Mach Translat (2015) 29:163–187
DOI 10.1007/s10590-015-9172-5

Complexity of alignment and decoding problems: restrictions and approximations

Noah Fleming[1] · **Antonina Kolokolova**[1] · **Renesa Nizamee**[1]

Received: 15 September 2014 / Accepted: 12 September 2015 / Published online: 21 September 2015

Abstract We study the computational complexity of the Viterbi alignment and relaxed decoding problems for IBM model 3, focusing on the problem of finding a solution which has significant overlap with an optimal. That is, an approximate solution is considered good if it looks like some optimal solution with a few mistakes, where mistakes can be wrong values (such as a word aligned incorrectly or a wrong word in decoding), as well as insertions and deletions (spurious/missing words in decoding). In this setting, we show that it is computationally hard to find a solution which is correct on more than half (plus an inverse polynomial fraction) of the words. More precisely, if there is a polynomial-time algorithm computing an alignment for IBM model 3 which agrees with some Viterbi alignment on $l/2 + l^{\epsilon}$ words, where l is the length of the English sentence, or producing a decoding with $l/2 + l^{\epsilon}$ correct words, then P = NP. We also present a similar structure inapproximability result for phrase-based alignment. As these strong lower bounds are for the general definitions of the Viterbi alignment and decoding problems, we also consider, from a parameterized complexity perspective, which properties of the input make these problems intractable. As a first step in this direction, we show that Viterbi alignment has a fixed-parameter tractable algorithm with respect to limiting the range of words in the target sentence to which a source word can be aligned. We note that by comparison, limiting maximal fertility—even to three—does not affect NP-hardness of the result.

✉ Antonina Kolokolova
kol@mun.ca

Noah Fleming
nrf171@mun.ca

Renesa Nizamee
mrn271@mun.ca

[1] Memorial University of Newfoundland, St. John's, NL, Canada

Keywords Viterbi alignment · Decoding · IBM Model 3 · Phrase alignment · Approximation · Lower bounds · Edit distance

1 Introduction

Two computational problems of much importance to machine translation (MT) are alignment and decoding. The former can be loosely defined as follows: given a pair of sentences in different languages, find the best match between the words in the sentences. The latter, even more computationally involved, is the task of finding, given a sentence in one language, its best translation to another language. There are a number of variants of both problems, with different constraints and underlying models. Ubiquitous as these problems are, they quickly become provably computationally intractable as the conditions specifying good alignments and decodings become more realistic. Contrary to this, there are heuristics for these problems which perform surprisingly well in practice (Koehn 2004; Udupa and Maji 2005; Ravi and Knight 2010).

In this paper, we consider the complexity of finding a near-optimal solution to the alignment and decoding problems in two possible settings. In addition, we try to prove formally which restrictions on the inputs can bring down their computational complexity.

Following the conventions in statistical MT (SMT), we view the problem of translating a sentence from a source language (such as French) to a target language (such as English) as finding the most likely sequence of English words $e = e_1 \dots e_l$ corresponding to the given French sentence $f = f_1 \dots f_m$. That is, the goal is to find e such that $Pr(e|f)$ is maximized. This is called the decoding problem. Applying Bayes' rule, we may then express $Pr(e|f)$ as $Pr(e) \cdot Pr(f|e)$. These probabilities are considered separately for the decoding problem.

A classic sub-task of the decoding problem is to compute the best alignment between two given sentences f and e. For most of this paper (except Sect. 4), we will view an alignment as a matching of words in one sentence to words in the other, where each French word is aligned with at most one English word, and each English word may be aligned with any number of French words, or with nothing. Thus, an alignment is expressed as a sequence of $|f| = m$ values $a = a_1, \dots, a_m$, where f_j aligned with e_i is represented by $a_j = i$; French words with no match in the English sentence e are interpreted as aligning with a dedicated 'null' word e_0. In Fig. 1, both of the French words 'les' and 'cheveux' align to 'hair', while 'will' does not generate a single French word.

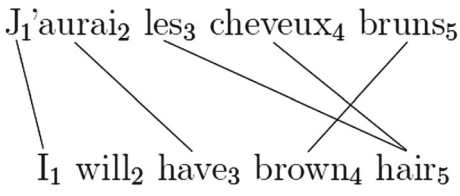

Fig. 1 Alignment between French and English sentences with $a_1 = 1, a_2 = 3, a_3 = 5, a_4 = 5, a_5 = 4$

The problem of finding an alignment with maximum $Pr(f|e)$ is the Viterbi alignment problem. The probability $Pr(f|e)$ is calculated as the sum the of probabilities of all possible alignments between e and f, $Pr(f|e) = \Sigma_a Pr(f, a|e)$.

Brown et al. (1993) developed five translational models, known as the IBM Models 1–5, which make up the most common approaches to calculating the *translation model* $Pr(f|e)$. Each of the five IBM models gives a way of computing $Pr(f|e)$ under increasingly complex assumptions. The computational complexity of these models also increases, with model 1 being the simplest. For models 1 and 2, Viterbi alignment can be calculated in polynomial time. However, decoding is hard for IBM Model 1 when considering a simple bigram model as the language model (Knight 1999). For Model 3 and above, Viterbi alignment, and thus decoding, are NP-hard (Udupa and Maji 2006). The most common approach for handling NP-hard problems is to search for approximate solutions. The complexity of computing an approximate solution for various approximation parameters and definitions of approximation varies widely among computational problems. For instance, problems such as the Traveling Salesperson problem with Euclidean distance, while still an NP-hard problem, allows for arbitrarily good approximations. In contrast, problems such as MaxClique (given a graph, find a complete subgraph of maximum size) is hard to approximate even to within a polynomial factor of the optimal value. To the best of our knowledge, the complexity of approximating a solution to the Viterbi alignment and decoding problems for models 3 and higher remains an open problem.

Usually, an approximation algorithm produces a solution with a value sufficiently close to the value of an optimal solution, e.g. an alignment with probability at least half that of the optimal. Unfortunately, an alignment which is close in value can be very different from an optimal alignment. Therefore, we look at the complexity of approximating a solution to the Viterbi alignment with respect to a somewhat different metric; rather than finding an alignment with the closest value to the optimal, we consider an "approximate" alignment to be one which is not too different structurally from an optimal alignment. Alternatively, this can be viewed as an optimal alignment with a fairly small number of mistakes. This is equivalent to the problem of recovering an optimal alignment given its noisy or corrupted representation, akin to the error-correction setting.

Another approach to MT known as "phrase-based alignment" (or "forced decoding") attempts to break both sentences into contiguous phrases, and then align these phrases. The intuition is that phrases, rather than aligning word-to-word as is done in with the IBM models, would provide a more realistic translation. DeNero and Klein (2008) demonstrate the hardness for phrase-based alignment, as well as discussing a number of heuristics for handling this problem. As heuristics give no guarantee of their solution being close to optimal, the complexity of approximating an optimal alignment to the phrase alignment problem remains an interesting open problem.

Though inapproximability and NP-hardness of a problem does imply that in the worst case, this problem will quickly become intractable as the input size grows, these worst case instances may not be that common in practical applications. Natural as it would be to analyse the complexity of these problems on average, such distributional analysis is notoriously difficult, as distributions occurring in practice are often not well defined, or too complex to obtain results.

Parameterized complexity is a more fine-grained approach to computational complexity which is aimed at handling such situations. Loosely speaking, parameterized complexity looks at a variety of properties of the problem to see which of them can and cannot be restricted (or occur restricted in the practical setting) to produce faster algorithms. More precisely, a problem is k-fixed-parameter tractable if there exists an algorithm with running time $g(k) \cdot n^c$; here, k can be a list of several parameters. Thus, if k is small in instances occurring in practice, this algorithm will perform efficiently.

An analysis of restrictions of alignment problems and the effect of such restrictions on computational complexity was done by Søgaard (2009). He examined the alignment problem for synchronous grammars, in particular inversion transduction grammars and two-variable binary bottom-up non-erasing range concatenation grammars. He shows that even though the universal recognition problem for these grammars is polynomial-time decidable, requiring alignments to satisfy additional conditions, in particular that of being one-to-one, makes the problem NP-hard. Thus, relaxing these conditions brings the complexity down.

1.1 Our results

We show that not only is computing the fertilities (the number of French words aligned to each English word) in Viterbi alignment from English sentence e to French sentence f with the parameters of IBM model 3 NP-hard, but computing this list of fertilities with fewer than $l/2-l^\epsilon$ mistakes is not possible in polynomial time unless P = NP. This result holds even when mistakes of insertion and deletion are allowed, and with bounds on L_1 and (for $\sqrt{l/2-l^\epsilon}$) L_2 norms of the vector of errors in fertilities. Moreover, this result holds even when the fertilities are restricted to have a value between 0 and 3.

A similar result holds for relaxed decoding (computing the sentence alignment pair (e, a) of maximum likelihood given f), where computing e with at most $l/2 - l^\epsilon$ mistakes is intractable unless P = NP, again allowing insertions, deletions and incorrect words as mistakes. The latter result holds even without considering the order of words in e, i.e. producing a list of words in e for the most probable (e, a) pair. Here, l is the length of the optimal e; as a function of $m = |f|$, the error bound is $m/4 - m^\epsilon$.

