A probabilistic model of Ancient Egyptian writing

Mark-Jan Nederhof and **Fahrurrozi Rahman**
School of Computer Science
University of St Andrews
North Haugh, St Andrews, KY16 9SX, UK

Abstract

This article investigates a probabilistic model to describe how signs form words in Ancient Egyptian writing. This applies to both hieroglyphic and hieratic texts. The model uses an intermediate layer of *sign functions*. Experiments are concerned with finding the most likely sequence of sign functions that relates a given sequence of signs and a given sequence of phonemes.

1 Introduction

Ancient Egyptian writing, used in Pharaonic Egypt, existed in the form of *hieroglyphs*, often carved in stone or painted on walls, and sometimes written on papyrus (Allen, 2000). Hieroglyphs depict people, animals, plants and various kinds of objects and geographical features. A cursive form of Ancient Egyptian writing, called *hieratic*, was predominantly written on papyrus. Most hieratic symbols can be seen as simplified hieroglyphs, to such an extent that it is difficult for the modern untrained eye to tell what is depicted. Because hieratic handwriting varied considerably over time, with notable differences between regions and scribes, the creation of computer fonts for hieratic is problematic, and consequently scholars commonly resort to publishing hieratic texts in a normalized hieroglyphic font. Since Version 5.2, Unicode contains a selection of 1071 hieroglyphs. Henceforth we will use the term *sign* to refer to a hieroglyph or a hieratic symbol.

The Ancient Egyptian language is in the family of Afro-Asiatic languages, which includes the Semitic languages (Loprieno, 1995). As in scripts of several Semitic languages (e.g. Hebrew, Arabic, Phoenician), only consonants are written. Modern scholars use between 24 and 25 letters to transliterate Egyptian texts in terms of these consonants.

Most are written as Latin characters, some with diacritical marks, plus aleph ꜣ and ayin ꜥ. An equal sign is commonly used to precede suffix pronouns; thus *sḏm* means "to hear" and *sḏm=f* "he hears". A dot can be used to separate other morphemes; for example, in *sḏm.tw=f*, "he is heard", the morpheme *.tw* indicates passive.

The Ancient Egyptian writing system itself is a mixture of phonetic and semantic elements. The most important are *phonograms*, *logograms*, and *determinatives*. A phonogram is a sign that represents a sequence of one, two or three letters, without any semantic association. A logogram represents one particular word, or more generally the lemma of a word or a group of etymologically related words. A determinative is commonly written at the end of a word, following phonograms, to clarify the meaning of a word; in their most obvious use, determinatives disambiguate between homophones, or more precisely, different words consisting of the same consonants. In addition, there are *typographical* signs, for example, three strokes that indicate the plural form of a noun (also used for collective nouns). More classes of signs can be distinguished, such as the *phonetic determinatives*, which tend to be placed near the end of a word, next to normal determinatives, but their function is phonetic rather than semantic, i.e. they repeat letters already written by phonograms.

What makes automatic analysis of Ancient Egyptian writing so challenging is that there was no fixed way of writing a word, so that table-lookup is largely ineffective. Even within a single text, the same word can often be found written in three or more different ways. Moreover, one sign can often be used in different functions, e.g. as phonogram or as determinative. Some signs can be used as different phonograms with different sound values. Together with the absence of word boundary markers, this makes it even hard to segment a text into words.

Generalizing statements can be made about writings of words. Typically, either a word starts with a number of phonograms, covering all the letters of the stem, possibly some covered more than once, followed by one or more determinatives, or a word starts with a logogram, possibly followed by one or more phonograms especially for endings, possibly followed by one or more determinatives. More phonograms can follow the determinatives for certain suffixes. This coarse description is inadequate however to model the wide spectrum of writings of words, nor would it be sufficient to disambiguate between alternative analyses of one sequence of signs.

These factors motivate the search for an accurate and robust model that can be trained on data, and that becomes more accurate as more data becomes available. Ideally, the model should be amenable to unsupervised training. Whereas linguistic models should generally avoid unwarranted preconceptions, we see it as inevitable that our model has some knowledge about the writing system already built in, for two reasons. First, little training material is currently available, and second, the number of signs is quite large, so that the little training material is spread out over many parameters. The *a priori* knowledge in our model consists of a sign list that enumerates possible functions of signs and a formalization of how these functions produce words. This knowledge sufficiently reduces the search space, so that probabilistic parameters can be relatively easily estimated.

