- Relative Future
  - Verbs beginning with a consonant, lenite root + (e)as. e.g ghlanas
  - Verbs beginning with a vowel, add dh' to the root + (e)as. e.g dh'òlas
  - Verbs beginning with F, add dh' to a lenited root + (e)as. e.g dh'fhàgas

For all aspects of the future tense, the verb will be lenited (assuming lenition is possible) when it is put in the negative. e.g) ghlan, fhàg

3.5 Conditional

- Independent:
  - For consonants, lenite the root and add (e)adh / (a)inn / (e)amaid to the end
  - For vowels and FH + vowels, add Dh' to root and add (e)adh / (a)inn / (e)amaid to the end
- Dependent:
  - root + (e)adh / (a)inn / (e)amaid
  - will lenite after negative
- Conditional examples:
  - An cuireadh? An òladh?

○ Chuireadh	Dh’òladh
○ Chuirinn	Dh’òlainn
○ Chuireamaid	Dh’òlamaid
○ Cha chuireadh	Chan òladh
○ Cha chuirinn	Chan òlainn

<u>4. Irregular Verb rules</u>

Unfortunately the above rules do not work for the irregular verbs in Scottish Gaelic, of which there are only 10, which is relatively few in comparison to other languages, for example English which has over 450 irregular verbs, although only around 200 of those are in common use. In order to implement these rules into a Finite State Transducer I may have to deal with them each individually. Please find below all 10 irregular verbs, with a table (I did not number the tables as there are ten of them, and each only deal with one verb, stated above them) showing all of their possible morphological alterations:

Rach - Going

	Past	**Future**	**Conditional**
Independent	Chaidh	Thèid	Rachadh / Rachainn / Rachamaid
Dependent	Deach	Tèid	Rachadh / Rachainn / Rachamaid

Imperative Singular	Thalla
Imperative Plural	Thallaibh

Faic - Seeing

	Past	**Future**	**Conditional**
Independent	Chunnaic	Chì	Chìtheadh / Chìthinn / Chìtheamaid
Dependent	F(h)aca	F(h)aic	F(h)aiceadh / F(h)aicinn / F(h)aiceamaid

Imperative Singular	Faic
Imperative Plural	Faicibh

Cluinn - Hearing

	Past	**Future**	**Conditional**
Independent	Chuala	Cluinnidh	Chluinneadh / Chluinninn / Chluinneamaid
Dependent	C(h)uala	Cluinn	C(h)luinneadh / C(h)luinninn / C(h)luinneamaid
Relative Future		Chluinneas	

Imperative Singular	Cluinn
Imperative Plural	Cluinnibh

Faigh - Getting/Finding

	Past	**Future**	**Conditional**
Independent	Fhuair	Gheibh	Gheibheadh / Gheibhinn / Gheibheamaid
Dependent	D’fhuair	Faigh	F(h)aigheadh / F(h)aighinn / F(h)aigheamaid

Imperative Singular	Faigh
Imperative Plural	Faighibh

Thig - Coming

	Past	**Future**	**Conditional**
Independent	Thàinig	Thig	Thigeadh / Thiginn / Thigeamaid
Dependent	Tàinig	Tig	Tigeadh / Tiginn / Tigeamaid

Imperative Singular	Thig
Imperative Plural	Thigibh

Dèan - Doing/Making

	Past	**Future**	**Conditional**
Independent	Rinn	Nì	Dhèanadh / Dhèanainn / Dhèanamaid
Dependent	D' rinn	Dèan	Dèanadh / Dèanainn / Dèanamaid

Imperative Singular	Dèan
Imperative Plural	Dèanaibh

Ruig - Reaching/Arriving

	Past	**Future**	**Conditional**
Independent	Ràinig	Ruigidh	Ruigeadh / Ruiginn / Ruigeamaid
Dependent	D' ràinig	Ruig	Ruigeadh / Ruiginn / Ruigeamaid

Relative Future		Ruigeas	

Imperative Singular	Ruig
Imperative Plural	Ruigibh

Thoir - Giving/Taking

	Past	**Future**	**Conditional**
Independent	Thug	Bheir	Bheireadh / Bheirinn / Bheireamaid
Dependent	Tug	Toir	Toireadh / Toirinn / Toireamaid

Imperative Singular	Thoir
Imperative Plural	Thoiribh

Abair - Saying

	Past	**Future**	**Conditional**
Independent	Thuirt	Their	Theireadh / Theirinn / Theireamaid
Dependent	Tuirt	Abair	Abradh / Abrainn / Abramaid

Imperative Singular	Abair
Imperative Plural	Abraibh

Beir - Bearing/Taking Hold Of

	Past	**Future**	**Conditional**
Independent	Rug	Beiridh	Bheireadh / Bheirinn / Bheireamaid
Dependent	D' rug	Beir	B(h)eireadh / B(h)eirinn / B(h)eireamaid
Relative Future		Bheireas	

Imperative Singular	Beir
Imperative Plural	Beiribh

4.1 Verb 'to be'

Past tense:	Independent -> Bha	Dependent -> Robh
Present tense:	Independent -> Tha	Dependent -> eil / bheil
Future tense:	Independent -> Bithidh	Dependent -> Bi

	Relative Future -> Bhitheas
	* Cha lenites dependent -> Bhi
Conditional:	Independent -> Bhitheadh Dependent -> Bitheadh
	*Can be a synthetic verb (Subject and Verb in one) :
	Independent -> Bhithinn / Bhitheamaid
	Dependent -> Bithinn / Bitheamaid
	** Again Cha lenites dependent

5. Lexicon

The initial lexicon I shall make for this dissertation shall consist of forty words, all ten irregular verbs, then ten verbs starting with a consonant, ten starting with a vowel, and ten starting with an 'f', in order to get a good variety of words to show the different rules for verb morphology in Scottish Gaelic. Every verb shall be shown with the root and verbal noun of the verb.

Consonant verbs:

1.	glan,	glanadh
2.	cuir,	cur
3.	ceannaich,	ceannach
4.	coimhead,	coimhead
5.	coisich,	coiseachd
6.	leugh,	leughadh
7.	sgrìobh,	sgrìobhadh
8.	cluich,	cluich
9.	dùin,	dùnadh
10.	tuig,	tuigsinn

Vowel verbs:

1.	èist,	èisteachd
2.	iarr,	iarraidh
3.	ith,	ithe
4.	obraich,	obair
5.	òl,	òl
6.	iasgaich,	iasgach
7.	òrdaich,	òrdachadh
8.	innish,	innse
9.	itealaich,	itealaich
10.	ainmich,	ainmeachadh

F verbs:

1.	fàg,	fàgail
2.	fosgail,	fosgladh
3.	freagair	freagairt
4.	foghlaim,	foghlaim
5.	fàs,	fàs
6.	fan,	fantainn / fantaill
7.	fuaraich,	fuarachadh
8.	fuaigh,	fuaigheal
9.	fuirich,	fuireach
10.	fòn,	fòn

Irregular verbs:

1.	abair,	ràdh
2.	beir,	breith
3.	cluinn,	cluinntinn
4.	dèan,	dèanamh
5.	faic,	faicinn
6.	faigh,	faighinn
7.	rach,	dol
8.	ruig,	ruigsinn
9.	thig,	tighinn
10.	thoir,	toirt

