COPiL CAMBRIDGE OCCASIONAL PAPERS in LINGUISTICS
Volume **6**, Article 5: 131–175, 2012 | ISSN 2050-5949

COMPARATIVE METHOD ALGORITHM*

SOPHIA GILMAN
University of Cambridge / Yale University

ABSTRACT In this paper, I outline a way of automating the Comparative Method, with the aim of expediting the process and providing a new outlook on the 'reality of reconstructions' controversy. Taking as input word lists in IPA format, the Comparative Method Algorithm (henceforth CMA) walks through the steps of the classic comparative method in a transparent way and outputs reconstructions of words in the proto-languages and the proto-language's sound system, along with the cognate sets and correspondences that were found in the process. Furthermore, at each step, the algorithm provides a degree-of-certainty estimate (DOCE) for its conclusions, which can be used as an evaluation of the 'reality' of a reconstruction. To help deal with the issue of a uniform reconstructed language, the algorithm also outputs more than one option for each reconstructed word and proto-language sound system, using the DOCE estimate to determine which candidates are best.

1 PRELIMINARIES

1.1 Two goals of the Comparative Method Algorithm

In the last few years, several attempts have been made to automate parts of the phonological comparative method[1], a process historical linguists use to

* I would like to thank everyone who contributed to and made this project possible: Moreno Mitrovic, my supervisor, without whose help this project would never have gotten off the ground or been completed. Melanie Bell, my second supervisor, for all of her help with the phonology aspects of this project. Mattis List, Jeff Mielke, Sven Grawunder, Juliette Blevins, Andy Wedel, and Mikael Parkvall for kindly responding to my queries about the current states of various databases and projects described in this paper. Mattis List for all of the e-mail discussion and advance information on his forthcoming article. Conrad Nelson and Olga Gilman for discussion of algorithmic ideas. My friends and family (especially Mihoko Takeuchi) for putting up with endless and incomprehensible rants about the Comparative Method Algorithm.

1 There have not yet been attempts to automate morphological or syntactic reconstruction, nor will I attempt to do so in this paper. As Bourchard-Cote et al. point out, "phonological changes are generally more systematic than syntactic or morphological changes," and it is only this systematicity that makes automation possible (Bourchard-Cote et al. 2008: 1). The next few years, however, may see a development of automations of the moprhological or syntactical aspects of the Comparative Method.

reconstruct sounds and whole words in the parent- or proto- language of a given set of related languages. This comparative method has been the "single most important tool" of the historical linguist since the 19th century and has seen "great success," especially in the realm of phonology (Trask 1996). While the comparative method is effective, it is also exceedingly time-consuming and faces several significant theoretical issues. So far, computational methods have been applied only to the first of these problems: various algorithms have been designed to model various parts of the comparative method and thus speed up the process (e.g. Oakes 2000; Kondrak 2002; etc.).

Following in their footsteps, I present here a Comparative Method Algorithm (henceforth CMA) that models most of the classic comparative method. A central goal of the CMA, like with earlier projects, is to expedite the process of reconstruction more than was possible with previous algorithms by adapting the best aspects of said previous methods and by making use of recently created databases of diachronic change. However, as I hope to show, the CMA can not only expedite the linguist's work considerably but also help provide a solution to one of the main problems with the classical comparative method — evaluating the reality of reconstructions. In what follows, I describe this problem further (section 1.2) and present in brief the two solutions that the CMA proposes (section 1.3). With these goals and solution ideas established, I will then describe the general scope of the CMA project(section 1.4): what the input into the CMA would be (section 1.4.1) and the extent to which the CMA models the classic method in its entirety (section 1.4.5).

1.2 The 'reality of reconstructions' problem

The issue of assessing the reconstructions generated by the comparative method haunts the classical literature on the topic (Trask 1996; Crowley 1998; Fox 1995). In what sense do reconstructed proto-words such as $*plHno$ actually represent a real language that someone somewhere once spoke? (Trask 1996). Formalists argue that such reconstructed forms are mere short-hand for describing what related languages have in common. Others believe they reflect something closer to true phonetic forms. Most believe that reconstructed forms have some unspecified (and unspecifiable?) degree of phonetic reality (Trask 1996). None of these answers are particularly specific — the question is essentially unresolved.

Moreover, reconstructed forms are unrealistic in at least one respect: there is only one form reconstructed for any given semantic meaning. In other words, reconstructed forms create the illusion that language is uniform, that it has no dialects — a clearly false statement (Pulgram 1959). But the mechanical Comparative Method can, by definition, yield only one form for each word.

The Pulgram dilemma lies in interpreting this single output: is that form representative of a particular dialect, an underlying form, some cross-dialect average, or none of the above? There seems to be no consensus on the answer to this question or on the reality of reconstructions in general.

1.3 What kind of a solution does the CMA propose?

One of the main goals of the Comparative Method Algorithm I develop in this paper is to provide a new way of looking at these problems (the other is expediting the process). I hope to do this through two innovations in my algorithm: Degree of Certainty Estimates (henceforth DOCEs) as described in sections 1.3.1 and 1.3.2 and a multiple output version of reconstruction (section 1.3.5)

1.3.1 The Degree of Certainty Estimate

The basic idea behind the Degree of Certainty Estimate (adapted from a similar but inimplemented idea in Steiner et al. 2011) is to provide a sort of empirical measure of how real a reconstruction is. For example, if four related words in four different daughter-languages all start with a /t/, a linguist would be fairly confident in reconstructing a /t/ in the parent-language. Moreover, all but the most determined formalists would likely agree that the sound in the parent-language was a /t/. In such a case, the DOCE would be very high. On the other hand, if the first sound in each of the four daughter-languages was different (e.g. /h/, /s/, /z/, and /ʃ/), the linguist would be less confident and the formalists might argue that the reconstruction is less likely to represent a real sound. The more compelling the evidence (and thus the higher the DOCE), the more likely the linguistics community would be to accept the reconstruction as reflecting an actual reality.

Of course, in the classic (i.e. performed by human) Comparative Method, the decision making process is more complex than in the simplistic example above; it consists of a series of steps, at each of which the quality of the evidence (and thus the confidence of the linguist or algorithm) may vary. Much like the human historical linguist, the CMA walks through these steps, and, at each step, the CMA not only finds an answer (e.g. pick a proto-sound) but also provides the DOCE, an estimate of how good the evidence involved in this particular decision was. Combining the DOCEs from all the steps, the final DOCE associated with the reconstruction will thus reflect the quality of the evidence for that reconstruction, and, by extension, be a measure of how 'real' a reconstruction is. Of course, calculating the DOCE for reconstructions is not a way to put an end to the 'reality of reconstruction' debate, but such an

empirical approach may provide a new way of looking at the situation.

In order for the DOCEs of the CMA to reflect with any degree of accuracy the reality of reconstructions by the classic Comparative Method, the CMA must, as much as possible, both model the comparative method minutely (with all of its flaws) and remain otherwise pre-theoretical. The next section is dedicated to discussing these two heuristics in more detail.

1.3.2 Two Heuristics for CMA design

1.3.3 Modeling the Method precisely — flaws included

As stated earlier, one goal of the CMA and the DOCEs in the CMA is to determine how well reconstructions reflect a previous reality. In other words, the DOCEs calculated by the CMA are, in a sense, evaluating the classical comparative method. To meet this goal, the CMA will attempt to model the classic Comparative Method (both strong sides and flaws) as closely as possible, so that the DOCEs actually reflect the effectiveness of the classic method, not just that of the CMA.

As Moreno Mitrovic (personal communication) points out, making an algorithm to evaluate a method does not *necessarily* require one to make the algorithm model the flaws of the original method: there may be ways to fix the flaws of the method without ruining the evaluative aspect. On the other hand, modeling the method precisely *is* a simple and certain way of making accurate evaluations. Finding approaches that both retain the evaluative aspect and amend the method would thus be a good subject for future work, but such work is beyond the scope of the present paper.

The basic result of this decision is that the present version of the CMA does *not* attempt to solve any of the issues with the classic Comparative Method *except* attempting to resolve the 'reality of reconstructions' debate. Thus, for instance, the Comparative Method makes some flawed assumptions (e.g. that language splits are sudden as per the tree model hypothesis) and that sound change is always regular (the Neogrammarian hypothesis) (Campbell 1998). The CMA will make the exact same assumptions, because to depart from them would be to depart from modeling the Comparative Method.

The only exception to this rule is that the CMA is able to provide a resolution to Pulgram's Dilemma, because this resolution in no way affects the calculations of the DOCEs (discussed in detail in section 1.3.5).

Overall, however, the CMA must strive to model the classic method as closely as possible, and this heuristic will come into play in many of the design decisions discussed in the implementation sections (sections 2 and 3).

1.3.4 Remaining Pre-Theoretical

Moreover, it is insufficient for the CMA to merely model the Comparative Method precisely — there are several aspects of the Comparative Method that are simply underdescribed. Thus, for example, a human linguist compares sounds to determine how 'similar' they are or how likely one is to become another (i.e. the 'naturality' of sound changes). The classic descriptions of the method (e.g. Trask 1996, Campbell 1998) do not explain how that is to be done. A human linguist has intuitions and experience to guide him. The algorithm, on the other hand, is equipped with neither and must perforce find some way of operationalizing these decisions.

In doing so, however, the CMA must, as much as possible, avoid introducing any theoretical or other assumptions not already in classic Comparative Method. Any new assumptions will necessarily affect the DOCE calculations, and thus the DOCEs will then reflect the effectiveness of the Comparative Method only given that assumption, rather than the effectiveness of the method in general. Furthermore, if the CMA were to adopt any particular theoretical approach (e.g. sounds are similar if they share certain articulatory features), the DOCEs would become immediately much less useful to any linguists not subscribing to the given theory.

Therefore, since the CMA should ideally reflect only the effectiveness of the Comparative Method and since its output should, ideally, be useful for any historical linguist, the second heuristic in CMA design is to remain pre-theoretic, attempting to use only tools approved by a linguistic consensus and relying as much as possible on empirical data.

Both of these heuristics (modeling the method as closely as possible and remaining pre-theoretic) will be assumed from here on out and be used in design decisions throughout the rest of this paper. Having established these heuristics, we can now turn to how the CMA proposes to deal with second issue (after the 'reality of reconstruction' debate) that it is meant to provide a new outlook on: Pulgram's Dilemma.

1.3.5 Multiple Outputs - resolving Pulgram's Dilemma

Pulgram's dilemma, as described earlier, is that the classic Comparative Method only produces a single reconstructed form for any word, which cannot accurately reflect the reality that languages are by no means uniform. One potential resolution would be to have the CMA produce multiple reconstructed forms. With the DOCE estimates, this solution could be actualized without tampering with any other part of the classical Comparative Method.

Consider again the situation where the four daughter-words for which the

CMA must reconstruct a proto-form begin with four different sounds (e.g. /h/, /s/, /z/, and /ʃ/). The CMA would consider several different options for the proto-sound and assign to each one a DOCE estimate. If only one output is required, the CMA will simply output the proto-sound with the best DOCE. However, to resolve Pulgram's Dilemma, the CMA instead gives several variants for the proto-sound: any proto-sound option whose DOCE clears a certain threshold is output to the user or just any proto-sounds with tying DOCEs. This way, each reconstructed proto-word could have several possible variants, modeling the lack of uniformity in a real language. As Steiner et al. (2011) point out, such multiple outputs based on DOCEs (or, as they say, "statistical output") would be "an even better approximation of the proto-forms as traditionally possible." Whether this variety will be in any way a satisfactory model of dialectal variation remains to be seen, but, much like with DOCE estimates, the CMA at least offers another innovative way of approaching the problem.