In our framework, a *sign function* is formally identified by the combination of (a) the one or more signs of its writing, (b) its class, which could be 'phonogram', 'logogram', 'determinative', etc., (c) zero, one or two values, depending on the class. One example is the phonogram function for sign ⌓ with sound value *r*. There is a logogram function for the same sign, with as value the lemma *r3*, "mouth". A typographical function for the three stokes may have a semantic value 'plural' and a phonetic value that is the masculine plural ending -*w*.

The problem we will address in the experiments is guessing the sign functions given the signs and the letters. This is related to the problem of automatically obtaining transliteration from hieroglyphic text. As far as we are aware, the earliest work to attempt this was Tsukamoto (1997). It relied on simple Unix applications such as 'grep' and 'sed'. The same problem was addressed by Rosmorduc (2008), using manually produced rewrite rules. Further work along these lines by Barthélemy and Rosmorduc (2011) uses two approaches, namely cascades of binary transducers and intersections of multitape transducers, with the objective to compare the sizes of the resulting automata.

A more modest task is to automatically align given hieroglyphic text and transliteration, as considered by Nederhof (2008), who used an automaton-based approach with configurations, similar to that in Section 4, except that manually determined penalties were used instead of probabilities.

Relating hieroglyphic texts and their Egyptological transliteration is an instance of relating two alternative orthographic representations of the same language. The problem of mechanizing this task is known as machine transliteration. For example, Knight and Graehl (1998) consider translation of names and technical terms between English and katakana, and Malik et al. (2008) consider transliteration between Hindi and Urdu. Another very related problem is conversion between graphemes and phonemes, considered for example by Galescu and Allen (2002).

Typical approaches to solve these tasks involve finite-state transducers. This can be justified by the local dependencies between input and output, that is, ultimately the transliteration can be broken down into mappings from at most n to at most m symbols, for some small n and m. For Ancient Egyptian however, it is unclear what those bounds on n and m would be. In this sense, Ancient Egyptian may pose a challenge to the Regularity hypothesis from Sproat (2000). For this reason we do not exclusively rely on finite-state methods in this paper.

2 Sign list

Essential to the application of our model is an annotated sign list. We have created such a list in the form of a collection of XML files.[1] Apart from being machine-readable, these files can also be converted to human-readable web pages. Among other things, the files gather knowledge about the various functions of the 1071 signs from the

---

Unicode repertoire, gathered from a number of sources, the foremost of which is Gardiner (1957). The annotated sign list is necessarily imperfect and incomplete, which is due to inadequacies of the Unicode set itself (Rosmorduc, 2002/3; Polis and Rosmorduc, 2013), as well as to the nature of Ancient Egyptian writing, which gave scribes considerable freedom to use existing signs in new ways and to invent new signs where existing signs seemed inadequate. We have furthermore ignored the origins of signs, and distinguish fewer nuances of sign use than e.g. Schenkel (1971).

Our functions are divided into logograms, determinatives, phonograms, phonetic determinatives and typographical signs. The typographical signs include for example the three strokes that indicate plurality or collectivity. Another example is the single stroke, which can function as 'semogram marker', that is, it indicates that a neighboring sign has a semantic function (as logogram or determinative) rather than a function as phonogram. Other examples of typographical signs include the numerals.

We further distinguish between determinatives that are used with a class of semantically related words, and determinatives that are specific to one word. For example, the "tree" determinative ⟨⟩ is used with various nouns related to trees, plants and wood, whereas the determinative depicting a mooring post ‖ is only used to write the word *mnjt*, "mooring post". Where a determinative is specific to one word, the same sign can often be used as logogram as well, that is, the sign can be used to write a word without accompanying phonograms. For example, 𓈙 can as logogram stand on its own for *ḫr*, "to fall", but it is determinative in 𓎡 𓈙, *ḫr*, where it is preceded by two phonograms for *ḫ* and *r*, respectively.

A few combinations of signs have a single function as a group. For example, ⟨⟩⟨⟩ is a phonogram *nn*, whereas an isolated ⟨⟩ can only be a phonogram *nḫbt*. The combination of signs 𓀀𓀀𓏤 is a determinative for a "group of people".

The sign list contains (very rudimentary) information about the morphological structure of words written by logograms, in particular the stem and the gender (of nouns). The motivation is that this is necessary in order to match sign occurrences to

transliterations. For example, the information that the word *nmtt*, "step", denoted by the logogram 𓌡, is feminine can be used to infer that uses of the logogram in plural writings should be matched to *nmtwt*, "steps", with the feminine plural ending -*wt* in place of the feminine singular ending -*t*. Similarly, the logogram 𓂧, for *ḫnj*, "to row", is accompanied by information that its stem is *ḫn*, so we can identify the use in the writing of *ḫn=f*, "he rows", without the weak consonant *j*, which disappears in most inflections.

