The University of York Department of Computer Science

Submitted in part fulfilment for the degree of BEng.

Unsupervised Learning of the Word Morphology of Indigenous Languages

Joseph Leakey

April 30, 2018

Supervisor: Jeremy L. Jacob

Abstract

"Morphological analysis" describes the machine-learning and study of **morphologies**, which formally classify the structures of spoken languages. Various unsupervised methods achieving this are reviewed, and some of the author's observations are considered in the subsequent design of three learning implementations based on these methods. Two of the implementations are extended to test separate theories presented by the author, and all three are also tested to observe how accurately they can infer morphologies from unannotated English; Cornish and Scottish Gaelic corpora. The results of these tests are discussed, with the optimal conditions for "eager suffix-matching" and "entropic locking" — both concepts described in the material — also being judged. The report ends with some of the author's reflections on this project, including their suggestions for further work.

Contents

List of Figures

List of Definitions

- **Morphology**: A classification of strings reflecting the structure of a spoken language
- **Morpheme**: An ordered string defined in a morphology
- **Boundary/split**: A position in a word marking morphemes' ends and divisions
- **Corpus**: A subset of a spoken language's member set
- **Unsupervised learning**: Machine-learning on non-annotated data
- **Stem**: A substring of a whole word beginning from position 0
- **Suffix**: A substring of a whole word ending at the last position
- **Counterpart**: The substring left after discounting a stem or a suffix from a word
- **Inflexion**: The adjustment of a morpheme to change a word's grammatical form
- **Derivation**: The adjustment of a morpheme to change a word's meaning
- **Precision**: The proportion of boundaries in an estimated morphology that are "correct" (with respect to a verified source)
- **Recall**: The proportion of correct boundaries in a verified morphology that also exist in an estimated morphology
- **Resolution**: The average number of unique singular characters that succeed an X-long stem in words across a given corpus
- **Pure method**: An established and unmodified method of morphological learning
- **Indigenous language**: A spoken language that's native to a paricular geographical region

1 Introduction

The (Computer Science) field of morphological analysis aims to enable and advance the machine-learning of spoken languages' **morphologies**, primarily by developing algorithms to infer them from corpora of words. Such learning can either be "supervised" with annotations about words' structures, or otherwise: this project concerns procedures of the latter type.

A morphological learner aims to break down words like the following...

{blind, blinded, blinding, exhaust, exhausted, exhausting}

...into their constituent components...

{blind, blind/ed, blind/ing, exhaust, exhaust/ed, exhaust/ing}

...so that they can then be represented as collections of **morphemes**: meaningful units of language [1].

Prefixes	blind-, exhaust-
Suffixes	-NULL, -ed, -ing

1.1 Motivations

The most obvious application of morphological analysis is in natural language processing (NLP), wherein the recognition of morphemes is necessary to set up inflexive and derivational relationships [2]. For instance: the derivational link between "method" and "methodology" only surfaces after morphological analysis, so the latter may be mistakenly interpreted as an inflexive alteration of the former (as opposed to a word with an entirely distinct meaning) without it.

Beyond this, classifying the morphologies of lesser-spoken languages can help to preserve them. Many indigenous languages are spoken by miniscule populations around the world, and some may be the last living speakers of their respective tongues. Once the morphologies of these languages can be learned, however, their grammars and lexica can then be formalised and kept alive for future generations to learn.

1.2 Limitations

No more than seven months were allocated for this project's completion, so the author wasn't able to review the entirety of the present field in that time. Nevertheless, much of it was still documented in chapter 2. The scope of the tests discussed in chapter 5 was also limited by this definite time-frame.

Tests on indigenous corpora were limited to two written in Cornish [3] and Scottish Gaelic [4], and these were sourced without structural annotations. This meant that any tests measuring precision and recall metrics could only be performed with a pre-annotated English corpus [5]. Tests were also performed with a standard desktop computer, without access to a high-performance workstation machine.

1.3 Direction

The general aims of the project were to explore the present field of morphological analysis and to contribute to it, primarily by designing and implementing original variations (or "extensions") of previous works. Relevant literature was reviewed and used to define the project's practical goals: the resulting implementations were then developed and tested before some conclusions and other closing thoughts were presented.

1.4 Statement of Ethics

As this was a project centered around language, the primary ethical considerations at hand were for **protecting identities** and **the fair use of data**. No datums pertaining to individuals or any other representations of sensitive information were handled, and each of the corpora used in tests were sourced legally and responsibly from public domains [3] [4] [5].

While there are no plans to make this project's implementations available for public use as of April 2018, care will need to be taken in clarifying their capabilities (and crediting anyone who influenced their development) should they ever be released. Although unlikely, the *professional* use of these implementations could cast a heavy influence on the preservation of languages for which existing documents are already sparse: especially if they introduce inaccuracies in doing so.

2 Literature Review

This chapter begins with an overview of the problem domain before discussing the challenges it presents and some of the existing methods for overcoming them.

2.1 Understanding morphologies

A "**morphology**" is a classification of meaningful linguistic units — called "morphemes" — reflecting the structure of a language and how their words are formed [1]. Morphologies are not to be confused with languages (which are systems facilitating complex communications) or dictionaries (which compile and categorise *whole* words), although both of these are supported by morphologies. Whole words are produced by concatenating and merging morphemes, as in the exemplary case of the English word "literature": a concatenation of the stem "liter-" (meaning "writing") and the English suffix "-ature" (signifying "a product of" something).

Morphemes can be of any length and individual morphemes can even pass for words: the latter being especially true in ideographic languages, which associate individual symbols with distinct meanings [6]. With that said, few single-morpheme terms exist in many languages so the computer science field of "morphological analysis" (which can include the present topic of morphological learning) remains significant in both scope and complexity.

The biggest challenges of computational morphological analysis involve the notion of "meaning" and the absolute lack thereof in unannotated strings. Unless the entire domain of meaning covered by a language can be encoded and related to morpheme strings, no machine can infer it: one cannot naturally conclude what the meaning of "litera-" is without some form of manual provision, even if the process of obtaining it isn't supervised itself. Hence, computational morphological learning is limited to heuristic means of inference that recognise morphemes through observations about lingual data and structure.

While the inability to infer meanings from raw strings complicates the present task overall, it also removes any need to distinguish the different meanings of identical morpheme strings. A single morpheme can indeed possess multiple meanings: this is frequently evident amongst occurrences of the English suffix "-er" [7], which completes comparatives (such as "harder" or "faster") **and** agent nouns (such as "worker" or "hunter") alike. However, morphological learners need not be concerned with meaningful distinctions like these for as long as they are unable to infer any meanings at all.

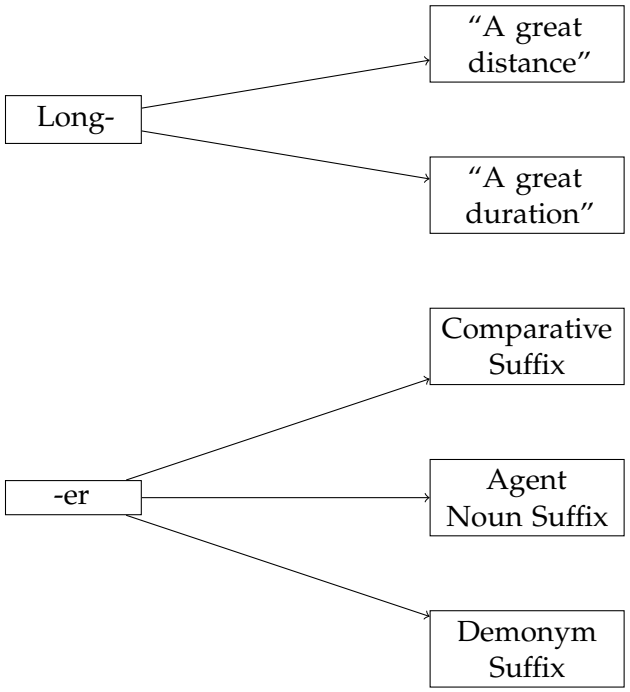


Figure 2.1: A mapping demonstrating that individual morphemes can possess numerous meanings

One side-effect of ignoring meanings obfuscates the distinction between derivational and inflectional morphemes. Derivational morphemes alter words' inherent meanings while inflectional morphemes simply change grammatical forms to reflect persons, numbers, tenses, cases and the like [1] [8]. Some morphemes possess separate derivational and inflectional meanings which are cast aside by heuristic morphological learners:

morphemes like "-er" [7], which can be a derivational agent noun suffix (as in the case of "writ**er**"; **a person who** writes) *or* an inflectional comparative verb suffix (as in the case of "stronger": the comparative form of "strong") amongst other things.

This distinction is nevertheless notable because the various inflectional forms of words with identical derivations are particularly indicative of morphological boundaries. Given the English word "work" and the inflectional past-tense form "worked", it's possible to infer "work-" and "-ed" as two separate morphemes simply by comparing their structures: no semantic knowledge is neeeded to reach this conclusion.

This is exactly the reasoning employed in the unsupervised learning of word morphologies: a process that's entirely analytical without semantic information to hand. It still remains a non-trivial task because a corpus of words could be broken down into constituent morphemes in an extremely large variety of ways, as the following example will demonstrate.

Consider this small set of English words.

{architect, architecture, architected, architecting,
post, posture, posted, posting}

If it's split up like this...

{architect, architect/ure, architect/ed, architect/ing,
post, post/ure, post/ed, post/ing}

...then it's possible to infer this (miniscule) morphology.

Stems	architect-, post-
Suffixes	-NULL, -ure, -ed, -ing

However, merely moving two of these boundaries by singular positions...

{architect, architec/ture, architec/ted, architect/ing,
post, post/ure, post/ed, post/ing}

...will change the morphology significantly, raising stem/suffix counts and introducing spurious morphemes that could be deemed as being invalid by most.

Stems	architec-, architect-, post-
Suffixes	-NULL, -ture, -ted, -ure, -ed, -ing

2.2 Reviewing known methods of unsupervised learning

2.2.1 Methods involving word comparisons

The oldest tool for morphological learning to be covered here is an "analogous" concept coined by F. de Saussure in his "*Course in General Linguistics*" [6]. He defines an "**analogy**" as a structure that relates pairs of words with one-another to establish their commonalities and differences, as is done here with the Latin accusatives "honorem" and "oratorem" and the derived terms "orator" and "honor".

oratorem : orator = honorem : **honor**

Figure 2.2: An analogy that produces the Latin word "honor", as described by F. de Saussure [6]

This example describes the production of "honor" as a replacement for the original term "honos": a transition made to preserve the regularity of Latin [6]. Hence, while analogies do not inherently produce extraneous words, they *are* representations of lingual regularity defining substrings from which other words can then be formed.