For phrase alignment, more specifically perfect (bijective) phrase alignment, an analogous hard problem is determining phrase boundaries (spans). There, we show the hardness of finding a solution with at most $l/2-l^\epsilon$ mistakes for the phrase-to-word alignment, and $2(l+m)/3 - (l+m)^\epsilon$ for the phrase-to-phrase setting.

Finally, we recast the known fact that restricting the distortion makes Viterbi alignment for model 3 tractable as a fixed parameter tractability result. That is, we show an algorithm with the running time of the form $g(k)n^c$ for model 3 Viterbi alignment with the "distortion" parameter k: that is, for any j, $f_j \in [\max\{0, l/m(j-k)\}, \min\{l/m(j+k), l\}]$ range ($k = 0$ is monotone alignment).

2 Viterbi alignment and relaxed decoding for IBM Model 3

Given a French sentence f, the goal of a translation process is to produce an English sentence e, such that $Pr(e|f) = Pr(f|e)Pr(e)/Pr(f)$ is maximized. $Pr(f)$ will be

constant and so can be omitted; it is enough to maximize $Pr(f|e) \cdot Pr(e)$. Here we call $Pr(e)$ the *language model* probability and $Pr(f|e)$ the *translation model* probability. We will follow the notation and definitions of Brown et al. (1993) and Udupa and Maji (2006) for the rest of this section.

Central to the models of Brown et al. is the notion of an *alignment* between the source English sentence $e = e_1 \dots e_l$ and the target French sentence $f = f_1 \dots f_m$. This is represented as $a = a_1, \dots a_m$, $a_j \in \{0, \dots, l\}$, where $a_j = i$ denotes that a French word f_j is aligned with an English word e_i. When $a_j = 0$, there is no English word corresponding to f_j, or, equivalently, it is considered to be aligned to the "null word" e_0. Note that this notion of alignment is a one-to-many relation, where an English word can align with several French words, but any French word must be aligned with at most one word in the English sentence.

We phrase our original model of translation in terms of alignments, $Pr(f|e) = \Sigma_a Pr(f, a|e)$. Here, the probabilities $Pr(f, a|e)$ are given by the translation model. Given the translation model, we define a *Viterbi alignment* to be the alignment a^* between given English and French sentences, e and f, for which the probability $Pr(f, a|e)$ is maximized; $a^* = \text{argmax}_a Pr(f, a|e)$.

Following Udupa and Maji (2006), we define *exact decoding* as the problem, given some French sentence f, of producing the English sentence e^* for which $Pr(f, a|e) \cdot Pr(e)$ is maximized. In the *relaxed decoding* problem, we look for the pair (e, a) of an English sentence and a corresponding alignment which maximizes $Pr(f, a|e)$. That is, $(e^*, a^*) = \text{argmax}_{(e,a)} Pr(f, a|e)$.

Solving $Pr(f, a|e)$ consists of deciding on the length m of f, choosing which French words can be a translation of each e_i, and finally specifying the order of these words in the French sentence. We will let $a_1^{j-1} = a_1, \dots, a_{j-1}$ and $f_1^{j-1} = f_1, \dots, f_{j-1}$ denote the choices of the English words aligned to the first $j-1$ positions of the French sentence, and the French words translated from them respectively. From this we can obtain (1):

$$Pr(f, a|e) = Pr(m|e) \Pi_{j=1}^{m} Pr(a_j|a_1^{j-1}, m, e) Pr(f_j|a_1^j, f_1^{j-1}, m, e). \quad (1)$$

The five IBM models defined in Brown et al. (1993) differ in the assumptions of the dependencies in these expressions. In particular, our focus will be Model 3, which is based on the following generative process:

1. Choose the number of words ϕ_i that each English word e_i aligns to (called the *fertility* of e_i). This value can be 0, and will depend only on the identity of e_i. This choice will be made according to the distribution $n(\phi_i|e_i)$, called the *fertility model*.
2. Choose the number of French words that are not aligned to any English word, $n(\phi_0|\Sigma_{i=1}^{l}\phi_i)$. This, along with the previous step, determines the length m of the French sentence.
3. For each English word, select a set of ϕ_i French words which will be the translation of that English word according to the *lexicon model* $t(f_j|e_{a_i})$. Model 3 assumes that this probability does not depend on the positions of English or French words in the sentences.

4. Determine where to place these generated French words, according to the *distortion model* $d(j|i, m, l)$, in order to obtain the resulting French sentence. Here, the probability that an English word at position i is translated into a French word at position j depends only on i and lengths of e and f.
5. Place French words corresponding to e_0 into the remaining positions uniformly at random.

With these definitions and assumptions, the Model 3 formula for $Pr(f, a|e)$ becomes (2):

$$\begin{aligned} &Pr(f, a|e) \\ &= n(\phi_0|\Sigma_{i=1}^{l}\phi_i) \cdot \Pi_{i=1}^{l} n(\phi_i|e_i)\phi_i! \cdot \Pi_{j=1}^{m} t(f_j|e_{a_j}) \cdot \Pi_{j:a_j=i>0} d(j|i, m, l) \end{aligned} \tag{2}$$

2.1 NP-hardness of Viterbi alignment for IBM Model 3

NP-hardness results on variants of decoding problems first appeared in Knight (1999), who showed that for a simple bigram language model, decoding is hard for IBM Model 1 even without considering fertilities.

Udupa and Maji (2006) present a thorough complexity analysis of a number of computational problems for IBM Model 3 when considering only the translation model (without appealing to any language model to prove hardness). In particular, they show NP-hardness of Viterbi alignment and relaxed decoding for Model 3 (and thus models 4 and 5) by a reduction from the SetCover problem.

For our results, it is more convenient to show NP-hardness of Viterbi alignment for model 3 by a reduction from a version of the Satisfiability problem. In general, *SAT* is defined as follows: given a propositional formula $F(x_1, \ldots, x_l)$, determine whether there exists an assignment of values true/false to $x_1, \ldots, x_l$ that causes F to evaluate to true (we will write "0" for false, and "1" for true). This problem remains NP-complete even when F is restricted to be a 3CNF formula: $F(x_1, \ldots, x_l) = \wedge_{i=1}^{m}(l_{i1} \vee l_{i2} \vee l_{i3})$, where each l_{ij} is either a variable or a negation of a variable. With the 3CNF restriction, the problem is known as 3*SAT*. Such triples $(l_{i1} \vee l_{i2} \vee l_{i3})$ are called clauses. Clauses with 2 or 1 literals are permitted as well.

Another NP-hard variant of this problem, 1in3SAT, asks whether there exists an assignment that not only makes F true, but also makes true exactly one l_{ij} for each i. The problem 1in3SAT remains NP-hard even if each variable x_i occurs only positively and at most 3 times.

With this, define *Monotone Cubic 1in3SAT* (MC1in3SAT) as follows: given a propositional formula in 3CNF $F(x_1, \ldots, x_l)$ where each variable occurs only positively and each variable has at most three occurences in the formula, determine whether there exists an assignment of values 0, 1 to $x_1, \ldots, x_l$ such that exactly one variable in each clause is set to 1.

Example 1 Formula (3):

$$(x_1 \vee x_2) \wedge (x_1 \vee x_3) \wedge (x_1 \vee x_2 \vee x_4) \tag{3}$$

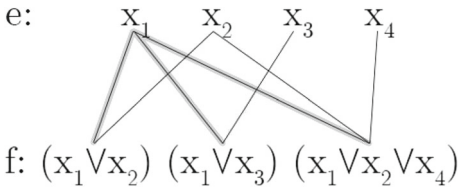

Fig. 2 An alignment problem instance obtained from formula (3) in Example 1, with non-zero probability pairs indicated by the *thin lines*, and an optimal alignment by *highlighted lines*. Note that alignment produced by $Viterbi3$ will have a higher score if it sets fewer variables to 1, favouring 1000 over 0110

is a satisfiable MC1inSAT formula, with a satisfying assignment setting $x_1 = 1$ and the rest to 0, or another $x_1 = 0$, $x_2 = 1$, $x_3 = 1$, $x_4 = 0$. These assignments can be represented by binary strings 1000 and 0110, respectively (cf. Fig. 2 for an example). However, formula (4):

$$(x_2 \vee x_3) \wedge (x_1 \vee x_2) \wedge (x_1 \vee x_3) \wedge (x_1 \vee x_2 \vee x_4) \tag{4}$$

is unsatisfiable, as the first three clauses cannot be satisfied by any combination of assignments to the variables x_1, x_2, x_3.

Let $Viterbi3Dec(n, t, d, f, e, p)$ be a decision version of finding a Viterbi alignment: rather than finding an alignment with maximal probability, we ask whether there exists an aligment with probability greater than p, for a given p. The results extend to optimization versions of both problems, where an optimization version of $MC1in3SAT$ asks for a correct assignment with a minimal number of non-zero variables.