3 Corpus

There is currently only one comprehensive corpus of Late Egyptian, which is still under development (Polis et al., 2013). Corpora of Middle Egyptian, the object of our study, are scarce however. Moreover, we are not aware of any available corpora of hieroglyphic texts in which each sign is annotated with its function. One attempt in that direction was reported by Hannig (1995, p. XXXV), with the objective to determine the ratios of frequencies of four main classes of signs, using the first 40 lines of the text of Sinuhe.

It follows that in order to train and test our model, we had to create our own annotated corpus.[2] As yet, it is of modest size, including just two classical texts, known as The Shipwrecked Sailor (P. Leningrad 1115) and Papyrus Westcar (P. Berlin 3033). For the convenience of annotation of the text with sign functions, the text was linearized, that is, information about horizontal or vertical arrangement of signs was discarded. Whereas the positioning of signs relative to one another can be meaningful, our current models do not make use of this; if necessary in the future, the exact sign positions can be extracted from another tier of annotation.

We normalized the text by replacing graphical variants, such as 𓂺 and 𓂾, by a canonical representative, using machine-readable tables that are part of our sign list. We also replaced composite signs by smallest graphemic units. For example, we replaced a single sign consisting of three strokes (typographical sign for plurality or collectivity) by three signs of one stroke each. Motivations for this include convenience and uniformity:

---

in typeset hieroglyphic texts one may prefer to use three separate strokes and fine-tune the distance between them to obtain a suitable appearance.

Disregarding damaged parts of the manuscripts, the segmented texts of The Shipwrecked Sailor and Papyrus Westcar comprise 1004 and 2669 words, respectively. These were annotated with functions, using a customized, graphical tool. In this tool one can select known functions for signs, as present in the XML files mentioned in Section 2, but the tool also gives the option to create new functions that are not covered by the sign list.

Per word, we have the sequence of signs of the hieroglyphic writing, the sequence of letters of the transliteration, and a sequence of functions. Each function in this sequence is linked to one or more signs, and in the case of e.g. a phonogram or logogram, a function is also linked to one or more letters. Different kinds of functions were already discussed in Section 2. In addition to these, there is the artificial 'spurious' function, which is used when a sign has no obvious purpose. There is also the 'mult' function, which is linked to repeated ('multiple') occurrences of signs, to denote duals or plurals. (Ancient Egyptian had a dual form for nouns, next to the plural form; e.g. the dual noun *rdwj*, with masculine dual ending *-wj*, means "pair of legs"; see Figure 1a.)

An important document in this process was an annotation manual that helped to disambiguate contentious cases, of which there were many. For example, we have characterized a phonogram earlier as a sign used to represent sound value, rather than a semantic relation to a certain concept. However, in many cases it is not immediately obvious whether two words that share letters are semantically (etymologically) related. One example is the sign ⌐, which is primarily used as logogram for *nṯr*, "god". It is also used in the writing of the word *snṯr*, "incense", and one may naively interpret it as phonogram there. Although the exact etymology and even the correct transliteration of this word are uncertain, it is highly likely that the sign is not merely chosen for its sound value, but for its semantic relationship to *nṯr*, "god", perhaps derived from *nṯr* with the causative prefix *s-*, or perhaps in combination with the verb *sn*, "to smell", as suggested by de Vartavan (2010). Hence the sign must certainly be analyzed as logogram rather than phonogram in both *nṯr* and *snṯr*.

We have as far as possible relied on conven-
tional wisdom, but on several occasions we had to resort to informed guesses, making additions to the annotation manual to ensure consistency.

Some more examples of annotations are given in Figure 1. In (b) we see use of the three strokes to indicate plural, with masculine plural ending *-w*; the singular noun is *ḏbꜥ*, "finger". In (c), a logogram consisting of two signs represents the stem of the verb *m33*, "to see". A phonogram writes the second letter of the stem once more. Of the morpheme *.tw*, which indicates passive, only the *t* is represented by a sign. Illustrated in (d) is that one letter may be represented by several phonograms. In particular, consecutive phonograms may overlap one another in the letters they cover.