6. Finite State Transducers

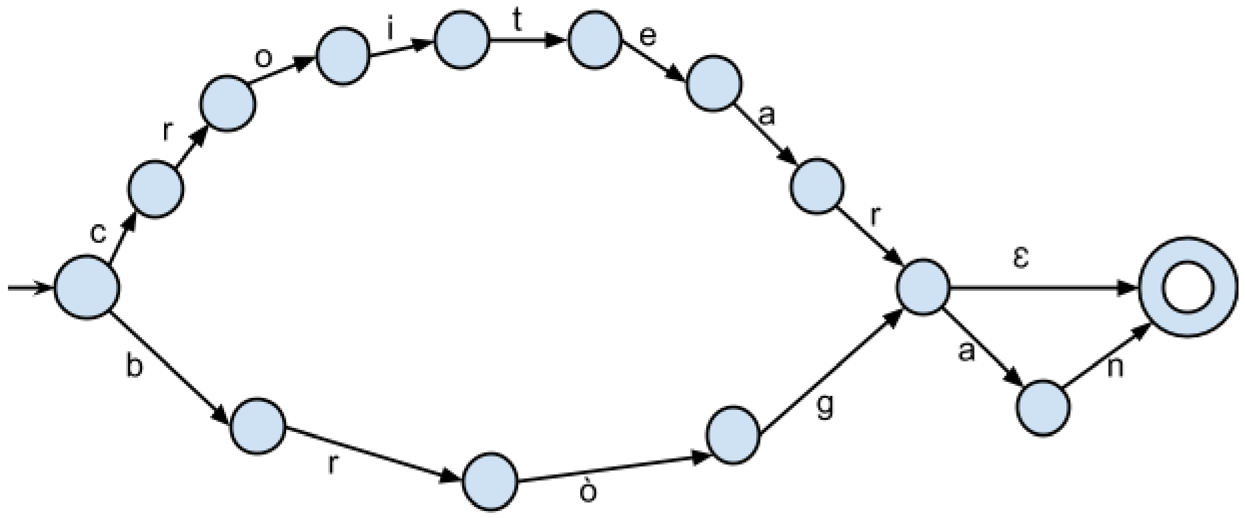
In order to understand what finite state transducers are, I must first give you a brief outline as to what morphology is. Morphology is the study of the way in which words are made up from smaller meaning-bearing units, which are known as *Morphemes* (Jurafsky, D. & Martin, J. H, 2000, p 59). An example of Morphemes in the Scottish Gaelic language can be seen in the word “Croitearan” which means “crofters”, in this word the morphemes are “croitear” and “-an”, similarly the morphemes for its English equivalent are “crofter” and “-s”. It must be noted that my example “croitearan” can actually be split up as “croit” + “(e)ar” + “an”, just as “crofters” can be split into “croft” + “er” + “s”, but since for this dissertation I am not dealing with derivational morphology I shall be ignoring this and the “-ar” suffix. Morphemes are often split into stems and affixes, with stems being the morpheme with the most meaning, and affixes supplying additional meanings of different kinds, therefore in my previous example, “croitear” would be the stem, as it gives us the main meaning, while “-an” is an affix which shows us that the word is in the plural.

Affixes themselves can be split up even further into different categories, such as prefixes, suffixes, infixes and circumfixes. Prefixes and suffixes are common on the Scottish Gaelic language, a prefix is where the affix is placed before the stem of the word, for example the "dh'" in the word "dh'ith" which indicates that "ith" (to eat) is in the past tense. A suffix is when the affix is placed at the end of the stem, these are very common in Scottish Gaelic, and "-an" is a perfect example from our previous example "croitearan" which indicates that the stem is in the plural. Prefixes and suffixes are considered to be concatenative morphology, as a word is composed of a certain number of morphemes concatenated or put together. A word in Scottish Gaelic can be composed using more than one affix, for example "dh'òladh" (would drink), the suffix "adh" tells us that the word is in the conditional tense, and the prefix "dh'" tells us that it is in it's independent form.

A finite state transducer should be able to take in a word, which can be comprised of any number of morphemes, and be able to morphologically parse that word and output it. For example if our example word "croitearan" were to be put into an FST, the output would be "croitear +N +PL" and if "dh'òladh" were to be put in, the output would be "òl +V +Indep +Conditional". In order to build any morphological parser, you need three things:

1. A lexicon, which is a collection of stems and affixes, along with information about them. I have listed the words that will initially be in my lexicon already.
2. Morphotactics, which state the order in which morphemes can form a word
3. Orthagraphic rules, these are spelling and grammar rules, which I have already listed above for Scottish Gaelic verbs.

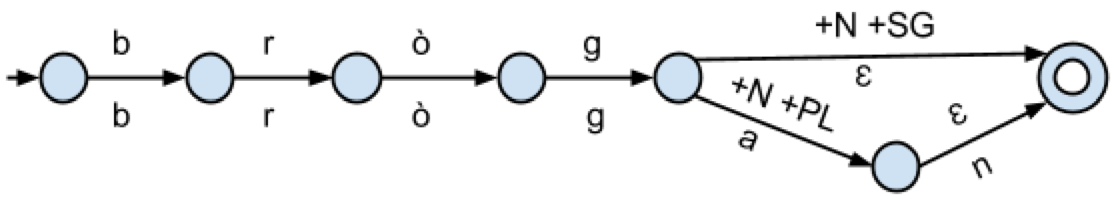
An FSA, or finite state automaton is used as one of the most common ways of modelling morphotactic rules. An FSA is a basically a decision tree structure where you want to validate an input word against a list, for example figure 1 is an FSA that would accept both the plural and singular forms of the words "croitear" and "bròg" (shoe) :



(Fig 1)

If you were to put the word "croitearan" into this FSA it would accept it as it would be able to traverse through each of the points of the FSA to the acceptance state, which is shown in my example by the double elipse, and words which make it from the initial point to the acceptance state of an FSA are called *the language accepted* by the FSA, so the language accepted by this FSA would be {croitear, croitearan, bròg, brògan}. An FSA is therefore able to represent a Lexicon and do morphological recognition.

Finite state transducers (FST) are similar to FSA, except they are based on two-level morphology. "Two-level morphology represents a word as a correspondence between a lexical level … and the surface level" (Jurafsky, D. & Martin, J. H, 2000, p 71). The surface level is the actual spelling of the word, while the lexical level is about the morphemes that make up the word. An FST maps between these two levels using an FSA, "thus we usually visualize an FST as a two-tape automaton which recognizes or generates pairs of strings" (Jurafsky, D. & Martin, J. H, 2000, p 71). For example, figure 2 is a simple FST which has the accepted language {bròg, brògan} and which will output the morphemes needed to make up those words if one of them is entered:



(Fig 2)

So if you were to enter “bròg” as the input to this FST, “bròg +N +SG” would be the output, and if you were to enter “brògan” as the input, “bròg +N +PL” would be the output.

For my FST I shall be creating a lexicon of stems along with their affixes and information about them such as tense and dependency, as well as replace rules which will be able to implement spelling rules and mutations, which can be represented by graphs or regular expressions. These will be implemented using Xerox Finite State Tools, the *lexc* tool will be used to code the lexicon transducer, and the *xfst* tool will be used to code the replace rules transducer.

7. Implementation

When implementing my transducer, I had to first decide exactly what aspects the lexicon transducer would have to deal with and what aspects the replace rules would have to deal with. In order to identify this, I created lists of all the changes that could happen to a single verb, along with the morphological tags I wanted to be attached to them, I then colour coded what aspects would be dealt with by the lexicon and what aspects would be left for the replace rules, for example:

glan ->	ghlan	+Past +Reg

glan	+Future +Dep +Reg
glanaidh	+Future +Indep +Reg
ghlanas	+Relative +Future +Reg
ghlanadh	+Conditional +Indep +Reg
ghlanainn	+Conditional +Indep +Reg +1P +Sg
ghlanamaid	+Conditional +Indep +Reg +1P +Pl
glanadh	+Conditional +Dep +Reg
glanainn	+Conditional +Dep +Reg +1P +Sg
glanamaid	+Conditional +Dep +Reg +1P +Pl
glanaibh	+Pl +Imp +Reg
glan	+Sg +Imp +Reg

Where red represents morphological aspects that will be dealt with using replace rules, and green represents the morphological aspects that will be dealt with using the lexicon transducer. The aspects that the lexicon transducer takes care of are all examples of affixation, suffixation in all of the above examples, this is because the lexicon transducer can append suffixes onto the surface level and morphological tags onto the lexical level. Replace rules are used, for lenition in the above examples, because a part of the verb must be changed on a spelling level, not just have some bit added onto it.