Theorem 1 *$MC1in3SAT$ is polynomial-time reducible to $Viterbi3Dec$.*

Proof The reduction of $MC1in3SAT$ to $Viterbi3Dec$ proceeds as follows. Given an $MC1in3SAT$ formula, $F(x_1, \ldots, x_l)$ on l variables and m clauses,

1. Associate variables $x_1, \ldots, x_l$ with words $e_1, \ldots, e_l$, and clauses $c_j = (l_{j1} \vee l_{j2} \vee l_{j3})$ with $f_1, \ldots, f_m$.
2. Set $p = 0$.
3. For each i, let $num(x_i)$ be the number of occurrences of x_i in F. Set $d(j|i, m, l) = 1$ for any j, i, m, l, and the other model parameters as

$$n(\phi_i|e_i) = \begin{cases} 1/2\phi_i! & \text{if } \phi_i = num(x_i) \\ 1 & \text{if } \phi_i = 0 \\ 0 & \text{otherwise} \end{cases}$$

$$t(f_j|e_i) = \begin{cases} 1 & \text{if variable } x_i \text{ occurs in clause } c_j \\ 0 & \text{otherwise} \end{cases}$$

Suppose there is a $Viterbi3Dec$ alignment with a non-zero probability. Consider fertilities $\phi_1, \ldots, \phi_l$ of $e_1, \ldots, e_l$. Now, set $x_i = 1$ if $\phi_i > 0$, and $x_i = 0$ otherwise.

By definition of the fertility model, each x_i either satisfies none or all of the clauses in which it occurs; in the latter case, it will be aligned with all f_j corresponding to these clauses, guaranteeing that no other variable in these clauses is set to 1. Furthermore, each x_i is only aligned with clauses in which it occurs. As we did not include e_0, each f_j is aligned with some e_i. Therefore, fertilities of the Viterbi alignment translate into a correct assignment for the original $F(x_1, \ldots, x_l)$. For the other direction, a correct assignment to F gives an alignment with non-zero probability by construction. □

Note that in this reduction, just as in the reduction from SetCover of Udupa and Maji (2006), the only information that is needed from the Viterbi alignment is the list of fertilities. Thus, a 'hard part' of computing a Viterbi alignment is producing a list $(\phi_0, \ldots, \phi_l)$ of fertilities of $e_0, \ldots, e_l$ from the most probable alignment. We call this problem the fertility problem for model 3: *Fertility*3(f, e, n, t, d); the (optimization version of the) above reduction gives NP-hardness of this problem.

As in Udupa and Maji (2006), modifying $n(\phi|e)$ to assign 0 fertility to English words that do not align with anything, and noting that no word should appear more than once in an optimal solution, we obtain the following corollary for the relaxed decoding problem:

Corollary 1 *Relaxed decoding is NP-hard by a reduction from optimization version of MC*1*in*3*SAT.*

Proof Given an instance of *MC*1*in*3*SAT*, $F(x_1, \ldots, x_l)$ on l variables and m clauses, set the model parameters as we did in Theorem 1, with the exception that we set the fertility model to be:

$$n(\phi_i|e_i) = \begin{cases} 1/2\phi_i! & \text{if } \phi_i = num(x_i) \\ 0 & \text{otherwise.} \end{cases}$$

This assigns 0 fertility to English words which do not align to any French word, and therefore e can only contain words from the set $\{e_1, \ldots, e_l\}$, corresponding to the variables in F, as the fertilities of all other English words must be 0 by the way we have set the model parameters. Furthermore, no word will appear more than once in an optimal alignment, as replacing duplicates of a word by a single word, and connecting all of the alignments of the duplicates to the single word would increase the score of the English sentence e. Therefore, the proof proceeds as in Theorem 1. □

Note that in this reduction, as in the SetCover reduction in Udupa and Maji (2006), the order of words in e does not affect the score of a sentence, as the distortion model $d(j|i, m, l) = 1$ for all j, i, m, l, effectively saying that the distortion, or probability that a word at location i is aligned to a word at location j, is uniform over m and l.

3 Structure inapproximability

One natural measure of a quality of an alignment is the number of 'mistakes' of some form it has compared to the closest optimal alignment.

Consider a translation of "the weather is not so good" that happened to produce "Le temps ttt n'est pas si bon". As this French sentence has a typo ("ttt"), the probability of this translation could be quite low, for example with $t(f_3|e) = 0$. However, intuitively this is still a good translation, with just one mistakenly inserted low-probability word. This suggests that approximate decoding producing outputs that are 'fairly close' to a good translation might be more useful than just producing a decoding which has a relatively high (if suboptimal) probability. In this section we show that for some choice of parameters, finding a good approximation of this form, at least for Hamming distance and edit distance functions, is essentially as hard as solving the original problem.

3.1 Hamming distance approximation

Recall that Hamming distance $d_H(y, z)$ between two strings y, z of the same length is the number of locations on which they differ. For binary strings, Hamming distance is equivalent to the L_1 norm. A random n-bit string has expected Hamming distance of $n/2$ from any other length-n binary string; over an alphabet of k symbols, the expected agreement is n/k. For variants of the SAT problem, we can encode an assignment to the variables $x_1, \ldots, x_n$ as a binary string $s_1, \ldots, s_n$, where $s_i = 0$ if variable x_i is set to 0, and $s_i = 1$ otherwise.

Lemma 1 (Sheldon and Young 2013) *If a satisfying assignment for a SAT problem on l variables can be approximated to within Hamming distance $l/2 - l^\epsilon$ for some $\epsilon > 0$, then P = NP.*

Proof First, note that it is enough to have an algorithm determining the value of one variable; the formula is then simplified and the process is repeated until the whole assignment is revealed. The proof proceeds by amplifying an arbitrary variable x_i $l^{1/\epsilon}$ times, i.e. introducing $l^{1/\epsilon}$ new variables z_k and adding clauses stating that they are equivalent to x_i. Now, if there is a polynomial-time algorithm that is guaranteed to return a witness within $l/2-l^\epsilon$ Hamming distance of a satisfying assignment, then such a string will be correct on the majority of copies of x_i. Taking the majority thus gives the correct value of this variable, and repeating the process l times, substituting computed values on each iteration, results in a satisfying assignment. If there is no satisfying assignment, then this can be verified by checking that the computed values for $x_1 \ldots x_l$ do not satisfy the formula. The resulting algorithm for SAT runs in time $l + l^{1/\epsilon}$ times the running time of the assumed polynomial-time approximation algorithm, which is polynomial when ϵ is constant. □

Corollary 2 *MC1in3SAT cannot be approximated to within Hamming distance $n/2 - n^\epsilon$ for any constant $\epsilon > 0$ unless $P = NP$.*

Proof The non-monotone case follows directly from Lemma 1, by noting that the condition that x_i is equivalent to all z_k can be expressed as $(x_i \rightarrow z_1) \wedge (z_1 \rightarrow z_2) \wedge \cdots \wedge (z_{n^{1/\epsilon}-1} \rightarrow z_{n^{1/\epsilon}})$. This only uses two occurrences of each z_k, and each clause $(\neg z_k \vee z_{k+1})$ can be satisfied by exactly one literal. As this construction needs

an extra copy of x_i, one copy of x_i in the original part of the formula can be replaced with z_2, restoring the property that there are at most three occurrences per variable.

Finally, to deal with monotonicity, use an even number of z_k variables and change the equivalence clauses to $(z_k \vee z_{k+1})$. As exactly one of these variables can satisfy each clause, in a correct solution all odd-numbered z_k will have values opposite to that of x_i, and all even-numbered z_k equivalent to x_i. Now, as before, taking the majority of values of z_{2k+1} and negations of z_{2k} produces the correct value of x_i. □

This, together with the reduction from *MC*1*in*3*SAT* to *Viterbi*3, and thus the problem of finding the fertility values in Viterbi alignment, gives the following result. Here, we encode the fertilities of the English words $\phi_1, \ldots, \phi_l$ as a string $s = s_1, \ldots, s_l$, where $s_i = \phi_i$.

Theorem 2 *If there exists a polynomial-time algorithm $A(f, e, n, t, d)$ that correctly computes at least $l/2 + l^\epsilon$ fertility values in a Viterbi alignment, for some constant $\epsilon > 0$, then $P = NP$.*

Proof Apply the above reduction from *MC*1*in*3*SAT* to *Viterbi3Dec* to the formula from Corollary 2. This formula is initially on N variables with added $N^{1/\epsilon}$ copies of x_i, for the total of $l = N + N^{1/\epsilon}$. The fertility of each of e_{N+k} corresponding to a z_k for $k > 1$ is either 2 or 0. Now, suppose that an algorithm $A(f, e, n, t, d)$ computes the array of fertilities with at most $l/2 - l^\epsilon$ mistakes. As there are at least $(N + N^{1/\epsilon})/2 + (N + N^{1/\epsilon})^\epsilon \geq (N + N^{1/\epsilon})/2 + N$ correct fertilities, even if all mistakes are among fertilities for words corresponding to z_k, more than half of their values must be correct, allowing us to recover the value of x_i. □

Note that this result extends to the L_1-norm of the difference vector between correct and approximate lists of fertility values. If there are k wrong values in the array of fertilities, then, since fertilities are non-negative integers, the L_1 norm of the difference vector is at least k. Thus, the L_1-norm of differences bounded by $l/2 - l^\epsilon$ gives at least $l/2 + l^\epsilon$ correctly computed fertility values. The same reasoning applies to the L_2 norm, as again the worst-case number of mistakes given a value of the norm is when each incorrect position contributes 1 to the norm. However, in this case the bound on the L_2 norm has to be $\sqrt{l/2 - l^\epsilon}$.

As fertilities can be recovered from Viterbi alignment, computing a sequence of m numbers which agrees with an optimal alignment in more than $l/2 + l^\epsilon$ places is already NP-hard. More precisely, if $a = a_1 \ldots a_m$ is an optimal alignment, and there is a way to compute $b = b_1 \ldots b_m$ which differs from a on at most $l/2 - l^\epsilon$ positions, then at most $l/2 - l^\epsilon$ e_i's can have incorrect fertility in b. Therefore, the value of x_1 can be recovered by looking at appropriate fertilities, as before.