In our annotation, each sign is connected to exactly one function. We have imposed this constraint to simplify our model, which is discussed in the next section. In the case of repeated signs that indicate dual or plural, only second and possibly third occurrences are connected to the 'mult' function, as in Figure 1a, whereas the first occcurrence is connected to a 'log' or 'det' function (and on rare occasions a 'phon' function) as appropriate.

4 Model

The examples in the previous section show that a sequence of functions can be the intermediary between a sequence of hieroglyphic signs and a sequence of letters of the transliteration. We assume that each function is associated with one sign or several signs occurring consecutively, which means that the sequence of signs is formed simply by concatenating the signs gathered one by one from the sequence of functions. In terms of automata, one can think of functions as appending signs at the end of a tape, or in other words, the tape head moves strictly left-to-right.

The relation between the sequence of functions and the transliteration is more involved however. In fact, the sequences of functions as we have presented them provide insufficient information to fully specify the sequence of letters. In the example of Figure 1c, the letters coming from the respective functions would string together to *m33tf* rather than *m3.tw=f*. In order to complement the information contained in a sequence of functions so as to fully specify the transliteration, we have added two new types of functions. The first we call *epsilon-phonograms*. Such a function acts

a: "my (two) legs"

phon   phon   det   mult(2)   log

r   *d*   *w*   *j*   =   *j*

b: "your fingers"

log   typ(plur)   phon

ḏ   *b* ^{c} *w*   =   *k*

c: "he is seen"

log   phon   phon   phon

m   3   .   *t*   *w*   =   *f*

d: "water"

phon   phon   phon   phon   phon   det(liquid)
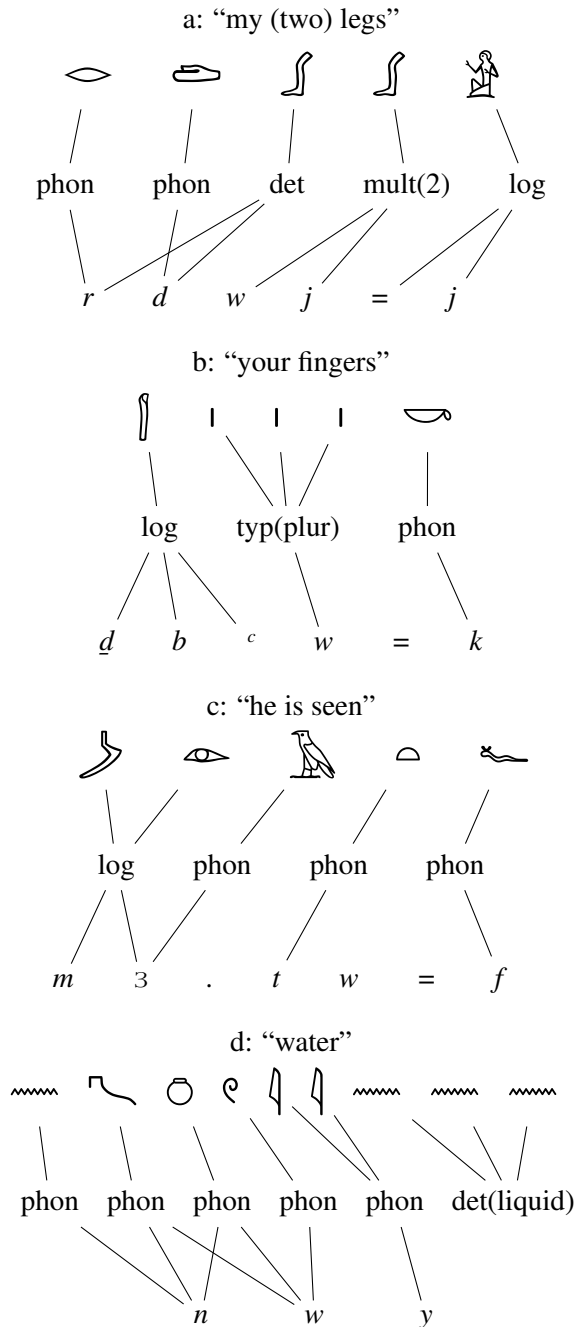
n   *w*   *y*

Figure 1: Annotations in the corpus, all from the Shipwrecked Sailor.

as a phonogram in the sense that letters are generated, but it does not correspond to any hieroglyphic signs (in other words, it corresponds to the empty, or epsilon string of signs).