For regular verbs I noticed that this would work for any of them, using the lexicon to append a suffix onto the end of the verb, and the replace rules to perform lenition (or to add Dh' for verbs beginning with vowels, or Dh'fh for verbs beginning with f). In order to know when to use the replace rules, I decided to use replace rule triggers. Replace rule triggers are when you concatenate a tag onto the lower level string in your lexicon transducer (^Len for lenition for example), which then acts as a trigger for a replace rule which will perform the morphological change outside of the lexicon. This means that whenever ^Len appears in the lower level string when passing through the replace rule transducer, its rule for lenition will activate. For further information on how lexc works, check out the Finite-State Lexicon

Compiler website by Karttunen(1993).

7.1 Lexicon transducer - lexc

One of the main differences for how the lexicon should perform with regular verbs, is whether to concatenate the string with a broad or a slender suffix, therefore my lexicon shall initially split regular verbs up based on whether they are slender or broad, from there it will add the appropriate morphological tags and suffixes. For example, here is a simplified version of how my Lexicon will deal with regular verbs:

```
LEXICON Verbs

! Stem                      ContinuationClass           Gloss

glan                        Broad;                      ! clean
cuir                        Slender;                    ! put

LEXICON Broad

+Past+Reg:^Len              #;
+Future+Indep+Reg:aidh      #;

LEXICON Slender

+Past+Reg:^Len              #;
+Future+Indep+Reg:idh       #;
```

From this simple version of my lexicon, you can see examples of replace rule triggers, concatenation of morphological tags, continuation class division and stem suffixation.

7.2 Replace rules

Xfst is able to read regular expressions, and these are what I will be using to code my replace rules. The replace rules will be in the following:

String -> replacement string || Left context _ Right context;

Where the '_' represents where the replaced string is to be positioned. The symbol '.#.' can be used to define whether the replaced string should be put at the start or at the end of the string, for example if the string you wanted to replace had to be at the start of the string, you would put :

a -> A || .#. _ ;

Meaning that 'a' would only change to 'A' if it were at the beginning of the string, similarly:

a -> A || _ .#. ;

Means that 'a' would only change to 'A' if it were at the end of the string.

7.3 Regular verbs

The only morphological aspect that I need to implement with replace rules is lenition, varying slightly as to how to deal with verbs that start with consonants, vowels or f, but no matter what letter the verb starts with it will need to be altered whenever the replace rule trigger ^Len appears. Therefore I need to initially define vowels, and lenitable consonants for the transducer, the transducer will then be able to use these definitions to see when it should alter a verb. An example of how to define a set in xfst can be seen here with vowels:

```
define Vowel                    a | à | o | ò | u | ù | i | ì | e | è;
```

A simple example of a replace rule for lenition, will need to lenite only when there is the correct replace rule trigger present, and only when the first letter of the verb is a lenitable consonant, for example:

```
[..] -> h || .#. lenitableCons _ ?+ %^Len;
```

[ ] is used in regular expressions to denote empty strings, [..] however limits the expression to denote one empty string, this means that it will only put in one 'h' after the first letter, so long as it is a lenitable consonant, and so long as the string contains the trigger '^Len', instead of an infinite amount of h's, which it would be able to do if I had used [ ] as there are an infinite number of empty strings between the first consonant and the rest of the string. This is an example of how crucial it is to get the grammar of the regular expression correct.

'%' is used in regular expressions to literalise symbols which have special functions, I needed to use it for my replace rule trigger, as '^' has a special function in regular expressions, it is used when comparing greater than and less than, when checking the number of concatenations performed on a string, therefore I needed to literalise it so that the replace rule would recognise the replace rule trigger '^Len'.

After going through all of the replace rules, my transducer then has to take the replace rule trigger off of the verb if it is there, to make sure that it deals with verbs properly. This is done very simply by stating that the trigger, ^Len, is replaced with an empty string always, and have it as the last rule in the transducer, so that it doesn't get rid of the trigger before the trigger needs to be used. This rule can be seen here:

```
%^Len -> [ ];
```

By not having a right or a left context, the rule is stating that this replacement will always happen, in every context. The files containing my full lexicon transducer and replace rule transducer can be found attached to the appendix of this dissertation. On them you can see the different ways I dealt with verbs which start with vowels and similar problems.

7.4 Irregular verbs

Implementing the irregular verbs was much more difficult than the regular verbs as they did not follow any rules, therefore I had to make many replace rules just for a single condition of one of the verbs. To make it cleaner and easier to see how the lexicons were working, I created a second lexicon for irregular verbs, which I can also join with my replace rules transducer. This just means that I shall have to test them separately. Initially I looked at all tenses and morphological alterations of irregular verbs to see if I could find some regularity in them. I found that the irregular verbs are most regular, in the sense that they follow regular rules, in the imperative, where the only occasions that the regular rules aren't followed are for the imperative plural of "abair" and both forms of the imperative for "rach". The table below demonstrates where else irregular rules follow regular rules:

Table 8

	Future	**Conditional**
Dependent	Abair, beir, cluinn, dèan, faic, faigh, ruig	beir, cluinn, dèan, rach, ruig, faic
Independent	beir, cluinn, ruig	beir, cluinn, dèan, rach, ruig, thuig

Relative	beir, cluinn, ruig	

I had to make a separate continuation class for each verb, each which contained the twelve or thirteen (depending on whether there was a relative future or not) different morphological tags to create all of the different forms of the verb, and if the verb had a regular aspect, it would have a continuation class to a regular verb lexicon for the specific tense it was regular in.

I then looked at the rest of the irregular verbs, and saw that because they were so irregular, that I would have to use replace rules on the entire word to change it from one form to another, I therefore had to make many replace rules, and use replace rule triggers to set them off. I could use the same replace rule trigger for every irregular verb in the same tense, for example ^PastIndep, as my replace rules would only activate if that trigger were there, and if it was the exact word it was looking for, for example:

```
a b a i r ->  t h u i r t || _ %^PastIndep;
```

This replace rules states that “abair” will always be replaced with “thuirt” if there is the replace rule trigger ^PastIndep concatenated onto the end of the verb, which is where it will always be concatenated. This requires a lot of work and a lot of replace rules, but it does enable my transducer to recognise the different forms of the irregular verbs. I also then have to take away all of the triggers once they are finished being used, similarly to what I did with the lenition trigger:

```
%^PastIndep -> [ ];
```

8. Problems encountered

One of the major problems that I encountered while creating and testing the transducers was that lexc and xfst were not recognising accented characters (À, Ò, Ù, Ì, È, à, ò, ù, ì, è for Scottish Gaelic). This was because xfst needs files to be UTF-8 character encoded in order to recognise many characters, and the text editor I was using was saving the files with ISO-8859-15 encoding, once I changed this xfst recognised and included words which had an accented character. I had encountered this problem before with Irish accented characters with a fada (á,ó,ú…), which I was able to solve, but a further problem arose with Scottish Gaelic, although the characters were now recognised and included, it could not print them back to me, a different character was always in the place of an accented character, it was always the same character for each accented character so I was still able to test if the replace rules and lexicon were working, but it is not perfect and I was not able to fix it, this is a part of my FST that could be improved upon.

