A Polynomial-Time Dynamic Programming Algorithm for Phrase-Based Decoding with a Fixed Distortion Limit

Yin-Wen Chang
Google Research, New York
yinwen@google.com

Michael Collins*
Google Research, New York
mjcollins@google.com

Abstract

Decoding of phrase-based translation models in the general case is known to be NP-complete, by a reduction from the traveling salesman problem (Knight, 1999). In practice, phrase-based systems often impose a hard distortion limit that limits the movement of phrases during translation. However, the impact on complexity after imposing such a constraint is not well studied. In this paper, we describe a dynamic programming algorithm for phrase-based decoding with a fixed distortion limit. The runtime of the algorithm is $O(nd!lh^{d+1})$ where n is the sentence length, d is the distortion limit, l is a bound on the number of phrases starting at any position in the sentence, and h is related to the maximum number of target language translations for any source word. The algorithm makes use of a novel representation that gives a new perspective on decoding of phrase-based models.

1 Introduction

Phrase-based translation models (Koehn et al., 2003; Och and Ney, 2004) are widely used in statistical machine translation. The decoding problem for phrase-based translation models is known to be difficult: the results from Knight (1999) imply that in the general case decoding of phrase-based translation models is NP-complete.

The complexity of phrase-based decoding comes from reordering of phrases. In practice, however, various constraints on reordering are often imposed in phrase-based translation systems. A common constraint is a "distortion limit", which places a hard constraint on how far phrases can move. The complexity of decoding with such a distortion limit is an open question: the NP-hardness result from Knight (1999) applies to a phrase-based model with no distortion limit.

This paper describes an algorithm for phrase-based decoding with a fixed distortion limit whose runtime is linear in the length of the sentence, and for a fixed distortion limit is polynomial in other factors. More specifically, for a hard distortion limit d, and sentence length n, the runtime is $O(nd!lh^{d+1})$, where l is a bound on the number of phrases starting at any point in the sentence, and h is related to the maximum number of translations for any word in the source language sentence.

The algorithm builds on the insight that decoding with a hard distortion limit is related to the bandwidth-limited traveling salesman problem (BTSP) (Lawler et al., 1985). The algorithm is easily amenable to beam search. It is quite different from previous methods for decoding of phrase-based models, potentially opening up a very different way of thinking about decoding algorithms for phrase-based models, or more generally for models in statistical NLP that involve reordering.

2 Related Work

Knight (1999) proves that decoding of word-to-word translation models is NP-complete, assuming that there is no hard limit on distortion, through a reduction from the traveling salesman problem. Phrase-based models are more general than word-to-word models, hence this result implies that phrase-based decoding with unlimited distortion is NP-complete.

Phrase-based systems can make use of both *reordering constraints*, which give a hard "distortion limit" on how far phrases can move, and *reordering models*, which give scores for reordering steps, often penalizing phrases that move long distances. Moses (Koehn et al., 2007b) makes use of a distortion limit, and a decoding algorithm that makes use

* On leave from Columbia University.

Transactions of the Association for Computational Linguistics, vol. 5, pp. 59–71, 2017. Action Editor: Holger Schwenk.
Submission batch: 10/2016; Revision batch: 11/2016; Published 2/2017.

of bit-strings representing which words have been translated. We show in Section 5.2 of this paper that this can lead to at least $2^{n/4}$ bit-strings for an input sentence of length n, hence an exhaustive version of this algorithm has worst-case runtime that is exponential in the sentence length. The current paper is concerned with decoding phrase-based models with a hard distortion limit.

Various other reordering constraints have been considered. Zens and Ney (2003) and Zens et al. (2004) consider two types of hard constraints: the IBM constraints, and the ITG (inversion transduction grammar) constraints from the model of Wu (1997). They give polynomial time dynamic programming algorithms for both of these cases. It is important to note that the IBM and ITG constraints are different from the distortion limit constraint considered in the current paper. Decoding algorithms with ITG constraints are further studied by Feng et al. (2010) and Cherry et al. (2012).

Kumar and Byrne (2005) describe a class of reordering constraints and models that can be encoded in finite state transducers. Lopez (2009) shows that several translation models can be represented as weighted deduction problems and analyzes their complexities.[1] Koehn et al. (2003) describe a beam-search algorithm for phrase-based decoding that is in widespread use; see Section 5 for discussion.

A number of reordering models have been proposed, see for example Tillmann (2004), Koehn et al. (2007a) and Galley and Manning (2008).

DeNero and Klein (2008) consider the phrase alignment problem, that is, the problem of finding an optimal phrase-based alignment for a source-language/target-language sentence pair. They show that in the general case, the phrase alignment problem is NP-hard. It may be possible to extend the techniques in the current paper to the phrase-alignment problem with a hard distortion limit.