3.2 Edit distance approximation

Let edit distance $d_E(y, z)$ be the number of insertions, deletions and replacements of symbols needed to convert y into z. Edit distance, even though in some respects related to Hamming distance, nevertheless has a very different behaviour. For example, a string 01010101 and a string 10101010 have the maximal Hamming distance of $n = 8$, yet

their edit distance is just 2, corresponding to deleting a 0 in front and inserting it in the back of the string. For Hamming distance, a random string is expected to be within $n/2$ from any string, but it is not clear what the expected edit distance is between two random strings. If two strings are far in the edit distance though, they are far in the Hamming distance. Therefore, lower bounds on edit distance approximability imply lower bounds for the Hamming distance, but the reverse is not immediate.

However, in the case where one of the strings is a string of all 0s or all 1s then the two notions coincide, as long as the length of the approximating string is the same. Indeed, even edit distance with transpositions to a string of all 1s from any given string is equivalent to Hamming distance. In addition, an edit distance between a string y of l 1s (or 0s) and an arbitrary string z of length m is bounded by the length difference plus number of 0s in z if $m < n$, and by maximum of length difference and number of 0s in z if $m > n$.

As edit distance is defined for strings, an encoding of a solution to a problem has to be specified more carefully than the 'number of mistakes' framework we used for Hamming distance. We try to use natural witnesses and encodings of solutions, for example, a satisfying assignment to a formula represented as a binary string of 0/1 values of $x_1, \ldots, x_l$, or a characteristic string of the cover for MinSetCover problem.

Lemma 2 *If there is a polynomial-time algorithm that, for some constant $\epsilon > 0$, can approximate a satisfying assignment to SAT instance on l variables to within edit distance $l/2 - l^{\epsilon}$, then P = NP.*

Proof Consider an "amplified" instance of SAT from the proof of Hamming distance inapproximability of SAT (Lemma 1). Assume, without loss of generality, that the variable x_1 is being amplified. That is, the instance contains $l^{1/\epsilon}$ new variables equivalent to x_1, and clauses enforcing this equivalence.

A string encoding a satisfying assignment to such an instance consists of either $l^{1/\epsilon} + 1$ 0s or $l^{1/\epsilon} + 1$ ones, together with $l - 1$ arbitrary bits. Moreover, we can assume that all values of the copies of x_1 are together, for example as the first $l^{1/\epsilon}$ positions in the assignment. Now, suppose there is an algorithm that approximates the satisfying assignment above, with $l^{1/\epsilon}$ copies of x_1, to within edit distance $N/2 - N^{\epsilon}$, where $N = l + l^{1/\epsilon}$. Let y' be a string returned by the approximation algorithm and y the corresponding optimal solution. Consider only the first $l^{1/\epsilon}$ positions in y', those corresponding to the copies of x_1. Without loss of generality, assume that $x_1 = 1$ in y. These positions can be changed to 0 (to obtain y') by either a replacement or an insertion/deletion pair moving values of the remaining $l - 1$ variables into the first $l^{1/\epsilon}$ positions. However, as discussed above, in this case the number of insert/delete pairs is at least as large as the number of replacements. Therefore, the same argument as for the Hamming distance applies, and bounding the edit distance between y and y' by $N - N^{\epsilon}$ means that the majority of the copies of x_1 in y' have a correct value.□

Note also that this argument works even if transposition operations are allowed.

Corollary 3 *If there is a polynomial-time algorithm that, for some constant $\epsilon > 0$, can approximate a satisfying assignment to MC1in3SAT instance with l variables to within edit distance $l/2 - l^{\epsilon}$, then P = NP.*

Proof Consider an amplified instance of *MC*1*in*3*SAT*, with variable x_1 being amplified. That is, there are new variables $z_1, \ldots, z_{l^{1/\epsilon}}$ and clauses $(x_1 \vee z_1)$, $(z_1 \vee z_2) \ldots (z_{l^{1/\epsilon}-1} \vee z_{l^{1/\epsilon}})$. Rather than viewing the representation of a satisfying assignment as assignments to $z_1 \ldots z_{l^{1/\epsilon}}$ followed by assignments to $x_1 \ldots x_l$, we write all variables that would obtain the same value as contiguous substrings. For example, rename the variables so that the string encoding the assignment starts with all z_{2k}, followed by $x_1 \ldots x_l$, followed by all z_{2k-1}, $1 \leq k \leq 1/2(l^{1/\epsilon})$ (assuming, without loss of generality, that $l^{1/\epsilon}$ is even).

If an optimal assignment sets $x_1 = 1$, it also sets all even-numbered $z_k = 1$ and all odd-numbered $z_k = 0$. Therefore, a correct assignment encoding will have $1/2l^{1/\epsilon}+1$ 1s at the start, followed by an arbitrary sequence of $l - 1$ bits, followed by $1/2l^{1/\epsilon}$ 0s. Now, there are two long monotone strings to which the argument before can be applied. Note that any t operations of insertion, deletion and replacement that result in a string of the same length as before would corrupt at most t locations in the monotone blocks in the front and in the back. Now, as before, we are guaranteed that there are $1/2(l+l^{1/\epsilon})+(l+l^{1/\epsilon})^{\epsilon} > 1/2(l+l^{1/\epsilon})+l$ correct locations, so even if all $x_1 \ldots x_l$ fall on correct locations, that leaves $> 1/2(l+l^{1/\epsilon})$ correct z_k's, from which the value of x_1 can be recovered. □

From this, using the same reasoning as for Hamming distance approximation, but with the order of the variables in the list of fertilities as in the witness for *MC*1*in*3*SAT* above (even copies, then original variables, then odd copies), we obtain a corollary that approximating a string encoding fertilities up to edit distance $l/2 + l^{\epsilon}$ is not possible unless $P = NP$. One may object to treating a sequence of numbers as a string over a finite alphabet, yet in our reductions all fertilities are bounded by 3.

Although edit distance does not appear to be the most natural metric when considering two strings of fertilities, it is far more natural when considering the decoding problem, where we measure the distance between two sentences. Our edit distance inapproximability results for decoding follow from those for alignment, and thus we begin by considering edit distance for alignment.

Corollary 4 *An optimal array of fertilities, as well as an optimal solution* $a_1 \ldots a_m$ *to* $Viterbi3$ *is not approximable to within an edit distance of* $l/2 - l^{\epsilon}$ *for any* ϵ *unless* $P = NP$.

Proof The argument above addresses the proof for fertilities. For the alignment a, the proof immediately follows from the Hamming distance inapproximability of Viterbi3, as insertions and deletions change at most one fertility each. Furthermore, allowing corrupting a by incrementing or decrementing all a_j starting with a given j as a form of spurious insertion or deletion in e, can be handled by the same argument as for the array of fertilities. □

The notion of edit distance approximation is more appropriate for decoding than for alignment. There, an approximate e can have a different length than an optimal e, in addition to having different words (replacement) and word order (insertions/deletions).

Theorem 3 *If there is a polynomial-time algorithm that can produce a sentence e' within edit distance $L/2 - L^{\epsilon}$ of some optimal sentence e of length L, or within $m/4 - m^{\epsilon}$ for $|f| = m$, then $P = NP$.*

Proof Suppose that there is a polynomial-time algorithm which produces a string $e' = e'_1 \dots e'_{L'}$ which is within edit distance $L/2 - L^{1/\epsilon}$ from some optimal $e = e_1 \dots e_L$.

Consider an amplified instance of $MC1in3SAT$ on l original variables reduced to the relaxed decoding problem as described above, except rather than adding $l^{1/\epsilon}$ new variables and clauses, add $2l^{1/\epsilon}$ of each. Then, the corresponding sentence $f_1 \dots f_m$ will have $m \leq 3l + 2l^{1/\epsilon}$ words.

As words with fertility 0 do not become part of e, the correct e has $l^{1/\epsilon}$ words corresponding to either z_{2k} or z_{2k+1} variables, $k \leq l^{1/\epsilon}$, as well as at most l words corresponding to $x_1, \dots, x_l$. To recover the value of x_1, we check whether the majority of z_k in e have even or odd values of k.

Our reduction, similarly to Udupa and Maji (2006), is designed to disregard the order of words. This is because we have set the distortion model, which encodes the probability of a word at location i aligning to a word at location j, to be uniformly $1, d(j|i, m, l) = 1$. That is, all sentences with a given set of words $\{e_1, \dots, e_s\}$, have the same score. Moreover, as the probability of a sentence increases when decreasing the number of words in a valid translation, no word in e repeats. Therefore, e can be viewed as a set of words.

Thus, only changing, adding or removing words corresponding to z_ks would affect recovering the value of x_1. The smallest number of operations needed to make the majority of these indicate the opposite value of x_1 is $1/2 \cdot l^{1/\epsilon}$ replacements. Thus, $L/2 - L^{\epsilon}$ bound works for the same reason as before.