Thus, by using DOCEs and giving multiple outputs, the CMA provides a novel empirical approach to the issues of the 'reality of sound reconstructions' and 'Pulgram's Dilemma'. With these basic solutions in mind, we now take a closer look at the scope of the CMA project: what is the input into the method and how much of the Comparative Method does it model? (section 1.4).

1.4 The Scope of the CMA

As stated in the introduction, all the previous efforts at automating the comparative method model some, but not all, the steps of the method. The CMA is designed to model nearly the entire method, more than any previous endeavor save Oakes' in 2000. However, it is still up to the user linguist to prepare the input data. Because the CMA does not, after all, possess all the knowledge of the user linguist, this data must be prepared in a very specific way, and section 1.4.1 explains the three main requirements imposed on the word lists the user feeds into the algorithm: IPA orthography (section 1.4.2), morphologically simple 'words' only (section 1.4.3), and no borrowed words (section 1.4.4). After the input requirements for the CMA are thus both stated and explained in section (section 1.4.1), I proceed to explain in slightly more detail what the full model of the comparative method offers (section 1.4.5).

1.4.1 Input into the CMA

The basic input into the classical comparative method is a set of word lists from languages which the linguist suspects to be related. Thus, the user linguist must first have some hunch that a set of languages are related. Then, he

must craft the word lists that are the input into the CMA, where a word list in any given language contains words, stems, or morphemes in together with their meanings. These word lists, notably, must be more carefully pruned than those a human linguist might use. Specifically, all the word lists must be in the IPA orthography (further discussed in section 1.4.2), they must be simple morphological forms unless the user is comfortable with unparsed forms being used (section 1.4.3), and borrowings must be excluded or the possibility that they may muck up the analysis must be permissible (section 1.4.4).

Furthermore, since the data is still likely to be less than ideal, the user should associate a DOCE with the data that would reflect the following considerations: how likely is it to be flawed, to what extent does it have uncertainties, what is the chance that borrowings remain, and are any of the words likely to reflect to onomatopoeia (such words are likely to be similar to words in other languages and yet not be cognate with them) (Trask 1996). All of these are considerations which the algorithm cannot calculate, and, since the final DOCE should still reflect them to be maximally useful, the user should supply that information. The user would thus, depending on his preference, be able to specify a DOCE for the whole data, but can also provide special DOCEs for specific words (e.g. if they might be borrowings), and/or specific characters (e.g. if the user is uncertain that he picked the right IPA character for some character in the field notes). Any overall DOCEs will simply be averaged over all the smaller parts (e.g. the DOCE for the whole data will be averaged over all characters). Since, at the minimum, the user only needs to provide their best guess at a DOCE for the whole data, this should not be too onerous.

Unlike the requirements to specify DOCEs, the requirements for the wordlists, however, may well seem inconvenient. Unfortunately, very similar requirements usually hold for the currently available algorithms. In the next three sections, I will explain why each of these conditions must hold in the CMA: section (section 1.4.2) discusses the IPA requirement (section 1.4.3), addresses the parsing requirement, (section 1.4.4) deals with constraint against including borrowed words.

1.4.2 IPA orthography

All reconstruction algorithms, including the CMA, are forced to require all the input to be in the same orthography. Otherwise, the computer simply has no way of comparing words from different wordlists, which is an integral part of the comparative method. Some algorithms (e.g. Steiner et al. 2011) allow the user linguist to use any orthography. Such algorithms, however, cannot take into any relationships between particular sounds in their analysis, since the

algorithm has no information on what sounds are represented by the orthography. The CMA, on the the other hand, wants to model as closely as possible the human linguist who considers such concepts as sound similarity when deciding if two words are likely to be cognate or look at the general likelihood of a particular sound change (e.g. he considers that /s/ becomes /h/ more often than /h/ becomes /s/) when trying to pick a proto-phoneme. If the CMA is to have any knowledge that could imitate a linguist's experience and intuition in such considerations, all the input must be in a single orthography - the same orthography as the one used in giving the CMA said knowledge (see sections 2 and 3 for more discussion of how such prior knowledge is incorporated into the CMA). Since the IPA is the most universally accepted system, it is the natural candidate (IPA Handbook 1999). Furthermore, at least for this first version of the CMA, the IPA, if it is the system picked, should be used without diacritics, because if (for instance) one were to try to calculate the probability of change between any two IPA symbols with any two diacritics, one would need to store a more probabilities than is within the power of any modern processor. (For a more complex discussion on choosing the IPA as the orthography for the CMA, please see Gilman (2012b)).

1.4.3 Why input words must be morphologically simple

The human linguist looking at field notes usually parses the words, separating out the morphemes and performing the Comparative Method on each morpheme individually. The CMA, unlike the human linguist but like all other reconstruction algorithms,[2] is unable to identify prefixes, suffixes, or other morphemes (i.e. to do morphological parsing). Thus, if words are not morphologically simple, they will still be treated as such by the CMA, and the reconstructions may reflect this mistake.

The basic reason for this is that the human linguist usually has some prior knowledge about the language (or at least related languages) as well as about general morphological theory. The CMA cannot have any information about the former, and the current model does not have any information about the latter. In theory, it would be possible to create a morphological parser that

2 Many current algorithms use a method called local alignment to be able to find cognates in word lists despite possible complex morphology (e.g. List 2012; Steiner et al. 2011). This method essentially ignores sequences that do not tend to match up; for example, if dealing with the words xi-bar-te and bar-ko it would align the two 'bar's and ignore the rest). Ideally, we could use such methods to discard or separate out the different morphemes of any given word. However, such a method will very often isolate parts of morphemes and thus cannot be used for actual word parsing. Because of this, the reconstruction step must still operate on the full input word, and thus the full input word must be morphemically simple or treatable as such.

works on any language without any prior information, and some work has been done in this area (e.g. Kim et al. 2011). However, such algorithms tend to use much bigger word lists than those normally used in the comparative method (e.g. 100,000 words in the Kim et al. algorithm) making it difficult to apply such algorithms in the CMA (Kim et al. 2011). Therefore, the reconstruction work done by the CMA is purely phonetic and the input must be adjusted accordingly.

1.4.4 Why the input must be vetted for borrowings in advance

Borrowing is one of the main issues within the classic Comparative Method: when two words are similar enough to be cognates, a linguist must be sure to check that neither of the words is actually a borrowing from the other language, as that is a frequent alternative explanation for such similarity (Trask 1996). To avoid this problem, linguists commonly use word lists with only the vocabulary items that tend to be resistant to borrowing (e.g. Swadesh lists) but caution is still necessary (Trask 1996). Identifying the borrowings that might lurk even in such lists is a task even the human linguist often has difficulty performing and that often involves much background knowledge. As Campbell puts it, linguists "must resort to other techniques which are not formally part of the comparative method for dealing with borrowing" (1998: 147).

Of the purely reconstructive algorithms currently available, none model these techniques. Steiner et al. argue that "loanwords should in principle be detectable [...] by using characteristic sub-regularities in the sound correspondences as indications for different strata in the lexicon" (2010, p.4) but do not provide a way to automate this. On the other hand, the work of Ringe et al. in the related realm of phylogenetics suggests ways of tracking borrowing (2006). While it would be in principle possible to integrate phylogenetics methods into the CMA, this has not yet been done and remains a task for the future. Thus, currently, the CMA does not provide a way of tracking borrowing and the user linguist must prune the data for borrowing himself.

Thus, even though the CMA places stringent requirements on the input data, these requirements are the same as those for all the currently available algorithms that offer a similar level of analysis to the CMA. With these not outstanding requirements for input data, however, the CMA models more of the comparative method than any previous algorithm (with the exception, again, of Oakes 2000).

1.4.5 What the CMA Models

Given the input data, the CMA models the entirety of the comparative method. The method itself can be roughly divided into two steps: finding sets of cognates (related words in the daughter languages) and consistent sound correspondences,[3] and reconstructing various aspects of the languages: proto-sounds, proto-sound systems, and proto-words (Durie and Ross 1996; Fox 1995). The first of these steps has seen significant work in the last few years, and the main goal of the CMA is to advance the state of the art by integrating all the best aspects of existing algorithms and using previously unused diachronic databases (see section 2 for further discussion)

If the first step has been successfully modeled in the past, the second step — reconstructing aspects of the proto-language — is nearly entirely unexplored. Bouchard-Cote et al. published a series of articles on the topic (2007, 2008, and 2009), but their method bears essentially no resemblance to the classic comparative method. The only other attempt was that of Oakes in 2000, but this algorithm was fairly unsuccessful as both the author himself and Kessler point out (Oakes 2000; Kessler 2005). One of the reasons for this lack of success, as Kessler (2005) points out, was that Oakes' methods for evaluating how good a potential proto-sound is and more specifically the likelihood of a given sound change were both fairly crude. This flaw the CMA will attempt to remedy. On the other hand, there is a flaw that persists in the CMA: the algorithm only models sound changes that involve one sound at a time. Thus, sound changes such as metathesis (the transposition of characters in a word, such as the /t/ and /s/ in 'tangis' and 'sangit'), hapology (the loss of a syllable followed or preceded by a similar sounding syllable), and reduplication (the repetition of a certain part of a word emphasis, such as repetition of the initial syllable for emphasis) cannot be modeled here (Oakes 2000; 236). (To some extent, this problem also effects cognate identification, and none of the algorithms available for that model those changes either). This defect, unfortunately, cannot be remedied by the CMA. On the other hand, the CMA, coming nearly twelve years after Oakes, is able to take advantage of recent advances in theory to create an algorithm that models the human linguist more closely, uses a more refined scheme for comparing sounds, and provides multiple options for proto-sound system systems and proto-words (see section 3 for further discussion)

Thus, the CMA, as described in sections 2 and 3, is an up-to-date algorithm

3 A 'sound correspondence' or 'correspondence set' is "a set of 'cognate' sounds; the sounds found in the related words of cognate sets which correspond from one related language to the next because they descend from a common ancestral sound. (A sound correspondence is assumed to recur in various cognate sets)." (Campbell 1998, p.112)

that is able to model the entirely of the comparative method algorithm. This algorithm successfully outputs several options for reconstructed sounds, proto-forms, and the sound sound-system for the proto-language, all equipped with DOCE estimates. Moreover, The CMA is thus by no means a black box: the user linguist would also have access to the cognate sets and correspondence sets, with their respective DOCE estimates. Because of this, the user-linguist can use the CMA both to create reconstructions and to assist him/her through any particular stage of the process.

1.5 Summary of §1

In this section, we have outlined the goals and scope of the Comparative Method Algorithm, a computational implementation of the full classic phonological Comparative Method. This CMA will be a new development in the computational historical linguistics field in three different ways. First, while there have been many efforts to automate the first step of this process (finding cognates and correspondences), the CMA will be only the second effort to provide a way of automating the entire method including the reconstruction of proto-sounds, with the first being Oakes' effort in 2000. Furthermore, the CMA will output Degree of Certainty Estimates (DOCEs) for each proposed reconstruction, and this will provide a new way of discussing the 'reality of reconstruction' issue. Finally, the CMA, unlike the classic method or any previous algorithm, will be able to generate several forms of any reconstructed word, which would be a hypothetical solution to the issue of uniform reconstructed languages.

In order for these goals to be met, the CMA must follow two heuristics in its design: it must model the classic method as closely as possible and retaining an empirical, pre-theoretic approach throughout.

Given that these are the goals and heuristics, we have also established the input it requires. The CMA, as discussed in section 1.4.1, takes in monomorphemic word lists of potentially related languages in IPA orthography that have been vetted for borrowings. Furthermore, the user provides a DOCE for either the whole input set, individual words, or even individual characters, as he might choose.