Various methods for exact decoding of phrase-based translation models have been proposed. Zaslavskiy et al. (2009) describe the use of traveling salesman algorithms for phrase-based decoding. Chang and Collins (2011) describe an exact method based on Lagrangian relaxation. Aziz et al. (2014) describe a coarse-to-fine approach. These algorithms all have exponential time runtime (in the length of the sentence) in the worst case.

Galley and Manning (2010) describe a decoding algorithm for phrase-based systems where phrases can have discontinuities in both the source and target languages. The algorithm has some similarities to the algorithm we propose: in particular, it makes use of a state representation that contains a list of disconnected phrases. However, the algorithms differ in several important ways: Galley and Manning (2010) make use of bit string coverage vectors, giving an exponential number of possible states; in contrast to our approach, the translations are not formed in strictly left-to-right ordering on the source side.

3 Background: The Traveling Salesman Problem on Bandwidth-Limited Graphs

This section first defines the bandwidth-limited traveling salesman problem, then describes a polynomial time dynamic programming algorithm for the *traveling salesman path problem* on bandwidth limited graphs. This algorithm is the algorithm proposed by Lawler et al. (1985)[2] with small modifications to make the goal a *path* instead of a *cycle*, and to consider directed rather than undirected graphs.

3.1 Bandwidth-Limited TSPPs

The input to the problem is a directed graph $G = (V, E)$, where V is a set of vertices and E is a set of directed edges. We assume that $V = \{1, 2, \ldots, n\}$. A directed edge is a pair (i, j) where $i, j \in V$, and $i \neq j$. Each edge $(i, j) \in E$ has an associated weight $w_{i,j}$. Given an integer $k \geq 1$, a graph is **bandwidth-limited** with bandwidth k if

$$\forall (i, j) \in E, \quad |i - j| \leq k$$

The *traveling salesman path problem* (TSPP) on the graph G is defined as follows. We will assume that vertex 1 is the "source" vertex and vertex n is the "sink" vertex. The TSPP is to find the minimum cost directed path from vertex 1 to vertex n, which passes through each vertex exactly once.

[1] An earlier version of this paper states the complexity of decoding with a distortion limit as $O(I^3 2^d)$ where d is the distortion limit and I is the number of words in the sentence; however (personal communication from Adam Lopez) this runtime is an error, and should be $O(2^I)$ i.e., exponential time in the length of the sentence. A corrected version of the paper corrects this.

[2] The algorithm is based on the ideas of Monien and Sudborough (1981) and Ratliff and Rosenthal (1983).

3.2 An Algorithm for Bandwidth-Limited TSPPs

The key idea of the dynamic-programming algorithm for TSPPs is the definition of equivalence classes corresponding to dynamic programming states, and an argument that the number of equivalence classes depends only on the bandwidth k.

The input to our algorithm will be a directed graph $G = (V, E)$, with weights $w_{i,j}$, and with bandwidth k. We define a 1-n *path* to be any path from the source vertex 1 to the sink vertex n that visits each vertex in the graph exactly once. A 1-n path is a subgraph (V', E') of G, where $V' = V$ and $E' \subseteq E$.

We will make use of the following definition:

Definition 1. *For any* 1-n *path* H*, define* H_j *to be the subgraph that* H *induces on vertices* $1, 2, \ldots j$*, where* $1 \leq j \leq n$*.*

That is, H_j contains the vertices $1, 2, \ldots j$ and the edges in H between these vertices.

For a given value for j, we divide the vertices V into three sets A_j, B_j and C_j:

- $A_j = \{1, 2, \ldots, (j-k)\}$ (A_j is the empty set if $j \leq k$).
- $B_j = \{1 \ldots j\} \setminus A_j$.[3]
- $C_j = \{j+1, j+2, \ldots, n\}$ (C_j is the empty set if $j = n$).

Note that the vertices in subgraph H_j are the union of the sets A_j and B_j. A_j is the empty set if $j \leq k$, but B_j is always non-empty.

The following Lemma then applies:

Lemma 1. *For any* 1-n *path* H *in a graph with bandwidth* k*, for any* $1 \leq j \leq n$*, the subgraph* H_j *has the following properties:*

1. *If vertex* 1 *is in* A_j*, then vertex* 1 *has degree one.*
2. *For any vertex* $v \in A_j$ *with* $v \geq 2$*, vertex* v *has degree two.*
3. H_j *contains no cycles.*

Proof. The first and second properties are true because of the bandwidth limit. Under the constraint of bandwidth k, any edge (u, v) in H such that $u \in A_j$, must have $v \in A_j \cup B_j = H_j$. This follows because if $v \in C_j = \{j+1, j+2, \ldots n\}$ and $u \in A_j = \{1, 2, \ldots j-k\}$, then $|u - v| > k$. Similarly any edge $(u, v) \in H$ such that $v \in A_j$ must have $u \in A_j \cup B_j = H_j$. It follows that for any vertex $u \in A_j$, with $u > 1$, there are edges $(u, v) \in H_j$ and $(v', u) \in H_j$, hence vertex u has degree 2. For vertex $u \in A_j$ with $u = 1$, there is an edge $(u, v) \in H_j$, hence vertex u has degree 1. The third property (no cycles) is true because H_j is a subgraph of H, which has no cycles. □

It follows that each connected component of H_j is a directed path, that the start points of these paths are in the set $\{1\} \cup B_j$, and that the end points of these paths are in the set B_j.