The general format of a **Saussurian analogy** is given below. Note that an associated "suffix" can also be prefix or even a circumfix (like in the valid relation between "protect" and "**un**protect**ed**"), but it **cannot** be an infix: the relation between "h**#**ave" ("have") and "h**e**ave" would be deemed to be invalid, even though empty strings are typically deemed to be valid components in analogies [6].

$$\begin{matrix}\text{Stem}_1 + \text{Suffix}_1 : \\ \text{Stem}_1 + \text{Suffix}_2\end{matrix} = \begin{matrix}\text{Stem}_2 + \text{Suffix}_1 : \\ \text{Stem}_2 + \text{Suffix}_2\end{matrix}$$

Figure 2.3: The standard expression of a valid analogy, as defined by F. de Saussure in his "*Course in General Linguistics*" [6]

protect : **un**protect**ed** = affect : **un**affect**ed**

Figure 2.4: An example of a valid analogy in which Stem_1 = "protect", Stem_2 = "affect", $\text{Suffix}_1 = \emptyset$ and Suffix_2 = {"un-", "-ed"}

Once again, note that analogies do not take substrings' meanings into account. This means that valid analogies may link words through recurrent morphemes that are identical in form but varied in meaning. Analogies might also indicate irregular word-forms from time-to-time, though identifying and correcting these isn't of the present task's concern.

solute : **ab**solute = normal : **ab**normal

Figure 2.5: An analogy created in part from a pairing of "-solute" with the prefix "ab-", which means differently in "absolute" to how it does in "abnormal"

fear : fear**ed** = bear : bear**ed**

Figure 2.6: An example of an analogy that produces the irregular past-tense form of "bear" (for which the regular counterpart is "bore")

This Saussurian model can indicate similarities between words that may correspond to morphemes, but it crucially isn't bound to a heuristic that can **discover** said similarities. It is, of course, completely possible to brute-force analogous discovery by comparing each possible pair of words with each other and respectively evaluating all of their possible split configurations. However, this tends to be painfully inefficient — T. Bevan's brute-forcing analogous producer took roughly 2 hours and 45 minutes to evaluate a corpus of just 250 English words [9] — so restricting search-spaces is highly advisable for learning morphologies from particularly large corpora.

Brill [10] sought to do this by exclusively considering analogies mapped around singular stems; identifying sets of suffixes falling within length restrictions (put in place to inhibit spurious relationships) that also shared common counterparts. The notion of an analogy revolving around a single component is mostly sound as longer stems/suffixes are inherently less likely to appear in other words — the longer a substring is, the less of a chance there is that any random sequence of characters will match it — but any stems/suffixes for which all known counterparts are too long will clearly be uncategorisable through this approach.

Meanwhile, Schone and Jurafsky [11] sought to identify and relate sets of stems with recurring suffix combinations (and vice-versa) with "tries" [12]. Tries are defined in more detail within section 2.2.2: what's relevant here is that subtries correspond to groups of suffixes following a single stem [12], so recurring subtries indicate suffix groups that are common to numerous stems. Schone and Jurafsky use sufficiently common suffix-groups to establish hypothetical production rules which can then be refined further, following a process not unlike one proposed by Neuvel and Fulop in a paper on the "*Unsupervised Learning of Morphology without Morphemes*" [13].

In that publication, Neuvel and Fulop adapt the theory of "whole-word morphology" (established by Ford and Singh [14], and itself built upon the Saussurian model) to develop **a procedure for generating words from other words without needing to consider individual morphemes**. While this might appear to depart from the present subject, the "**rules**" entailed by WWM can often indicate morphemes because — much like Saussurian analogies — they're formed from similarities between words.

$$|\,\text{**\#\#\#\#\#ed}\,|_{\text{Verb}} \leftrightarrow |\,\text{**\#\#\#\#\#ing}\,|_{\text{Verb}}$$

Figure 2.7: An example of a whole-word morphological rule [14], relating the English verb suffixes "-ed" and "-ing"

WWM rules are presented as pairs of strings made up of standard characters (indicating required matches); hashes (indicating *required* instantiations); stars (indicating *possible* instantiations) and associated word-types [14]. The rule shown in figure 2.7 states that if a verb contains 7-to-9 characters and ends with "-ed", another verb of 8-10 characters with an identical stem and the alternative suffix of "-ing" can be deemed valid and produceable; **or vice-versa**. For instance: if the English words "moulded" and "accounting" were evaluated against this rule, "moulding" and "accounted" would be flagged as valid words.

Neuvel and Fulop's procedure identifies similar words; the differences between them and the similarities *between those differences* to produce WMM rules of the given format [13]. They provide an example of...

{receive, reception, conceive, conception, deceive, deception}

...being resolved to...

$$|\,\text{*\#\#ceive}\,|_{\text{Verb}} \leftrightarrow |\,\text{*\#\#ception}\,|_{\text{Singular Noun}}$$

. . . in this manner; firstly by identifying the common differentiators "-ive" and "-ption" and then by recognising the common substring "ce-" in the counterpart stems {rece-, conce-, dece-} to indicate the common definite substring "-ception". The auxiliary symbols were then judged by comparing the counterpart stems' lengths.

Neuvel and Fulop reported that their implementation of this process was at least 70-to-82 percent accurate in producing new words from 3000-word samples taken from *Le petit prince* and *Moby Dick* [13], but these samples *were* tagged with word-type annotations. Type information — let alone morphological data — is sparse for indigenous languages, so the effectiveness of this procedure with such indigenous corpora raises interest.

One final comparative learning process of note was proposed by C. Jacquemin in a paper on "*Guessing Morphology from Terms and Corpora*" [15]. Jacquemin's focus on whole-word relations is much like that shown by Neuvel and Fulop, but his (earlier) work sticks closer to de Saussure's analogous principles by emphasising **conflations of multi-word terms**: that being the relation of word groups through textual structure over mere substring recurrences.

By evaluating "windows" of words in structured pieces of literature, Jacquemin's method can infer the similarities and differences relating multi-word phrases like "*sensible conflation*" and "*sensibly conflated*"; recognising the degrees of similarity binding each former and latter pair before matching like-for-like pairs of conflations. Their differences are later expressed and refined as "**signatures**", which are much like WMM rules but for multi-word phrases in this context. According to Jacquemin's definition [15], the signature relating the aforementioned phrases is. . .

$$\{\langle\text{-e, -ion}\rangle, \langle\text{-y, -ed}\rangle\}$$

. . . and indicates that for any two-word phrase with one sequence of suffixes, another phrase with identical stems and the other sequence of suffixes can be produced.

Jacquemin's work is not investigated further in this report as it relies on corpora being structured, and structure corpora can be difficult to obtain for rarer languages. The use of global structure as a matching tool is nevertheless fascinating, and the notion of a signature will yet prove useful in consoldiating words' relationships during one of the learning tasks explored later in the report.

2.2.2 Methods involving local analysis

One of the most well-known morphological learning procedures was set out by Zellig Harris in "*From Phoneme to Morpheme*" [16]. As the title implies, it was adapted from a phonological variant and justified by Harris with similar reasoning: specifically with the notion that points of high successor variation in a string correspond to positions where segmentations would be most suitable.

The **Harrisian segmentation algorithm** has two stages. Firstly, a set of strings are compared so that all unique stems occurring across the set can eventually be mapped to their respective **successor** characters. This requires passing left-to-right through each index in each word and matching each resultant stem to all other words with identical stems, obtaining and mapping their (unique) successors in turn [16]. Evaluating the three stems in "end" against...

$$S = \{\text{eager, earned, end, endless, engender, enjoy}\}$$

...like this would return the mappings...

$$\text{e-} \rightarrow \{\text{a, n}\}_2$$
$$\text{en-} \rightarrow \{\text{d, g, j}\}_3$$
$$\text{end-} \rightarrow \{\text{l, \#}\}_2$$

...wherein the hash symbols indicate complete words' ending points and the subscripted integers indicate unique successor counts.

Once a complete set of stem-to-successor mappings is obtained, the quantities of successors following each point in each word are compiled into "**successor quantity distributions**": the indexes of these SQDs' local maxima (or of any values exceeding a pre-determined "cutoff" value) are then judged to be their associated words' morphological boundary positions [16]. Across the example set S, the SQD of "end" resolves to...

$$\text{SQD}_S(\text{“end”}) = \langle 2, \underline{3}, 2 \rangle$$

...indicating that two successors follow character 1 (after "e-"), three follow character 2 (after "en-") and two follow character 3 (after "end-"). If local maxima were to be inferred as boundary indicators, the index of 2 (after "en-") corresponding to the 3 value in SQD_S("end") would be returned as one morphological boundary position in "end". This same index would also be returned if any SQD values over 2 were inferred as boundary indicators.

Note that in the context of SQD analysis, only the local maxima at the beginning and the end of a plateau — and **not** any in between them — can be inferred as boundary indicators. Furthermore, this does **not** apply for such points that coincide with words' ends. For instance: the local maxima of the SQD...

$$\langle 1, \underline{2}, 2, \underline{2}, 1 \rangle$$

...would be deemed to exist at indexes 2 and 4, but the single local maximum in the alternative SQD...

$$\langle 2, 2, \underline{2}, 1, 1 \rangle$$

...would be deemed to exist at index 3.

The complete Harrisian segmentation algorithm simply runs through this process for each word in a set and returns their respective estimated boundary positions [16]. It's possible to accelerate the initial mapping process by instead compressing corpora into **tries**: depth-wise tree-like representations of string sets, as defined by D. Knuth in "*The Art of Computer Programming*" [12] and advocated for Harrisian analysis by D. Kazakov and S. Manandhar [17].

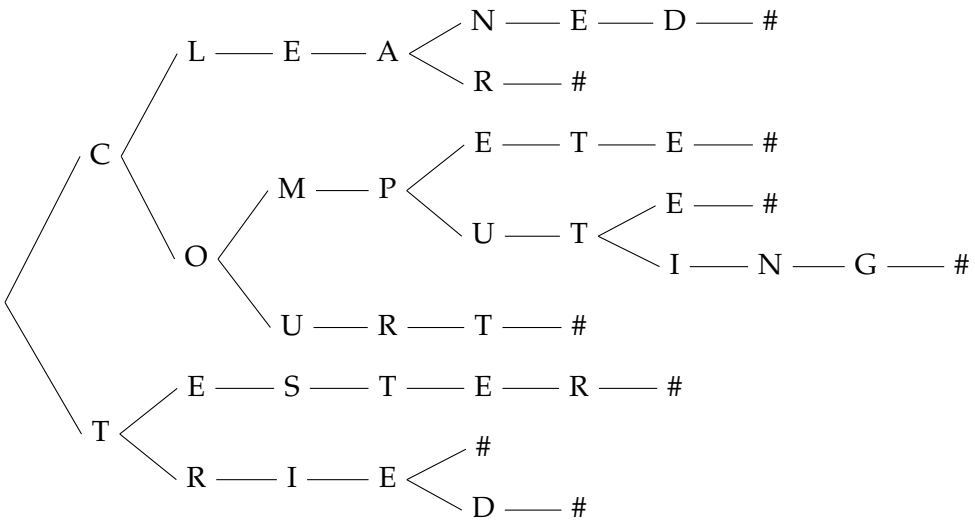


Figure 2.8: An example of a trie [12] encoding the small corpus {cleaned, clear, compete, compute, computing, court, tester, trie, tried}, wherein the hash symbol (#) indicates a word's ending

A trie is a tree of characters in which a character at level X represents the X-long string of characters going to it from the root, and each character's children are the unique characters which can succeed that X-long string across the source corpus [12]. Ergo, whole corpus words correspond to complete trie paths and stems' successor quantities are encoded as branching factors. The "a" node in figure 2.8 represents the stem "clea-" and branches to "n" and "r" nodes, indicating that "clea-" can be followed by either of those characters in the source corpus. It also has a branching factor of 2, mirroring the fourth values in $SQD_{2.8}$("cleaned") and $SQD_{2.8}$("clear").