Armed with information about the goals, heuristics, and input into the CMA, we can now look at the way the CMA implements both steps of the classic Comparative Method: finding cognates and correspondences (section 2) and reconstructing the proto-sounds, proto-words, and proto-sound system (section 3).

2 Cognates and Correspondences

The first step of the Comparative Method is the search for cognates and correspondence sets, and the goal of this section is to propose an algorithm that would accurately model the comparative method, be as pre-theoretic as possible, and calculate the relevant DOCEs along the way. As mentioned in the previous section, there has been a considerable amount of literature on the topic over the past nearly 100 years, and the main goal of the present proposal is to show that an algorithm fitting those three heuristics can be assembled from the various current proposals. Since my aim is mainly to prove that this algorithm will meet all three heuristics, the focus will be on the way the algorithm implements the work of the human linguist, not on the more purely computational aspects of the algorithm. At any given step, I will provide at least one example of implementation in the previous literature, however, to illustrate that the task is plausible.

To contextualize the discussion, section 2.1 provides a detailed discussion of how the human linguist approaches the topic. Thereafter, section 2.2 ascertains the overall outline of the algorithm, after a discussion of the options available in the literature. Finally, the last three sections (2.3, 2.4, and 2.5) concern the salient steps and aspects of this algorithm and presents the CMA approach in context of the literature. While many works will be referred to throughout this discussion, four recent proposals that capture the main advantages of past work and meet these criteria to some large degree will be used as the main models for the CMA algorithm: Kondrak and Hauer (2011), Delmesteri (2011), Steiner et al. (2011), and List (forthcoming) (in two different versions).

2.1 The method of a human historical linguist

Armed with a set of word lists in the hypothetically related languages, the linguist searches for cognates and correspondence sets, using a somewhat recursive process (List forthcoming). First, the linguist creates an initial list of potential cognate words based on a number of factors. He considers the words' similarity in meaning and form (i.e. sound and string similarity), being careful to take into account the probability of chance resemblances (Trask 1996; Wilkins 1995). He also looks at the plausibility of the phonological pathways implied by his word pairings (i.e. if he pairs /het/ and /ket/, he considers whether /h/ and /k/ could have come from the same proto-sound) (Fox 1995; Wilkins 1995). In looking at plausibility, he is also considers the context of the changes (including prosodic ones), since many sound changes are much more likely in specific contexts (List 2012). In looking at these factors, he is also careful to look into any possibility of faulty data as well the possibilities that

borrowings or onomatopoetic words may have found their way into his word lists (Fox 1995). (In the CMA, these precautions are taken care of by the user linguist).

Using the initial list of cognates, the linguist creates a list of the regular sound correspondences among those cognates. Hiding behind the word 'regular' is again a list of factors that the linguist considers before adding a correspondence set to his list: How sure is he that the words from which the correspondence set came were cognate? across how many word sets does the correspondence hold? How many languages of the ones considered does it involve? (List 2012; Steiner et al. 2011).

When the list is complete, the linguist can edit or re-create the cognate list, now using the correspondence sets, a more accurate measure than mere similarity of form and meaning, as the basis of evaluating how likely the words are to be cognate. As Trask points out, the great strength of the comparative method is that the systematic sound correspondences, rather than more flimsy measures, are the basis for similarity evaluation (Trask 2000 cited in List 2012). Thereafter, he can again update the correspondence set list, and the process continues until no further changes to either the cognates or the correspondences are being made or until a sufficiently satisfactory result is reached (List 2012 forthcoming).

This is the process that the CMA is meant to model. In the next section, the question of how the algorithm ought to work, on the most general level, is addressed. Since, in the next section and thereafter, I propose to build the algorithm on the basis of four recent models mentioned above, please refer to Table 1 throughout the next four sections.

	algorithm aspect	Kondrak and Hauer	Delmestri	Steiner et al.	List version 1	List version 2	CMA
1	Number of times data is processed	once	once	recursive loop until convergence	twice	twice	recursive loop until convergence
2	Effects of chance accounted for by	thresholds	thresholds	correspondences and thresholds	correspondences and thresholds	correspondences and thresholds	correspondences and thresholds
3	Similarity of form measured by	string similarity	alignment	alignment	alignment	alignment	string similarity and alignment
4	Semantic similarity considered	no	no	yes	no	no	yes
5	Learning from correspondences	N/A	none	yes	yes	yes	yes

Table 1 Summary of the Differences Between the Algorithms Considered in this section, including the CMA

2.2 The General Outline of the Algorithm

(See rows 1-2 of Table 1)

The human linguist, as we saw in section 2.1, employs a recursive (circular) pattern in the cognates and correspondences search. Most of the earlier algorithms, however, do not do this, nor do they consider it: they fix on some phonetic and/or 'plausibility of pathway' method for comparing words and establishing cognates, determine correspondences based on those cognates, and consider the process done (e.g. Kondrak an Hauer 2011; Delmestri 2011; Oakes 2000). These methods thus do not consider regular correspondences in establishing cognates at all, thus not incorporating what Trask calls the chief advantage of the comparative method (1996). Moreover, using correspondences is one of the main ways the comparative method avoids falling for chance resemblances between words, and thus these algorithms face additional danger in that realm — they try to avoid chance resemblances only by setting high thresholds vis a vi how similar the two words must be (see row 2 of Table 1). In addition, most algorithms entirely ignore semantic similarity, though both sound similarity (i.e. similarity of form) and 'plausibility of pathways' may be included. An alternative method (used by List 2012) is to establish cognates at first based purely on sound similarity measures, determine correspondences, and then re-determine cognates incorporating the new data. This method is obviously better but still considers the data only twice and does not consider semantic similarity.

Finally, Steiner et al.'s algorithm remedies these faults. Their algorithm is similar to List's, except that it is equipped to continue iterating through the cycles until satisfaction is reached. With slight modification, it could also be equipped to stop only when no additional changes are made. This would then be the more accurate model of the human linguist, who may check his cognate list against his correspondences more than once. Furthermore, Steiner et. al's algorithm takes into account similarity of meaning, which is also done by the human linguist. Thus, the Steiner et al. (2011) algorithm will be the basic model used in the CMA.

In the next few sections, I will present the detailed model of the CMA algorithm for finding cognates and correspondences, following the general outline in Figure 1.

Thus, the algorithm will first look at all the pairs of words in the word lists to compare their form (section 2.3) in two different ways (by looking at string similarity (section 2.3.1) and by performing pair-wise alignment (section 2.3.2)) before determining a way to combine the two methods to give each pair of words a combined DOCE score (section 2.3.3). When all words are paired and each has a determined DOCE for its alignment, each word pair will also

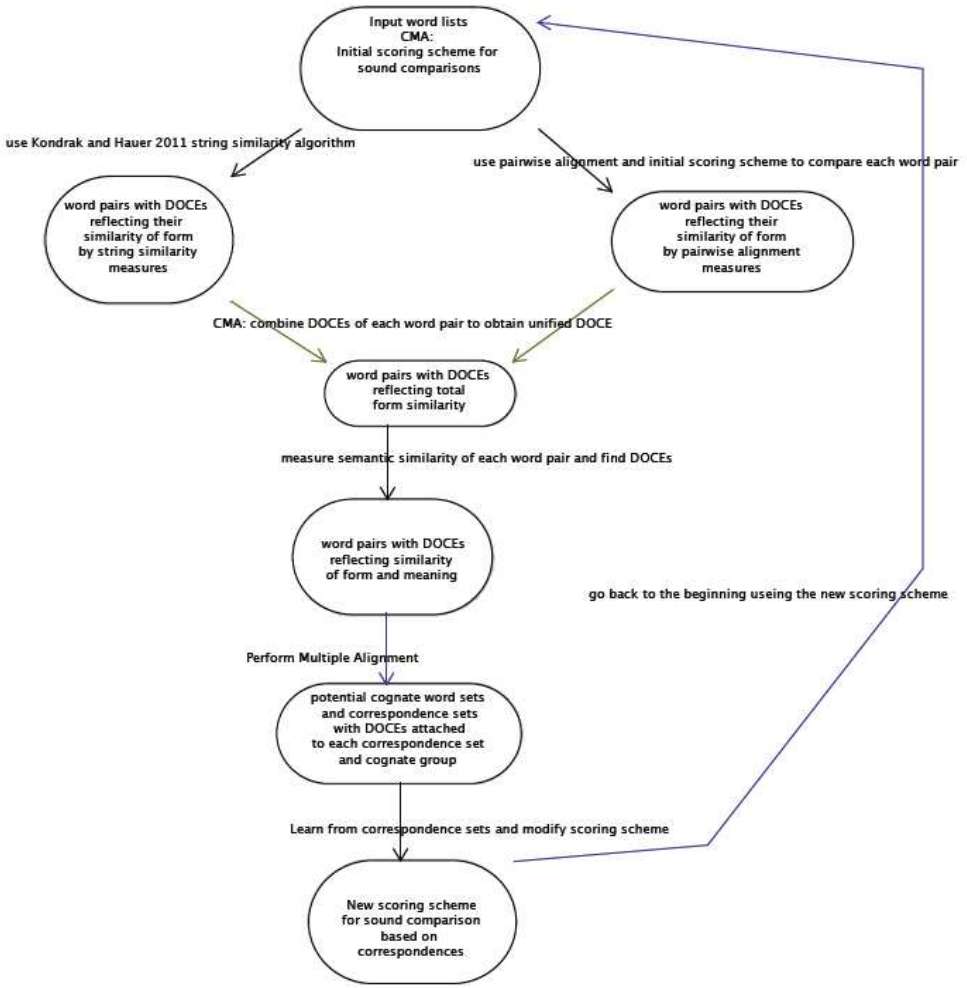

Figure 1 Outline of the CMA Cognates and Correspondences Algorithm

be measured for semantic similarity (section 2.4), as a human linguist would consider it, and this measurement will be added to the word pair's overall DOCE. Finally, the word pairs will undergo multiple alignments, in which their individual DOCEs will be used to combine word pairs into larger cognate groups, which will then be aligned to generate the set of correspondence sets (section 2.5). Once the correspondences are found, the algorithm can learn from them (section 2.6) and then the process restarts. The difference is that the scoring scheme used to compare the forms of the word pairs now has

information from the previously found correspondence sets. This iterative process can then continue until running it again generates no further changes.

What follows is the discussion of the specifics of these general steps, and, throughout this discussion, the questions of how to remain pre-theoretic, how to best model the comparative method, and how to calculate the relevant DOCEs will also be addressed.

2.3 Form Similarity

(See row 3 of Table 1)

The first step of all such algorithms is to compare the words in terms of form, as that is the most important measure, until sound correspondences become available. There have been two major types of approaches to measuring suc: looking at string similarity alone and performing alignment with a sound similarity scoring scheme (Delmestri 2011). Both are effective and both model the human linguists in different ways, so that, after discussing both (section 2.3.1 and 2.3.2), I will show a way to combine the two for the purposes of CMA (section 2.3.3)

2.3.1 String Similarity

The first approach is to compare the words as strings of symbols, looking at such measures as the number of n-grams (sequences of n letters) they share, the number of identical first characters, etc.(Delmestri 2011). In such approaches, any two characters are considered either different or identical, with no account taken of degree of similarity. This last aspect of the method is, thus, a false model of the human linguist, who considers *how* similar sounds are and even how likely they were to come from the same sound. On the other hand, since the human linguist does look at the words as strings as well as at individual characters, it is plausible that such characteristics as number of identical initial characters play a part in his decision. Furthermore, this kind of method is purely pre-theoretic in terms of phonetic theory, as it does not even take into account any specific way of comparing sounds — the only place where theoretical bias (in the form of different phonetic theories) may enter into the form similarity discussion. Finally, such methods have been reasonably effective (Delmestri 2011).