We now define an equivalence relation on subgraphs. Two subgraphs H_j and H'_j are in the same equivalence class if the following conditions hold (taken from Lawler et al. (1985)):

1. For any vertex $v \in B_j$, the degree of v in H_j and H'_j is the same.
2. For each path (connected component) in H_j there is a path in H'_j with the same start and end points, and conversely.

The significance of this definition is as follows. Assume that H^* is an optimal 1-n path in the graph, and that it induces the subgraph H_j on vertices $1 \ldots j$. Assume that H'_j is another subgraph over vertices $1 \ldots j$, which is in the same equivalence class as H_j. For any subgraph H_j, define $c(H_j)$ to be the sum of edge weights in H_j:

$$c(H_j) = \sum_{(u,v) \in H_j} w_{u,v}$$

Then it must be the case that $c(H'_j) \geq c(H_j)$. Otherwise, we could simply replace H_j by H'_j in H^*, thereby deriving a new 1-n path with a lower cost, implying that H^* is not optimal.

This observation underlies the dynamic programming approach. Define σ to be a function that maps a subgraph H_j to its equivalence class $\sigma(H_j)$. The equivalence class $\sigma(H_j)$ is a data structure that stores the degrees of the vertices in B_j, together with the start and end points of each connected component in H_j.

[3] For sets X and Y we use the notation $X \setminus Y$ to refer to the set difference: i.e., $X \setminus Y = \{x | x \in X \text{ and } x \notin Y\}$.

Next, define Δ to be a set of 0, 1 or 2 edges between vertex $(j+1)$ and the vertices in B_j. For any subgraph H_{j+1} of a 1-n path, there is some Δ, simply found by recording the edges incident to vertex $(j+1)$. For any H_j, define $\tau(\sigma(H_j), \Delta)$ to be the equivalence class resulting from adding the edges in Δ to the data structure $\sigma(H_j)$. If adding the edges in Δ to $\sigma(H_j)$ results in an ill-formed subgraph—for example, a subgraph that has one or more cycles—then $\tau(\sigma(H_j), \Delta)$ is undefined. The following recurrence then defines the dynamic program (see Eq. 20 of Lawler et al. (1985)):

$$\alpha(j+1, S) = \min_{\Delta, S' : \tau(S', \Delta) = S} \big(\alpha(j, S') + c(\Delta)\big)$$

Here S is an equivalence class over vertices $\{1 \dots (j+1)\}$, and $\alpha(S, j+1)$ is the minimum score for any subgraph in equivalence class S. The $\min$ is taken over all equivalence classes S' over vertices $\{1 \dots j\}$, together with all possible values for Δ.

4 A Dynamic Programming Algorithm for Phrase-Based Decoding

We now describe the dynamic programming algorithm for phrase-based decoding with a fixed distortion limit. We first give basic definitions for phrase-based decoding, and then describe the algorithm.

4.1 Basic Definitions

Consider decoding an input sentence consisting of words $x_1 \dots x_n$ for some integer n. We assume that $x_1 = \texttt{<s>}$ and $x_n = \texttt{</s>}$ where <s> and </s> are the sentence start and end symbols respectively. A phrase-based lexicon specifies a set of possible translations in the form of phrases $p = (s, t, e)$, where s and t are integers such that $1 \leq s \leq t \leq n$, and e is a sequence of $m \geq 1$ target-language words $e_1 \dots e_m$. This signifies that words $x_s \dots x_t$ in the source language have a translation as $e_1 \dots e_m$ in the target language. We use $s(p)$, $t(p)$ and $e(p)$ to refer to the three components of a phrase $p = (s, t, e)$, and $e_1(p) \dots e_m(p)$ to refer to the words in the target-language string $e(p)$. We assume that $(1, 1, \texttt{<s>})$ and $(n, n, \texttt{</s>})$ are the only translation entries with $s(p) \leq 1$ and $t(p) \geq n$ respectively.

A *derivation* is then defined as follows:

Definition 2 (Derivations)**.** *A derivation is a sequence of phrases $p_1 \dots p_L$ such that*

- $p_1 = (1, 1, \texttt{<s>})$ *and* $p_L = (n, n, \texttt{</s>})$.
- *Each source word is translated exactly once.*
- *The distortion limit is satisfied for each pair of phrases p_{i-1}, p_i, that is:*

$$|t(p_{i-1}) + 1 - s(p_i)| \leq d \ \ \forall \, i = 2 \dots L.$$

where d is an integer specifying the distortion limit in the model.