Harrisian analysis can operate quickly and with decent accuracy if it's implemented well [9], but it's not without issues. M. Hafer and S. Weiss were (respectfully) critical of it in "*Word Segmentation by Letter Successor Varieties*" [18] and pointed out a few shortcomings: most notably that longer stems' successors tend to vary less, meaning that the procedure can have difficulty identifying later splits in words.

To explain why this is, we shall define "**resolution**" as the average quantity of successors following an X-long stem across a corpus. The resolution of a longer stem will never exceed that of a shorter stem with the same initial characters, as the chance for any random sequence of initial characters to match a word's beginning naturally decreases as the size of the sequence increases. More words begin with "e-" than "en-", because the set of words beginning with "e-" is made up of every word beginning with "en-" united with every word beginning with "ea-"; "eb-"; and so on.

I	1	2	3	4	5	6	7	8	9	10
R	9.88	8.65	3.67	1.98	1.39	1.17	1.01	0.91	0.87	0.79

Figure 2.9: The (declining) resolutions (R) measured after the first 10 levels (I) of the trie constructed from a 54626-word English corpus

So as words increase in size and gain more morphemes, their Harrisian trie distributions naturally branch less. This aligns with the notion of a morpheme as a unit of meaning [1]: the more morphemes a word contains, the more specialised it is and the less that meaning can be extended with further morphemes. Hence: if longer stems are less likely to preface multiple successors, the valid suffixes indicated by *some* of

these longer stems are less likely to be signalled by local maxima.

Harris himself did propose **reverse analysis** to negate this issue [16]. This involves defining predecessor mappings for unique suffixes *in addition* to standard successor mappings, prior to passing through each word *in both directions* and merging each resultant pair of boundary position distributions as one sees fit. While reverse-analysis can cover for the weaknesses of forward-analysis in identifying late splits [16] [18], each standard method of merging Harrisian split distributions can compromise results; whether by discarding too many correct splits through intersection (to the disadvantage of a process that can split individual words numerous times over), or by preserving too many incorrect splits through union or crossover [9].

Therefore, room remains for a more optimal modification of the Harrisian algorithm that identifies more late splits correctly. The present paper attempts to justify one in chapter 3.1, making use of "eagerness" (which the original algorithm lacks) to apply observations about some words in splitting up others.

Hafer and Weiss also pointed out that the recognition of straight successor quantities as boundary indicators fails to take *non-unique* successor occurrence quantities into account, meaning that boundaries with many unique successors — but only a few that occur far more frequently than the rest — are always likely to be deemed correct when they may not be (due to the possibility of the lesser-occurring successors hailing from anomalous terms like abbreviations) [18].

To counter this issue with the Harrisian procedure, Hafer and Weiss proposed the use of "**entropy**" as an alternative boundary indicator. In classic information theory, entropy was defined by C. Shannon as the quantisation of the randomness present in a system of many possible outcomes [19]. It's commonly expressed as...

$$H = - \Sigma_{i = 1 \rightarrow n} (p_i \log_2 p_i)$$

...where H is the Shannon entropy of a system; i is one of n possible outcomes and p_i is the probability of that outcome occurring [19]. By calculating the Shannon entropy across the set of successors following a given stem, the in-corpus variation between those successors can be quantised such that H is maximised when each successor occurs equally.

After being modified to calculate **successor entropy distributions** (SEDs) instead and to return the indexes of sufficient entropy values, the Harrisian procedure achieved much better precision and recall metrics in the tests run by Hafer and Weiss [18]. This modification wasn't replicated

in the present project's own Harrisian implementation, but the value of entropy as an indicator of next- and last-letter variation was recognised and used to form the foundation of another implementation.

Other authors have also explored different options for morphological learning via local analysis. H. Déjean's work [20] adapts the Harrisian method to learn morphologies through **frequential analysis**, proceeding by finding the most common stems and endings across an entire corpus and recursively matching *sets* of these with shared counterparts to build up a compositional morphology. Déjean claims that this procedure is particularly adept at differentiating the minor inflectional and derivational differences between words with very similar morphemes: he notes that his implementation identified {start/ed, startl/ed, startl/ing} in one case [20], clarifying the distinction between "start-" and "startl-" in a manner that the Harrisian method would be unlikely to replicate.

D. Bernhard later extended [21] Déjean's work, adapting his method's initial discovery phase to recognise stems and suffixes based on drops in **mean transitional probability distributions**. J. Saffran et. al. [22] defined the "transitional probability" of a character Y with respect to a substring X as...

$$\frac{\#XY}{\#X}$$

...where #S is the frequency of a string S in a given corpus. Given that characters with high transitional probabilities are likely to follow their predecessors, Bernhard observed that boundaries after which no characters possessed dominant transitional probabilities — *boundaries after which no particular characters were likely to follow* — corresponded to points of high successor variation that could suitably bootstrap Déjean's procedure [21].

2.2.3 Methods involving global optimisation

Instead of evaluating individual words with respect to others, some learning methods seek to determine "**configurations**" of morphological boundaries. A configuration is a unique set of boundaries applied over an entire corpus, and learners which manipulate these to optimise corpus-wide metrics are known as "**global optimisation**" methods.

In the present field, global optimisation methods are typically based on J. Rissanen's "**minimum description length**" (MDL) principle [23]. This is the notion that the optimality of data corresponds to its shortest

and/or smallest possible representation(s), and it's adopted by learning methods that seek to compress corpora as much as possible. The notion is justified by the idea that morphemes should be repeatable as distinct units of meaning [1].

J. Goldsmith presents one such learning model [24], which aims to compress corpora into the smallest possible sets of **signatures**. Generally speaking, a signature is a data-structure that compresses morphological information: a very simple one is given in figure 2.10. Goldsmith presents heuristics to produce initial signature sets *and then* refine these sets through careful pointer manipulation, judging each and every proposed change by formulae returning sets' compressed lengths. This process can gradually develop a layered structure including signatures that directly point to other signatures, neatly accounting for invalid combinations (like "bagg-s", with respect to the signature in figure 2.10) and regular forms (like "brok-en", which is the regular form of "break-ed") as a result [24].

block-		
break-		-age
broker-	$\leftrightarrow$	-ing
cover-		-s
coin-		-#
drain-		

Figure 2.10: A simple example of a signature binding a set of stems to a set of suffixes

Two other MDL-based optimisation methods were later presented by M. Creutz and K. Lagus [25]. Instead of linking substring sets through signatures, their work involves the **progressive segmentation** of words and components at optimal positions: judged as such recursively with a complete MDL-based cost function in one case, and linearly with maximum likelihood estimations in another.

The former's cost function accounts for whole source texts *and* for any candidate morphemes yielded, so any candidate splits must *further* compress those texts in order to even be considered over leaving words whole. In this method, an online search algorithm locates and consolidates each component's most cost-reducing splits recursively — occasionally pausing to re-evaluate earlier splits against the collective cost of the morphemes yielded after it — up until no more splits are deemed to be

worthwhile, at which point the resultant configuration is returned [25].

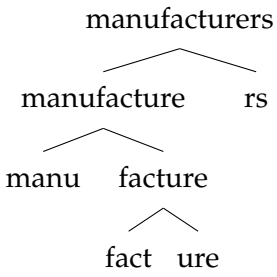


Figure 2.11: A tree representing the morphology of "manufacturers", as would be indicated by the recursive segmentation method defined by Creutz and Lagus [25]

The latter procedure repeatedly and randomly segments words into complete sets of morphemes, estimating their probabilities using a maximum likelihood function to identify the most morphologically likely sets of splits in each word. This function estimates morphemes' probabilities based on how often they occur over whole source texts in proportion to all other estimated morphemes; hence, split configurations yielding morphemes that all occur reasonably frequently will be preferred over more "uneven" alternatives [25].

The last learning method to be reviewed is a **genetic method structured around the naïve theory of morphology** by D. Kazakov [8]. This method is "naïve" because it assumes that all words are comprised of singular stems and singular suffixes [8], meaning that it only seeks to return one-split-per-word and judge the utility of a configuration merely by the summation of its constituents' lengths. Kazakov does note that his procedure can be repeated to identify numerous candidate boundaries for individual words [8]: but, as is promptly discussed, this may not be feasible for particularly large corpora.

Kazakov's algorithm is **genetic**, meaning that it attempts to recombine and mutate populations of "**individuals**" over set numbers of generations to optimise them [26]. The general procedure for genetic algorithms is given in figure 2.13: this was adapted for morphological learning by Kazakov as follows.

The genetic procedure takes a corpus of words and produces a range of randomised mappings from the corpus to index sets. These mappings

are the "individuals" of the procedure, and each word in an individual is mapped to an index between (or equivalent to) 1 and the word's length. Note that no words can be mapped to 0, meaning that boundaries *before* whole words are deemed invalid (but not boundaries *after* whole words) [8]. A mapping inherently yields a set of stems and suffixes, as is demonstrated in figure 2.12.

Word	**Length**	**Boundary**
bl/ind	5	2
bl/inded	7	2
exh/austs	8	3
exhau/sting	10	5
...	...	...
Total Length	30+	

Stems	**Suffixes**
bl-	-ind
...	-inded
exh-	-austs
exhau-	-sting
...	...
10+	18+

Figure 2.12: Table depicting a (partial) individual of word-to-index mappings and their resultant components

Each "**generation**" (or recursive cycle) [26] of the procedure evolves the population through a number of steps. Firstly, each individual's "**fitness**" is determined with an appropriate fitness function. The naïve fitness function used by Kazakov is expressible as...

$$N_{max} - N_{individual}$$

...where N_{max} is the total count of characters across a source corpus, and $N_{individual}$ is the total count of characters across the sets of unique stems and suffixes yielded by an individual [8]. This function would return the fitness of figure 2.12's partial individual as...

$$30 - (10 + 18) = \underline{2}$$

Then, pairs of parent individuals are selected to produce "child" individuals though "**sexual recombination**" (or "crossovers") [26]. Fitter individuals are more likely to be selected as parents at this stage, whether by direct fitness proportions (as in "roulette-wheel" selection [26]) or by some other selection procedure. In a morphological learning process, recombination amounts to merging pairs of mappings using appropriate functions [8].

Once children are born from recombining parents, they're randomly **mutated** [26]. In this context, a mutation is a random one-index shift in a

word's mapping [8]. To demonstrate: mutating the mapping of "blind" in figure 2.12 could change the word's mapped index to 1 or 3, depending on the randomised direction of the mutation. To preserve any fitness gains obtained through recombination, mutations typically occur with low probabilities [26].