It is more natural to express this inapproximability in terms of m rather than L, as m is the function of the input. However, as the number of mistakes tolerated in e is $1/2l^{1/\epsilon}$, and f contains more than $2l^{1/\epsilon}$ words, the edit distance for e' from which x_1 can still be recovered is $m/4 - m^{\epsilon}$. □

3.3 Previous work on structure approximation

Motivated by cognitive psychology applications, Hamilton et al. (2007) presented a variant of approximation which they called a *structure approximation*. This framework extends the classical notion of approximation (finding a solution which is close in value to an optimal solution) to the problem of finding a solution which is close according to a specified metric. More precisely, the description of a problem includes a distance function $d(y, z)$ which may depend on the input to the problem. An approximate solution y is considered good if $d(y, z)$ is sufficiently small for some optimal solution z. This can be seen as a generalization of the standard notion of approximation, as the distance $d(y, z)$ can be defined as a logarithm of the ratio of values of solutions y and z. In a follow-up paper (Rooij and Wareham 2012), this approach was applied to other problems such as the coherence model of belief fixation in cognitive science.

Hamilton et al. (2007) present a number of lower bound results for arbitrary distance functions, among which the most prominent is the Hamming distance. This is a very natural metric for comparing how close two solutions encoded as binary

strings are. For example, in the Hamming approximation for Max3SAT, a solution would be considered close to an optimal solution if it differs from it in few variable assignments, even if these variable assignments dramatically decrease the number of satisfied clauses.

Several other papers include results that can be interpreted as lower bounds for structure approximability with respect to Hamming distance. The reconstruction of a partially specified NP witness, considered in Gal et al. (1999), is probably the first result along these lines. There, they show that it is possible to reconstruct a satisfying assignment to a formula from $N^{1/2+\epsilon}$ bits of a satisfying assignment of a related (though larger) formula. Their proofs rely on erasure codes, so ϵ is a fixed parameter.

Kumar and Sivakumar (1999) showed that for any NP problem there is a verifier with respect to which all solutions are Hamming-far from each other: if one makes the witnesses to be encodings of natural witnesses to the original problem by some error-correcting code, the verifier decodes the witness and then checks it using the original verifier. Then, list-decoding allows one to find a correct codeword for the witness from a string which is within $n/2 + n^{4/5+\gamma}$ Hamming distance from it. Following this, Feige et al. (2000) show that some natural verifiers (e.g. binary strings directly encoding satisfying assignments for variants of SAT) are often hard to approximate to within Hamming distance $n/2 - n^{\epsilon}$ for some ϵ dependent on the underlying error-correcting code. Guruswami and Rudra (2008) improve this ϵ to $2/3 + \gamma$, but on the negative side argue that methods based on error-correcting codes can only give bounds up to $n/2 - O(\sqrt{n \log n})$.

The recent paper of Sheldon and Young (2013) settles much of the Hamming distance approximation question, providing the lower bounds of $n/2 - n^{\epsilon}$ for any ϵ for many of the problems considered in Feige et al. (2000), as well as upper bounds of $n/2$ for several natural problems including Weighted Vertex Cover, and a surprising $n/2 + O(\sqrt{n \log n})$ lower bound for the universal NP-complete language. The latter result they extend to existence of such very hard to approximate verifiers for all paddable (in the sense of Berman and Hartmanis 1977) NP languages, improving on Kumar and Sivakumar (1999). Their proof techniques avoid error-correcting codes altogether, instead combining amplification with search-to-decision (Turing) reductions and downward self-reducibility.

In this paper we considered Hamming distance and edit distance as our measures of closeness. Exploring the setting of structure approximation further, it would be interesting to see whether there is a generic way of building a lattice of hardness implications for various metrics. We conjecture, in particular, that any metric with a certain "locality property" (that is, one "unit of change" only affects a small, though not necessarily constant number of positions) should be inapproximable by generalizing Hamming distance results. Alternatively, one wonders whether there is a non-trivial, practically interesting metric for which there is indeed a fast approximation algorithm for any NP-hard problem. In that respect, considering various metrics and their interrelation with respect to computational problems is a promising area with a possibility for new approaches to computational problems from a wide variety of fields.

4 Phrase-based alignment

Phrase-based alignment is another approach used in SMT systems (Koehn and Marcu 2003; DeNero 2010). Whereas in IBM model 3 a list of French words aligned with a given English word does not have to be contiguous, but each French word is aligned with only one English word, now both English and French sentences are split into non-empty contiguous phrases, which are then aligned bijectively; each English phrase is aligned with one, and only one, French phrase. This is also known as "forced decoding" for phrase-based translation: that is, given a pair (e, f) of English and French sentences, the goal is to find the highest-scoring derivation of a phrase-based translation model. Here, we study the complexity of this problem, with the goal of identifying the hardness of parts of this task in the context of structure approximation. For simplicity, our focus will be on the phrase-based alignment rather than on decoding.

Heuristics have been a popular approach for phrase alignment, used both as a direct application of a heuristic and in the context of modelling a problem in a Integer Linear Programming framework, and then invoking heuristics-based solvers for ILP. In particular, hill climbing has been used in Marcu and Wong (2002), Och and Ney (2003), Birch et al. (2006) and simulated annealing in MacCartney et al. (2008) to solve the problem of partitioning strings into phrases. However, although useful in practice, such heuristic algorithms give no guarantee of the closeness to optimality.

4.1 Weighted sentence alignment

Following DeNero and Klein (2008), we formally define a *weighted sentence alignment (WSA)* problem as follows. Let e and f be sentences. The phrases in e are represented by a sequences of words between positions i to j in e called spans, denoted by $[i : j)$; f is represented by $\{[k : l)\}$ in the same fashion. A link is an aligned pair of phrases $([i : j), [k : l))$. An alignment is a set of links such that every word (token), in either sentence, occurs in exactly one link (here, we treat each occurrence of a word as a separate word). A weight function $\phi : \{([i : j), [k : l))\} \to \mathbb{R}$ assigns a weight to each link. A total weight of an alignment a, denoted $\phi(a)$, is a product of weights of its links. Now, an optimization version of the weighted sentence alignment problem asks, given (e, f, ϕ), to find the alignment with the maximum weight. A decision version of this problem can be stated as finding an alignment a of weight $\phi(a) \geq 1$.

In a more general statement of the problem, in particular in the natural language inference setting (MacCartney et al. 2008), the original sentence (text) can contain much more information than the resulting sentence. However, it can be reduced to the bijective case by padding the target sentence with null words (half the number of words of the original sentence suffices), and setting the weight of links between any phrase over the null words and any phrase of the original sentence to be 1, and weight of any link with a phrase involving both null and non-null words to be 0.

We will also consider a restricted version of this problem where only the French sentence is split into phrases, each aligned with exactly one English word. Even though it is not as relevant from a practical point of view, it simplifies hardness proofs.

Theorem 4 (DeNero and Klein 2008) *The decision version of the WSA problem is NP-complete.*

Proof DeNero and Klein (2008) show NP-hardness of WSA by the following reduction from 3SAT. Let F be a formula with n variables and m clauses. The construction will produce an instance I of WSA consisting of sentences e and f, and a function ϕ such that there is an alignment of weight (at least) 1 in I if and only if F is satisfiable.

Let sentence e consist of blocks of words as follows, with one word for each occurrence of a literal: $x_i^1 \ldots x_1^{p_i} \bar{x}_i^1 \ldots \bar{x}_i^{q_i}$, where p_i and q_i are the number of positive and negative occurrences of x_i in F, respectively. Thus, the length of e will be $\leq 3m$, with equality if every clause in F contains exactly 3 literals. Now, the sentence f will contain two types of words. The first m words, $c_1 \ldots c_m$, will correspond to the clauses of F. They will be followed by 'slack words' $s_1 \ldots s_n$, one for each variable in F. Finally, the function ϕ will only have values 0 and 1, and it will have the value 1 in two cases: (i) if the link is of the form (c_i, l_k), where literal l_k occurs positively in clause c_i (for all occurrences of l_k). This will be used to align each clause with a literal that makes it true; (ii) each slack variable s_i corresponding to a variable i will be aligned with all possible substrings of $x_i^1 \ldots x_1^{p_i} \bar{x}_i^1 \ldots \bar{x}_i^{q_i}$ in which either all positive or all negative copies of the variable (or both) are present. For example, if there is one positive occurrence of x_i and two negative occurrences of x_i, then the links with $\phi([i:j),[k,l)) = 1$ have $f_{[k:l)} = s_i$ and $e_{[i:j)}$ either $x_i \bar{x}_i \bar{x}_i$, or $\bar{x}_i \bar{x}_i$, or $x_i \bar{x}_i$, or x_i. The first one covers both positive and negative, the second covers all negative, and the last two all positive occurrences of the literal. These slack variables are needed to ensure that either only positive or only negative literals are left unmatched to be aligned with clause words.

To see that this reduction works, note that a satisfying assignment becomes an alignment in which every clause word is matched with one literal that makes it true (starting from the front of the block for positive and end of the block for negative), and slack variables cover the literals that remain unmatched to clauses. For the other direction, note that there is exactly one link for each slack variable: if it is matched with a block that contains all positive occurrences of the corresponding variable in F, the corresponding variable can be set to false, otherwise it can be set to true (if it is matched with the block containing all occurrences, then either assignment works).

Assuming that F has exactly 3 variables per clause, $|e| = 3m$, $|f| = m + n$, and $|\phi| \leq (3m)^2(m+n)^2$, and so the resulting instance is polynomial size, and the reduction runs in polynomial time.