Of the algorithms represented here, Kondrak and Hauer's algorithm follows this string similarity path. However, rather than taking any particular measure (for, as Kondrak and Dorr determined, a group of string similarity measures works better than any single one of them), Kondrak and Hauer found the subset of sound similarity measures used in the literature that, when taken

together, produce the best effect (Kondrak and Dorr 2004; Kondrak and Hauer 2011). (Specifically, they used the minimum edit distance, longest common prefix length, number of common bigrams, the length of each word, and the difference in length between the longer and the shorter word (2011: 867)). In thus drawing from the past accomplishments in the area, Kondrak and Hauer's algorithm can be considered a sort of synthesis of previous work and will thus be used as the CMA string similarity algorithm.

If this algorithm is used, how could the DOCE be calculated? This algorithm outputs a 'score' for each word pair that indicates how good of a cognate candidate that word pair is, based on the combination of these metrics (Kondrak and Hauer 2011). This score could be fairly easily normalized to a percentage — a DOCE that would reflect the similarity of the words. Notably, however, Kondrak and Hauer's algorithm (like all similar algorithms), does not help determine correspondences — it only determines which words are similar enough in form to be cognates. Nonetheless, since this kind of algorithm is often represented in the literature, to some degree accurately models the human linguist, is reasonably effective, and is pre-theoretic — it should be included in the CMA in some form. Before considering how to best do this, however, we want to look at the other type of form similarity algorithm - pairwise alignment.

2.3.2 Alignment Models

The other type of approach is based on measuring similarity through alignment and on the idea that sounds may have different degrees of similarity. This type of algorithm allows the CMA to model the human linguist in a different way — it can take account, like the human linguist, similarities between sounds and even of probable pathways between sounds. All three algorithms (including both versions of the List algorithm) besides Hauer and Kondrak's belong to this type. Any such algorithm has two main components: a scoring scheme for comparing two sounds (discussed briefly in section 2.3.2) and a structural aspect, which uses that scoring scheme to align words and find correspondence sets (discussed in section 2.3.2) (List forthcoming).

The Scoring Scheme In creating the scoring scheme, the goal, as before, is to model the human linguist as closely as possible while relying as much as possible on empirical data and remaining pre-theoretic. Unfortunately, most previous efforts do not fully meet either one or the other of these goals. Some intentionally create a scoring scheme too simple to even approximately model a linguist's knowledge and intuition (e.g. Steiner et al. 2011); others commit to a particular phonetic theory, not fulfilling the pre-theoretic requirement

(see Kessler (2005) for examples); yet others base their scoring scheme, to some large degree, on intuition, which lacks the empiricism the CMA hopes to embody (e.g. List forthcoming version 2).

Instead of adopting any of these options, the CMA proposes to create a new scoring scheme based on recently developed diachronic databases: the Brown et al. database of sound correspondences (Brown et al. 2011), the UNIDIA database of sound changes recorded in the literature (DiaDM 2012), and Engstrand et al.'s database of sound changes recorded in the literature (Hamed and Flavier 2009). Up till now, only List has uses a database-centered approach (List forthcoming), but he only uses the sound correspondence database. The CMA, on the other hand, will base its scoring scheme on three databases, thus using a much broader scope of data. The use of these databases will allow the CMA to create a scoring scheme that calculates the degree of sound similarity based on empirical data and takes into account the precise context (i.e. IPA characters or prosodic boundaries before and after the sound) of sound correspondences, something that has not been done previously, because only the UNIDIA and Engstrand et al. database, that have previously not been used, record context at all. Unfortunately, since the focus of this paper is on the flow of the overall algorithm, it is not possible to present the full details of the scoring scheme calculations in this paper. Instead, further discussion of the choice to not use phonetic theory, a detailed discussion of the databases, and the method used to calculate the similarity degree are available in Gilman (2012b). For the purposes of the current discussion, it is mainly relevant that the CMA provides a new scoring scheme that is based on empirical databases and that generates a degree of similarity between 0 and 1 for any two IPA characters, as well as for some particular contexts of any given change (the contexts in the scoring schema are those discussed in the literature as relevant).

The Alignment Algorithm Once the algorithm is armed with a scoring scheme, it finds the phonetic alignment for all pairs of words. While there are many options for this step of the algorithm, the differences between them are mainly computational. They are all fairly pre-theoretic, and, while they do model the human linguist in slightly different ways, a full discussion of this is beyond the scope of this paper. Since the choice of algorithm is mostly computational, I propose to use the List model for the first version of the CMA, simply because it is open source and available online (http://lingulist.de/lingpy) (List forthcoming). The general outline of the algorithm, however, is the same across many different implementations, and it is presented as such below.

First, it considers each pair of words and uses dynamic programming[4] to

4 Dynamic programming is essentially a method for breaking a given problem down into

find the way to align the two words, so that they have the highest possible overall score (List 2012). This overall score is calculated by combining the scores associated with the different character pairings. Thus, for example, if the words are /king/ and /pig/, the algorithm would align them as in Table 2. Assuming that the similarity metric somehow took into account that /p/ and /k/ are more similar than /n/ and nothing, the scores for each pair might be something as in line three of Table 2, and the overall score would thus be 0.5+1+0+1 = 2.5

Word1	k	i	n	g
Word2	p	i	[gap]	g
Alignment score	0.5	1	0	1

Table 2 Aligning 'king' and ''pig'

If, on the other hand, the words were /king/ and /pit/, the alignment would depend on whether the scoring scheme said /g/ and /t/ or /n/ and /t/ were more similar. If /g/ and /t/, were more similar, the alignment in the left of Table 3 would be preferred to that in right of Table 3, since the score from the left part of Table 3 alignment is better (1.8 rather than 1.6). 1.8 would thus also be the resulting 'sound similarity' score for that potential cognate pair.

Word1	k	i	n	g	Word1	k	i	n	g
Word2	p	i	[gap]	t	Word2	p	i	t	[gap]
Alignment score	0.5	1	0	0.3	Alignment score	0.5	1	0.1	0

Table 3 Aligning 'King' and ''Pig' in two Ways

This score, once it is calculated can be normalized against the 'perfect score' (i.e. the score that the pair would have received if the words were identical) and thus converted into a DOCE estimate for the 'form similarity' of the word pair.

Furthermore, in doing the alignment, the algorithm also identifies the regular sound correspondences necessary — they are simply the columns in the

small steps. For a good explanation of how dynamic programming is used in application to alignment, see List forthcoming.

tables above. We thus have the lists of correspondences to be used in the next step in the process, and the DOCE scores associated with them (i.e. their individual scores). Before multiple alignment or semantic similarity are introduced, however, we want to consider how best to combine these alignments and DOCEs with the string similarity measures discussed in section 2.3.1.

2.3.3 Combining the string similarity and alignment models

Both of these types of algorithms have seen success in various recent work, and they model slightly different aspects of a human linguist's intuition about similarity of form: the string similarity models reflect the ways a human might look at the strings themselves and the alignment model focuses more closely on sound similarity. Since the models are slightly different, it is also plausible that they may work most effectively on slightly different types of word pairs. Thus, both from a 'modeling the human' and from an effectiveness point of view, it would make sense to integrate both models into the CMA, something that has not, to the best of my knowledge, been done previously.

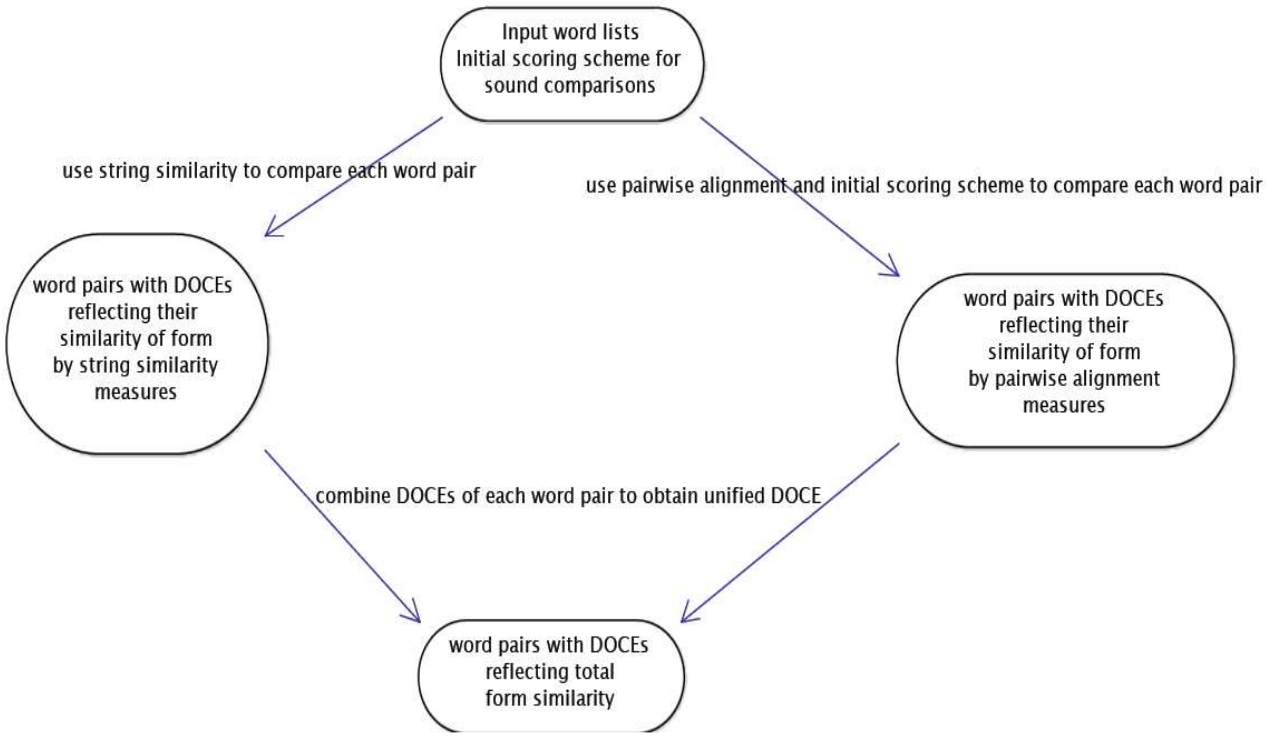

Figure 2 Combining String Similarity and Alignment

How can these two methods be combined? (Throughout the following discussion, refer to Figure 2) Both algorithms calculate the similarity score and DOCE for each word pair, so the combined DOCE score of the two word pairs could be used to determine the overall DOCE. In combining the DOCEs, it is probable that the two models should have different weights, depending on which one is found to be more effective. Thus, hypothetically, an equation such as (1) should be used to calculate the overall DOCE, where $DOCEtotal$ is the overall DOCE, $weightSim$, $doceSim$, $weightAl$, and $doceAl$ are respectively the weights and DOCEs associated with the string similarity and algorithmic word comparison measures.

(1) $$DOCEtotal = weightSim \times doceSim + weightAl \times doceAl$$

These total DOCEs would thus reflect the similarity in form of any given word pair. Having measured similarity of form for word pairs in this new way, the CMA can now incorporate the semantic similarity of the word pairs into their similarity DOCEs (section 2.4), before moving on to grouping the pairs into larger cognate sets (section 2.5).