9. Conclusion

In this dissertation I have designed and implemented a morphological analyser for Scottish Gaelic verbs using a Finite State Transducer. I hope to have described a FST clearly, both in terms of what it does and how it does it, I have been largely successful with my implementation, but there are a few areas where my work could be improved upon in the future, its inability to print accented characters for example.

Through testing I also found out that it was not correctly dealing with “ dh’ “ and would therefore not recognise any verbs beginning with dh’, this is a big fault and I unfortunately noticed it too late. I do not believe it is a fault with my coding of the lexicon, but maybe in the replace rules, or maybe in the command line of xfst, somewhere I did not include ‘ as a character that would be accepted, given time I believe this could be fixed.

My FST does not deal with the passive forms of verbs, it does not deal with the negative, I also ran out of time to implement in the verbal noun versions of each verb. Another problem with my FST is that it will only recognise verbs which have been entered as stems into the lexicon, this means that it is labour intensive if you wish to have my FST recognise many verbs. A way to improve this would be to create a semi-automatic program which could take verb stems from a corpus or online dictionary and insert them into my lexicon.

In my opinion, minority languages like Scottish Gaelic benefit greatly from technological resources like this, and I would encourage further work to be put into this area. In a paper on FST's for Irish Uí Dhonnchadha, Caoilfhionn Nic Pháidín and Josef Van Genabith state that "Minority languages must endeavor to keep up with the language technology advances available to the more dominant languages if they are to survive and prosper in the modern world." (Ui Dhonnchadha, C. Nic Phaidin and J. Van Genabith, 2003) Scottish Gaelic is a language in trouble, with the 2011 census showing a further drop in speakers from 59,000 to 58,000 as reported by the BBC (2013), which when taken with the overall population figure from the same census, 5,295,000 means that just over 1% of the population of Scotland can speak Scottish Gaelic. The census showed that the rate of decline was decreasing but this is still very worrying, it means that current language policy for Scottish Gaelic might not be working, and if we are not careful Scottish Gaelic could become a dead language. This is why it is important now, more than ever to use the language technology available to us to help preserve the Scottish Gaelic language in some way, so that at least if it dies out, people will have the proper resources to learn it and possibly revive it in the future, resources such as FST's, online dictionaries and voice synthesisers, such as the abair project (abair.ie) a voice synthesiser for the Irish language, which can help people get the correct phonetics and pronunciation. These technologies also serve useful for people who wish to learn the language but are not in Scotland, where it is easiest to do so, thus making the language more globally accessible.

There are people who are against this kind of opinion, people who want Scottish Gaelic to be a community language, and would rather see it die than see it become a language that is just studied by a few as a hobby, or seen as the Latin of Scotland. There are people who are also against all of the government funding that goes into technological advances for Scottish Gaelic, this can be seen when in July it was reported by the Scottish Express (2013) that £2 million was to be spent on developing a Gaelic online dictionary, much controversy surrounded this, with Robert Oxley of the Taxpayer's Alliance even stating that "Many will have sympathy with a desire to preserve culture, but when school-leavers can't do the basics this has the whiff of a vanity project."(Scottish Express, 2013). There are famous linguists who would also possibly be against technological advances for Scottish Gaelic, Fishman for example who favours a bottom up approach would be much more concerned with developing Gaelic as a community and family language long before investing money in technological advances.

There are also many who would support the technological advance of minority languages, including the Scottish government, and the current good news is that an online part-of-speech tagger and gold standard corpus for Scottish Gaelic is currently being developed (THE CARNERGIE TRUST FOR UNIVERSITIES OF SCOTLAND, 2009) which will greatly prosper Scottish Gaelic and help it keep up with the technology of more majority languages.

Appendix

In this appendix, I have included the actual files which go into making my FST, as well as screenshots of the xfst in action, and test results.

- Dissertation.txt

```
define Vowel                [a|à|o|ò|u|ù|i|ì|e|è];
define lenitableCons        [b|c|d|f|g|m|p|t];
define VowelUpper           [A|À|O|Ò|U|Ù|I|Ì|E|È];
define lenitableConsUpper [B|C|D|F|G|M|P|T];

read lexc < DissertationLexicon.txt
define LEX;
read regex < DissertationRules.regex
define RUL;
read regex LEX .o. RUL;
save Dissertation.fst;
```

This file is saved as a simple .txt file, it can be altered at the “read lexc <” line to read in either the regular or irregular verbs lexicon. This is the file that brings the other files together and creates the FST in xfst. As soon as xfst is open, the command “source < Dissertation.txt” will build the FST, assuming all of the appropriate files are in the same directory.

- DissertationLexicon.txt

```
Multichar_Symbols
+Reg +Past +Sg +Pl +1P +Imp +Dep +Indep +Future +Conditional +Broad +Slender +Relative ^Len

LEXICON Root
        Verbs;

LEXICON Verbs

! Stem                  ContinuationClass               Gloss

glan                    Broad;                          ! clean
cuir                    Slender;                        ! put
ceannaich               Slender;                        ! buy
coimhead                Broad;                          ! watch
coisich                 Slender;                        ! walk
```

```
leugh                           Broad;                                  ! read
sgrìobh                 Broad;                                  ! write
cluich                          Slender;                                ! play
dùin                            Slender;                                ! close
tuig                            Slender;                                ! understand
èist                            Slender;                                ! listen
iarr                            Broad;                                  ! ask
ith                             Slender;                                ! eat
obraich                 Slender;                                ! work
òl                              Broad;                                  ! drink
iasgaich                        Slender;                                ! fish
òrdaich                 Slender;                                ! order
innis                           Slender;                                ! tell
itealaich                       Slender;                                ! fly
ainmich                 Slender;                                ! name
fàg                             Broad;                                  ! leave
fosgail                         Slender;                                ! open
freagair                Slender;                                ! answer
foghlaim                        Slender;                                ! learb
fàs                             Broad;                                  ! grow
fan                             Broad;                                  ! stay
fuaraich                Slender;                                ! cool/ease
fuaigh                          Slender;                                ! sew/stitch
fuirich                         Slender;                                ! stay/reside
fòn                             Broad;                                  ! phone


LEXICON Broad

+Past+Reg:^Len                          #;
+Sg+Imp+Reg:0                   #;
+Future+Dep+Reg:0                       #;
+Future+Indep+Reg:aidh          #;
+Future+Relative+Reg:as^Len     #;
+Conditional+Reg:0                      BroadConditional;
+Pl+Imp+Reg:aibh                        #;

LEXICON Slender

+Past+Reg:^Len                          #;
+Sg+Imp+Reg:0                   #;
+Future+Dep+Reg:0                       #;
+Future+Indep+Reg:idh           #;
+Future+Relative+Reg:eas^Len  #;
+Conditional+Reg:0                      SlenderConditional;
+Pl+Imp+Reg:ibh                         #;

LEXICON BroadConditional
```

```
+Indep:adh^Len                  #;
+Indep+1P+Sg:ainn^Len           #;
+Indep+1P+Pl:amaid^Len                  #;
+Dep:adh                                #;
+Dep+1P+Sg:ainn                         #;
+Dep+1P+Pl:amaid                        #;

LEXICON SlenderConditional

+Indep:eadh^Len                         #;
+Indep+1P+Sg:inn^Len            #;
+Indep+1P+Pl:eamaid^Len                 #;
+Dep:eadh                               #;
+Dep+1P+Sg:inn                          #;
+Dep+1P+Pl:eamaid               #;
```

All information necessary about this file can be found in the write up section of the dissertation.