The second newly added type of function we call a *jump*. Here we conceive of the transliteration as a tape that is being written left-to-right, but with the possibility that the tape head moves backward, by applying a jump with a negative value. Thereafter, some letters already written may be written again, or rather, scanned, as letters on the tape cannot be changed after they have been written. By applying a jump with a positive value, the tape head may also move forward, across letters already written, but not across unwritten tape squares at the right end of the tape.

A further constraint is that jumps may only be applied immediately preceding a function that (re)writes one or more letters. This is to exclude spurious ambiguity. For example, without this constraint, we could at will apply a jump preceding a determinative or following a determinative; the constraint excludes the former. Similarly, an epsilon-phonogram may not be immediately followed by a jump.

With such constraints, an annotation from our corpus can be extended in exactly one way with epsilon-phonograms and jumps to account for the relation between signs and letters. For example, the sequence of functions from Figure 1c is extended to have a jump one position backward following the logogram. Further, three epsilon-phonograms for '.', '*w*' and '=', are inserted between the existing phonograms. In the example of Figure 1d, several jumps with values -1 and -2 are needed.

To make the preceding more precise, we introduce the concept of *configuration*, which gathers three kinds of information, viz. (a) the letters already written on the tape, (b) the current position of the tape head, and (c) a collection of Boolean flags. Initially, the tape is empty, the tape head is at the beginning of the tape, and all flags are **false**. One of the flags indicates whether the preceding function was a jump, another indicates whether the preceding function was an epsilon-phonogram. These two flags are needed to exclude spurious ambiguity, as explained earlier. One more flag will be discussed later.

In a given configuration, only a subset of functions is applicable. For example, if the tape contains *n* and the input position is 0, then a phonogram with sound value *nw* is applicable, resulting in tape content *nw* with the input position 2. However, in the same configuration, with tape content *n* and input position 0, a phonogram with sound value *t* is not applicable, as letters on the tape may not be changed.

In general, every function has a *precondition*, that is, a set of constraints that determines whether it is applicable in a certain configuration, and a

Phonogram with sound value γ.

Pre	γ is prefix of β or β is prefix of γ.
Post	If β was prefix of γ and $\beta\delta = \gamma$, then δ is added at end of tape. Input position incremented by $|\gamma|$. 'jump' and 'epsphon' flags set to **false**.

Logogram for word γ.

Pre	$i = 0$ or (stem after causative prefix) $i = 1$ and $\alpha = s$. β is prefix of γ.
Post	For δ such that $\beta\delta = \gamma$, δ is added at end of tape. Input position incremented by $|\gamma|$. 'jump' and 'epsphon' flags set to **false**.

Determinative not specific to any word.

Pre	'jump' and 'epsphon' flags are **false**.
Post	Tape and input position unchanged.

Phonetic determinative with sound value γ.

Pre	γ is substring of α.
Post	Tape and input position unchanged.

Jump with value j.

Pre	'jump' and 'epsphon' flags are **false**. $0 \le i + j \le |\alpha\beta|$.
Post	Input position becomes $i + j$. 'jump' flag set to **true**.

Table 1: Preconditions and postconditions of the most important functions. We assume the tape content is $\alpha\beta$ and the input position is $i = |\alpha|$, i.e. the length of α. In other words, the tape content following the input position is β.

postcondition, which specifies how its application changes the configuration. The most important functions are characterized in this manner in Table 1.

Note that epsilon-phonograms together with the 'spurious' functions guarantee that the model is robust against mistakes (either by the modern encoder or by the ancient scribe) and against gaps in our knowledge of the writing system. That is, given a sequence of signs and a sequence of letters, we can always find at least one sequence of functions that connects the two together.

Our model has specialized functions in place of the generic typographical functions as they occur in the corpus. For example, the three strokes, for 'plurality or collectivity', in the model correspond to several different functions with different pre-
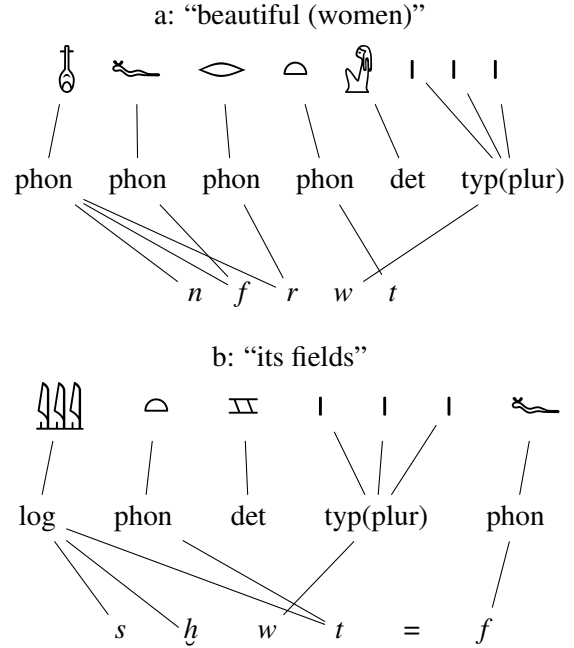