2.4 Semantic Similarity

(See row 4 of Table 1)

In addition to similarity in form, semantic similarity is always a factor in the human linguist's analysis. For the CMA, this factor is not particularly significant, since the CMA mostly relies on word lists where meanings are likely to be nearly identical. For Steiner et al., on the other hand, this is especially important, since their algorithm takes as input any dictionary type word lists, not just carefully pruned ones such as those usually used in the classical comparative method. However, even in the classical method, it is possible for words to have slightly different definitions in the word lists, and thus there should be some way of calculating a DOCE penalty for potential cognates whose meanings differ to some degree. The two solutions considered here are those employed by Steiner et al. (section 2.4.1) and one employed by Kondrak 2011 (section 2.4.2). The goal, as before, is to choose the method that models the human linguist as closely as possible.

2.4.1 The Steiner et al. method

Steiner et al. essentially create matrices of meanings, where the distance between two meanings depend on how similar in form the words for them are

across many languages. The intuition behind this method is that "similar meanings have a larger probability to be expressed similarly in human language than different meanings" (Steiner et al. 2011: 10). The data set for the many languages can be either "the word lists of [29] Indo-European languages from the Intercontinental Dictionary Series (Key and Comrie 2007)" or taken from the data set itself, if it is large enough (Steiner et al. 2011: 10). The distance, thus calculated, can be converted into a percentage by normalizing it against whatever distance is deemed sufficiently large to reflect no semantic similarity. This percentage, then, which can be subtracted from the positive DOCE estimate calculated for the two words during the 'sound similarity' step, thus punishing semantically divergent words.

If we are to evaluate how closely this method models the human linguist, two problems arise. The first is that semantic similarity is here, in a way, modeled on similarity of form, and this assumption does not model accurately the human linguist, who presumably considers shades of actual semantic meaning (Trask 1996). In addition, the CMA should be usable for any language family, which makes using data from the Indo-European languages only potentially problematic. On the other hand, the input data sample, which is likely to consist only of short Swadesh-like word lists and may feature only a few languages, would also be a bad source for creating such matrices. Thus, Steiner et al.'s solution, while plausible, does not seem to be ideal for the CMA.

2.4.2 The Kondrak Method

An alternative solution, as developed by Kondrak, uses WordNet to calculate a sort of semantic distance between the words (2011). WordNet includes a lexical hierarchy of word senses expressing relationships such as hypernymy and hyponymy, which allows one to track the distance between any two senses (Wordnet 2012). The distance can be converted into a DOCE penalty, just as it would be with the Steiner et al. method.

How accurately does this method model the human linguist? As Kondrak himself points out, the WordNet database allows one to account for certain types of semantic changes — including generalization, specialization, synechdoche — but not others, such as metonomy, melioration/pejoration, and metaphor (Kondrak 2011). Nonetheless, this method is at least based on types of actual semantic change as considered by the linguist, which makes it preferable. On the other hand, using WordNet, developed based purely on English meanings, only aggravates the problems of using Indo-European data, as in Steiner et al.'s solution (WordNet 2012). However, this same kind of bias exists for the historical linguist, who can only rely on the semantic similarity in the glosses (i.e. in the linguist's native language). In fact, using English

only is more realistic than using a full set of Indo-European languages, as such knowledge is not, presumably, usually within the grasp of the human linguist. Thus, while this is a problem, it models a flaw that exists in the classic comparative method. Since the goal of the CMA is to model the comparative method as closely as possible, this is a good thing in context of the CMA. Thus, overall, the Kondrak WordNet model is a more CMA-appropriate way to model semantic similarity, and this method shall be used in place of the original Steiner et al. method within the CMA.

Using the form similarity comparison method developed in section 2.3 and this semantic similarity method, the CMA will thus identify pairs of words with DOCEs reflecting how similar they are in both form and meaning. Equipped with these DOCEs, the algorithm can now proceed with multiple alignment.

2.5 Multiple Alignment

The next step is multiple alignment, where the full cognate sets are brought together and aligned, and the correspondence sets are found. Here, again, there are a variety of options, which differ mainly in computational ways and model the linguist approximately equally well. The CMA (at least in this first version) will simply use List's model as the most easily available (see List forthcoming for details). At the end of such a process, the CMA has a full list of cognate sets, each one presented as in Table 4 below, with the columns representing the correspondence sets.

Language1	b	i	z
Language2	p	e	[gap]
Language 3	p	i	ʃ

Table 4 A sample alignment

Furthermore, at this step, the DOCE associated with each correspondence set is calculated. When a linguist considers the reliability of a correspondence set (regularity aside), he looks at the sounds that must be considered similar, the number of languages in the correspondence set (i.e. the number of languages which supply an IPA character rather than NULL as their contribution), and his confidence in his own alignment. All of these factors must thus be included in the overall DOCE, and each would bear its own weight. Notably, however, the similarity of the sounds is already integrated, in the CMA's case, into the DOCE of the alignment (i.e. the linguist's confidence

in his alignment). Thus, there are only two categories. The exact weights are not easily calculable, so they should be picked very approximately and fine-tuned for the best performance after the algorithm is completed. Suppose thus, that each consideration has a weigh, and that the weights add up to 1: $weightN$ is the weight associated with the number of languages in the set, and $weightC$ is the DOCE of the cognate set as calculated by the multiple alignment algorithm. With these weights in mind, the overall DOCE might be calculated using something like the formulation in equation (3), where the weights are as defined below, $langinv$ is the number of languages involved in the correspondence set, $langtot$ is the number of languages being compared in this instance, and $CognateDOCE$ is the DOCE of the cognate set overall.

$$DOCEcorresp = weightN \times \frac{langinv}{langtot} + weightC \times CognateDOCE \tag{2}$$

If this formula is used, each correspondence set, as well as each cognate set, now is both found and has a DOCE that incorporates the previous DOCEs. Notably, the DOCEs associated with the correspondence sets will *not*, in this first version of the CMA, be used in the next step (learning from correspondences), but they will become very important in the discussion of actually reconstructing proto-sounds in section 3. The next step, meanwhile, is to learn from the correspondences.

2.6 Learning from the Correspondences

(See row 5 of Table 1)

The final step of the process is learning from the correspondences established to create a new scoring scheme. Since regular correspondences are, as Trask (1996) points out, more important than similarity of form and meaning, they will now play the greatest role in the scoring schema. Meaning and form similarity should not, however, disappear completely, as those categories are still important (see below for formulaic discussion).

In learning from the actual correspondences (as opposed to calculating their DOCEs as discussed in section 2.5), the main factor is the regularity of the correspondences. There are a range of different formulas used to calculate the probabilities of transition; the decision is mainly computational, and, while there may be subtle differences in how an equation models the human linguist, a discussion of such differences is beyond the scope of the current paper. For this version of the CMA, Delmestri's PAM-like matrix formulas, a substitution matrix tool from bioinformatics, will be used, as Delmestri deems them to be the most effective (at least in her algorithm) (Delmestri 2011). This formula

thus yields the degree of similarity between two sounds, as predicted by the regular correspondences found.

The only remaining step is then to integrate the new knowledge learned from correspondences with the old information on sound similarity. Most previous algorithms seem to neglect to do this, but this step is nonetheless critical to modeling the human historical linguist, who does not forget everything he knows about likely correspondences and plausible pathways after considering the correspondences. On the other hand, as Trask points out, the correspondences do play significantly less weight, so that the weights of the two considerations will be unbalanced (at a first approximation, one could guess 30% for values from the old scoring schema and 70% for the new values). Thus, the formulas for the final probabilities might be calculated as in equation 3 below, where the variables new the probability ($ProbNew$), the weight of the old knowledge ($WeightOld$), the old probability ($ProbOld$), the weight of the new information found out from the correspondence sets ($weightCorresp$), and the probability derived from the correspondences ($ProbCorresp$)

(3) $$ProbNew = weightOld \times ProbOld + weightCorresp \times ProbCorresp$$

With these equations, the new scoring scheme is created, and it can be used to restart the process of finding Cognates and Correspondences. This process, then, of finding cognates and correspondences can be repeated until the new scoring schema does not differ from the old. At this point, the correspondence sets and their DOCEs can be considered as set in stone, and the next step — reconstruction of proto-sounds — can begin.

2.7 Summary of §2

In this section, a model has been presented for finding the Cognates and Correspondences from a set of word lists. This method, outlined below in figure 3 is a combination of established practices and innovative ideas. In the figure 3, elements of this algorithm that arise from previously established models have a shaded comment.

As can be seen in the figure, the overall multiple alignments and DOCEs are calculated with a combination of string similarity and alignment algorithms, which is a new structural idea that the CMA brings to the field. Furthermore, the CMA uses a new scoring scheme for the alignment algorithm (as described in briefly section 2.3.2 and in more detail Gilman 2012b) that relies on new diachronic databases thus being simultaneously more pre-theoretic than previous scoring schemes and modeling the human linguist better, in dealing with both

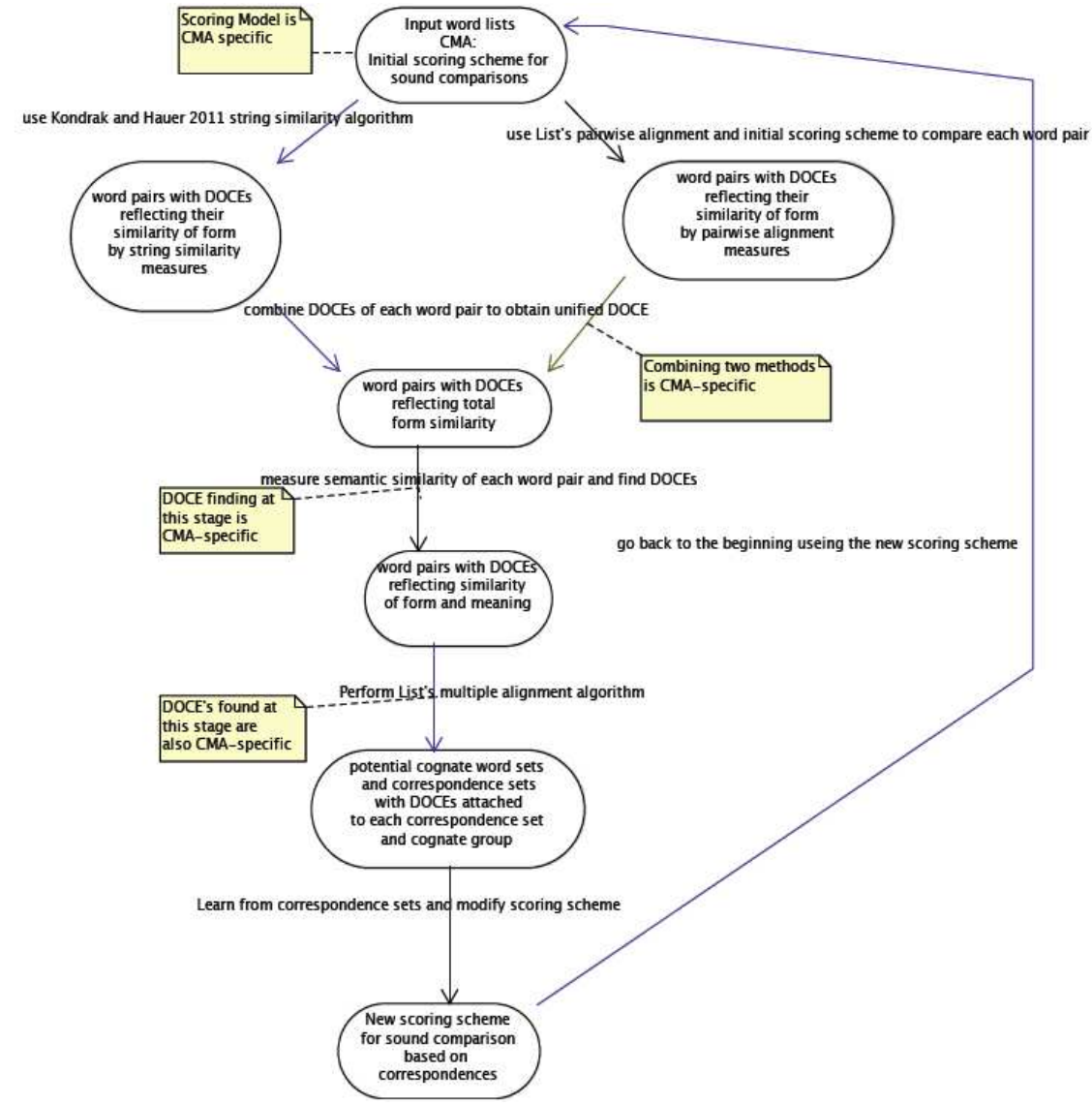

Figure 3 Outline of the CMA Cognates and Correspondences Algorithm

pure probability of corresondence and the associated 'plausibility of pathways' (i.e. using sound change databases). Finally, thougout the algorithm, the CMA preserves DOCE estimates that will play the key role in this algorithm, as it is the DOCEs calculated here will be used to calculated the final DOCEs

associated with reconstructed proto-words, which, in turm, will be used to provide another outlook on the reality of reconstruction discussion. Thus, overall, the CMA version of form comparison algorithm provides a pre-theoretic and accurate-modeling-of-human-linguist algorithm that contains two previously untried ideas in addition to the DOCEs that are the main contribution of this algorithm.