Finally, the new population's fitnesses are evaluated and the process is repeated over a set number of generations. The healthiest observed individual is returned at the very end, as shown in figure 2.13.

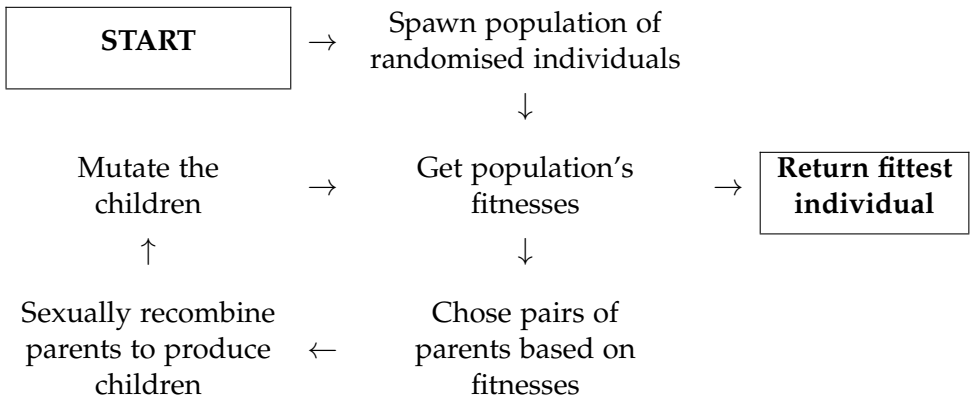


Figure 2.13: The general procedure undertaken by a genetic algorithm [26]

Along with mapping only one split to each word, Kazakov's method — while sound as an MDL-based process — can struggle to evolve particularly large individuals. Each one has to be at least as large as the corpus spawning it, so reasonably-sized populations can take *lots* of time to recombine; mutate and measure for fitness values in turn. This is also to say nothing of how many single-mapping configurations there can exist across even reasonably-sized corpora, for which even just a fraction could take *many* generations for a randomly-directed procedure of this kind to evaluate.

Given how time-complex it can be, it's understandable why Kazakov chose to design this genetic procedure around a naïve fitness function: a more complex variant wouldn't be feasible for all but the smallest of corpora. Goldsmith nevertheless points out [24] that naïve learners may be more inclined to return incorrect segmentations if they can compress corpora further, so tests with alternative MDL-based cost functions (like those covered in this chapter) may be worth following up on.

3 Problem Definition and Task Design

The remainder of the paper documents the design; implementation and testing of the morphological learning programs mandated by the project. With no specific goals for the present author to attain beyond those implied by the project's title, the decision was taken to **structure practical work around proposed extensions to two reviewed learning methods and tests of another with indigenous corpora**.

As the author, I personally justified this direction as a reinforcement of the key concepts learned from the literature review *and* as an opportunity to review my own observations around those concepts. Such observations require rigorous analysis and empirical evidence to truly hold, and I particularly appreciated that fact as a relative newcomer to the present field.

The three learning methods implemented and extended here align with the three main domains of morphological learning: through word comparisons, through local analysis and through global optimisation. There are many methods in all three domains, including some which haven't been reviewed in this paper for the sake of time. Those chosen to be extended and tested were picked based on how "extendable" they are and on my ideas/criticisms, which are discussed further as I propose and justify each individual extension throughout this chapter.

All original algorithms and extensions were subject to appropriate tests, measuring metrics from precision and recall to fitness gains and generation accuracies depending on each test's context. These testing procedures and the corpora used in them are discussed in further detail throughout chapter 5.

The remainder of this chapter will present and attempt to justify the foundations of my work.

3.1 Task 1: Extending Harris' local segmentation method to perform eager suffix-matching

In chapter 2.2.2, the Harrisian segmentation method [16] was discussed. It was emphasised that while the original algorithm is based on solid reasoning derived from similar work in phonological analysis [16], the low "resolutions" of longer stems presented problems in identifying splits nearing longer words' ends. Because longer stems tend to possess fewer successors, later boundaries can be difficult to identify in certain words and the original procedure's accuracy suffers for it.

Reverse analysis was also brought up as a "fix" for this problem, but merging splits obtained from forward- and reverse-analysis can be problematic for the quality of certain estimations. Therefore, a different solution — one aiming to exploit the Harrisian procedure's resolution drop-off for an advantage — was explored instead.

If longer stems are followed by fewer morphemes, then it stood to reason that **any** variations following them are more likely to indicate actual morphological boundaries. This is because smaller groups of successors are less likely to share similarities, meaning that they're more likely to pass for morphemes and the boundaries indicating them are more likely to be correct.

This reasoning isn't intuitive, so it will be demonstrated. Consider the words...

{compute, computing, court}

...taken from the example trie given in figure 2.8. The prefix "co-" is common to all three strings and is followed by "m" and "u" across the set, but it doesn't actually align with the proper stems "com-" and "court": the variation only exists because those two morphemes share the same first two letters. Now consider the stem "comput-", which is followed by the suffixes "-e" and "-ing". These marks the last variation points in both "compute" and "computing" respectively, and both suffixes are indeed proper.

This is clearly a very small example, and average suffix-counts-per-stem can increase with corpora sizes - meaning that similarities may still tie even the shortest suffixes together in larger corpora - but the point remained. **The theory suggested that words' final variation points are likely to indicate their suffixes**, so an implementation of the Harrisian segmentation algorithm was created and extended to perform "**eager suffix-matching**" (ESM) for the purpose of verifying it.

3.1.1 Eager Suffix Matching

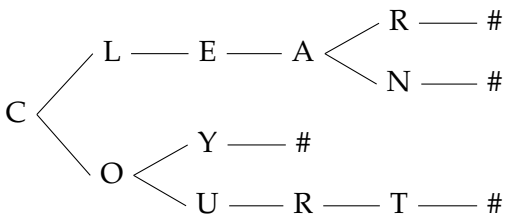
In machine-learning, the property of "**eagerness**" is exhibited when information obtained from one phase of learning is used to assist another [17]. In morphological learning, eagerness can henceforth be described as the use of auxilliary word data to refine the analysis of further unseen words.

The counterpart to eagerness is "**laziness**" [17], and Harris' algorithm qualifies as being lazy because it operates exclusively on a local level: when a word is being segmented, only that word is considered [16]. Successor quantities are derived from other words, but their respective splits are never considered again after being estimated.

By contrast, ESM **does** consider other words' SQDs when determining final splits. This simple procedure operates as follows...

- Locate each word's final point of variation
- Record the (unique) suffix produced by the substring going from that point to the end of the word
- Order the list of eagerly-obtained suffixes by their lengths (from the longest to the shortest) **or** by their occurrence counts (from the most common to the least common)
- For each word in the corpus, find the first suffix in the list that can be matched to it and record the resultant boundary position
- Evaluate each word as usual up until their re-matched suffixes

To better understand how ESM operates, consider the trie shown below.



With ESM, the following suffixes would be eagerly obtained from this small trie and then procedurally re-matched to each word in the associated corpus. Re-matches would be biased towards longer or more frequently-occurring suffixes: both biases were tested in the experiments documented throughout chapter 5.2.

{-urt, -r, -n, -y}

The hypothesis suggested that ESM should identify suffixes more effectively than forward-analysis, while also preserving the utility of the standard Harrisian procedure in identifying words' earlier boundaries. To see how this held up, check chapter 5.2.

3.2 Task 2: Using entropy to influence and accelerate Kazakov's global optimisation method

In chapter 2.2.3, Kazakov's naïve genetic optimisation method [8] was covered. Although the naïve theory of morphology aligns neatly with the MDL principle's focus on data compression [23], genetic algorithms are ill-suited for evolving *extremely* large arrays of index mappings. This was evident in the performance of Kazakov's genetic implementation, which took eight-and-a-half hours to evolve 120-word individuals over 2000 generations in one case [8].

The focus of this task was therefore on optimising this promising learning method so that it could be performed on larger corpora. One clear-cut way of doing this is by restricting the algorithm's **search-spaces**: artificially limiting pools of evaluatable configurations such that fitter individuals can be found in fewer generations. In a randomised process of this kind, restrictions are achievable by **locking** mappings deemed to be optimal by a certain metric, such that they cannot be further mutated afterwards.

With some way to infer the optimality of a mapping like...

{doing $\rightarrow$ 2 {"do-", "-ing"}}

...it could be preserved across generations, thereby limiting the enclosing individual's search-space *and* sustaining the individual's fitness. Given that the naïve fitness function benefits from segmentations around points with maximal next- and last-letter variations [8], the naïve optimality of a boundary could therefore be quantified by a total measure of the variation around it... and this is suitably provided by **entropy**.

3.2.1 Entropic locking of boundaries

As mentioned in chapter 2.2.2, Hafer and Weiss [18] experimented with using successor entropies [19] to judge Harrisian splits. Given an English

corpus, the successor entropy of a stem (s) occuring in that corpus is defined as...

$$SH(s) = -\sum_{i=1\to 26} (p_i(s + \alpha(i)) \log_2 p_i(s + \alpha(i)))$$
$$(PH(s) = -\sum_{i=1\to 26} (q_i(\alpha(i) + s) \log_2 q_i(\alpha(i) + s)))$$

...where SH quantifies the successor entropy of s; i is an index in the ISO basic Latin alphabet [27]; $\alpha(i)$ is the corresponding letter; and $p_i(s + \alpha(i))$ is the proportion of non-unique occurrences of the concatenated stem $(s + \alpha(i))$ with respect to all possible $(s + \alpha)$ occurrences [19]. The corresponding definition of **predecessor entropy** (PH) is also given, wherein $q_i(\alpha(i) + s)$ instead refers to a proportion of non-unique *suffix* occurrences. To clarify the meaning of $p_i(s + \alpha(i))$: across the set...

{steak, steal, steam, stem, stemmed}

... $p_i(stea) = 0.6$ and $p_i(stem) = 0.4$.

The distribution of successor entropies across a word peaks when many successors follow a stem equally as often as one-another [18]. The same is true for predecessor entropies when many unique characters precede a suffix equally as often. A boundary exhibiting significant next- and last-letter variations will correspond to high successor and predecessor entropies respectively, and the naïve theory of morphology suggests that these are also the boundaries which *should* maximise compression [8].

Ergo, the resulting hypothesis suggested that locking mappings to boundaries exhibiting high **collective entropy** — that being the sum of the successor and predecessor entropies around a given boundary — *should* restrict individuals' search-spaces to retain more optimised configurations, thereby accelerating fitness gains throughout the process and enabling fitter individuals to be discovered within fewer generations. To test this, an original recreation of Kazakov's implementation was developed and extended to perform **entropic locking**: this is documented in chapter 4.3.

$$CH(word, i) = SH(word_{1\to i}) + PH(word_{i\to LEN(word)})$$
$$1 \leq i \leq LEN(word)$$

Figure 3.1: The definition of collective entropy, wherein $word_{x\to y}$ is a substring of the word going from position x to position y

When an entropic locking threshold (ELT) is set, this new program will calculate each encountered boundary's collective entropies (CEs)

according to the definition given in figure 3.1. If a mapping's CE is found to exceed the ELT, it will be locked and safeguarded from mutations until the evolutionary procedure is completed or until the mapping is replaced through a crossover.