Given a derivation $p_1 \dots p_L$, a target-language translation can be obtained by concatenating the target-language strings $e(p_1) \dots e(p_L)$.

The scoring function is defined as follows:

$$f(p_1 \dots p_L) = \lambda(e(p_1) \dots e(p_L)) + \sum_{i=1}^{L} \kappa(p_i) + \sum_{i=2}^{L} \eta \times |t(p_{i-1}) + 1 - s(p_i)| \quad (1)$$

For each phrase p, $\kappa(p)$ is the translation score for the phrase. The parameter η is the distortion penalty, which is typically a negative constant. $\lambda(e)$ is a language model score for the string e. We will assume a bigram language model:

$$\lambda(e_1 \dots e_m) = \sum_{i=2}^{m} \lambda(e_i | e_{i-1}).$$

The generalization of our algorithm to higher-order n-gram language models is straightforward.

The goal of phrase-based decoding is to find $y^* = \arg\max_{y \in \mathcal{Y}} f(y)$ where $\mathcal{Y}$ is the set of valid derivations for the input sentence.

Remark (gap constraint): Note that a common restriction used in phrase-based decoding (Koehn et al., 2003; Chang and Collins, 2011), is to impose an additional "gap constraint" while decoding. See Chang and Collins (2011) for a description. In this case it is impossible to have a dynamic-programming state where word x_i has not been translated, and where word x_{i+k} has been translated, for $k > d$. This limits distortions further, and it can be shown in this case that the number of possible bitstrings is $O(2^d)$ where d is the distortion limit. Without this constraint the algorithm of Koehn et al. (2003) actually fails to produce translations for many input sentences (Chang and Collins, 2011).

$H_1 = \langle \pi_1 \rangle =$ $\langle\langle (1, 1, \text{<s>}) \rangle\rangle$
$H_3 = \langle \pi_1 \rangle =$ $\langle\langle (1, 1, \text{<s>})(2, 3, \text{we must}) \rangle\rangle$
$H_4 = \langle \pi_1 \rangle =$ $\langle\langle (1, 1, \text{<s>})(2, 3, \text{we must})(4, 4, \text{also}) \rangle\rangle$
$H_6 = \langle \pi_1, \pi_2 \rangle =$ $\langle\langle (1, 1, \text{<s>})(2, 3, \text{we must})(4, 4, \text{also}) \rangle, \langle (5, 6, \text{these criticisms}) \rangle\rangle$
$H_7 = \langle \pi_1, \pi_2 \rangle =$ $\langle\langle (1, 1, \text{<s>})(2, 3, \text{we must})(4, 4, \text{also}) \rangle, \langle (5, 6, \text{these criticisms})(7, 7, \text{seriously}) \rangle\rangle$
$H_8 = \langle \pi_1 \rangle =$ $\langle\langle (1, 1, \text{<s>})(2, 3, \text{we must})(4, 4, \text{also})(8, 8, \text{take})(5, 6, \text{these criticisms})(7, 7, \text{seriously}) \rangle\rangle$
$H_9 = \langle \pi_1 \rangle =$ $\langle\langle (1, 1, \text{<s>})(2, 3, \text{we must})(4, 4, \text{also})(8, 8, \text{take})(5, 6, \text{these criticisms})(7, 7, \text{seriously})(9, 9, \text{</s>}) \rangle\rangle$

Figure 1: Sub-derivations H_j for $j \in \{1, 3, 4, 6, 7, 8, 9\}$ induced by the full derivation $H = \big\langle\big\langle (1, 1, \text{<s>})(2, 3, \text{we must})(4, 4, \text{also})(8, 8, \text{take})(5, 6, \text{these criticisms})(7, 7, \text{seriously})(9, 9\text{</s>}) \big\rangle\big\rangle$. Note that H_j includes the phrases that cover spans ending before or at position j. Sub-derivation H_j is extended to another sub-derivation H_{j+i} by incorporating a phrase of length i.

4.2 The Algorithm

We now describe the dynamic programming algorithm. Intuitively the algorithm builds a derivation by processing the *source-language sentence* in strictly left-to-right order. This is in contrast with the algorithm of Koehn et al. (2007b), where the *target-language sentence* is constructed from left to right.

Throughout this section we will use π, or π_i for some integer i, to refer to a sequence of phrases:

$$\pi = \big\langle p_1 \dots p_l \big\rangle$$

where each phrase $p_i = (s(p_i), t(p_i), e(p_i))$, as defined in the previous section.