3 Phonological Reconstruction

After the 'cognates and reconstructions' step, the comparative method goes into its final phase: reconstructing protophonemes. The input for the sound reconstruction step is the set of regular correspondences and the cognate sets found by the cognates algorithm, along with their DOCE estimates. The output should be proto-sounds, the sound system of the proto-language, and proto-wordforms — all with DOCE estimates. Furthermore, to satisfy the second goal of the CMA, this final algorithm componenet should output several solution options, thus resolving Pulgram's paradox of a uniform reconstructed language. At the same time, as with the previous step, the goal is to remain pre-theoretic and model the human linguist as closely as possible.

The only real basis for creating such an algorithm is currently Oakes algorithm from 2000. Notably, there was one other algorithm that aimed to reconstruct proto-sounds, by Bouchard-Cote et al. (2007; 2008; 2009), but it does not model the comparative method. Since, in order to be useful in evaluating the reality of reconstructions, the CMA algorithm must model the comparative method as closely as possible, there is very little that the CMA can gain from the Bourchard-Cote et al. proposal. Instead, the algorithm I propose is roughly based on Oakes, but modifies it in several critical ways. Notably, Oakes does not reconstruct the proto-sound system, nor, of course, does he deal with multiple outputs.

Thus, this discussion, much as the discussion in the previous section, will first introduce the human method for this step (section 3.1) and then take apart the three sections of the algorithm — establishing the proto-phonemes (section 3.2), reconstructing the proto-sound systems (section 3.3), and reconstructing proto-words based on the proto-sound-systems (section 3.4)

3.1 The method of the human historical linguist

At the reconstruction step of the process, the human linguist goes through two steps: he reconstructs proto-sounds (and proto-sound system) and then uses that system to reconstruct the proto-words.

In the first step, the human linguist, considers many factors in looking at

candidate proto-sound: explanatory power (explain all the data), majority (if one sound predominates in the correspondence set), economy (as few changes as possible between the proto- and current sounds), naturality (sounds changes must be such as occur in history), a preference for reconstructing the minimal number of phonemes, a preference for creating a balanced system balance, a preference for not reconstructing any phonemes foreign to the language family, an awareness that some phonemes may have been lost through sound mergers, and the possibility of parallel development (Crowley 1992; Lass 1993; Trask 1995; Fox 1995). To simulate the human, a reconstructing algorithm must, ideally, attempt to consider all (or at least most) of these factors as well.

Once all the proto-sounds are established, the sound system, the list of unique sounds reconstructed, can be assemebled.

Finally, the last stage is the reconstructing proto-words stage — a fairly mechanical process for both the human linguist and the algorith. In fact, it is simply a matter of plugging the found proto-sounds in for the sound correspondence groups. This step, both for the human and for the computer, is fairly simple.

The only other addition is that the CMA must deal with Pulgram's paradox, which is not part of the standard human linguist's work. For that purpose, multiple proto-sounds (and thus proto-sound systems) will be generated, from which the proto-words are later composed. Thus, the basic aim of the reconstruction step would be accomplished.

In what follows, I will accordingly present the three steps of this algorithm: determining the proto-sounds (section 3.2), creating the sound system (section 3.3), and generating proto-words based on these systems proto-words (section 3.4).

In all of these sections, I attempt to outline the CMA version of this algorithm in as much detail as possible, pseudo-code included. This attempt is provided here, because the CMA is only the second method since Oakes to deal in reconstruction, and thus there is a heavier burden of proof in terms of showing that such an algorithm, meeting all the heuristics required here, is possible. The pseudo-code blocks in the sections that follow should illustrate that the algorithm is entirely possible. The caveat, however, is that this particular method may not be the best possible method — it is merely *a method* that meets the heuristics of remaining pre-theoretic and modeling the human linguist reasonably well while also calculating DOCEs and providing multiple outputs.

3.2 Reconstructing Proto-Sounds

3.2.1 The Overall Outline

The basic idea of the CMA reconstruction algorithm is surprisingly simple, closely modeling that of Oakes as shown in Algorithm 1 (Oakes 2000). For every correspondence set, the CMA goes through every candidate proto-sound (i.e. every IPA character), considers how likely it is to be the proto-sound, and assigns it a DOCE reflecting how good of a proto-sound it would be. The IPA character(s) with the best DOCE score(s) win and are considered proto-sound options.

The plural here refers to the multiple outputs discussed in the introduction. There are several potential ways of generating these multiple outputs: one could preserve only the best set number of candidates (e.g. 2), only preserve multiple proto-sound candidates when there is a tie in the DOCE (or something very close to a tie where the DOCEs of the top two candidates are only some small percentage apart), or preserve all candidates above a certain DOCE threshold (e.g. 40%). The second method will be adopted: the algorithm shown below (Algorithm 1) preserves only tied DOCEs. While this is not necessarily the best method, it is the simplest and is a reasonable illustration of the sorts of DOCEs the CMA may want to preserve.

This simplistic algorithm approximation declares three initially empty variables to represent the best proto-sound candidate ($BestCandidate$), the other candidate with the same DOCE score ($OtherBestCandidate$), and the DOCE of these two best candidates ($BestCandidateDOCE$) (lines 3-5 in Algorithm 1). After finding the DOCE for each candidate (i.e. the $CurrentCandidateDOCE$ for the $CurrentCandidate$ by using the 'findDOCE' function in the algorithm at line 11 that will be defined in section 3.2.3), it then compares this DOCE to the previous best DOCE (line 9). If the new DOCE is found to be better, then it is declared the new $BestCandidateDOCE$, and the associated candidate is declared the $BestCandidate$ (line 10-11). If the $CurrentCandidate$ has the same DOCE as the previous best candidate, it is assigned to the $OtherBestCandidate$ slot (12-14). Notably, this means that if there are more than two candidates with identical DOCEs, the first and last ones will be the ones picked. Since they are picked arbitrarily, however, this is an acceptable way to proceed, as long as the goal is to pick only two candidates. Later versions of the CMA may modify this part. With the current strategy, however, the algorithm knows what the best candidates for the proto-sound are and what their common DOCE is at the end of the for-loop above.

The main difference between this model and the work of the human linguist is that the human linguist may not feel the need to be quite that meticulous

Algorithm 1 The Very Basics of the CMA Sound Reconstruction Algorithm

```
    for all sound correspondences do
        {STEP 0: Initialize Variables}
 3:     BestCandidate = NULL
        BestCandidateDOCE = 0
        OtherBestCandidate = NULL
 6:     CurrentCandidate = NULL {Constantly changes inside loop}
        CurrentCandidateDOCE = 0 {Constantly changes inside loop}
        for all IPA characters do
 9:         {STEP 1: calculate the DOCE score for each candidate proto-sound}

            CurrentCandidate = CurrentIPACharacter
            CurrentCandidateDOCE = calculateDOCE(CurrentCandidate)
12:         {STEP 2: add the current candidate to the best candidate couple, if
            it is better than or ties with the previous best candidate}
            if CurrentCandidateDOCE > BestCandidateDOCE then
                BestCandidateDOCE = CurrentCandidateDOCE
15:             BestCandidate = CurrentCandidate
            else if CurrentCandidateDOCE = BestCandidateDOCE then
                OtherBestCandidate = CurrentCandidate
18:     return BestCandidate, OtherBestCandidate, BestCandidateDOCE
```

in his processing, as he usually has some guiding intuition as to what candidates are important. Since the CMA has no intuitions, however, this step is necessary. Furthermore, this slight mismodeling should have no effect on the algorithm's ability to find the same answer as the linguist and is thus not in violation of the heuristic. Thus, with this general algorithm in mind, we can discuss what exactly the criteria for determining the DOCE of a candidate proto-sound are (section 3.2.2) before presenting a detailed outline of the *calculateDOCE* function from Algorithm 1 that is the lynchpin of this algorithm (section 3.2.3).

3.2.2 The Criteria

In calculating the DOCE of a given candidate proto-sound, the goal of the CMA should be to exactly model the human linguist and include all seven of the criteria mentioned in the human's methodology. While this is, unfortunately, outside the range of possibility for the CMA at the moment, the

CMA models six of the eight characteristics (as opposed to Oakes' four[5]). Conveniently, the CMA (unlike Oakes) calculates the score for each criterion separately (except naturality and economy, which are combined), which allows the weights of the different criteria to be adjusted to best model the human linguist. Accordingly, I discuss each criteria individually in this section before discussing the algorithm for calculating the numbers involved and combining these criteria into a final DOCE estimate for each candidate proto-sound (section 3.2.3).

Naturality 'Naturality,' for the human linguist, is a matter of considering how likely one sound is to change into another. There are, as with sound similarity, a theoretical and an empirical approach to measuring the probability of sound change. Oakes basis his analysis on one particular theory: changes are considered probable if they are listed in Crowley's textbook as natural (Oakes 2000). To remain pre-theoretic and quantitative in analysis, the CMA instead models its scoring schema of sound probability on the three diachronic databases used for the sound similarity scoring scheme. Since these databases record either the sound changes (with relevant contexts) found in the literature or the probability of a sound correspondence (which is the result of a sound change), they allow for the creation of a fairly precise scoring scheme that gives the probability of any sound change (normalized to be between 0 and 1) sans context as well as in several contexts specifically mentioned in the literature (See Gilman 2012b for more detailed discussion of why the empirical approach is preferrable, why sound correspondences are relevant, and how exactly the probabilites are calculated). This scoring scheme will thus be used to determine the naturality of a sound change.

Majority 'Majority' is the simplest of the human linguist's considerations: the more daughter-phonemes the candidate proto-phoneme is identical to, the more likely it is to be correct (Trask 1996). The DOCE of this category will be a simple percentage of agreement (e.g. 20% if 1 of the 5 daughter phonemes matches the candidate proto-phoneme). While one could (as Oakes does) consider 'majority' to be part of 'naturality,' the human linguist, at least according to stanard textbooks, attaches a special weight to daughter-sounds and candidate proto-sounds being identical, and thus the CMA likewise gives this consideration a separate weight (Trask 1996).