Beyond the stated hypothesis, it also remained to be seen whether further fitness gains by this genetic procedure could actually translate into more accurate outputs. To root this out, outputs' precision and recall values were measured in addition to fitness gains during the tests documented in chapter 5.3.2.

3.3 Task 3: Implementing Neuvel and Fulop's analogous word production method for indigenous corpora

The final task of the project was somewhat simpler than the other two but remained necessary as an exploration of the present field. Here, the aim was to observe whether Neuvel and Fulop's unique method for *producing* words [13] could remain useful when applied to indigenous corpora in which linguistic information may be incomplete. This method offers great potential in identifying unknown words from others, which can be particularly useful in aiding linguistic understanding and discovery.

After Neuvel and Fulop's WWM-based [14] production algorithm was implemented, it was applied to independently-randomised corpora subsets and all resultant rules and generations were observed. Generated words were also compared with all complete corpora to judge the accuracy of the implementation: more information about this is presented in chapter 5.4.2.

It should be clarified that this project's implementation doesn't bind word-types to rule components, as the WWM theory typically mandates [13] [14]. This aspect was touched upon in chapter 2.2.1 and wasn't carried over because word-type information is sparse for indigenous languages, as is evident with the corpora used for chapter 5's tests.

4 Implementation

All three of the implementations discussed in this section were written from scratch by the present author. To save space, output logs are given in the place of actual code.

4.1 A note on programming languages

Each of the learning programs documented here was written in **Python**: a lightly-structured, type-free language that's ideal for scripting purposes. If they were being developed for professional purposes, more research and care would have gone in to choosing a language with fewer overheads and more features to complement machine-learning. However, the present author felt that it was ultimately much more practical to spend time developing implementations over learning unfamiliar languages for small computational gains: especially given that none of the tests documented in chapter 5 were *primarily* concerned with run-times, and that proportional run-time *differences* were arguably more useful when it came to comparing extensions and methods.

4.2 Implementation 1: A Harrisian learner supporting eager suffix-matching

This first implementation recreated Harris' local analysis method [16], offering means to create nodes corresponding to characters and to link them together into Harrisian tries. While the code behind it is of my own creation, the underlying structure is loosely based on the work of T. Bevan [9].

Like in his program, *Node* objects are set up to store representative characters upon instantiation. Nodes also possess "child" lists containing their children, and trie structures can be set up by instantiating child nodes within their parents' successor sets. If a whole string of characters is to be appended to a node — as may be the case with a trie's root node — then the implementation will recursively append nodes representing

each character to their predecessors' child-sets, prior to appending a null node (indicating a word's ending with the "#" symbol) after the final character. This is demonstrated visually by figure 4.1.

Full corpora can be represented as tries by instantiating specialised root nodes and subsequently appending their members with the *appendstring()* method. If an attempt is made to re-append an existing child to a node, it won't be duplicated but it may still be accessed to append new suffixes: this ensures that words with identical stems can be represented without duplicating their commonalities.

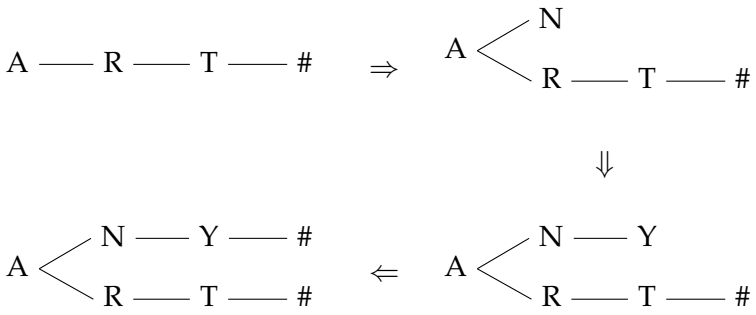


Figure 4.1: A visual demonstration of how a word can be added to a trie structure

The global *Harris()* method conducts Harrisian analysis exactly as it was described in chapter 2.2.2: by mapping each observable stem in a corpus to their unique successors (through the construction of a trie in this case) and estimating resultant morphological boundary positions through observations on all words' successor quantity distributions. This implementation estimates boundary positions from **local maxima**, which runs counter to the original algorithm's use of "cutoff" thresholds [16] [18]: local maxima sampling is utilised instead so that peaks in successor variations occurring at low-resolution[1] indices can be recognised as potential boundary locations.

To facilitate eager suffix-matching, a *finalbranch()* method was written for the *Node* class. This returns the index of the final branch in the trie path corresponding to a supplied word (and going out from the addressed node). The aforementioned *Harris()* method contains a boolean-type *ESM*

[1] "Resolution" is defined in chapter 2.2.2 and in the preliminary list of definitions.

parameter that — when True — will perform Harrisian analysis on a corpus *with preliminary suffix-matching*, as it is defined in chapter 3.1. A further *frequencymatching* integer-type parameter can impose a floor on the lengths of candidate suffixes obtained through this process, such that any eagerly-obtained suffixes shorter than *frequencymatching* in length won't be recorded for frequency-wise suffix re-matches. When this latter parameter is set to 0, length-wise re-matching is performed instead.

4.2.1 Demonstration

This exeuction log demonstrates the functionality of the core *Node* class, which supports the construction of tries. Here, a root node is established and a list of words are iteratively appended to produce the structure shown in figure 2.8. From this root, any and all can be accessed and words' successor quantity distributions can be calculated.

```
>>> corpus = ["cleaned", "clear", "compete", "compute",
   "computing", "court", "tester", "trie", "tried"]
>>> root = HarrisNode.start()
>>> root.appendstrings(corpus)
>>> print(root)
START | ['c', 't']
>>> print(root.trie())
('START', [('c', [('l', [('e', [('a', [('n', [('e', [('d',
   ['#'])])]), ('r', ['#'])])])]), ('o', [('m', [('p',
   [('e', [('t', [('e', ['#'])])])]), ('u', [('t', [('e',
   ['#']), ('i', [('n', [('g', ['#'])])])])])])])]), ('u',
   [('r', [('t', ['#'])])])])]), ('t', [('e', [('s', [('t',
   [('e', [('r', ['#'])])])])])]), ('r', [('i', [('e', ['#',
   ('d', ['#'])])])])])])

>>> print(root.child("c"))
c | ['l', 'o']
>>> print(root.child("trie"))
e | ['#', 'd']

>>> print(root.distribution("computing"))
[2, 2, 1, 2, 1, 2, 1, 1, 0]
>>> print(maxima(root.distribution("computing")))
[2, 4, 6]
>>> print(segment("computing",
   (maxima(root.distribution("computing")))))
['co', 'mp', 'ut', 'ing']
```

```
>>> print(root.finalbranch("computing"))
6
```

4.3 Implementation 2: A genetic learner supporting entropic locking

The second implementation recreated Kazakov's genetic optimisation method [8], which gradually evolves populations of singular-boundary mappings over set numbers of generations. This is the largest of the three presented here and was only guided by the general framework shown in figure 2.13: ergo, any similarities between it and Kazakov's original implementation are purely coincidental as the present writer has not witnessed the latter.

It's structured around an *Individual* class, which encapsulates a mapping between words and singular boundary positions in the form of a dictionary. Individuals behave as they do in the original algorithm, which was described in detail during chapter 2.2.3: the internal *shift()* method facilitates mutations and the internal *fitnessabsolute()* method returns individuals' naïve fitness values.

Word	**Index**
sky/ward	3
theor/ies	-5
r/esident	1
inte/rrupt	4
...	...

Figure 4.2: A partial mapping encapsulated within an *Individual*, wherein the boundary mapped to "theories" is locked in place

Entropic locking is supported as an option, and a few methods were written to enable this. The *Individual* class' internal *lock()* method can lock a mapping in place by inverting the associated index's sign, as negative indices signify locked mappings in this system. This representation holds up because splits at zeroth positions are not permitted, as per Kazakov's original implementation [8].

The global *prefixentropy()*, *suffixentropy()* and *collectiveentropy()* methods were also written to calculate splits' collective entropies (as they are

defined in chapter 3.2). These are only ever called by the primary *genetic()* method if the corresponding *entropythreshold* parameter is set to any value above zero.

The main *genetic()* method enacts Kazakov's genetic learning procedure, which operates by spawning a randomised population of individuals and...

- ...calculating their respective fitnesses...
- ...selecting parents through **tournaments**...
- ...breeding their children through **uniform crossovers**...
- ...and randomly mutating their children...

...iteratively over a set number of generations [8] [26]. If entropic locking is enabled, boundaries exhibiting sufficient collective entropies are locked in place just before the rest are mutated.

Parents breed **uniformly**, meaning that they randomly intertwine their respective mappings to produce their children. Parents are also selected through **tournaments**: groups of randomly-selected individuals in which the fittest qualify for sexual recombination [28]. These tournaments are upheld over more conventional "roulette-wheel selections" (across probabilistic distributions that are proportional to individuals' respective fitnesses [26]) because variations in fitness values tend to be slight, and tournaments ensure that due biases can be lent towards selecting healthier individuals.

4.3.1 Demonstration

This execution log demonstrates the utility of the *Individual* class at the heart of the genetic procedure. It begins with the spawning of an individual with random mappings, and then demonstrates a boundary being shifted; another being locked in place; and the individual's fitness being calculated. The calculation of a stem's successor entropy is also shown.

```
>>> corpus = ["chanta", "chantai", "chantais", "chantait",
   "chanter"]
>>> ind = Individual(corpus)
>>> print(ind)
chan|ta [4]
```

```
chantai| [7]
cha|ntais [3]
cha|ntait [3]
chante|r [6]

>>> ind.shift("chantai", -1)
'chanta|i'

>>> ind.lock("chantait")
>>> print(str(ind.boundary("chantait")))
-3
>>> print(str(ind.boundaryabsolute("chantait")))
3
>>> ind.shift("chantait", 1)
'chantait': Boundary has been locked at position [3]

>>> print(ind.n())
33
>>> print(ind.nmax())
36
>>> print(ind.fitnessabsolute())
3

>>> print(stementropy(corpus, "chant"))
0.7219280948873623
```

4.4 Implementation 3: A WWM-based word-production tool

The final implementation recreated the WWM production tool established by Neuvel and Fulop [13], complete with means of inferring rules from existing corpora. It can perform comparisons to infer "**relationships**", which are henceforth defined in this context as pairs of stems or suffixes sharing groups of comparisons. It can then construct WWM rules [14] from the similarities and differences across these relationships, and these can then be applied to generate new words from existing ones. An example of a suffix-linking relationship is given in figure 4.3.