We overload the s, t and e operators, so that if $\pi = \big\langle p_1 \dots p_l \big\rangle$, we have $s(\pi) = s(p_1)$, $t(\pi) = t(p_l)$, and $e(\pi) = e(p_1) \cdot e(p_2) \dots \cdot e(p_l)$, where $x \cdot y$ is the concatenation of strings x and y.

A *derivation* H consists of a single phrase sequence $\pi = \big\langle p_1 \dots p_L \big\rangle$:

$$H = \pi = \big\langle p_1 \dots p_L \big\rangle$$

where the sequence $p_1 \dots p_L$ satisfies the constraints in definition 2.

We now give a definition of *sub-derivations* and *complement sub-derivations*:

Definition 3 (Sub-derivations and Complement Sub--derivations)**.** *For any $H = \big\langle p_1 \dots p_L \big\rangle$, for any $j \in \{1 \dots n\}$ such that $\exists i \in \{1 \dots L\}$ s.t. $t(p_i) = j$, the sub-derivation H_j and the complement sub-derivation $\bar{H}_j$ are defined as*

$$H_j = \langle \pi_1 \dots \pi_r \rangle, \quad \bar{H}_j = \langle \bar{\pi}_1 \dots \bar{\pi}_r \rangle$$

where the following properties hold:

- *r is an integer with $r \geq 1$.*
- *Each π_i for $i = 1 \dots r$ is a sequence of one or more phrases, where each phrase $p \in \pi_i$ has $t(p) \leq j$.*
- *Each $\bar{\pi}_i$ for $i = 1 \dots (r-1)$ is a sequence of one or more phrases, where each phrase $p \in \bar{\pi}_i$ has $s(p) > j$.*
- *$\bar{\pi}_r$ is a sequence of zero or more phrases, where each phrase $p \in \bar{\pi}_r$ has $s(p) > j$. We have zero phrases in $\bar{\pi}_r$ iff $j = n$ where n is the length of the sentence.*
- *Finally, $\pi_1 \cdot \bar{\pi}_1 \cdot \pi_2 \cdot \bar{\pi}_2 \dots \pi_r \cdot \bar{\pi}_r = p_1 \dots p_L$ where $x \cdot y$ denotes the concatenation of phrase sequences x and y.*

Note that for any $j \in \{1 \dots n\}$ such that $\nexists i \in \{1 \dots L\}$ such that $t(p_i) = j$, the sub-derivation H_j and the complement sub-derivation $\bar{H}_j$ is not defined.

Thus for each integer j such that there is a phrase in H ending at point j, we can divide the phrases in H into two sets: phrases p with $t(p) \leq j$, and phrases p with $s(p) > j$. The sub-derivation H_j lists all maximal sub-sequences of phrases with $t(p) \leq j$. The complement sub-derivation $\bar{H}_j$ lists all maximal sub-sequences of phrases with $s(p) > j$.

Figure 1 gives all sub-derivations H_j for the derivation

$$\begin{aligned} H &= \big\langle\big\langle p_1 \dots p_7 \big\rangle\big\rangle \\ &= \big\langle\big\langle (1, 1, \text{<s>})(2, 3, \text{we must})(4, 4, \text{also}) \\ &\qquad (8, 8, \text{take})(5, 6, \text{these criticisms}) \\ &\qquad (7, 7, \text{seriously})(9, 9, \text{</s>}) \big\rangle\big\rangle \end{aligned}$$

As one example, the sub-derivation $H_7 = \langle \pi_1, \pi_2 \rangle$ induced by H has two phrase sequences:

$$\pi_1 = \big\langle (1, 1, \text{<s>})(2, 3, \text{we must})(4, 4, \text{also}) \big\rangle$$
$$\pi_2 = \big\langle (5, 6, \text{these criticisms})(7, 7, \text{seriously}) \big\rangle$$

Note that the phrase sequences π_1 and π_2 give translations for all words $x_1 \dots x_7$ in the sentence. There

are two disjoint phrase sequences because in the full derivation H, the phrase $p = (8, 8, \text{take})$, with $t(p) = 8 > 7$, is used to form a longer sequence of phrases $\pi_1\, p\, \pi_2$.

For the above example, the complement sub-derivation $\bar{H}_7$ is as follows:

$$\bar{\pi}_1 = \big\langle (8, 8, \textsf{take}) \big\rangle$$
$$\bar{\pi}_2 = \big\langle (9, 9, \texttt{</s>}) \big\rangle$$

It can be verified that $\pi_1 \cdot \bar{\pi}_1 \cdot \pi_2 \cdot \bar{\pi}_2 = H$ as required by the definition of sub-derivations and complement sub-derivations.