5 Oakes (2000) takes his list of criteria from Crowley. The list here, on the other hand, is a compilation of the lists from Crowley, Trask, Fox, and Lass.

Economy This criterion is, in this model, part of 'naturality.' For the human linguist, the 'economy' criterion is based on the assumption that certain changes take more than one step (e.g. for /b/ to become /h/, the linguist might say that first /b/ became /p/ and then /p/ became /h/) (Trask 1996). This would be, in a sense, because a change from /b/ to /h/ would be considered implausible. The CMA, however, only reconstructs a single proto-phoneme for each set of daughter-languages, at least at this stage. Thus, instead of measuring economy directly, the CMA measures it indirecly through naturality: a change such as /b/ to /h/ would have a lower probability in the sound change scoring scheme for 'naturality' described above, and thus /b/ as the proto-phoneme would receive a lower DOCE, reflecting both the unnaturalness of the change and its lack of economy. Ideally, it would be good to be able to measure this criterion separately, as the human linguist does consider it as different from 'naturality.' One solution would be to separately determine the weights the human attaches to 'naturality' and 'economy' and then assign their sum to the CMA 'naturality,' which, in the CMA, reflects both of these criteria.

No foreign sounds This criterion implies that no sounds foreign to the language family should be reconstructed (Lass 1993). The algorithm will, at the beginning, create a list of sounds in the daughter languages. If a sound outside of those is being reconstructed, it will receive a 0 DOCE score for the 'no foreign sounds' category.

Lost information The human linguist is aware that some phonemes may have been lost through sound mergers and that parallel development may be mistaken for a common ancestroy (Trask 1996). In any comparative method model, these are inherent dangers. The CMA model, however, can account for the issue, to some degree, by imposing a DOCE cost across the board. While this will not affect the winning reconstructed choice, it would affect the overall DOCE, thus bearing on the final question of how real a reconstruction is.

The only criteria not included in this model are considerations of reconstructing the minimum number of phonemes and those of system balance. The issue with both criteria is that it would require the algorithm to either find all possible proto-sound systems or run a very memory-heavy dynamic programming algorithm leading to the most economical and balanced sound system. While both are in principle possible, they are potentially beyond the strength of modern computers. That is, thus, a remaining flaw in the CMA. There is another other potential problem, mentioned in Oakes (2000) and unaccounted for

both there and in this version of the CMA: multiple correspondences may generate the same proto-sound. Opinions seem to differ on the extent to which this is allowable, as Trask (1996), for example, reconstructs the same proto-sound for different correspondences whereas Crowley (1998) recommends avoiding this if at all possible. For the sake of simplicity, the CMA currently allows multiple correspondences to yield the same proto-sound. A further version of the CMA may add an option for reconstructing proto-sounds such that each correspondences has a unique proto-sound. Aside from accounting for these two categories, however, the CMA sound reconstruction algorithm meets the necessary requirements: it calculates the relevant DOCEs, allows for multiple ouput by preserving more than one candidate proto-sound, takes into consideration an additional criterion that the Oakes' algorithm did not (number 5 on the list), allows more criteria to have separate weights, and remedies the issue with theoretical bias in naturality. The next question is how exactly such an algorithm would work.

3.2.3 Combining the Criteria into an Algorithm

The goal of this section is to determine how exactly the criteria are combined to find the DOCE of a proto-sound candidate, to define the $calculateDOCE$ function used in Algorithm 1. Such a function must calculate the individual scores for each of the criteria listed in section 3.2.2 (accomplished in Steps 1-2 of the function as outlined algorithm 2 before combining them into the final DOCE score in Step 3 in 2).

The way in which each criterion is calculated is slightly different. Majority and Naturality (which subsumes Economy) are the most complex calculations. For each proto-sound candidate, the algorithm must go through each daughter sound and calculate the Majority and Naturality scores for that candidate-daughter pair, which is accomplished in Step 1. The Majority score is simply 0 for non-identical sound pairs and 1 for identical ones, as shown in lines lines 8-9 of Algorithm 2. The Naturality score is slightly more complex: it is the probability recorded for that proto-sound — daughter-sound (and possibly context) pairing in the scoring schema, which I label in the algorithm as $ScoringScheme(candidate, daughtersound)$ in line 10 of Algorithm 2. As the algorithm calculates each score, it adds that score to the previous score (which is the sum of all the previous scores for that criterion for the different daughter-phonemes) obtaining an overall sum for all the daughter sound (see lines 9-10 of figure 2). (The values for both Majority and Naturality are initially 0, as they are defined as such when the variables are initialized in Step 0). Then, in Step 2, the algorithm divides that sum by the total number of daughter-sound, thus finding the overall scores for Naturality and Majority.

The calculations for the 'No foreign sounds' and the 'Lost information' criteria are much simpler and can both be done in Step 2, outside the daughter-sound loop (lines 12-15 in figure 2. The 'No foreign sounds' criteria is simply 1 if the candidate proto-sound is a sound in any of the daughter languages ($PhonemesNative$ stores all such sounds) and 0 otherwise. Lastly, the number associated with the 'Lost information' criterion reflects only the DOCE the user associated with their data and thus is pre-set at the beginning of this algorithm, needing no further calculation. Thus, after Steps 1 and 2, the scores associated with each criterion are known.

Finally, in Step 3, the combined DOCE score of the candidate proto-sound is calculated based on a simple weighting schema shown in line 17 of Algorithm 2. Each criterion has a weight, and these weights determine the role of each score. Notably, there is a component in that equation that is not one of the criteria: the DOCE of the correspondence itself is as calculated in section 2. Thus, using the DOCE values from section 2 and the criteria for proto-sound evaluation — all of which are scores are between 0 and 1 — the CMA calculates the overall DOCE, also thus a number in that range.

When this DOCE is found, we can use it to compare proto-sounds as discussed in Algorithm 1. In technical terms, the DOCE of a given candidate, the output of Algorithm 2 is the output of the $calculateDOCE$ function, used in Algorithm 1, the only part of Algorithm 1 that was not previously fully described. With Algorithm 1 thus fully described, the next step is to form a set of alternative sound systems based on the ties (section 3.3).

Algorithm 2 The *calculateDOCE* function that determines the DOCE of a given proto-sound candidate

{STEP 0: initializing the variables}
$PhonemesNative$ = All the phonemes that are in at least one of the daughter languages
$numdaught$ = The number of daughter sounds involved in the comparison

$Majority = 0$
$Naturality = 0$
$Foreign = 0$
{STEP 1: Go through each daughter sound in the correspondence set to find the 'majority' and 'naturality' scores}
for all daughter sounds in the correspondence set **do**
  if daughter sound and candidate are the same **then**
    $Majority = Majority + 1$ {Otherwise, we would add 0, i.e. do nothing}
  $Naturality = Naturality + ScoringScheme(candidate, daughtersound)$
{STEP 2: Deal with 'foreign' criteria and normalize in-loop values}
if candidate in $PhonemesNative$ **then**
  $Foreign = 1$ {Otherwise, it remains 0}
$Majority = Majority \div numdaught$
$Naturality = Naturality \div numdaught$
{STEP3: Calculate the Overall DOCE of the proto-sound candidate}
$candidateDOCE = Majority \times MajorityWeight + Foreign \times ForeignWeight + Naturality \times NaturalityWeight + LostInfo \times lostInfoWeight + CorrespondenceDOCE \times CorrespondenceDOCEWeight$
return $candidateDOCE$

3.3 Sound Systems

At this stage, the CMA has a list of correspondences, each paired with one or two best proto-sounds. To eventually get multiple ouput options in the proto-word lists, the CMA must now generate multiple sound systems. There are two basic approaches for finding the sound systems. One could make a separate sound system for each possible combination of proto-sounds, such that if there were two correspondence sets and each had two best proto-sounds, one would have four sound systems. The problem with this method is that it couldyield a very large number of sound systems: if there was a coincidence for every sound correspondence (highly unlikely), there could be as many as $2^{(NumberOfSoundCorrespondences)}$ sound systems. The alternative method would be to simply calculate two sound systems: one based on the original best candidate and the other based on the tying candidate (with original candidates used wherever no candidate tied). For the sake of simplicity, the CMA here uses the second method to generate two sound systems, though the modifications to use a different method or provide the reader with options could easily be made.

The algorithm in Algorithm 3 implements the two-system method. After initializing two empty sound systems, it goes through each sound correspondence and fills in the sound systems, after checking that that sound is not in the sound system yet (e.g. lines 6-7 of Algorithm 3). With each added sound, the number of the sounds in that sound system (i.e. $soundnumOriginal$ and $sundnumTying$) grow by 1 AS IN LINE (these numbers will be needed later to calculate the overal DOCE of the sound system) When filling in the sound system based on the tying ($OtherBestCandidate$) proto-sound candidates (lines 8-13 in Algorithm 3), the algorithm first checks that such a tying candidate exists (i.e. is not NULL) — it is used if it exists, and the original best candidate is used otherwise.

One notable thing is that this sound system stores three different things: it stores the sound correspondence the proto-sound is based on complete with its original DOCE as calculated in section 2 (*correspondence*), the proto-sound itself ($BestProtoSound$ or $OtherBestCandidate$), and the DOCE of that proto-sound ($BestProtoSoundDOCE$). All of that information is necessary for finding the proto-words and calculating their DOCEs (see section 3.4).

Once these systems are complete, the algorithm calculates their overall DOCEs: the average of all the DOCEs of the proto-sounds (as shown in lines 21-26 of Algorithm 3). Thus, the DOCE of the sound system includes the DOCEs of the proto-sounds themselves ($BestProtoSoundDOCE$), which, in turn, include the DOCEs associated with the correspondences ($CorrespondenceDOCE$). The DOCEs of the sound system thus still reflect the entire chain of the process

at the point when the sound systems are found. The final step is then finding the different versions of the proto-words and their correspondoning DOCEs (section 3.4).

Algorithm 3 Algorithm for Generating Sound Systems

{STEP 0: Initialize two empty sound systems}
$SoundSystemOriginal = NULL$
$SoundSystemTying = NULL$
$soundnumOriginal = 0$
$soundnumTying = 0$
{STEP 1: Put the proto-sounds into two sound systems}
for all correspondences **do**
  if $BestProtoSound$ not in $SoundSystemOriginal$ **then**
    Add the *correspondence*, its $BestProtoSound$, and its $BestProtoSoundDOCE$ to $SoundSystemOriginal$
    $soundnumOriginal = soundnumOriginal + 1$
  if $OtherBestCandidate \neq NULL$ **then**
    {i.e. if there is a tying candidate}
    if $OtherBestCandidate$ not in $SoundSystemTying$ **then**
      Add the *correspondence*, its $OtherBestCandidate$, and its $BestProtoSoundDOCE$ to $SoundSystemTying$
      $soundnumTying = soundnumTying + 1$
  else
    if $BestProtoSound$ not in $SoundSystemTying$ **then**
      Add this *correspondence*, its $BestProtoSound$, and its $BestProtoSoundDOCE$ to $SoundSystemTying$
      $soundnumTying = soundnumTying + 1$
{Step 2: Calculate DOCEs for the sound systems and return sound system}
$SystemOriginalDOCE = 0$
$SystemTyingDOCE = 0$
for all proto-sounds in $SoundSystemOriginal$ **do**
  $SystemOriginalDOCE = SystemOriginalDOCE + BestProtoSoundDOCE$
$SystemOriginalDOCE = SystemOriginalDOCE \div soundnumOriginal$

for all proto-sounds in $SoundSystemTying$ **do**
  $SystemOriginalDOCE = SystemOriginalDOCE + ProtoSoundDOCE$
$SystemOriginalDOCE = SystemOriginalDOCE \div soundnumTying$
return $SystemOriginal, SystemOriginalDOCE, SystemTying, SystemTyingDOCE$

3.4 Proto-Words

After the reconstruction of sounds is complete, there remains a final step: the reconstruction of the proto-words. This step, for both the human linguist and the CMA, is quite simple. The only difference between the work of the human linguist and that of the CMA, is that the CMA goes through this process several times: once for each sound system available, regardless of how many have been created (in this case, two). Thus, the multiple outputs are generated in a very simple way: each proto-word is reconstructed with as many options as there are sound systems, each time using the algorithm in Algorithm 4.