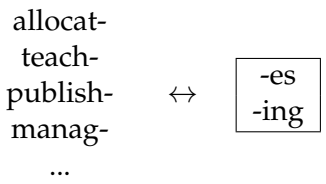


Figure 4.3: A simple example of a relationship linking the suffixes "-es" and "-ing"

It was previously touched upon that information about word-forms is sparse for indigenous languages, so the decision was taken to ignore them when constructing WWM rules. This means that any pair of stems or suffixes with at least two common counterparts can form the foundation of a rule, regardless of their inflections and/or derivations.

The implementation infers relationships as follows. For each position in each string throughout a corpus, it obtains a candidate stem-suffix pair and then attempts to match both the stem and the suffix to all other equally-long stems and suffixes across the corpus. If **both** the stem and the suffix each occur more than a specified number of times, they'll be recorded in global stem/suffix sets respectively.

Once all candidate stems and suffixes are obtained, each unique pair of stems (and subsequently each unique pair of suffixes) will be checked for relationships. If more than a specified number of counterparts can be matched to **both** stems/suffixes in any given pair, that relationship will be recorded and a WWM rule will be produced. Each and every WWM rule can then be tested against every word in the original corpus, and any "new" words can then be saved if they're produced.

The *Relationship* class was written to provide a basic structure for relationships, and the global *relationships()* method is the one that uncovers them. This class includes an internal *rule()* method which produces WWM rules (as two-tuples of strings in the format detailed during chapter 2.2.1 [14]) from relationships, and the global applyrule() method tests these against words: returning modified strings for words that can be matched to rules, or nothing for those that can't.

4.4.1 Demonstration

This is a log of the outputs produced by the implementation in response to some basic calls. The *relationships()* method searches for relationships

in the provided set: the "1" argument specifies that a stem must be matchable with at least 1 *other* word (and likewise for a corresponding suffix) for a pairing to be recorded, and the "3" argument specifies that a relationship must tie a pairing to at least three counterparts before it can be returned.

```
>>> relset = relationships(["receive", "reception",
   "conceive", "conception", "deceive", "deception",
   "honor", "honorem", "orator", "oratorem", "bake",
   "baked", "charge", "charged"], 1, 3)
>>> print(relset[0])
Ending A: -ive
Ending B: -ption
Stems: ['rece', 'conce', 'dece']

>>> print(relset[0].rule())
('*##ceive', '*##ception')

>>> applyrule("perceive", relset[0].rule())
perceive -> per[ceive] -> perception | ('*##ceive',
   '*##ception')
'perception'

>>> applyrule("perception", relset[0].rule())
perception -> per[ception] -> perceive | ('*##ceive',
   '*##ception')
'perceive'

>>> applyrule("preconceive", relset[0].rule())
# Nothing returned: "preconceive" is too long to
# be matched

>>> applyrule("ceive", relset[0].rule())
# Nothing returned: "ceive" is too short to be
# matched

>>> applyrule("wrong", relset[0].rule())
# Nothing returned
```

5 Testing and Evaluation

5.1 Testing resources

Three corpora containing English, Cornish and Scottish Gaelic words were sourced for the following experiments. These languages were chosen for analysis based on the availability of contents written in them, which varies from language to language.

Morphological data for indigenous languages is extremely sparse, so only the English corpus was annotated with constituents' morphological boundary positions. These are represented by the indexes of their positions: for instance, "document/ation" was annotated with the number 8 to indicate that a boundary exists just before the word's eighth letter (where "d" is the zeroeth letter). Such annotations were obviously not made available to the learning programs used and were only considered for precision and recall value calculations.

The (American) English corpus was sourced from the *English Lexicon Project* by Washington University in St. Louis [5] and contained 54626 words in total. The Cornish corpus was sourced from the Cornish Language Partnership's *Cornish Dictionary* [3] and contains 9645 words in total. The Scottish Gaelic corpus was sourced from the University of Glasgow's *Digital Archive of Scottish Gaelic* [4] and contains 230976 words in full, although this was filtered down to a randomly-selected subset of 5028 words for the sake of manageability. This Scottish Gaelic corpus did contain a non-zero quantity of English words that were added by accident, though the vast majority of these were filtered out using the aforementioned English corpus.

The English corpus was also reduced to independently-randomised subsets of 100; 500; 1000 and 500 words: all of which were then left unchanged (and with their boundary annotations preserved) for all tests thereafter, excluding experiment 4.

All tests were run with a machine powered by an *AMD Ryzen 7 1700* CPU clocked to 3.7GHz across all 8 cores. This is considerably less powerful than a typical research-grade machine but it proved to be sufficient for the tests discussed here.

5.2 Experiment 1: Measuring the utility of eager suffix-matching on an English corpus prior to Harrisian analysis

5.2.1 Methodology

Each English corpus was segmented by implementation 1 without eager suffix-matching (ESM); with length-wise ESM; with frequency-wise ESM; and again with frequency-wise ESM but with an additional 2-character floor imposed on matchable suffixes' lengths. The resulting segmentation listings were compared against the boundary annotations in the original corpora to judge their precision (P); recall (R) and last-split match (LM) percentages, where the last property is defined as the proportion of words for which their predicted and "correct" final boundary positions are identical.

5.2.2 Results

Size	Without ESM				With ESM (By Length)			
	P (%)	R (%)	LM (%)	T (s)	P (%)	R (%)	LM (%)	T (s)
100	11.60	8.39	3.00	0.016	10.95	9.68	4.00	0.016
500	12.59	9.58	3.40	0.078	11.64	14.17	4.80	0.078
1000	17.35	13.92	6.80	0.063	12.98	15.96	5.30	0.219
5000	24.59	24.39	17.22	0.500	18.22	23.36	9.30	1.922
54626	37.53	54.16	49.40	5.641	22.98	33.03	16.15	32.906

Size	With ESM (By Freq. [ML = 1])				With ESM (By Freq. [ML = 2])			
	P (%)	R (%)	LM (%)	T (s)	P (%)	R (%)	LM (%)	T (s)
100	11.51	10.32	4.00	0.016	11.51	10.32	4.00	0.016
500	22.68	29.66	29.40	0.063	21.49	27.95	27.40	0.063
1000	39.25	53.97	65.90	0.141	25.58	34.08	33.70	0.172
5000	40.07	61.67	68.94	0.688	32.69	48.39	47.90	0.479
54626	37.26	67.85	52.69	8.484	31.91	53.90	42.30	9.219

Figure 5.1: The full set of results for experiment 1, showing precision (P); recall (R) and last-split match (LM) percentages for each estimated split configuration and the amount of time (T) taken by the Harrisian implementation to produce them

In general, it appears that the accuracy of the Harrisian process scales up with word-counts; as does the utility of frequency-wise ESM, **but only to a certain limit**. The latter observation is perhaps the biggest takeaway from these results.

In both sets of segmentations obtained with frequency-wise matching, the proportions of "correct" suffix boundaries identified across the 5000-word corpus (*ML1* $\rightarrow$ *68.94%; ML2* $\rightarrow$ *47.90%*) are higher than those identified across the full-size lists (*ML1* $\rightarrow$ *52.69%; ML2* $\rightarrow$ *42.30%*), indicating that the performance of frequency-wise ESM doesn't scale linearly with corpora sizes and eventually plateaus. This is likely because it gravitates towards matching more common suffixes (such as "-ed" and "-ing"), whereas larger corpora are more likely to include more specialised words with rarer suffixes that probably wouldn't be matched correctly in this manner.

ESM may also be less useful with larger corpora because — being inherently more varied — stems occurring in them are more likely to be paired with multiple suffixes, meaning that late splits in their Harrisian tries (corresponding to suffix boundary positions) will be more commonplace and easier to identify as local maxima. "Condemn/ing"; "condemn/ation" and "condemn/ed" are all in the full corpus and can be suffix-matched there through local maxima analysis, whereas only the last word exists in the 5000-word set and requires ESM to be suffix-matched with the "-ed" suffix correctly.

The effectiveness of length-wise ESM — or lack thereof — is also notable. Across all metrics, it returned almost entirely inferior results compared to the other three procedures; perhaps because the algorithm inferred longer suffixes from more unique words (with fewer trie splits) and mistakenly matched them to words sharing those longer endings but possessing shorter suffixes in actuality. This appears to be the case when considering the word "innovator", for which the suffix "-or" was identified correctly by the pure non-matching algorithm but mistaken for the longer "-ator" suffix by the length-wise ESM procedure.

Beyond ESM: **the recall metrics are roundly higher than the precision metrics**, suggesting that the Harrisian algorithm tends to "over-predict" boundary positions where none actually exist. This is subverted for the 100-word corpus, probably because the comparative lack of words translates into fewer trie splits and fewer identifiable split positions in turn. Run-times were mostly neglible, though length-wise ESM does appear to slow the process substantially by evaluating longer (and therefore less frequent) potential suffix matches first.

5.3 Experiment 2: Measuring the utility of entropic locking in a naïve genetic global corpus optimisation process

5.3.1 Methodology

Implementation 2 was run with various combinations of ELTs (entropic locking thresholds) and random mutation probabilities. Runs with the full-size English corpus evolved 8 individuals over 20 generations born from tournaments of 3, whereas runs with the smaller corpora evolved 32 individuals over 100 generations born from tournaments of 8. All runs' fitness distributions were recorded and their healthiest individuals were returned as split configurations to be compared against their respective originals (yielding the precision and recall metrics in figure 5.2).

Because the genetic process is somewhat non-deterministic [8], each and every run probably would have returned different split configurations if they were repeated: bear this in mind when considering these results. With that said, repeated runs of certain configurations returned very similar fitness distributions so the decision to avoid repeating all runs (in the interest of time) was made under the assumption that repeats would yield similar metrics.

5.3.2 Results

The figures obtained from this experiment are too plentiful to evaluate in detail, so analysis will again be limited to general trends.

Generally, **it takes more generations to return optimal split mappings for larger corpora** because there are more split permutations on larger individuals. The results of runs performed *without* entropic locking demonstrate this as a trait of the pure[1] algorithm by presenting an observable decline in accuracy metrics for increasingly large corpora, though it's worth pointing out that the full-size corpus was only evolved over 20% of the generations that the rest experienced. Even so, the relative fitnesses gains observed during the pure run on the full-sized corpus rose at a much slower rate than with the smaller corpora; and if individuals' fitnesses do indeed reflect their accuracies (which appears to be likely given these results), then it would have had to have been evolved over far more than 100 generations to produce a reasonably healthy child without

[1] A "pure" run of the genetic implementation ignores entropic locking.

entropic locking.