- DissertationIrregLex.txt

```
Multichar_Symbols
+Irreg +Past +Sg +Pl +1P +Imp +Dep +Indep +Future +Conditional +Broad +Slender +Relative
^PastIndep ^PastDep ^ImpPl ^ImpSg ^FutIndep ^FutDep ^CondInd ^CondIndSg ^CondIndPl
^CondDep ^CondDepSg ^CondDepPl +Pres ^Pres +FutRel

LEXICON Root
        Verbs;

LEXICON Verbs

! Stem                  ContinuationClass               Gloss

abair                   abair;                          ! say
beir                    beir;                           ! catch/bear
cluinn                  cluinn;                         ! hear
dèan                    dèan;                           ! make/do
faic                    faic;                           ! see
faigh                   faigh;                          ! find/get
rach                    rach;                           ! go
ruig                    ruig;                           ! arrive/reach
thig                    thig;                           ! come
thoir                   thoir;                          ! bring/give/take
tha                     tha                             ! to be
```

```
LEXICON abair
+Irreg:0                                                    regFutureDep;
+Indep+Past+Irreg:^PastIndep                                #;
+Dep+Past+Irreg:^PastDep                                            #;
+Imp+Sg+Irreg:0                                                     #;
+Imp+Pl+Irreg:^ImpPl                                        #;
+Indep+Future+Irreg:    ^FutIndep                                   #;
+Indep+Conditional+Irreg:^CondInd                                   #;
+Indep+Conditional+Irreg+1P+Sg:^CondIndSg                   #;
+Indep+Conditional+Irreg+1P+Pl:^CondIndPl                   #;
+Dep+Conditional+Irreg:^CondDep                                     #;
+Dep+Conditional+Irreg+1P+Sg:^CondDepSg                     #;
+Dep+Conditional+Irreg+1P+Pl:^CondDepPl                     #;


LEXICON beir
+Indep+Past+Irreg:^PastIndep    #;
+Dep+Past+Irreg:^PastDep                #;
+Irreg:0                        regImperative;
+Irreg:0                        regFutureDep;
+Irreg:0                        regFutureIndep;
+Irreg:0                        regFutureRel;
+Irreg:0                        regCondIndep;
+Irreg:0                        regCondDep;

LEXICON cluinn
+Indep+Past+Irreg:^PastIndep    #;
+Dep+Past+Irreg:^PastDep                #;
+Irreg:0                        regImperative;
+Irreg:0                        regFutureDep;
+Irreg:0                        regFutureIndep;
+Irreg:0                        regFutureRel;
+Irreg:0                        regCondIndep;
+Irreg:0                        regCondDep;

LEXICON dèan
+Indep+Future+Irreg:    ^FutIndep                           #;
+Indep+Past+Irreg:^PastIndep                        #;
+Dep+Past+Irreg:^PastDep                                    #;
+Irreg+Pl+Imp:aibh                                          #;
+Irreg+SG+Imp:0                                             #;
+Irreg:0                                            regFutureDep;
+Irreg:0                                            regCondIndepBroad;
+Irreg:0                                            regCondDepBroad;

LEXICON faic
+Indep+Past+Irreg:^PastIndep                                #;
```

```
+Dep+Past+Irreg:^PastDep                                #;
+Indep+Future+Irreg:    ^FutIndep                       #;
+Irreg:0                                      regImperative;
+Irreg:0                                      regFutureDep;
+Irreg:0                                      regCondDep;
+Indep+Conditional+Irreg:^CondInd                       #;
+Indep+Conditional+Irreg+1P+Sg:^CondIndSg     #;
+Indep+Conditional+Irreg+1P+Pl:^CondIndPl     #;

LEXICON faigh
+Indep+Past+Irreg:^PastIndep                  #;
+Dep+Past+Irreg:^PastDep                                #;
+Indep+Future+Irreg:    ^FutIndep                       #;
+Irreg:0                                      regImperative;
+Irreg:0                                      regFutureDep;
+Dep+Conditional+Irreg:^CondDep                         #;
+Dep+Conditional+Irreg+1P+Sg:^CondDepSg       #;
+Dep+Conditional+Irreg+1P+Pl:^CondDepPl       #;
+Indep+Conditional+Irreg:^CondInd                       #;
+Indep+Conditional+Irreg+1P+Sg:^CondIndSg     #;
+Indep+Conditional+Irreg+1P+Pl:^CondIndPl     #;

LEXICON rach
+Indep+Past+Irreg:^PastIndep                  #;
+Dep+Past+Irreg:^PastDep                                #;
+Indep+Future+Irreg:    ^FutIndep                       #;
+Dep+Future+Irreg:^FutDep                               #;
+Irreg:0                                      regCondIndepBroad;
+Irreg:0                                      regCondDepBroad;
+Imp+Sg:^ImpSg                                          #;
+Imp+Pl:^ImpPl                                #;


LEXICON ruig
+Indep+Past+Irreg:^PastIndep                  #;
+Dep+Past+Irreg:^PastDep                                #;
+Irreg:0                   regFutureDep;
+Irreg:0                   regFutureIndep;
+Irreg:0                   regFutureRel;
+Irreg:0                   regCondIndep;
+Irreg:0                   regCondDep;
+Irreg:0                   regImperative;

LEXICON thig
+Indep+Past+Irreg:^PastIndep                  #;
+Dep+Past+Irreg:^PastDep                                #;
+Indep+Future+Irreg:0                         #;
+Dep+Future+Irreg:^FutDep                               #;
```

```
+Dep+Conditional+Irreg:^CondDep                           #;
+Dep+Conditional+Irreg+1P+Sg:^CondDepSg              #;
+Dep+Conditional+Irreg+1P+Pl:^CondDepPl              #;
+Irreg:0                                             regCondIndep;
+Irreg:0                                             regImperative;

LEXICON thoir
+Indep+Past+Irreg:^PastIndep                         #;
+Dep+Past+Irreg:^PastDep                                  #;
+Indep+Future+Irreg:     ^FutIndep                        #;
+Dep+Future+Irreg:^FutDep                                 #;
+Indep+Conditional+Irreg:^CondInd                         #;
+Indep+Conditional+Irreg+1P+Sg:^CondIndSg            #;
+Indep+Conditional+Irreg+1P+Pl:^CondIndPl            #;
+Dep+Conditional+Irreg:^CondDep                           #;
+Dep+Conditional+Irreg+1P+Sg:^CondDepSg              #;
+Dep+Conditional+Irreg+1P+Pl:^CondDepPl              #;
+Irreg:0                              regImperative;

LEXICON tha
+Indep+Pres+Irreg:0                                       #;
+Dep+Pres+Irreg:^Pres                                #;
+Indep+Past+Irreg:^PastIndep                         #;
+Dep+Past+Irreg:^PastDep                                  #;
+Indep+Future+Irreg:     ^FutIndep                        #;
+Dep+Future+Irreg:^FutDep                                 #;
+Future+Relative:^FutRel                                  #;
+Indep+Conditional+Irreg:^CondInd                         #;
+Indep+Conditional+Irreg+1P+Sg:^CondIndSg            #;
+Indep+Conditional+Irreg+1P+Pl:^CondIndPl            #;
+Dep+Conditional+Irreg:^CondDep                           #;
+Dep+Conditional+Irreg+1P+Sg:^CondDepSg              #;
+Dep+Conditional+Irreg+1P+Pl:^CondDepPl              #;


LEXICON regFutureDep
+Future+Dep:0                         #;

LEXICON regFutureIndep
+Future+Indep:idh                               #;

LEXICON regCondIndep
+Cond+Indep:eadh^Len                  #;
+Cond+Indep+1P+Sg:inn^Len             #;
+Cond+Indep+1P+Pl:eamaid^Len                    #;

LEXICON regCondDep
+Cond+Dep:eadh                                  #;
```

```
+Cond+Dep+1P+Sg:inn                 #;
+Cond+Dep+1P+Pl:eamaid                      #;

LEXICON regFutureRel
+Future+Relative:eas^Len                    #;

LEXICON regImperative
+Sg+Imp:0                           #;
+Pl+Imp+Reg:ibh                     #;

LEXICON regCondIndepBroad
+Cond+Indep:adh^Len                 #;
+Cond+Indep+1P+Sg:ainn^Len          #;
+Cond+Indep+1P+Pl:amaid^Len         #;


LEXICON regCondDepBroad
+Cond+Dep:adh                       #;
+Cond+Dep+1P+Sg:ainn                        #;
+Cond+Dep+1P+Pl:amaid                       #;
```