We now state the following Lemma:

Lemma 2. *For any derivation $H = p_1 \dots p_L$, for any j such that $\exists i$ such that $t(p_i) = j$, the sub-derivation $H_j = \langle \pi_1 \dots \pi_r \rangle$ satisfies the following properties:*

1. *$s(\pi_1) = 1$ and $e_1(\pi_1) =$ <s>.*
2. *For all positions $i \in \{1 \dots j\}$, there exists a phrase $p \in \pi$, for some phrase sequence $\pi \in H_j$, such that $s(p) \leq i \leq t(p)$.*
3. *For all $i = 2 \dots r$, $s(\pi_i) \in \{(j - d + 2) \dots j\}$*
4. *For all $i = 1 \dots r$, $t(\pi_i) \in \{(j - d) \dots j\}$*

Here d is again the distortion limit.

This lemma is a close analogy of Lemma 1. The proof is as follows:

Proof of Property 1: For all values of j, the phrase $p_1 = (1, 1, \texttt{<s>})$ has $t(p_1) \leq j$, hence we must have $\pi_1 = p_1 \dots p_k$ for some $k \in \{1 \dots L\}$. It follows that $s(\pi_1) = 1$ and $e_1(\pi_1) =$ <s>.

Proof of Property 2: For any position $i \in \{1 \dots j\}$, define the phrase (s, t, e) in the derivation H to be the phrase that covers word i; i.e., the phrase such that $s \leq i \leq t$. We must have $s \in \{1 \dots j\}$, because $s \leq i$ and $i \leq j$. We must also have $t \in \{1 \dots j\}$, because otherwise we have $s \leq j < t$, which contradicts the assumption that there is some $i \in \{1 \dots L\}$ such that $t(p_i) = j$. It follows that the phrase (s, t, e) has $t \leq j$, and from the definition of sub-derivations it follows that the phrase is in one of the phrase sequences $\pi_1 \dots \pi_r$.

Proof of Property 3: This follows from the distortion limit. Consider the complement sub-derivation $\bar{H}_j = \langle \bar{\pi}_1 \dots \bar{\pi}_r \rangle$. For the distortion limit to be satisfied, for all $i \in \{2 \dots r\}$, we must have

$$|t(\bar{\pi}_{i-1}) + 1 - s(\pi_i)| \leq d$$

We must also have $t(\bar{\pi}_{i-1}) > j$, and $s(\pi_i) \leq j$, by the definition of sub-derivations. It follows that $s(\pi_i) \in \{(j - d + 2) \dots j\}$.

Proof of Property 4: This follows from the distortion limit. First consider the case where $\bar{\pi}_r$ is non-empty. For the distortion limit to be satisfied, for all $i \in \{1 \dots r\}$, we must have

$$|t(\pi_i) + 1 - s(\bar{\pi}_i)| \leq d$$

We must also have $t(\pi_i) \leq j$, and $s(\bar{\pi}_i) > j$, by the definition of sub-derivations. It follows that $t(\pi_i) \in \{(j - d) \dots j\}$.

Next consider the case where $\bar{\pi}_r$ is empty. In this case we must have $j = n$. For the distortion limit to be satisfied, for all $i \in \{1 \dots (r-1)\}$, we must have

$$|t(\pi_i) + 1 - s(\bar{\pi}_i)| \leq d$$

We must also have $t(\pi_i) \leq j$, and $s(\bar{\pi}_i) > j$, by the definition of sub-derivations. It follows that $t(\pi_i) \in \{(j - d) \dots j\}$ for $i \in \{1 \dots (r-1)\}$. For $i = r$, we must have $t(\pi_i) = n$, from which it again follows that $t(\pi_r) = n \in \{(j - d) \dots j\}$. $\square$

We now define an equivalence relation between sub-derivations, which will be central to the dynamic programming algorithm. We define a function σ that maps a phrase sequence π to its *signature*. The signature is a four-tuple:

$$\sigma(\pi) = (s, w_s, t, w_t).$$

where s is the start position, w_s is the start word, t is the end position and w_t is the end word of the phrase sequence. We will use $s(\sigma)$, $w_s(\sigma)$, $t(\sigma)$, and $w_t(\sigma)$ to refer to each component of a signature σ.

For example, given a phrase sequence

$$\pi = \big\langle (\textsf{1, 1, <s>})\ (\textsf{2, 2, we})\ (\textsf{4, 4, also}) \big\rangle,$$

its signature is $\sigma(\pi) = (\textsf{1, <s>, 4, also})$.

The signature of a sub-derivation $H_j = \langle \pi_1 \dots \pi_r \rangle$ is defined to be

$$\sigma(H_j) = \langle \sigma(\pi_1) \dots \sigma(\pi_r) \rangle.$$

For example, with H_7 as defined above, we have

$$\sigma(H_7) = \big\langle (1, \texttt{<s>}, 4, \textsf{also}), (5, \textsf{these}, 7, \textsf{seriously}) \big\rangle$$

Two partial derivations H_j and H'_j are in the same equivalence class iff $\sigma(H_j) = \sigma(H'_j)$.