Figure 2: Two annotations from Papyrus Westcar, showing the complications of the feminine plural.

conditions and postconditions. Firstly, the three strokes may be purely semantic, in the writing of a collective noun in singular form, where they do not represent any letters. If the plural strokes do signify plurality in the grammatical sense, they correspond to the -*w* ending of masculine plural, or to the -*wt* ending of feminine plural. The same holds for the 'mult' functions as they occurs in the corpus; they must be replaced by several different functions if we consider them at the granularity of preconditions and postconditions.

In our corpus we have linked functions marking plural only to the *w* from the ending, whether it is the -*w* ending of masculine plural or the *w* that is the first letter of the -*wt* ending of feminine plural. This is because the *t* of the feminine ending would normally be accounted for already by another sign, which could be a phonogram or logogram, as illustrated in Figure 2a and 2b.

The same two examples also illustrate the challenge that feminine plural poses to a left-to-right automaton model. When the feminine *t* is written to the tape, the sign corresponding to the *w* in front of the *t* is not seen until many steps later. One possible solution is to use lookahead, but this appears difficult to extend with probabilities. Instead, we have introduced an additional flag. This is set to **true** when a substring *wt* is seen in the input, together with a phonogram for *t* or a logogram for a

feminine noun. This flag indicates that an occurrence of a plural marker (the three strokes or repeated occurrences of determinatives) is required later for completing the analysis, by recognizing the end of the word, or for continuing the analysis into the next morpheme within the same word, e.g. a suffix pronoun =f as in Figure 2b.

A peculiar phenomenon in Egyptian writing is *honorific transposition*, which means that a sign or word is written first, even though its linguistic position is further to the end of a word or phrase. This applies in particular to gods and kings. For example, The Shipwrecked Sailor has *dw3.n=f n=j n͟tr*, "he thanked the god for me", with the sign for *n͟tr*, "god", written before the signs for *dw3.n=f n=j*. Where there is honorific transposition in the corpus spanning more than one word, all these words are put in the same segment. Our model presently does not capture honorific transposition however, which means accuracy is poor for the (few) cases in the corpus.

5 Probabilities

After having captured the relation between sequences of signs and sequences of letters solely in terms of sequences of functions, the next step is to estimate their probabilities. An obvious candidate is a simple N-gram model:

$$P(f_1^n) = \prod_i P(f_i \mid f_1^{i-1}) \approx \prod_i P(f_i \mid f_{i-N+1}^{i-1})$$

Here $f_1, \ldots, f_n$ is a sequence of functions, ending in an artificial end-of-word function, and f_i^j is short for $f_i, \ldots, f_j$. In our experiments, estimation of $P(f_i \mid f_{i-N+1}^{i-1})$ is by relative frequency estimation.

About 4000 functions are compiled out of the entries of the sign list. Added to this are dynamically created functions, such as numbers, epsilon-phonograms and jumps. Because little training material is available, this means a considerable portion of these functions is never observed, and smoothing techniques become essential. We use Katz's back-off (Katz, 1987) in combination with Simple Good-Turing (Gale and Sampson, 1995).

Functions are naturally divided into a small number of classes, such as the class of all phonograms and the class of all logograms. Using these classes as states, we obtain a second type of model in terms of (higher-order) HMMs (Rabiner, 1989; Vidal et al., 2005). For fixed N, and with c_i denoting the class of function f_i, we have:

$$P(f_i|f_{i-N+1}^{i-1}) \quad \approx \quad P(c_i|c_{i-N+1}^{i-1}) * P(f_i|c_i)$$

Estimation of both expressions in the right-hand side is again by relative frequency estimation, in combination with smoothing.

It should be noted that not all sequences of functions correspond to valid writings. Concretely, in the configuration reached after applying functions f_1^{i-1}, the preconditions of function f_i may not hold. As a result, some portion of the probability mass is lost in invalid sequences of functions. We see no straightforward way to avoid this, as the model discussed in Section 4, which allows jumps of the tape head, cannot be captured in terms of finite-state machinery.