The algorithm itself is fairly simple. Each 'word' in this case is the multiple-aligned cognate set found in section 3 (as for example in Table 5). Thus, for each column in Table 5, the algorithm in Algorithm 4 would reconstruct either a proto-sound or a gap (which is one of the IPA characters available as a proto-sound), thus, in the end reconstructing the entire word.

Language1	b	i	z
Language2	p	e	[gap]
Language 3	p	i	ʃ

Table 5 A sample alighnment

In the first step, the algorithm looks at each column in order, determining the right proto-sound and DOCE for each column by the simple expedient of checking what proto-sound is associated with that sound correspondence in sound system (lines 9-10 of Algorithm 4). As each proto-sound is found, it is added to the $WordString$ that is the proto-word and it's DOCE is added to the $WordStringDOCE$, which will (once normalized at line 13) become the DOCE of the proto-word (the $WordSTring$). At Step 2, every such word ($WordString$) and associated DOCE ($WordStringDOCE$) is added to the ultimate proto-word list ($WordList$) and its to-be-normalized DOCE ($WordListDOCE$). Finally, Step 3 finds the average and thus overall DOCE of each word list ($WordListDOCE$) and returns the word list (complete with proto-words and DOCEs for individual words) as well as the overall DOCE for the word list. Notably, again the DOCE for the word lists include the DOCE for the words, which include the DOCE for the proto-sounds, which include the DOCE for the sound correspondences. Thus, the final DOCEs do in fact include all the relevant DOCE information from throughout sections 2 and 3. Since the algorithm is run as many times as there are sound systems

(in this case 2), two or more alternative word lists will be generated, thus also providing the solution to Pulgram's Dilemma.

Algorithm 4 Algorithm for Reconstructing Proto-Words Based on a Given Sound System

{STEP 0: Initialize the previously found sound system and the empty word list}
$SoundSystem = CurrentSoundSystem$
$WordList = NULL$
for all words **do**
{STEP 1: Figure out each word and its associated DOCE}
$soundnum = 0$
$WordString = NULL$
$WordStringDOCE = NULL$
$wordnum = wordnum + 1$
for all correspondence sets in the word **do**
$AssociatedProtoSound$ = the proto-sound associated with that correspondence set in the sound system
$AssociatedProtoSoundDOCE$ = the $BestProtoSoundDOCE$ of that proto-sound
$WordString = WordString + AssociatedProtoSound$
$WordStringDOCE = WordStringDOCE + AssociatedProtoSoundDOCE$
$soundnum = soundnum + 1$
$WordStringDOCE = WordStringDOCE \div soundnum$ {STEP 2: Add each word and it's DOCE to the wordlist}
WordList = WordList + WordString
WordListDOCE = WordListDOCE +WordStringDOCE
{STEP 3: Average the DOCE of the word list to get the word list DOCE}

$WordListDOCE = WordListDOCE \div wordnum$
return $WordList, WordListDOCE$

3.5 Summary of §3

In this section, the CMA algorithm for the final step of the comparative method was developed: Algorithm 5 summarizes the methods the CMA uses to reconstructproto-sounds (Step 1 of Algorithm 5 below), proto-sound-systems (Step 2 of Algorithm 5), and proto-words (Step 3 of Algorithm 5). While the simplest decision possible was made in all circumstances, the detailed illustration of the algorithm in this section showed that it is possible to create a reconstruction algorithm that meets the two main goals of the CMA: it effectively calculate useful DOCEs (i.e. ones that include all previous DOCEs) and provides multiple reconstructions for the proto-sound-system and proto-words. Furthermore, the algorithm models the human linguist more carefully than the single previous attempt (by Oakes (2000)) in that it includes more of the criteria the human linguist uses in picking a proto-sound and assigns several of the criteria separate weights, so that their different importance to the linguist can be modeled (discussed in section 3.2.2). It also uses a database-based scoring schema for measuring the 'naturality' of a sound change, thus allowing the CMA algorithm to also be reasonably pre-theoretic. Thus, the current reconstruction CMA algorithm, as summarily outlined below in Algorithm 5, accomplishes all the goals set in the introduction and meets both of the main heuristics.

4 Conclusion

Thus, using previous methods and recently developed databases, we are able to create a Comparative Method Algorithm that models the full extent of the comparative method, produces DOCEs for every step of the way, and yields multiple outputs for reconstruction helping resolve Pulgram's dilemma. Currently, there are still a plethora of ways this CMA could be improved. To take one example, phylogenetic methods could be incorporated into the CMA to serve at least two different purposes. First, they could help deal with borrowings, as mentioned in the introduction. More importantly, creating a family tree of the set of languages being compared would allow the reconstruction stage to become more precise. For the criteria of majority, instead of considering each proto-sound in the correspondence set as having equal weight, the proto-sounds could be grouped according to the sub-groupings established by the phylogenetic method and thus be weighted more accurately. In addition, many of the computational steps discussed only briefly here should be considered farther and potentially modified to best fit the CMA. Finally, there are myriad improvements that could be made to the very basic algorithm proposed in § 3 for the Reconstruction Step from finding sound systems in a

Algorithm 5 Summary of the Reconstruction Step of the CMA

```
This algorithm starts out with aligned words and sound correspondences with their DOCEs
 {STEP 1: Find the Best Proto-Sounds}
3: for all sound corerspondences do
    for all IPA characters do
      Calculate the DOCE score of each IPA character
6:  Pick the two+ best ones
 {STEP 2: Create two+ Sound Systems}
 add the proto-sounds together with their DOCE scores to the relevant sound systems
 Calculate the overall DOCE of the sound system
9: return the sound system to the user
 {STEP 3: Find the Proto-Words with several alternatives for different sound systems}
 for all Sound Systems created at Step 2 do
    for all Cognates with Multiple Allignment found in section ?? do
12:   for all correspondence columns in those cognates do
        Pick the appropriate proto-sound and add it, with it's DOCE, to the proto-word
      Calculate the overall DOCE of the Proto Word
15:   Add the Proto Word to the List of Proto Words
    Calculate the Overall DOCE of the Lists of Proto Words
    return to the user the List of Proto-Words with the overall and word-individual DOCEs
```

different way to integrate system balance and minimum-number-of-phonemes considerations into the discussion of reconstructions.

In addition to such improvements, the outputs of the CMA (DOCEs and multiple-option reconstructed forms) will be interesting to study in and of themselves. It would be interesting to determine if and to what extent the multiple outputs exhibit similar patterns to a regular dialectal spread. The DOCEs, meanwhile, would give some estimation of how effective the Comparative Method is in general. A further step may be to modify the algorithm slightly to allow for testing of different theories within the framework. Thus, instead of using the combinations of databases for the initial substitution matrices for sound similarity and likelihood of sound change, one could base the matrices on specific databases or even different phonological theories and run the algorithm with the same data but different theories. Then, the extent to

which each version produces accurate reconstructions and the average values of the resulting DOCEs could be used as an indication of how good each theory and/or database is, or at least how appropriate it is the to task of reconstruction. Thus, this first approximation at the structure of a CMA opens up a plethora of new possibilities.

References

Blevins, Juliette. 2008. Naturalness and iconicity in language. In Klaas Willems & Ludovic De Cuypere (eds.), *Natural and unnatural sound patterns: A pocket field guide*, 121–148. Amsterdam: John Benjamins.

Bouchard-Coté, A., P. Liang, T. L. Griffiths & D. Klein. 2007. A probabilistic approach to diachronic phonology. In *Proceedings of the 2007 joint conference on empirical methods in natural language processing and computational natural language learning*, 887–896.

Bouchard-Côté, Alexandre, Thomas Griffiths & Dan Klein. 2008. A probabilistic approach to language. In *Proceedings of nips*, .

Bouchard-Côté, Alexandre, Thomas L. Griffiths & Dan Klein. 2009. Improved reconstruction of protolanguage word forms. In *Proceedings of human language technologies: The 2009 annual conference of the north american chapter of the association for computational linguistics*, .

Brown, Cecil H., Eric W. Holman & Søren Wichmann. 2011. Sound correspondences in the world's languages.

Campbell, Lyle. 1998. *Historical linguistics*. Edinburgh University Press.

Crowley, Terry. 1992. *An introduction to historical linguistics*. Oxford University Press.

Delmestri, Antonella. 2011. *Data driven models for language evolution*: University of Trento dissertation.

Engstrand, Olle, Petur Helgasson & Mikael Parkvall. 2008. The beginnings of a database for historical sound change. In *Papers from the 21st swedish phonetics conference*, 101–104.

Fox, Anthony. 1995. *Linguistic reconstruction: An introduction to theory and method*. Oxford University Press.

Gilman, Sophia. 2012. Operationalizing the intuitive aspects of the comparative method. Ms. Yale University. https://docs.google.com/document/d/1XMmRcKe2ry6DLAJQrgVkUqkbHqKfdbm_IDSdQkP4-g0/edit.

Hamed, Ben Mahe & Sabastien Flavier. 2009. Unidia: A database for deriving diachronic universals. In Monique Dufresne, Fernande Dupuis & Etleva Vocaj (eds.), *Historical linguistics 2007: Selected papers from the 18th international conference on historical linguistics* Current Issues in Linguistic Theory, 259–268. Amsterdam: John Benjamins.

Hamed, Ben Mahe & Sabastien Flavier. 2011. The diadm project: A web-based platform for diachronic data and models. http://www.diadm.ish-lyon.cnrs.fr/.

Hauer, Bradley & Grzegorz Kondrak. 2011. Clustering semantically equivalent words into cognate sets in multilingual lists. In *The 5th international joint conference on natural language processing*, .

Kessler, Brett. 2005. Phonetic comparison algorithms. *Transactions of the Philological Society* 103(2). 243–260.

Lass, Roger. 1993. How real(ist) are reconstructions. In C. Jones (ed.), *Historical linguistics: Problems and perspectives*, 156–189. Longman.

List, Johann-Mattis. 2012a. Lexstat: Automatic detection of cognates in multilingual wordlists. In *Proceedings of eacl, joint workshop of lingvis and unclh*, 117–125.

List, Mattis. 2012b. Sca: Phonetic alignment based on sound classes. Forthcoming.

Oakes, Michael P. 2000. Computer estimation of vocabulary in a protolanguage from word lists in four daughter languages. *Journal of Quantitative Linguistics* 7(3).

Pulgram, E. 1959. Proto-indo-european reality and reconstruction. *Language* 35(3). 421–6.

Steiner, Lydia, Peter F. Stadler & Michael Cysouw. 2011. A pipeline for computational historical linguistics. *Language Dynamics and Change* 1(1). 89–127.

Trask, R.L. 1996. *Historical linguistics*. Oxford University Press 2nd edn.

Sophia Gilman
University of Cambridge
Department of Theoretical
& Applied Linguistics
Faculty of Modern and Medieval Languages
Sidgwick Avenue
Cambridge, CB3 9DA

sophia.gilman@gmail.com

Yale University
Department of Linguistics
370 Temple St
PO Box 208366
New Haven, CT 06520-8366