- DissertationRules.regex

```
[..] -> h || .#. lenitableCons _ ?+ %^Len
.o.
[..] -> d h ‘ || .#. _ [Vowel | f h] ?+ %^Len
.o.
s h g -> s g || .#. _ ?+ %^Len
.o.
s h p -> s p || .#. _ ?+ %^Len
.o.
s h m -> s m || .#. _ ?+ %^Len
.o.
s h t -> s t || .#. _ ?+ %^Len
.o.
[..] -> h || .#. lenitableConsUpper _ ?+ %^Len
.o.
[..] -> D h ‘ || .#. _ [VowelUpper | f h] ?+ %^Len
.o.
S h g -> S g || .#. _ ?+ %^Len
.o.
S h p -> S p || .#. _ ?+ %^Len
.o.
S h m -> S m || .#. _ ?+ %^Len
.o.
S h t -> S t || .#. _ ?+ %^Len
```

```
.o.
%^Len -> [ ]
.o.
a b a i r ->  t h u i r t ||  _ %^PastIndep
.o.
a b a i r ->  t u i r t ||  _ %^PastDep
.o.
b e i r -> r u g || _ %^PastIndep
.o.
b e i r -> d ‘ r u g || _ %^PastDep
.o.
c l u i n n -> c h u a l a || _ %^PastIndep
.o.
c l u i n n -> c h u a l a || _ %^PastDep
.o.
d è a n -> r i n n || _ %^PastIndep
.o.
d è a n -> d ‘ r i n n || _ %^PastDep
.o.
f a i c -> c h u n n a i c || _ %^PastIndep
.o.
f a i c -> f h a c a || _ %^PastDep
.o.
f a i g h -> f h u a i r || _ %^PastIndep
.o.
f a i g h -> d ‘ f h u a i r || _ %^PastDep
.o.
r a c h -> c h a i d h || _ %^PastIndep
.o.
r a c h -> d e a c h || _ %^PastDep
.o.
r u i g -> r à i n i g || _ %^PastIndep
.o.
r u i g  -> d ‘ r à i n i g || _ %^PastDep
.o.
t h i g-> t h à i n i g || _ %^PastIndep
.o.
t h i g -> t à i n i g || _ %^PastDep
.o.
t h o i r -> t h u g || _ %^PastIndep
.o.
t h o i r -> t u g || _ %^PastDep
.o.
a b a i r -> a b r a i b h || _ %^ImpPl
.o.
r a c h -> t h a l l a i b h || _ %^ImpPl
.o.
r a c h -> t h a l l a || _ %^ImpSg
```

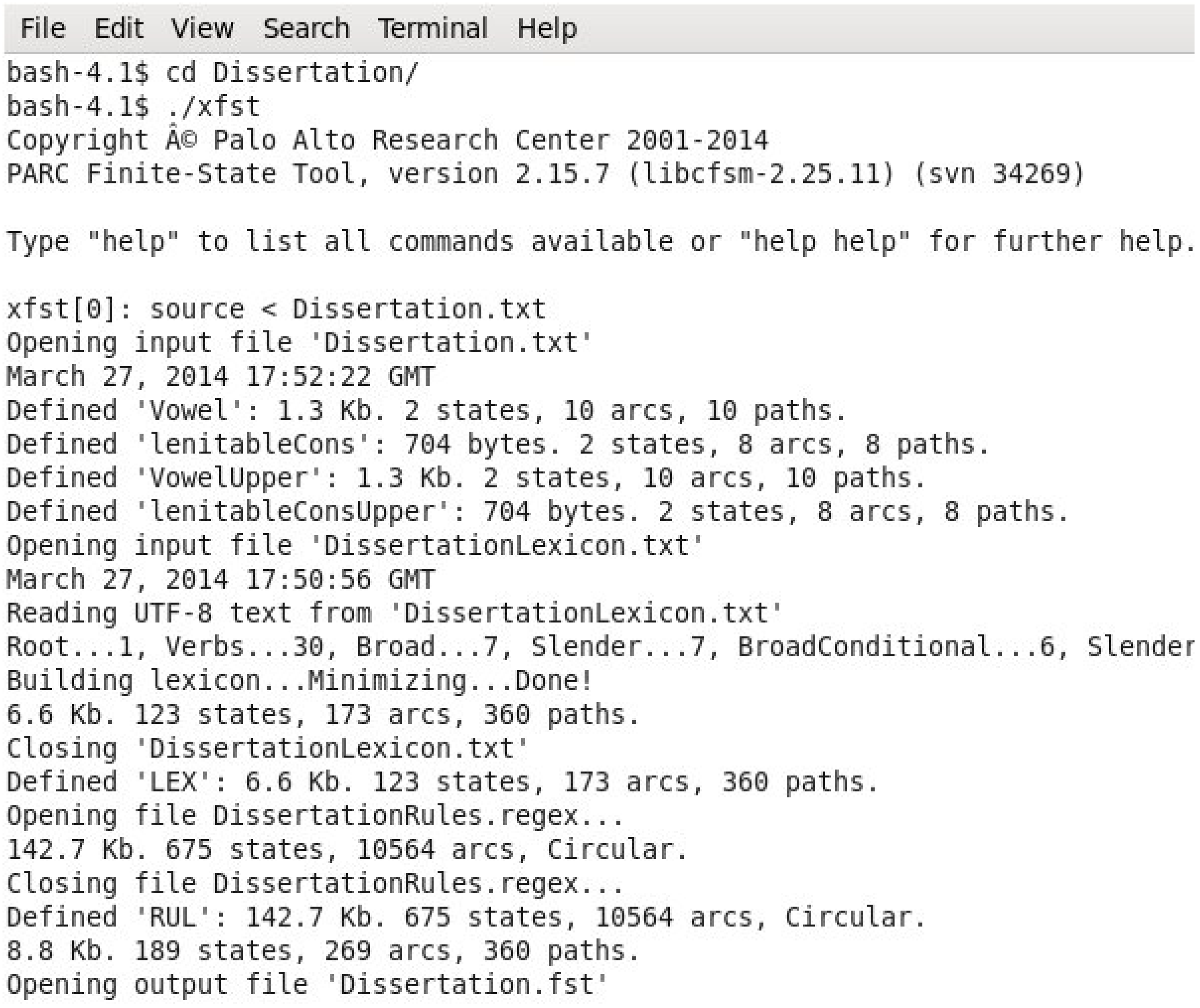
```
.o.
a b a i r -> t h e i r  || _ %^FutIndep
.o.
d è a n -> n ì || _ %^FutIndep
.o.
f a i c -> c h ì || _ %^FutIndep
.o.
f a i g h -> g h e i b h || _ %^FutIndep
.o.
r a c h -> t h è i d || _ %^FutIndep
.o.
t h o i r -> b h e i r || _ %^FutIndep
.o.
r a c h -> t è i d || _ %^FutDep
.o.
t h i g -> t i g || _ %^FutDep
.o.
t h o i r -> t o i r || _ %^FutDep
.o.
a b a i r -> t h e i r e a d h || _ %^CondInd
.o.
f a i c -> c h ì t h e a d h || _ %^CondInd
.o.
f a i g h -> g h e i b h e a d h || _ %^CondInd
.o.
t h o i r -> b h e i r e a d h || _ %^CondInd
.o.
a b a i r -> t h e i r i n n || _ %^CondIndSg
.o.
a b a i r -> t h e i r e a m a i d || _ %^CondIndPl
.o.
f a i c -> c h ì t h i n n || _ %^CondIndSg
.o.
f a i c -> c h ì t h e a m a i d || _ %^CondIndPl
.o.
f a i g h -> g h e i b h i n n || _ %^CondIndSg
.o.
f a i g h -> g h e i b h e a m a i d || _ %^CondIndPl
.o.
t h o i r -> b h e i r i n n || _ %^CondIndSg
.o.
t h o i r -> b h e i r e a m a i d || _ %^CondIndPl
.o.
a b a i r -> a b r a d h || _ %^CondDep
.o.
a b a i r -> a b r a i n n || _ %^CondDepSg
.o.
a b a i r -> a b r a m a i d || _ %^CondDepPl
```

```
.o.
f a i g h -> f h a i g h e a d h || _ %^CondDep
.o.
f a i g h -> f h a i g h i n n || _ %^CondDepSg
.o.
f a i g h -> f h a i g h e a m a i d || _ %^CondDepPl
.o.
t h i g -> t i g e a d h || _ %^CondDep
.o.
t h i g -> t i g i n n || _ %^CondDepSg
.o.
t h i g -> t i g e a m a i d || _ %^CondDepPl
.o.
t h o i r -> t o i r e a d h || _ %^CondDep
.o.
t h o i r -> t o i r i n n || _ %^CondDepSg
.o.
t h o i r -> t o i r e a m a i d || _ %^CondDepPl
.o.
t h a -> b h a || _ %^PastIndep
.o.
t h a -> e i l || _ %^Pres
.o.
t h a -> r o b h || _ %^PastDep
.o.
t h a -> b i t h i d h || _ %^FutIndep
.o.
t h a -> b i || _ %^FutDep
.o.
t h a -> b i  t e a d h || _ %^CondDep
.o.
t h a -> b i t h i n n || _ %^CondDepSg
.o.
t h a -> b i t h e a m a i d || _ %^CondDepPl
.o.
t h a -> b h i t h e a d h || _ %^CondInd
.o.
t h a -> b h i t h i n n || _ %^CondIndSg
.o.
t h a -> b h i t h e a d h || _ %^CondIndPl
.o.
%^Pres -> [ ]
.o.
%^PastIndep -> [ ]
.o.
%^PastDep -> [ ]
.o.
%^ImpPl -> [ ]
```

```
        .o.
        %^ImpSg -> [ ]
        .o.
        %^FutIndep -> [ ]
        .o.
        %^FutDep -> [ ]
        .o.
        %^CondDep -> [ ]
        .o.
        %^CondDepSg -> [ ]
        .o.
        %^CondDepPl -> [ ]
        .o.
        %^CondInd -> [ ]
        .o.
        %^CondIndSg -> [ ]
        .o.
        %^CondIndPl -> [ ];
```