No Ent. Locking (p(Mut.) = 1%)			
Size	**P (%)**	**R (%)**	**T (s)**
100	26.00	16.77	6.56
500	26.80	17.59	28.56
1000	23.00	14.62	72.30
5000	18.98	12.21	551.97
54626	17.70	11.45	1605.95

Ent. Locking Threshold = 5 (Size = 54626)			
p(Mut.)	**P (%)**	**R (%)**	**T (s)**
66	24.64	15.93	4937.20
100	32.16	20.80	5554.25

Entropic Locking Threshold = 4									
Size	**p(Mut.) = 1%**			**p(Mut.) = 66%**			**p(Mut.) = 100%**		
	P (%)	**R (%)**	**T (s)**	**P (%)**	**R (%)**	**T (s)**	**P (%)**	**R (%)**	**T (s)**
100	32.00	20.65	7.44	22.00	14.19	7.03	22.00	14.19	7.78
500	25.80	16.92	39.26	18.00	11.81	34.86	20.00	13.12	41.63
1000	22.20	14.11	82.64	19.90	12.65	88.63	20.20	12.84	92.38
5000	19.38	12.47	676.50	26.44	17.01	695.69	25.72	16.55	645.27
54626	18.40	11.90	4324.42	38.34	24.79	4885.47	41.49	26.64	4782.25

Entropic Locking Threshold = 3									
Size	**p(Mut.) = 1%**			**p(Mut.) = 66%**			**p(Mut.) = 100%**		
	P (%)	**R (%)**	**T (s)**	**P (%)**	**R (%)**	**T (s)**	**P (%)**	**R (%)**	**T (s)**
100	43.00	27.74	7.08	51.00	32.90	6.73	53.00	34.19	6.08
500	27.20	17.85	31.27	51.00	33.47	30.38	51.60	33.86	34.58
1000	28.70	18.25	81.47	50.20	31.91	63.92	53.00	33.69	75.59
5000	26.76	17.22	674.66	44.00	28.31	592.31	44.30	28.50	534.67
54626	18.46	11.94	4583.76	38.64	24.99	5039.55	39.94	25.83	6601.88

Figure 5.2: The split-comparison results of experiment 2, showing precision (P) and recall (R) percentages for each returned individual and the amount of time (T) taken by the genetic implementation to produce them

Outputs' proportional accuracies generally decreased as corpora sizes increased, but the full-size and 5000-word outputs yielded proportional *increases* over the smaller sets when the ELT (entropic locking threshold) was 4 and p(Mutation) was *at least* 66%. The full-size output was especially healthy in this case, given that it was produced after just 20 generations. For such larger inputs, it would appear that **this ELT of**

4 is quite close to a "sweet spot" that's high enough to lock at points of sufficient variation but low enough to prevent probable boundaries from shifting; thereby returning the greatest possible proportions of probable boundary positions for those larger lists. Furthermore, **this sweet spot appears to scale with corpora sizes** because lower word varieties lead to lower collective entropies on average: note that runs on the smaller corpora produced more accurate results when the ELT was 3.

The same trend was **not** subverted when the ELT was 4 and p(Mutation) was 1%, because low mutation chances inhibit in-word exploration and slow the discovery of optimal boundaries. Over more generations, those settings might well produce the most optimal results possible: the ELT is at just the right spot to lock a highest proportion of correct boundaries, and the low mutation probability would ultimately enable it to examine wider ranges of key split configurations that would otherwise be skipped over through more erratic mutation.

To verify that the full-size corpus' ELT sweet spot was closer to 4 than 5, additional runs were performed on that corpus with the ELT at 5 and p(Mutation) at both 66% and 100%. As expected, this ELT was too high to lock even some of the corpus' most probable boundaries and the resulting outputs' fitnesses suffered for it; even if it still locked some of the most probable boundaries in the set and produced better results than the corresponding pure run for that.

All runs' distributions of best fitnesses (observed after each of their respective generations) were also recorded, truncated and listed as percentage increases over starting fitnesses in figure 5.3. When the ELT was 4 and p(Mutation) was at least 66%, **the uptick in the larger outputs' relative accuracies was reflected by huge increases in proportional fitness gains**: in one case, going from the 1000-word run to the 5000-word run raised the corresponding proportional fitness increases from 25.23% to 56.25% respectively.

On the whole, **these fitness gains complement or even exhibit many of the trends in the accuracy metrics**: precision and recall values increase and decrease with relative fitness gains, smaller outputs (that are typically more precise than larger outputs after 100 generations) peak in fitness gains more quickly, and raising p(Mutation) when entropic locking is enforced drastically increases fitness gains in line with the accuracies of the larger outputs.

All outputs' precision percentages exceed their recall percentages, which is to be expected as this genetic procedure can only return one boundary position for each word in a corpus.

Beyond the precision and fitness metrics, **this genetic implementation proved to be roundly slower than the others**. Runs with the full-size corpus were restricted in scope because it already took long enough to evaluate smaller populations of 54646-word mappings over 20 generations, particularly when entropic locking was enabled due to how many stem and suffix comparisons it requires.

No Ent. Locking (p(Mut.) = 1%)		
Size	**δF (%)**	**ΔF (%)**
100	184.38	234.38
500	87.25	116.01
1000	55.99	75.26
5000	21.29	28.84
54626	0.46	0.85

Ent. Locking Threshold = 5 (Size = 54626)		
p(Mut.)	**δF (%)**	**ΔF (%)**
66	9.84	17.24
100	31.92	45.04

Entropic Locking Threshold = 4						
Size	**p(Mut.) = 1%**		**p(Mut.) = 66%**		**p(Mut.) = 100%**	
	δF (%)	**ΔF (%)**	**δF (%)**	**ΔF (%)**	**δF (%)**	**ΔF (%)**
100	217.24	231.03	64.10	64.10	48.72	48.72
500	98.93	128.62	28.24	31.56	29.93	29.93
1000	54.33	78.01	23.02	25.23	28.74	28.74
5000	22.89	33.48	50.58	56.25	52.64	56.57
54626	2.05	3.38	39.78	57.19	49.95	66.71

Entropic Locking Threshold = 3						
Size	**p(Mut.) = 1%**		**p(Mut.) = 66%**		**p(Mut.) = 100%**	
	δF (%)	**ΔF (%)**	**δF (%)**	**ΔF (%)**	**δF (%)**	**ΔF (%)**
100	221.88	231.25	164.86	170.27	185.29	197.06
500	86.33	114.66	114.93	125.69	121.58	129.45
1000	48.14	71.53	82.49	91.57	90.40	94.42
5000	24.32	36.15	64.60	68.39	65.28	67.49
54626	2.75	3.94	34.33	43.31	39.40	45.73

Figure 5.3: The fitness gains observed while evaluating each corpus during experiment 2, expressed as percentage increases over initial fitnesses at each run's mid-point (δF) and end (ΔF)

5.4 Experiment 3: Comparing implementations and extensions 1 and 2 across numerous languages

5.4.1 Methodology

Each corpus was evaluated once by the pure Harrisian implementation (H-P); the Harrisian implementation with frequency-wise ESM and no suffix length floors; the pure genetic implementation with p(Mutation) set to 1% (K-P); the genetic implementation with an ELT of 3 and p(Mutation) set to 66% (K-ENT3); and the same again but with an ELT of 4 (K-ENT4). The modified algorithms were chosen because their extensions were deemed to be optimal for corpora of various sizes, based on the findings of experiments 1 and 2.

5.4.2 Results

There are few concrete conclusions to reach here, but some are worthy of note. Perhaps the most interesting of them is that **the results for the Scottish Gaelic corpus are generally more inconsistent** than those for the similarly-sized English corpus, which could be attributed to a greater lack of similarities between words. This appears to be the case given the outcomes of experiment 4.

The results for the Cornish corpus are the most consistent of all three tested languages, though that one had almost twice as many words as the other two so — with more linguistic information encoded in it — trie maxima and peaks in collective entropy were always going to be easier to infer from that one.

In all three languages, the percentages of pure Harrisian splits in their respective suffix-matched results are *much* higher than those of the inverse. This is to be expected as the frequency-wise ESM method prefers to match more common suffixes, which tend to be shorter and with left-hand boundaries incidental with or *after* the right-most splits judged by the pure process.

Interestingly, more of the Harrisian results matched those of the entropy-locked genetic results than those of the pure genetic results. This could suggest a link — albeit a tenuous one — between Harrisian trie maxima and peaks of collective entropy in words, which makes sense: stems with more suffixes are naturally going to peak in next-letter collective entropies.

The pure genetic results align more with the suffix-matched results than

with the pure Harrisian results. Experiment 1's outcomes suggest that frequency-wise ESM is more likely to indicate final splits correctly across smaller corpora, so — if one also assumes that most of these splits are later than their pure Harrisian alternatives, because commonly-matched suffixes tend to be short — this may actually suggest that the pure genetic process is drawn towards indicating later splits.

English (5000)	*H-P*	*H-ML1*	*K-P*	*K-ENT3*	*K-ENT4*
H-P	-	98.47	12.26	26.03	19.72
H-ML1	63.45	-	12.46	27.16	16.55
K-P	18.90	29.80	-	13.32	13.32
K-ENT3	40.12	64.96	13.32	-	21.3
K-ENT4	30.40	39.58	13.22	21.3	-

Cornish (9645)	*H-P*	*H-ML1*	*K-P*	*K-ENT3*	*K-ENT4*
H-P	-	99.43	15.78	26.73	27.31
H-ML1	65.46	-	15.50	22.77	21.45
K-P	25.45	37.98	-	18.57	19.39
K-ENT3	43.12	55.80	18.57	-	32.29
K-ENT4	44.06	52.56	19.39	32.29	-

S-Gaelic (5028)	*H-P*	*H-ML1*	*K-P*	*K-ENT3*	*K-ENT4*
H-P	-	99.97	13.08	16.49	13.39
H-ML1	55.98	-	13.06	15.86	12.57
K-P	16.03	28.58	-	15.39	14.46
K-ENT3	20.21	34.71	15.39	-	20.58
K-ENT4	16.41	27.51	14.46	20.58	-

Figure 5.4: The full set of results for experiment 3, showing the percentages of splits identified by the left-hand implementations occuring in the outputs of the top-hand implementations

5.5 Experiment 4: Estimating the effectiveness of analogous word production across numerous languages

5.5.1 Methodology

Each complete corpus was reduced to five independently-randomised 1000-word subsets before being sequentially supplied to implementation 3 in turn. Stem-suffix pairings needed to respectively occur at least once elsewhere to be recorded, and relationships needed to share at least two counterparts produce WWM rules. Any and all WMM rules established from these relationships were applied to generate lists of "new" words from their respective sub-corpora: these lists were recorded and discussed as follows.

5.5.2 Results

Corpus	Run 1			Run 2			Run 3		
(1000)	G	GA (%)	T (s)	G	GA (%)	T (s)	G	GA (%)	T (s)
English	350	48.86	426.17	142	45.77	187.81	506	30.63	304.76
Cornish	3469	4.30	522.41	2522	4.44	797.00	1584	3.72	691.13

Corpus	Run 4			Run 5			Merged	
(1000)	G	GA (%)	T (s)	G	GA (%)	T (s)	G	GA (%)
English	1252	15.18	252.20	763	26.87	290.94	2995	25.94
Cornish	3591	2.87	728.14	3375	5.01	683.28	13655	4.07

Rules Produced						
Corpus (1000)	Run 1	Run 2	Run 3	Run 4	Run 5	Merged
English	8	12	16	10	12	58
Cornish	68	48	39	40	80	274

Figure 5.5: The full set of results for experiment 4, showing the numbers of words generated (G) at the end of each run; their proportions of correct generations (GA) and the amount of seconds (T) that they took to complete

As it stood, implementation 3 *appeared* to perform poorly with this experiment's testing corpora. The temptation may be to blame the implementation itself for this because it ignored word-types when setting rules, but note that the testing procedure was far from ideal as generations couldn't be checked against more comprehensive external sources. This means that some correct generations which didn't exist in the complete corpora — like "subtractions" from the English tests — were deemed to be incorrect, even though they obviously aren't.