6 Results

In our experiments, the training corpus was Papyrus Westcar and the test corpus was The Shipwrecked Sailor. We have considered but rejected the possibility of taking two disjoint parts of both texts together as training and test corpora, for example taking all odd words from both texts for training and all even words for testing. The argument against this is that many words occur repeatedly in the same text, and therefore there would be a disproportionate number of words that occur in both training and test material, potentially leading to skewed results.

Our objective is now to guess the correct sequence of functions, given the sequence of signs and the sequence of letters of a word. We determined recall, precision, and F-measure, averaged over all words in the test corpus. This was done after removing jumps and epsilon-phonograms, so that we could take the annotations from the corpus as gold standard. We have also ignored how functions are linked to letters; the main motivation for this was to be able to define a suitable baseline, as described next.

Among all sequences of functions that correspond to a given sequence of signs, the baseline model yields the one that maximizes the product of the (unigram) probabilities of those functions. Note that a function can correspond to one, two or more signs, so that all relevant partitions of the given sequence of signs need to be considered. As this ignores the letters altogether, the baseline is independent of the model of Section 4, avoid-

ing the intricacies of preconditions and postconditions.

For a concrete example, consider Figure 1b as gold standard. The 'relevant' items are (1) the logogram function of ⫾ for the lemma *dbꜥ*, "finger", tied to the first sign, (2) the typographical function of the three strokes, with meaning 'plural' and realised as letter *w*, tied to the next three signs, and (3) the phonogram function of ⌣ with sound value *k*, tied to the last sign. Recall and precision are 100% if 'retrieved' are exactly these three items.

We implemented the N-gram models and HMMs from Section 5. An acyclic finite automaton is first created, with states representing configurations together with the last $N - 1$ functions or classes. Transitions are labelled by functions, and have weights that are negative log probabilities determined by the chosen probabilistic model. Most of the functions directly come from the sign list. Other functions are dynamically constructed, on the basis of the input signs, as for example typographical functions representing numbers. Another example is the 'mult' function, which is generated if a pattern of one or more signs occurs two or more times. Final states correspond to configurations that have input pointers at the ends of the sequence of signs and the sequence of letters, and all Boolean flags set to **false**.

The shortest path from the initial state to a final state is extracted using the shortest-path algorithm of Dijkstra (1959). The labels on this path then give us the list of functions on the basis of which we compute recall and precision.

Results are given in Table 2. It is unsurprising that the models with $N = 1$ improve over the baseline. Although the baseline is also defined in terms of unigram probabilities, it ignores consistency of the sequence of functions relative to the letters. The first-order HMM performs better than the unigram model. This can be attributed to smoothing. For example, the unigram model will assign the same low probability to a spurious function unseen in the training material as to an unseen phonogram, although phonograms overall are far more likely. The first-order HMM however suitably models the low probability of the class of spurious functions.

For N greater than 1, the HMMs perform less well than the N-gram models. This suggests that the probabilities of functions depend more on the

	R	P	F1
baseline	86.0	86.0	86.0
N-gram			
$N = 1$	90.6	90.6	90.6
$N = 2$	94.4	94.4	94.4
$N = 3$	94.4	94.4	94.4
HMM			
$N = 1$	91.4	91.4	91.4
$N = 2$	91.8	91.8	91.8
$N = 3$	92.0	92.0	92.0
interpolation of N-gram and HMM			
$N = 1$	90.5	90.5	90.5
$N = 2$	94.8	94.8	94.8
$N = 3$	**95.0**	**94.9**	**94.9**

Table 2: Experimental results: recall, precision, F-measure.

exact identities of the preceding functions than on their classes. The best results are obtained with linear interpolation of the N-gram model and the HMM, weighted 9:1, for $N = 3$.

7 Conclusions and outlook

Our contributions include the design of an annotated corpus of sign use, allowing quantitative studies of the writing system, and serving to document rare uses of signs. The second main contribution is a probabilistic model of how signs follow one another to form words. The model is amenable to supervised training. Unsupervised training will be the subject of future investigation.

Next to automatic transliteration and alignment, our model was designed to improve the accuracy of a tool that automatically turns scanned pages of Sethe (1927) into the encoding of Nederhof (2002), using OCR and image parsing. We found that a bottleneck is that some common signs, such as 𓅬 (phonogram for ꜣ) and 𓅮 (phonogram for *tjw*), are indistinguishable in Sethe's handwriting. It remains to be investigated whether our probabilistic model would on average assign a higher probability to the correct sign in context.