As with the lexicons, all information about these replace rules can be found in the write up.

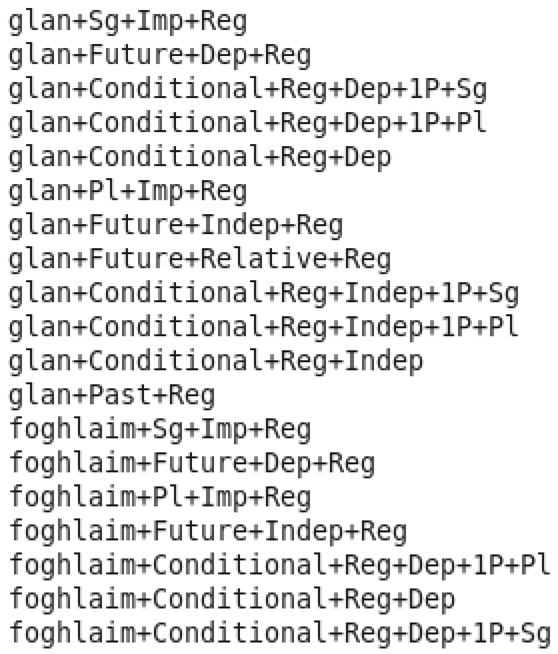
Once those files have been compiled into a FST in xfst you can type “lower” or “upper” to see the upper and lower levels of the morphological analysis, this will show you how many different morphs of the verbs there are along with all of their morphological tags. Here are some screenshots of xfst running, first compiling the FSt then checking the lower and upper:



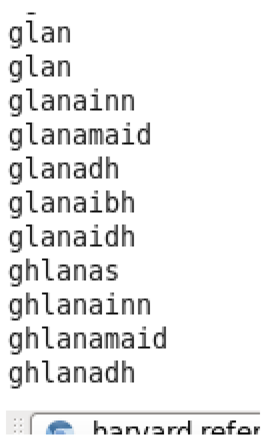
```
bash-4.1$ cd Dissertation/
bash-4.1$ ./xfst
Copyright Â© Palo Alto Research Center 2001-2014
PARC Finite-State Tool, version 2.15.7 (libcfsm-2.25.11) (svn 34269)

Type "help" to list all commands available or "help help" for further help.

xfst[0]: source < Dissertation.txt
Opening input file 'Dissertation.txt'
March 27, 2014 17:52:22 GMT
Defined 'Vowel': 1.3 Kb. 2 states, 10 arcs, 10 paths.
Defined 'lenitableCons': 704 bytes. 2 states, 8 arcs, 8 paths.
Defined 'VowelUpper': 1.3 Kb. 2 states, 10 arcs, 10 paths.
Defined 'lenitableConsUpper': 704 bytes. 2 states, 8 arcs, 8 paths.
Opening input file 'DissertationLexicon.txt'
March 27, 2014 17:50:56 GMT
Reading UTF-8 text from 'DissertationLexicon.txt'
Root...1, Verbs...30, Broad...7, Slender...7, BroadConditional...6, Slender
Building lexicon...Minimizing...Done!
6.6 Kb. 123 states, 173 arcs, 360 paths.
Closing 'DissertationLexicon.txt'
Defined 'LEX': 6.6 Kb. 123 states, 173 arcs, 360 paths.
Opening file DissertationRules.regex...
142.7 Kb. 675 states, 10564 arcs, Circular.
Closing file DissertationRules.regex...
Defined 'RUL': 142.7 Kb. 675 states, 10564 arcs, Circular.
8.8 Kb. 189 states, 269 arcs, 360 paths.
Opening output file 'Dissertation.fst'
```

Above you can see the fst being composed.