Therefore, WSA is NP-hard. As an alignment can be checked for validity (by asserting that each word appears exactly once) and the weight of the alignment can be computed in polynomial time, the decision version of WSA is NP-complete. □

Alternatively, NP-hardness of WSA can be shown by a reduction from the Vertex-Cover problem. There, we are given an undirected graph $G = (V, E)$ with n vertices and m edges, and asked whether there exists a subset of k vertices called a cover such that every edge has as its endpoint at least one vertex in the cover. In an optimization version, a minimal-size cover is sought.

To show *VertexCover* $\leq_p$ *WSA*, construct the instance as follows. The words of e will be blocks of copies of each vertex v_i, where the length of each such block is the

degree of v_i, denoted $deg(v_i)$, plus 1, so $|e| = 2m+n$. The words of f will be of three types. The first m words $c_1 \ldots c_m$ will correspond to edges of G; the next n words are the 'slack variables' $s_1 \ldots s_n$ covering leftover copies of vertices, with one extra copy always covered by s_i, and the final $n-k$ words $t_1 \ldots t_{n-k}$ in f will ensure that the size of the cover is at most k. Thus, $|f| = m+n+(n-k) = m+2n-k$. With this intuition, define ϕ so that $\phi(v_{i,j}, c_l) = 1$ if edge c_l has v_i as its endpoint (for each copy $v_{i,j}$ of v_i), then $\phi(v_{i,j} \ldots v_{i,\deg(v_i)+1}, s_i) = 1$ for each i and all j, $1 \leq j \leq deg(v_i)$. Finally, each t_l can cover the full block for every vertex (except for the last copy), so $\phi(v_{i,1} \ldots v_{i,deg(v_i)}, t_l) = 1$ for every t_l and every v_i.

If there is a vertex cover of size k in G, then an alignment in the constructed instance will link all vertices other than the k vertices in the cover with t-variables. It will link each edge with a copy of a vertex in the cover (in order starting from $v_{i,1}$), and variables s_i will be linked with a block of remaining copies of the corresponding vertices (consisting of at least one special copy, more if some edges have both endpoints in the cover). For the other direction, variables t_l denote vertices not in the cover, so the cover consists of the remaining vertices. If there is a cover of size smaller than k, then some s_i variables align with the whole block corresponding to such extra v_i, which is allowed by our definition of ϕ.

Now, note that the reduction above proves NP-hardness for a special case of the problem, namely the case where all phrases in f are single words, and ϕ takes 0 or 1 values. As we considered computing an array of fertilities in Viterbi alignment as a hard subproblem, here we will consider a subproblem of computing phrase boundaries in e for the optimal alignment with words in f.

Definition 1 (*PWSA*) The PWSA (for "phrase-to-words sentence alignment") problem is defined as follows. Given as input (e, f, ϕ) where $\phi : \{([i : j), [k : l))\} \rightarrow \{0, 1\}$, find a partition of e into phrases such that there is an alignment of weight 1 of phrases in this partition with words of f.

A solution w for PWSA will be a binary string $w_1 \ldots w_{|e|-1}$ such that if $[i : j)$ is a phrase in the optimal alignment, then $w_i = w_j = 1$, or $w_j = 1$ and $i = 0$, or $w_i = 1$ and $j = |e|$; and $\forall k, i < k < j$, $w_k = 0$. Note that w has to have $|f| - 1$ 1s for any valid alignment.

Given phrase boundaries, an optimal alignment can be found by the Hungarian algorithm (Kuhn 1955) in polynomial time, and from that a satisfying assignment to F can be recovered.

Alternatively, in an instance of PWSA the values of variables with more than two positive and two negative occurrences can be determined directly from w. Suppose a slack variable covers all positive occurrences of a variable v, and leaves out some negative occurrences. Then, there will be no splitting points within the block denoting the positive literals, but there will be as many splitting points for the negative literals as there are clauses which use them. From that, already, it can be inferred that the negative occurrences were used to satisfy the clauses, so the variable needs to be set to false. Thus, if a substring w_{ij} of w corresponding to a block of encoding a literal v (without the endpoints) is of the form $1111 \ldots 0000$, then we can immediately infer that $v = true$; otherwise if it is of the form $000 \ldots 1111$, $v = false$. It would not work if there is exactly one positive or negative occurrence of a variable, but this can

be resolved by modifying the reduction so that there is always an extra "$v_i \bar{v}_i$" (or a single dummy variable) in the middle of each block, and $\phi(x \ldots x) = \phi(\bar{x} \ldots \bar{x}) = 0$. Then, the partition of e uniquely specifies the optimal alignment.

4.2 Structure approximation for vertex cover

Recall that in the MinVertexCover problem the goal is to determine a minimal set of vertices such that every edge has at least one endpoint in the cover; the decision version VertexCover asks to determine whether there is a cover of size at most k. A natural witness to VertexCover is a binary string of length $n = |V|$, where a bit corresponding to a vertex is 1 iff that vertex is in the cover. In the Sheldon and Young (2013) proof of Hamming distance inapproximability of this problem, in an input graph a copy of an arbitrary vertex v is made and an even-length path on $\geq 2n^{1/\epsilon}$ vertices is added between v and its copy v'.

Now, as a (minimal) vertex cover of an even-length path consists of either all even or all odd vertices, we say that the original v is in the $k + n^{1/\epsilon}$ cover if all even vertices are in that cover, otherwise v is not in the cover. Then the argument proceeds by showing that the majority of the vertices on the path will be correctly placed by the same calculation as for SAT above.

Theorem 5 *Unless P = NP, no polynomial-time algorithm can approximate the natural witness to VertexCover within edit distance $n/2 - n^{\epsilon}$, for any constant $\epsilon > 0$.*

Proof Consider the Sheldon and Young (2013) construction described above, but with a different naming convention for the variables in the witness. Let variables $v_1 \ldots v_n$ be the original variables, v' a copy of a selected variable (e.g. of v_1), $u_1 \ldots u_{n^{1/\epsilon}}$ be even variables on the path from v to v' and $w_1 \ldots w_{n^{1/\epsilon}}$ be the odd variables on that pass. Now, in the witness the first $n^{1/\epsilon}$ positions will correspond to the u_i variables, followed by v_is, in turn followed by the w_is.

Now, the same kind of argument as before applies. The witness—a characteristic string of a vertex cover of size $K = k + n^{1/\epsilon}$—will be encoded by either a string of $n^{1/\epsilon}$ 0s followed by some string of length $n+1$ followed by $n^{1/\epsilon}$ 1s, or a similar string with 0s at the beginning and 1s at the end. Now, similar to the construction for the SAT problem in Lemma 2, we would like to argue that a sequence of $N/2 - N^{\epsilon}$ of arbitrary edit operations (insertions, deletions, replacements) would not result in any string that differs from the original on the u-part and w-part in more than $N/2 - N^{\epsilon}$ positions.

Consider a pair of insert/delete operations applied to the above string encoding a K-cover. Suppose (without loss of generality) that the correct string starts with 1s and ends with 0s. Consider deleting a value from the u part of the string and inserting it into the w part. Now, the middle part of the string (corresponding to the v variables) could become maximally far from the encoding of the K-vertex cover at that point, i.e. if it was of the form 01010101; however, to determine whether v is in the cover, only variables u_i's and w_j's are relevant. A pair of insert-delete operations then introduces at most one 0 into the u part (by shifting the v part into it), and at most one 1 into the w part by insertion. Therefore, the 'damage done' to these parts of the string is no more

than from performing two replacements, and the argument still applies to an already corrupted string.

Therefore, if there exists a structure approximation algorithm for vertex cover that can consistently return a string within edit distance $n/2 - n^\epsilon$ from an optimal cover, then this algorithm can be used to determine exactly whether any given variable is in the intended cover. By Turing/search-to-decision reduction, the actual cover can be computed. In this reduction, if a vertex was determined to be in the cover, then recurse on a graph without this vertex, and otherwise recurse on a graph without this vertex and all of its neighbours. □

4.3 Hamming distance and edit distance inapproximability of PWSA and WSA

In this section we will show that PWSA cannot be Hamming or edit distance structure approximated to within $n/2-n^\epsilon$, with respect to the witness defined above. From this, the structure inapproximability of WSA can be derived, albeit with weaker parameters. Note that a random string with $n/2$ 1s has expected Hamming distance $n/2$ from any given string with $n/2$ 1s; the larger disparity between the number of 0s and 1s gives a better expected Hamming distance. Thus, there is a randomized algorithm approximating PWSA to within Hamming distance $n/2$, but the results below show that doing better than that by a small inverse polynomial fraction is NP-hard.

Theorem 6 (Hamming inapproximability of PWSA) *Let (e, f, ϕ) be a valid input to PWSA. If there is a polynomial-time algorithm $A(e, f, \phi)$ computing a string w which is within Hamming distance $n/2-n^\epsilon$ of a witness for any constant $\epsilon > 0$, then P = NP.*

Proof We will show how to use such a structure approximation algorithm A for PWSA to compute the exact value of the first variable in F, in a manner similar to the proof of Hamming inapproximability of SAT.