The elephant in this room is the complete lack of results for the Scottish Gaelic corpus. The program failed to locate **any** stem-suffix pairings for which both the stem and the suffix occurred in the Scottish Gaelic corpus multiple times, and this may be attributable to the corpus' linguistic noise (which is discussed in chapter 5.1). Of course, the "full" corpus was a randomised reduction of a very large word-pool so it's not unreasable to expect a fraction of such a varied list to vary a lot in itself.

Tests with the Cornish corpus exhibited a contrasting quirk: it repeated more stems and suffixes **more frequently** than the English set, meaning that vastly more pairings; relationships and WWM rules were recorded from a smaller overall pool of words. This may not be a "problem" per se because of the verification issues mentioned two paragraphs ago, but less specialised rule-sets will naturally produce more spurious generations as the relative accuracy of the English generations (produced from fewer rules) implies.

As for the English outputs: most of the 58 generated rules related pairs of longer definite stems, but most of the "new" words were generated from less-specific rules relating shorter suffixes. These resemble rules 1 and 2 in figure 5.6 respectively, and it should be clarified that the Cornish rules universally took these forms too. Rules with fewer definite characters can be matched with more varied words and are therefore more applicable for widespread discovery, but — again — they're also more likely to produce incorrect forms, as is evident across productions with rules like |**#########| ↔ |**#########ly|.

1	('prevent#*', 'correspond#*')
2	('***#####ed', '***#####es')

Figure 5.6: Examples reflecting the two types of English WWM rules [14] produced by implementation 3

6 Conclusion

Above all else, this project served as a valuable opportunity for me — the present author — to explore an unfamiliar field in the domain of Computer Science through self-directed research and experimental rigour. I came away from it understanding a whole lot more than I did coming in: not just about the present field of research, but about the whole academic process and how knowledge is applied and shared throughout it.

The literature review that I undertook was substantial, and yet it doesn't come close to covering the entirety of the field at hand. The task of unsupervised morphological analysis is a big one and works pertaining to it are both plentiful and frequent: one notable piece focussing on "word embeddings" [29] was published fewer than three years prior to the time of this report. I remain confident that the literature review presented here does well to introduce the field, but I advise anyone taking an interest in it or undertaking a similar project to explore further.

The two extensions that were tested came from little more than my own intuition, but the tests revealed some fascinating conclusions that could well prove useful as the foundations of further developments. Eager suffix-matching is a simple process, but the results of experiment 1 suggest that it can indeed identify more suffixes correctly when only some are inferrable from successor quantity variations; usually when corpora are quite small. Paired with some kind of probabilistic function to measure candidate suffixes' likelihoods with respect to words' stems, it could yet prove to be quite the useful preface for small-scale Harrisian analysis.

The use of entropic locking in a naïve genetic process offers considerable potential, as it rapidly accelerated the fitness gains of larger mappings across the suite of genetic tests. Further testing is required to confirm how entropic-locking evolution holds up against pure evolution through larger generation counts, but it nevertheless raises the feasibility of evolving extremely large mappings through this naïve process. That may yet prove to be useful for big data operations or for learning with low-performance machines.

One admitted failing of this project is apparent as a general lack of

emphasis on indigenous languages. Despite being mentioned in the title, this project focussed far more on unsupervised learning and it showed with the relative defecit of tests involving the two sourced indigenous corpora. The quality of the Scottish Gaelic corpus was also not that great, with more useful insights being taken away from the cleaner Cornish corpus instead. With that said, the fact that implementations like these can operate on indigenous corpora at all indicates that any interest in preserving lesser-spoken languages through heuristic analysis is very much founded and worth acting on.

In summary: while this project wasn't of significant service to the present field, it nevertheless explores some interesting ideas that could later be expanded upon for more measurable progress in morphological analysis. I look forward to seeing if anyone does as time goes on.

6.1 Recommendations for Further Work

- Perform more comprehensive tests with entropic locking (on larger populations of larger individuals over more generations)
- Test whole-word morphological production of indigenous words using corpora annotated with word-types
- Explore further options for implementing morphological learners by evaluating (or even extending) different programming languages
- Test out Kazakov's genetic method with different fitness functions in place of the original's naïve function

Bibliography

[1] S. R. Anderson, *Morphology*, https://cowgill.ling.yale.edu/sra/morphology_ecs.htm, Accessed: Mar. 7, 2018.

[2] C. Jacquemin and E. Tzoukermann, 'NLP for term variant extraction: synergy between morphology, lexicon, and syntax', in *Natural Language Information Retrieval*, Springer, 1999, pp. 25–74.

[3] Cornish Language Partnership, *Cornish Dictionary*, http://www.cornishdictionary.org.uk/, Accessed: Apr. 4, 2018.

[4] University of Glasgow, *Digital Archive of Scottish Gaelic (DASG)*, http://dasg.ac.uk, Accessed: Apr. 5, 2018.

[5] D. A. Balota et al., 'The English Lexicon Project', in *Behavior Research Methods*, vol. 39, Aug. 2007, pp. 445–459. [Online]. Available: http://elexicon.wustl.edu/.

[6] F. de Saussure et al., *Course in General Linguistics*. New York, NY, USA: Philosophical Library, 1959.

[7] *Dictionary.com Unabridged*, http://www.dictionary.com/browse/-er, Accessed: Mar. 13, 2018.

[8] D. Kazakov, 'Unsupervised learning of naive morphology with genetic algorithms', in *ECML/Mlnet Workshop on Empirical Learning of Natural Language Processing Tasks, Prague*, Mar. 1997, pp. 105–112.

[9] T. Bevan, 'Unsupervised learning of the word morphology of indigenous languages', Apr. 2016.

[10] E. Brill, 'Some advances in transformation-based part of speech tagging', in *Proceedings of the Twelfth National Conference on Artificial Intelligence (Vol. 1)*, ser. AAAI '94, Seattle, Washington, USA: American Association for Artificial Intelligence, 1994, pp. 722–727, ISBN: 0-262-61102-3.

[11] P. Schone and D. Jurafsky, 'Knowledge-free induction of inflectional morphologies', in *Proceedings of the Second Meeting of the North American Chapter of the Association for Computational Linguistics on Language Technologies*, ser. NAACL '01, Pittsburgh, Pennsylvania: Association for Computational Linguistics, 2001, pp. 1–9. DOI: 10.3115/1073336.1073360.

[12] D. E. Knuth, *The Art of Computer Programming*, Second Edition. Redwood City, CA, USA: Addison Wesley, 1998, vol. 3: Sorting and Searching, ISBN: 0-201-89685-0.

[13] S. Neuvel and S. A. Fulop, 'Unsupervised learning of morphology without morphemes', in *Proceedings of the ACL-02 Workshop on Morphological and Phonological Learning–Volume 6*, ser. MPL '02, Stroudsburg, PA, USA: Association for Computational Linguistics, 2002, pp. 31–40. DOI: 10.3115/1118647.1118651.

[14] A. Ford and R. Singh, 'Propédeutique morphologique', *Folia Linguistica*, vol. 25, no. 3-4, pp. 549–576, 1991.

[15] C. Jacquemin, 'Guessing morphology from terms and corpora', in *Proceedings of the 20th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, ser. SIGIR '97, Philadelphia, Pennsylvania, USA: ACM, 1997, pp. 156–165, ISBN: 0-89791-836-3. DOI: 10.1145/258525.258557.

[16] Z. S. Harris, 'From phoneme to morpheme', *Language*, vol. 31, no. 2, pp. 190–222, 1955, ISSN: 00978507, 15350665.

[17] D. Kazakov and S. Manandhar, 'Unsupervised learning of word segmentation rules with genetic algorithms and inductive logic programming', *Machine Learning*, vol. 43, no. 1-2, pp. 121–162, Apr. 2001, ISSN: 0885-6125. DOI: 10.1023/A:1007629103294.

[18] M. A. Hafer and S. F. Weiss, 'Word segmentation by letter successor varieties', *Information Storage and Retrieval*, vol. 10, no. 11, pp. 371–385, 1974, ISSN: 0020-0271. DOI: https://doi.org/10.1016/0020-0271(74)90044-8.

[19] C. E. Shannon, 'A mathematical theory of communication', *Bell System Technical Journal*, vol. 27, no. 3, pp. 379–423, DOI: 10.1002/j.1538-7305.1948.tb01338.x.

[20] H. Déjean, 'Morphemes as necessary concept for structures discovery from untagged corpora', in *Proceedings of the Joint Conferences on New Methods in Language Processing and Computational Natural Language Learning*, ser. NeMLaP3/CoNLL '98, Sydney, Australia: Association for Computational Linguistics, 1998, pp. 295–298, ISBN: 0-7258-0634-6.

[21] D. Bernhard, 'Unsupervised morphological segmentation based on segment predictability and word segments alignment', in *Proceedings of the Pascal Challenges Workshop on the Unsupervised Segmentation of Words into Morphemes*, 2006.

[22] J. R. Saffran et al., 'Word segmentation: The role of distributional cues', *Journal of Memory and Language*, vol. 35, no. 4, pp. 606–621, 1996, ISSN: 0749-596X. DOI: 10.1006/jmla.1996.0032.

[23] J. Rissanen, 'Modeling by shortest data description', *Automatica*, vol. 14, no. 5, pp. 465–471, 1978, ISSN: 0005-1098. DOI: 10.1016/0005-1098(78)90005-5.

[24] J. Goldsmith, 'Unsupervised learning of the morphology of a natural language', *Computational linguistics*, vol. 27, no. 2, pp. 153–198, 2001.

[25] M. Creutz and K. Lagus, 'Unsupervised discovery of morphemes', in *Proceedings of the ACL-02 Workshop on Morphological and Phonological Learning - Volume 6*, ser. MPL '02, Stroudsburg, PA, USA: Association for Computational Linguistics, 2002, pp. 21–30. DOI: 10.3115/1118647.1118650.

[26] D. E. Goldberg, *Genetic Algorithms in Search, Optimization and Machine Learning*, 1st. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 1989, ISBN: 0201157675.

[27] International Organization for Standardization, *ISO/IEC 646:1991 - Information technology – ISO 7-bit coded character set for information interchange*, https://www.iso.org/standard/4777.html, Accessed: Apr. 30, 2018.

[28] B. L. Miller and D. E. Goldberg, 'Genetic algorithms, tournament selection, and the effects of noise', *Complex Systems*, vol. 9, pp. 193–212, 1995.

[29] R. Soricut and F. Och, 'Unsupervised morphology induction using word embeddings', in *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 2015, pp. 1627–1637.