We can now state the following Lemma:

Lemma 3. *Define H^* to be the optimal derivation for some input sentence, and H^*_j to be a sub-derivation of H^*. Suppose H'_j is another sub-derivation with j words, such that $\sigma(H'_j) = \sigma(H^*_j)$. Then it must be the case that $f(H^*_j) \geq f(H'_j)$, where f is the function defined in Section 4.1.*

Proof. Define the sub-derivation and complement sub-derivation of H^* as

$$H^*_j = \langle \pi_1 \dots \pi_r \rangle \quad \bar{H}^*_j = \langle \bar{\pi}_1 \dots \bar{\pi}_r \rangle$$

We then have

$$f(H^*) = f(H^*_j) + f(\bar{H}^*_j) + \gamma \tag{2}$$

where $f(\dots)$ is as defined in Eq. 1, and γ takes into account the bigram language modeling scores and the distortion scores for the transitions $\pi_1 \rightarrow \bar{\pi}_1$, $\bar{\pi_1} \rightarrow \pi_2$, $\pi_2 \rightarrow \bar{\pi}_2$, etc.

The proof is by contradiction. Define

$$H'_j = \pi'_1 \dots \pi'_r$$

and assume that $f(H^*_j) < f(H'_j)$. Now consider

$$H' = \pi'_1 \bar{\pi}_1 \pi'_2 \bar{\pi}_2 \dots \pi'_r \bar{\pi}_r$$

This is a valid derivation because the transitions $\pi'_1 \rightarrow \bar{\pi}_1$, $\bar{\pi_1} \rightarrow \pi'_2$, $\pi'_2 \rightarrow \bar{\pi}_2$ have the same distortion distances as $\pi_1 \rightarrow \bar{\pi}_1$, $\bar{\pi_1} \rightarrow \pi_2$, $\pi_2 \rightarrow \bar{\pi}_2$, hence they must satisfy the distortion limit.

We have

$$f(H') = f(H'_j) + f(\bar{H}^*_j) + \gamma \tag{3}$$

where γ has the same value as in Eq. 2. This follows because the scores for the transitions $\pi'_1 \rightarrow \bar{\pi}_1$, $\bar{\pi_1} \rightarrow \pi'_2$, $\pi'_2 \rightarrow \bar{\pi}_2$ are identical to the scores for the transitions $\pi_1 \rightarrow \bar{\pi}_1$, $\bar{\pi_1} \rightarrow \pi_2$, $\pi_2 \rightarrow \bar{\pi}_2$, because $\sigma(H^*_j) = \sigma(H'_j)$.

It follows from Eq. 2 and Eq. 3 that if $f(H'_j) > f(H^*_j)$, then $f(H') > f(H^*)$. But this contradicts the assumption that H^* is optimal. It follows that we must have $f(H'_j) \leq f(H^*_j)$. □

This lemma leads to a dynamic programming algorithm. Each dynamic programming state consists of an integer $j \in \{1 \dots n\}$ and a set of r signatures:

$$T = (j, \{\sigma_1 \dots \sigma_r\})$$

Figure 2 shows the dynamic programming algorithm. It relies on the following functions:

Inputs:

- An integer n specifying the length of the input sequence.
- A function $\delta(T)$ returning the set of valid transitions from state T.
- A function $\tau(T, \Delta)$ returning the state reached from state T by transition $\Delta \in \delta(T)$.
- A function $\texttt{valid}(T)$ returning TRUE if state T is valid, otherwise FALSE.
- A function $\texttt{score}(\Delta)$ that returns the score for any transition Δ.

Initialization:

$T_1 = (1, \{(1, \texttt{<s>}, 1, \texttt{<s>})\})$
$\alpha(T_1) = 0$
$\mathcal{T}_1 = \{T_1\}, \forall j \in \{2 \dots n\}, \mathcal{T}_j = \emptyset$
for $j = 1, \dots, n-1$
  for each state $T \in \mathcal{T}_j$
    for each $\Delta \in \delta(T)$
      $T' = \tau(T, \Delta)$
      if $\texttt{valid}(T')$ = FALSE: **continue**
      score $= \alpha(T) + \texttt{score}(\Delta)$
      Define t to be the integer such that
      $T' = (t, \{\sigma_1 \dots \sigma_r\})$
      if $T' \notin \mathcal{T}_t$
        $\mathcal{T}_t = \mathcal{T}_t \cup \{T'\}$
        $\alpha(T') =$ score
        $\texttt{bp}(T') = (\Delta)$
      else if score $> \alpha(T')$
        $\alpha(T') =$ score
        $\texttt{bp}(T') = (\Delta)$

Return: the score of the state $(n, \{(1, \texttt{<s>}, n, \texttt{</s>})\})$ in $\mathcal{T}_n$, and backpointers bp defining the transitions leading to this state.