Let F be a formula on n variables and m clauses. Choose k such that $n^k > 1.5m$. Now, augment F with $n^{k/\epsilon}$ copies of the dummy clause $(v \vee \bar{v})$ to obtain a new formula F'. If the reduction from Theorem 4 is applied to this F', it will have an effect of introducing $n^{k/\epsilon}$ copies of the literal v and $n^{k/\epsilon}$ copies of the literal $\bar{v}$ as additional words of e. That is, the first $n^{k/\epsilon} + p$ words of e will be copies of v, and the following $n^{k/\epsilon} + q$ words of e will be copies of $\bar{v}$, where p and q are the numbers of positive and negative occurrences of v in the original F. The clauses $(v \vee \bar{v})$ will become $n^{k/\epsilon}$ new words in f (say the first $n^{k/\epsilon}$ words of f). Finally, $\phi([i : j), [k : l))$ is defined as before with respect to the augmented formula. This amplification preserves the correctness of the reduction, as the link $([i : j), s_1)$ forces only copies of v or only copies of $\bar{v}$ to be used to satisfy the dummy clauses. Now, if w is a correct witness (of length $N = 3m + 2n^{k/\epsilon} - 1$) to this instance, the value of v can be determined immediately: if w starts with a string of at least $n^{k/\epsilon}$ 1s, then $v = true$, and if w starts with at least $n^{k/\epsilon}$ 0s, then $v = false$.

Suppose that there is an algorithm A that returns a 'corrupted' string w' which agrees with w on at least $N/2 + N^\epsilon$ bits. Here, we are not even concerned whether w' is a valid alignment (i.e. has $|f| - 1$ 1s); any such w' will work. That is, w' agrees with w on $(3m + 2n^{k/\epsilon} - 1)/2 + (3m + 2n^{k/\epsilon} - 1)^\epsilon \geq (3m + 2n^{k/\epsilon} - 1)/2 + n^k$ positions.

Now, suppose that all the errors lie within the $2n^{k/\epsilon}$ positions corresponding to extra copies of v and $\bar{v}$. Since we chose k such that $n^k > 1.5m$, and ignoring $-1/2$, there are at least $n^{k/\epsilon} + n^k - 1.5m > n^{k/\epsilon}$ correct bits in that block, i.e. more than half of the copies of v and $\bar{v}$ are computed correctly. Taking the majority now gives us the correct value of v. □

This result can be extended to show edit distance inapproximability of PWSA using the ideas from the edit distance inapproximability proof for VertexCover.

Corollary 5 *PWSA cannot be approximated in polynomial time to within edit distance* $n/2 - n^\epsilon$ *for any constant* $\epsilon > 0$ *unless* $P = NP$.

Proof We will use the same class of instances as in Theorem 6. Note that the substring of w that we are interested in is $w_1 \dots w_r$, where $r = 2n^{k/\epsilon} + p + q$, which is the block corresponding to the first variable v in F. In a correct witness, this substring is either of the form $1111\dots000000$ or $000\dots11111$, with the number of 0s and 1s at least $n^{k/\epsilon}$ each. Now, suppose an approximation algorithm A produces a string w' which is edit distance $N/2 - N^\epsilon$ of w; that is, w' can be converted to w with at most $N/2 + N^\epsilon$ insertion, deletion and replacement operations. Consider a substring $w'_1 \dots w'_r$ in w'. As for the case of VertexCover, we can argue that the Hamming distance between $w_1 \dots w_r$ and $w'_1 \dots w'_r$ is at most $N/2 - N^\epsilon$. Indeed, suppose for the sake of contradiction that the Hamming distance between $w_1 \dots w_r$ and $w'_1 \dots w'_r$ is greater than the edit distance between these two substrings. As they have the same size, the number of insertions is the same as the number of deletions. Now, it is sufficient to say that the pair insertion/deletion can introduce at most one 0 in the "1111…1" part, and at most one 1 in the "0000…000", by the same argument as in Theorem 5. Therefore, the Hamming distance inapproximability implies edit distance inapproximability with the same parameters. □

In the proofs above, we have shown inapproximability results for the problem PWSA, in which the second sentence is assumed to be partitioned as one word per phrase. A more realistic scenario would be to assume that the witness consists of the partition strings for both e and f (here, we are still assuming that ϕ takes values in $\{0, 1\}$). The corollary below shows that for a weaker bound, there is still an inapproximability. The weakening here comes from the fact that our block becomes a smaller fraction of the total length of the witness, since f contains $n^{k/\epsilon}$ words corresponding to the dummy clauses.

Corollary 6 *WSA with* $\phi \in \{0, 1\}$ *cannot be approximated to within Hamming distance or edit distance* $2n/3 + n^\epsilon$ *for any constant* $\epsilon > 0$.

Proof Consider the same reduction as before, but now the witness is of length $|e|+|f|$ and encodes partition into phrases of f as well as of e. Thus, the total length N of the witness becomes (ignoring "-1"s) $N = (3m+2n^{k/\epsilon})+(n^{k/\epsilon}+m+n) = 4m+3n^{k/\epsilon}+n$. If the calculation above is done with this value of N, then we end up with only $0.5n^{k/\epsilon}$ guaranteed correct positions in our $2n^{k/\epsilon}$ block of interest. We need $c, 0 < c < 1$, such that $N * c + N^\epsilon - (N - 2n^{k/\epsilon}) > n^{k/\epsilon}$; choosing $c = 2/3$ satisfies this condition. □

5 Fixed parameter tractability

The inapproximability results which we have derived appear to contradict the findings of Ravi and Knight (2010), which show that the popular hill-climbing algorithm implementation of IBM Model 3 in GIZA++ gives near-optimal solutions to Viterbi alignment. Such positive results suggest that in practice, the parameters which cause Viterbi alignment to be intractable are bounded.

Originally introduced by Downey and Fellows (1992), parameterized complexity aims to give a more fine-grained understanding of NP-hard problems and how to handle them. Motivated by the existence of problems which require exponential running time when their complexity is measured solely in terms of the input size, but for which there exists algorithms which are efficient in practice, parameterized complexity aims to isolate the key aspects (parameters) that make a problem intractable. If these parameters are then restricted, the problem becomes efficiently solvable.

More precisely, if a problem exhibits an algorithm whose running time can be written $g(k) \cdot n^c$ for some constant c, parameter k, and function g which depends only on k, then we say that this problem is $\{k\}$-fixed-parameter tractable (FPT), since restricting k will give an algorithm which is computationally efficient. We denote a problem P with the parameters $\{k\}$ by $\{k\}$-P. An example of such a parameter is the number of vertices with degree greater than 3 for the VertexCover problem.

Obviously, we could restrict all of the parameters of a problem in order to obtain an FPT algorithm, but this would bound the input size of an instance of the problem. Therefore, in parameterized complexity, the goal is to limit as few parameters as possible in order to obtain an FPT algorithm which is useful in practice. This approach has been shown to be extremely effective, deriving FPT algorithms for many classic NP-hard problems. Examples include an $O(1.27^k + |V|)$ FPT algorithm for $\{k\}$-Vertex Cover, where k is the maximum number of vertices in a cover, and $|V|$ is the number of vertices in the graph, or 2^l FPT algorithm for l the number of vertices with degree ≥ 3 (Cesati 2006).

The W-hierarchy was introduced in order to classify parameterized problems which are not fixed-parameter tractable by their inherent hardness in terms of weighted satisfiability problems for certain classes of circuits. We direct the reader to the survey on parameterized complexity (Buss and Islam 2008) for more information. The W-hierarchy is structured as in (5):

$$FPT \subseteq W[1] \subseteq W[2] \subseteq \cdots \subseteq XP \tag{5}$$

A natural parameter to consider would be the maximal fertility. Although in Viterbi alignment, the fertility of an English word is only bounded above by the length of the French sentence, rarely will it be the case that a word generates more than three words in its translation. Therefore, we define the parameter Φ—the maximum fertility—to be an upper bound on the number of French words which may be aligned to a single English word. That is, $\phi_i \leq \Phi$ for all $1 \leq i \leq l$. In fact, the implementation of IBM model 3 in Giza++ restricts the maximum fertility to be 9 (Ravi and Knight 2010).

Unfortunately, even limiting the fertilities to be less than or equal to 3 is not enough to give us a fixed-parameter tractable algorithm, since reduction from MC1in3SAT to

Viterbi3 restricts fertilities to be 3 or less. Therefore in order to obtain FPT algorithms, we must consider other parameters, potentially combining them with fertility.

The reduction to NP-hardness for Viterbi3 in Udupa and Maji (2005) reduces the SetCover problem to Viterbi3. This reduction relates the number of sets in the SetCover instance to the number of words in the English sentence with non-zero fertility. As Set Cover is $W[2]$-hard, restricting the number of words with non-zero fertility in Viterbi3, as well as restricting the length of the resulting sentence in relaxed decoding for model 3, does not make the problem computationally easier.

For many pairs of languages, such as English and French, it is natural to assume that any word in one sentence will be translated to a word 'not too far away' in the other. We define the distortion parameter k to be the radius in which an English word may translate to a French word. Formally, $\forall j,\ f_j \in [\max\{0, l/m(j-k)\}, \min\{l/m(j+k), l\}]$. In fact, in many practical implementations, k is limited to 4 (the "IBM constraint" Lopez 2008). This can be expressed in terms of the distortion model for IBM model 3 as $d(j|i,m,l) = 0 \text{ if } i \notin \lceil \frac{l}{m}(j \pm k) \rceil$.

The following dynamic programming algorithm, although not optimal, shows that $\{k\}$-Viterbi3 is fixed-parameter tractable.

Theorem 7 *$\{k\}$-Viterbi3 is fixed-parameter tractable.*

Proof We will describe a left-to-right dynamic programming algorithm which achieves a running time of $g(k)n^c$ for some function g and constant c.