Below you can see a screenshot of the upper level of the FST followed by the lower level

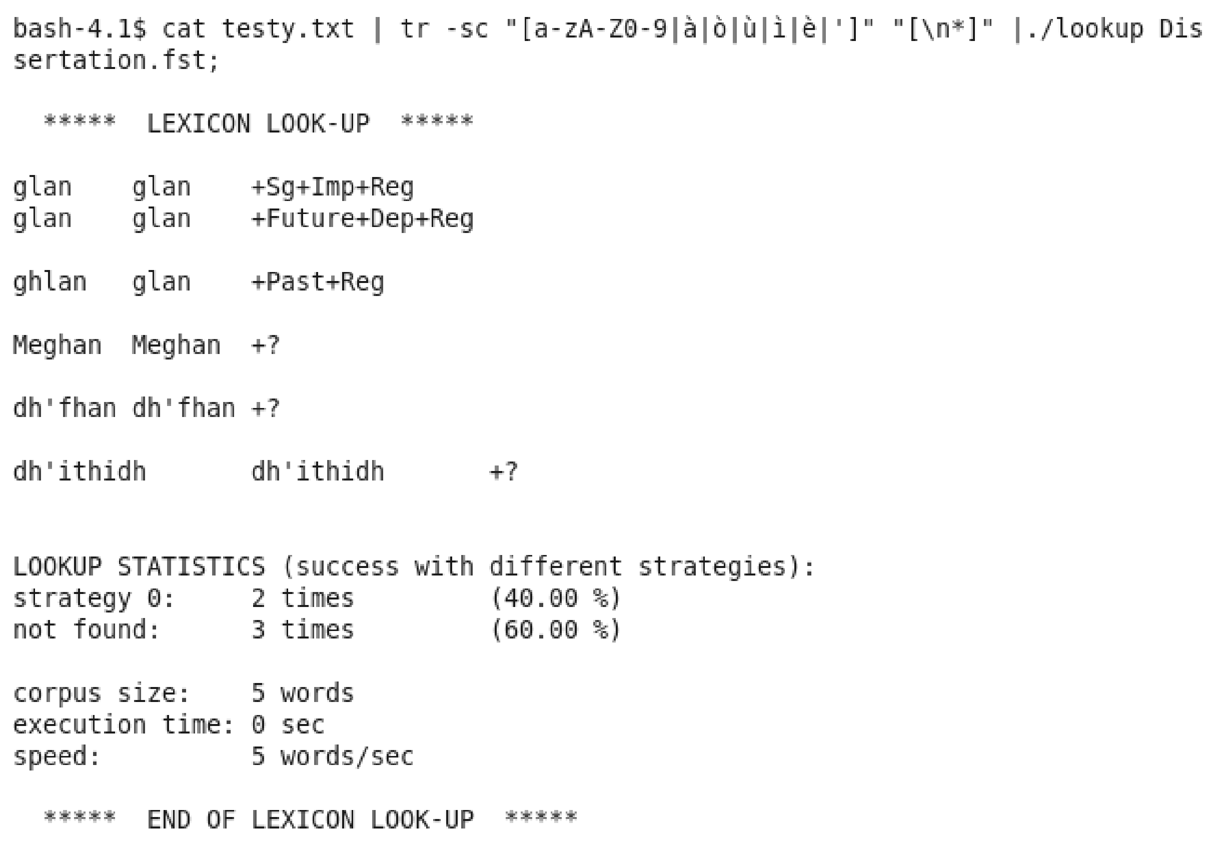


```
glan+Sg+Imp+Reg
glan+Future+Dep+Reg
glan+Conditional+Reg+Dep+1P+Sg
glan+Conditional+Reg+Dep+1P+Pl
glan+Conditional+Reg+Dep
glan+Pl+Imp+Reg
glan+Future+Indep+Reg
glan+Future+Relative+Reg
glan+Conditional+Reg+Indep+1P+Sg
glan+Conditional+Reg+Indep+1P+Pl
glan+Conditional+Reg+Indep
glan+Past+Reg
foghlaim+Sg+Imp+Reg
foghlaim+Future+Dep+Reg
foghlaim+Pl+Imp+Reg
foghlaim+Future+Indep+Reg
foghlaim+Conditional+Reg+Dep+1P+Pl
foghlaim+Conditional+Reg+Dep
foghlaim+Conditional+Reg+Dep+1P+Sg
```



```
glan
glan
glanainn
glanamaid
glanadh
glanaibh
glanaidh
ghlanas
ghlanainn
ghlanamaid
ghlanadh
```

Next there is a screenshot of the lookup process, where the FST is tested against a text file to see if it recognises any words, the text file is a test file that I created. In order to make sure that the lookup function will recognise all characters I have to include accented ones in the command - “cat testy.txt | tr -sc "[a-zA-Z0-9|à|ò|ù|ì|è|’]" "[\n*]" |./lookup Dissertation.fst;”



```
bash-4.1$ cat testy.txt | tr -sc "[a-zA-Z0-9|à|ò|ù|ì|è|']" "[\n*]" |./lookup Dis
sertation.fst;

  *****  LEXICON LOOK-UP  *****

glan    glan    +Sg+Imp+Reg
glan    glan    +Future+Dep+Reg

ghlan   glan    +Past+Reg

Meghan  Meghan  +?

dh'fhan dh'fhan +?

dh'ithidh       dh'ithidh       +?


LOOKUP STATISTICS (success with different strategies):
strategy 0:     2 times         (40.00 %)
not found:      3 times         (60.00 %)

corpus size:    5 words
execution time: 0 sec
speed:          5 words/sec

  *****  END OF LEXICON LOOK-UP  *****
```

As you can see the FST failed to recognise the dh’ and therefore did not recognise “dh’fhan” or “dh’ithidh” which it should have done. This is something I only realised in the test phase, otherwise I would have corrected it.

Bibliography

- Gillies, W. (2012) *Scottish Gaelic*. In Muller, N. & Ball, M. (eds). *The Celtic Languages.* Routledge.
- Beesley, K. R. & Karttunen, L. (2003) *Finite State Morphology -- The Book* CSLI Publications. [Online] Available from: http://www.stanford.edu/~laurik/fsmbook/home.html [Accessed: 27th March 2014].
- Watson, A. (2001) *The Essential Gaelic-English Dictionary.* Birlinn.
- Watson, A. (2005) *The Essential English-Gaelic Dictionary.* Birlinn.
- Ui Dhonnchadha, E., Nic Phaidin, C. & Van Genabith, J. (2003) *Design, Implementation and Evaluation of an Inflectional Morphology Finite State Transducer for Irish*. [Online] Machine Translation 18: 173-189. Available from: http://link.springer.com/article/10.1007%2Fs10590-004-2480-9 [Accessed: 27th March 2014].
- Fife, J. (1993) *Introduction*. In M. J. Ball and J. Fife (eds), *The Celtic Languages*, London: Routledge, pp. 3–25
- “Gàidhlig 1B, Oilthigh Ghlaschu, Roinn na Ceiltis is na Gàidhlig, Celtic and Gaelic, Sgoil nan Daonnachdan, School of Humanities” 2013
- Ò Maolalaigh, R. & MacAonghus, I. (2008) *SCOTTISH Gaelic IN TWELVE WEEKS*. Birlinn.
- Karttunen, L. (1993) *Finite-State Lexicon Compiler.* [Online] Available from: http://www.cis.upenn.edu/~cis639/docs/lexc.html#Regex [Accessed: 27th March 2014].
- Jurafsky, D. & Martin, J. H. (2000) *Speech and Language Processing - An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition*. Prentice Hall.
- BBC NEWS. (2013) *Census shows decline in Gaelic Speakers ‘slowed’.* [Online] Available from: http://www.bbc.co.uk/news/uk-scotland-highlands-islands-24281487 [Accessed: 27th March 2014].

- Scottish Express. (2013) *Fury over £2m for Gaelic online dictionary.* [Online] Available from: http://www.express.co.uk/scotland/417584/Fury-over-2m-for-Gaelic-online-dictionary [Accessed: 27th March 2014].
- THE CARNERGIE TRUST FOR UNIVERSITIES OF SCOTLAND. (2009) *An On-Line Part-Of-Speech (POS) Tagger and Gold-Standard corpus of Scottish Gaelic.* [Online] Available from: http://www.carnegie-trust.org/awards/research-grant-projects/an-on-line-part-of-speech-pos-tagger-and-gold-standard-corpus-of-scottish-gaelic-for-research-and-teaching.html#project-description [Accessed: 27th March 2014].
- Trinity College Dublin. (2014) *abair.ie The Irish Language Synthesiser.* [Online] Available from: www.abair.ie [Accessed: 27th March 2014].