References

J.P. Allen. 2000. *Middle Egyptian: An Introduction to the Language and Culture of Hieroglyphs*. Cambridge University Press.

F. Barthélemy and S. Rosmorduc. 2011. Intersection of multitape transducers vs. cascade of binary transducers: the example of Egyptian hieroglyphs transliteration. In *Proceedings of the 9th International Workshop on Finite State Methods and Natural Language Processing*, pages 74–82, Blois, France, July.

C.T. de Vartavan. 2010. Snt[r]/snt[r] means '[divine/godly] scent'. *Advances in Egyptology*, 1:5–17.

E.W. Dijkstra. 1959. A note on two problems in connexion with graphs. *Numerische Mathematik*, 1:269–271.

W.A. Gale and G. Sampson. 1995. Good-Turing frequency estimation without tears. *Journal of Quantitative Linguistics*, 2(3):217–237.

L. Galescu and J.F. Allen. 2002. Pronunciation of proper names with a joint n-gram model for bi-directional grapheme-to-phoneme conversion. In *Proceedings of the Seventh International Conference on Spoken Language Processing (ICSLP-2002)*, pages 109–112, Denver, CO, USA.

A. Gardiner. 1957. *Egyptian Grammar*. Griffith Institute, Ashmolean Museum, Oxford.

R. Hannig. 1995. *Grosses Handwörterbuch Ägyptisch-Deutsch: die Sprache der Pharaonen (2800-950 v.Chr.)*. Verlag Philipp von Zabern, Mainz.

S.M. Katz. 1987. Estimation of probabilities from sparse data for the language model component of a speech recognizer. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 35(3):400–401, March.

K. Knight and J. Graehl. 1998. Machine transliteration. *Computational Linguistics*, 24(4):599–612.

A. Loprieno. 1995. *Ancient Egyptian: a linguistic introduction*. Cambridge University Press.

M.G.A. Malik, C. Boitet, and P. Bhattacharyya. 2008. Hindi Urdu machine transliteration using finite-state transducers. In *The 22nd International Conference on Computational Linguistics*, volume 1, pages 537–544, Manchester, UK, August.

M.-J. Nederhof. 2002. A revised encoding scheme for hieroglyphic. In *Proceedings of the 14th Table Ronde Informatique et Égyptologie*, July. On CD-ROM.

M.-J. Nederhof. 2008. Automatic alignment of hieroglyphs and transliteration. In N. Strudwick, editor, *Information Technology and Egyptology in 2008, Proceedings of the meeting of the Computer Working Group of the International Association of Egyptologists*, pages 71–92. Gorgias Press, July.

S. Polis and S. Rosmorduc. 2013. Réviser le codage de l'égyptien ancien. Vers un répertoire partagé des signes hiéroglyphiques. *Document Numérique*, 16(3):45–67.

S. Polis, A.-C. Honnay, and J. Winand. 2013. Building an annotated corpus of Late Egyptian. In S. Polis and J. Winand, editors, *Texts, Languages & Information Technology in Egyptology*, pages 25–44. Presses Universitaires de Liège.

L.R. Rabiner. 1989. A tutorial on hidden Markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286, February.

S. Rosmorduc. 2002/3. Codage informatique des langues anciennes. *Document Numérique*, 6:211–224.

S. Rosmorduc. 2008. Automated transliteration of Egyptian hieroglyphs. In N. Strudwick, editor, *Information Technology and Egyptology in 2008, Proceedings of the meeting of the Computer Working Group of the International Association of Egyptologists*, pages 167–183. Gorgias Press, July.

W. Schenkel. 1971. Zur Struktur der Hieroglyphenschrift. *Mitteilungen des deutschen archäologischen Instituts, Abteilung Kairo*, 27:85–98.

K. Sethe. 1927. *Urkunden der 18. Dynastie, Volume I*. Hinrichs, Leipzig.

R. Sproat. 2000. *A Computational Theory of Writing Systems*. Cambridge University Press, Cambridge.

A. Tsukamoto. 1997. Automated transcription of Egyptian hieroglyphic texts: via transliteration using computer. *Journal of the Faculty of Culture and Education*, 2(1):1–40. Saga-University.

E. Vidal, F. Thollard, C. de la Higuera, F. Casacuberta, and R.C. Carrasco. 2005. Probabilistic finite-state machines — part II. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(7):1026–1039, July.