We construct a $2^{(2k)} \times l$ table with columns corresponding to the words $e_1, \ldots, e_l$ in the English sentence, and rows corresponding to the patterns of already aligned and not aligned words in the radius of k around position $j = m/l(i)$ in the French sentence. We call such unaligned positions 'blank words'. At each step, the algorithm proceeds by recording the optimal score thus far for alignments ending in each possible arrangement of blanks and non-blanks in the $2k$-length block around $j = m/l \cdot i$.

As the dynamic programming algorithm requires a table $2^{(2k)} \times l$ entries, and each entry can be computed in polynomial time, this algorithm runs in time $O(2^{(2k)} \cdot l)$. □

To illustrate how this algorithm works, let $l = m = 4$ and $k = 1$. Thus, each e_i can align to f_{i-1}, f_i, f_{i+1}. Consider a length $2k$ block of positions in f, with the $k+1$st position in the block corresponding to $j = m/l \cdot i$. Now, it is enough to know the probability of the best alignment of $f_1, \ldots, f_{j+k-1}$ ending in each of the potential pattern of aligned versus unaligned positions in the $2k$ block to compute the next iteration of the dynamic programming algorithm. In our example, it is enough to compute a matrix A of best probabilities for four patterns of the $2k = 2$ block, corresponding to (*blank*, *blank*), (*blank*, *word*), (*word*, *blank*), ($word1$, $word2$), for each i. Accordingly, before considering (for example) e_3, we have precomputed the best alignments of (e_1, e_2) to (f_1, f_2, f_3) for four cases: f_2, f_3 both unassigned, f_2 unassigned, f_3 unassigned, all three assigned. Now, the value of the best alignment involving e_3 and with (*blank*, *word*) pattern is the maximum of two choices: $P(a_2 = 3) \cdot n(\phi_3 = 2|e_3) \cdot P(a_4 = 3) \times A(2, \text{"}bb\text{"})$, and $n(\phi_3 = 1|e_3) \cdot P(a_4 = 3) \times A(2, \text{"}wb\text{"})$.

We summarize the parameterized complexity of Viterbi3 for the parameters considered above in Table 1.

Table 1 A summary of parameterized complexity of Viterbi3

Parameter	Complexity
k (distortion)	FPT
Φ (maximum fertility)	NP-hard for constant ≥ 3
l (no. of words with non-zero ϕ_i)	$W[2]$-hard

6 Conclusions

In this paper we considered the complexity of computing a close-to-optimal solution for Viterbi alignment and relaxed decoding for IBM model 3, as well as for forced decoding in the phrase-based translation model. For model 3 Viterbi alignment, computing correcty more than half of fertilities (plus an inverse polynomial fraction) is already intractable. For the relaxed decoding problem, so is computing more than half of the words of the resulting English sentence. Both of these results hold even if insertion and deletion errors are allowed, in addition to corrupting individual values. A similar result holds for the perfect phrase alignment problem as described in DeNero and Klein (2008); there, the computationally intractable component is computing correct phrase boundaries. We used the framework of Hamilton et al. (2007) and the techniques of Sheldon and Young (2013) for this task, in particular showing how the Hamming distance results of Sheldon and Young (2013) can be extended to edit distance for several problems.

From the practical point of view, it is more important to determine what restrictions and relaxations of problem specifications allow for efficient algorithms. We consider Viterbi alignment problem in the framework of parameterized complexity, and note that this problem is fixed-parameter tractable with respect to the distortion parameter, but intractable with maximal fertility or length of the decoded sentence as parameters. However, this is just the first step in this direction; we hope to perform a more thorough parameterized analysis of this and related problems in the future.

As integer linear programming-based heuristics seem relatively common in SMT, another direction for future work would be to analyse solutions produced by ILP and SDP algorithms, and show both upper and lower bound on approximation guarantee for such algorithms. Some LP relaxation-based approximation algorithms, in particular that for VertexCover, give good upper bounds on Hamming distance to an optimal solution. It would be interesting to see whether linear programming approximation algorithms for SMT problems give matching upper bounds to our inapproximability results, or better still, produce solutions close to the optimal under some conditions.

Acknowledgments We are very grateful to the anonymous referees and the editor of the Machine Translation journal for suggesting a more relevant setting to apply our techniques, and pointing us to the literature. We also want to thank Todd Wareham, Valentine Kabanets and Russell Impagliazzo for numerous discussions and suggestions, and to Venkat Guruswami for telling us about then-unpublished work of Sheldon and Young.

References

Berman L, Hartmanis J (1977) On isomorphisms and density of NP and other complete sets. SIAM J Comput 6(2):305–322

Birch A, Callison-Burch C, Osborne M, Koehn P (2006) Constraining the phrase-based, joint probability statistical translation model. In: HLT-NAACL 2006: proceedings of the workshop on statistical machine translation, New York, pp 154–157

Brown PF, Della Pietra VJ, Della Pietra SA, Mercer RL (1993) The mathematics of statistical machine translation: parameter estimation. Comput Linguist 19(2):263–311

Buss JF, Islam TM (2008) The complexity of fixed-parameter problems: guest column. SIGACT News 39(1):33–46. doi:10.1145/1360443.1360454

Cesati M (2006) Compendium of parameterized problems. http://www.sprg.uniroma2.it/home/cesati/research/compendium/compendium.pdf

DeNero J (2010) Phrase alignment models for statistical machine translation. PhD Thesis, UC Berkeley, CA

DeNero J, Klein D (2008) The complexity of phrase alignment problems. In: ACL-08: HLT. Proceedings of the 46th annual meeting of the association for computational linguistics on human language technologies: short papers, Columbus, pp 25–28

Downey RG, Fellows MR (1992) Fixed-parameter intractability. In: Proceedings of the seventh annual conference on structure in complexity theory, Victoria, pp 36–49

Feige U, Langberg M, Nissim K (2000) On the hardness of approximating NP witnesses. In: Approx 2000: approximation algorithms for combinatorial optimization. Proceedings of third international workshop. Lecture notes in computer science 1913. Springer, New York, pp 120–131

Gal A, Halevi S, Lipton RJ, Petrank E (1999) Computing from partial solutions. In: COCO '99: proceedings of the fourteenth annual IEEE conference on computational complexity, Atlanta, pp 34–45

Guruswami V, Rudra A (2008) Soft decoding, dual BCH codes, and better list-decodable ε-biased codes. In: CCC 2008: proceedings of the twenty-third annual IEEE conference on computational complexity, College Park, pp 163–174

Hamilton M, Müller M, van Rooij I, Wareham T (2007) Approximating solution structure. In: Demaine E, Gutin GZ, Marx D, Stege U (eds) Structure theory and FPT algorithmics for graphs, digraphs and hypergraphs, No. 07281 in Dagstuhl seminar proceedings. Internationales Begegnungs- und Forschungszentrum für Informatik (IBFI). Schloss Dagstuhl, Germany, Dagstuhl

Knight K (1999) Decoding complexity in word-replacement translation models. Comput Linguist 25(4):607–615

Koehn P (2004) Pharaoh: a beam search decoder for phrase-based statistical machine translation models. In: Machine translation: from real users to research: 6th conference of the Association for Machine Translation in the Americas. Springer, Berlin, pp 115–124

Koehn P, Och FJ, Marcu D (2003) Statistical phrase-based translation. In: HLT-NAACL 2003: conference combining human language technology conference series and the North American Chapter of the Association for Computational Linguistics conference series. Proceedings, Edmonton, pp 48–54

Kuhn HW (1955) The Hungarian method for the assignment problem. Naval Res Logist Q 2:83–97

Kumar R, Sivakumar D (1999) Proofs, codes, and polynomial-time reducibilities. In: COCO '99: proceedings of the fourteenth annual IEEE conference on computational complexity, Atlanta, pp 46–53

Lopez A (2008) Statistical machine translation. ACM Comput Surv 40(3):1–49

MacCartney B, Galley M, Manning CD (2008) A phrase-based alignment model for natural language inference. In: Proceedings of the conference on empirical methods in natural language processing. Association for Computational Linguistics, pp 802–811

Marcu D, Wong W (2002) A phrase-based, joint probability model for statistical machine translation. In: EMNLP-2002: proceedings of the 2002 conference on empirical methods in natural language processing, Philadelphia, pp 133–139

Och FJ, Ney H (2003) A systematic comparison of various statistical alignment models. Comput Linguist 29(1):19–51

Ravi S, Knight K (2010) Does giza++ make search errors? Comput Linguist 36(3):295–302

Sheldon D, Young NE (2013) Hamming approximation of NP witnesses. Theory Comput 9(22):685–702

Søgaard A (2009) On the complexity of alignment problems in two synchronous grammar formalisms. In: Proceedings of the third workshop on syntax and structure in statistical translation (SSST-3) at NAACL HLT 2009, Boulder, pp 60–68

Udupa R, Maji H (2005) Theory of alignment generators and applications to statistical machine translation. In: Proceedings of the 19th international joint conference on artificial intelligence, Edinburgh, pp 1142–1147

Udupa R, Maji HK (2006) Computational complexity of statistical machine translation. In: EACL-2006: 11th conference of the European chapter of the Association for Computational Linguistics, proceedings of the conference, Trento, pp 25–32

van Rooij I, Wareham T (2012) Intractability and approximation of optimization theories of cognition. J Math Psychol 56(4):232–247