Figure 2: The phrase-based decoding algorithm. $\alpha(T)$ is the score for state T. The $\texttt{bp}(T)$ variables are backpointers used in recovering the highest scoring sequence of transitions.

- For any state T, $\delta(T)$ is the set of outgoing transitions from state T.
- For any state T, for any transition $\Delta \in \delta(T)$, $\tau(T, \Delta)$ is the state reached by transition Δ from state T.
- For any state T, $\texttt{valid}(T)$ checks if a resulting state is valid.
- For any transition Δ, $\texttt{score}(\Delta)$ is the score for the transition.

We next give full definitions of these functions.

4.2.1 Definitions of $\delta(T)$ and $\tau(T, \Delta)$

Recall that for any state T, $\delta(T)$ returns the set of possible transitions from state T. In addition $\tau(T, \Delta)$ returns the state reached when taking transition $\Delta \in \delta(T)$.

Given the state $T = (j, \{\sigma_1 \dots \sigma_r\})$, each transition is of the form $\psi_1 \, p \, \psi_2$ where ψ_1, p and ψ_2 are defined as follows:

- p is a phrase such that $s(p) = j + 1$.
- $\psi_1 \in \{\sigma_1 \dots \sigma_r\} \cup \{\phi\}$. If $\psi_1 \neq \phi$, it must be the case that $|t(\psi_1) + 1 - s(p)| \leq d$ and $t(\psi_1) \neq n$.
- $\psi_2 \in \{\sigma_1 \dots \sigma_r\} \cup \{\phi\}$. If $\psi_2 \neq \phi$, it must be the case that $|t(p) + 1 - s(\psi_2)| \leq d$ and $s(\psi_2) \neq 1$.
- If $\psi_1 \neq \phi$ and $\psi_2 \neq \phi$, then $\psi_1 \neq \psi_2$.

Thus there are four possible types of transition from a state $T = (j, \{\sigma_1 \dots \sigma_r\})$:

Case 1: $\Delta = \phi\, p\, \phi$. In this case the phrase p is incorporated as a stand-alone phrase. The new state T' is equal to $(j', \{\sigma'_1 \dots \sigma'_{r+1}\})$ where $j' = t(p)$, where $\sigma'_i = \sigma_i$ for $i = 1 \dots r$, and $\sigma'_{r+1} = (s(p), e_1(p), t(p), e_m(p))$.

Case 2: $\Delta = \sigma_i\, p\, \phi$ for some $\sigma_i \in \{\sigma_1 \dots \sigma_r\}$. In this case the phrase p is appended to the signature σ_i. The new state $T' = \tau(T, \Delta)$ is of the form $(j', \sigma'_1 \dots \sigma'_r)$, where $j' = t(p)$, where σ_i is replaced by $(s(\sigma_i), w_s(\sigma_i), t(p), e_m(p))$, and where $\sigma'_{i'} = \sigma_{i'}$ for all $i' \neq i$.

Case 3: $\Delta = \phi\, p\, \sigma_i$ for some $\sigma_i \in \{\sigma_1 \dots \sigma_r\}$. In this case the phrase p is prepended to the signature σ_i. The new state $T' = \tau(T, \Delta)$ is of the form $(j', \sigma'_1 \dots \sigma'_r)$, where $j' = t(p)$, where σ_i is replaced by $(s(p), e_1(p), t(\sigma_i), w_t(\sigma_i))$, and where $\sigma'_{i'} = \sigma_{i'}$ for all $i' \neq i$.

Case 4: $\Delta = \sigma_i\, p\, \sigma_{i'}$ for some $\sigma_i, \sigma_{i'} \in \{\sigma_1 \dots \sigma_r\}$, with $i' \neq i$. In this case phrase p is appended to signature σ_i, and prepended to signature $\sigma_{i'}$, effectively joining the two signatures together. In this case the new state $T' = \tau(T, \Delta)$ is of the form $(j', \sigma'_1 \dots \sigma'_{r-1})$, where signatures σ_i and $\sigma_{i'}$ are replaced by a new signature $(s(\sigma_i), w_s(\sigma_i), t(\sigma_{i'}), w_t(\sigma_{i'}))$, and all other signatures are copied across from T to T'.

Figure 3 gives the dynamic programming states and transitions for the derivation H in Figure 1. For example, the sub-derivation

$$H_7 = \Big\langle \big\langle (1, 1, \text{<s>})(2, 3, \text{we must})(4, 4, \text{also}) \big\rangle, \big\langle (5, 6, \text{these criticisms})(7, 7, \text{seriously}) \big\rangle \Big\rangle$$

will be mapped to a state

$$\begin{aligned} T &= \big(7, \sigma(H_7)\big) \\ &= \big(7, \big\{(1, \text{<s>}, 4, \text{also}), (5, \text{these}, 7, \text{seriously})\big\}\big) \end{aligned}$$

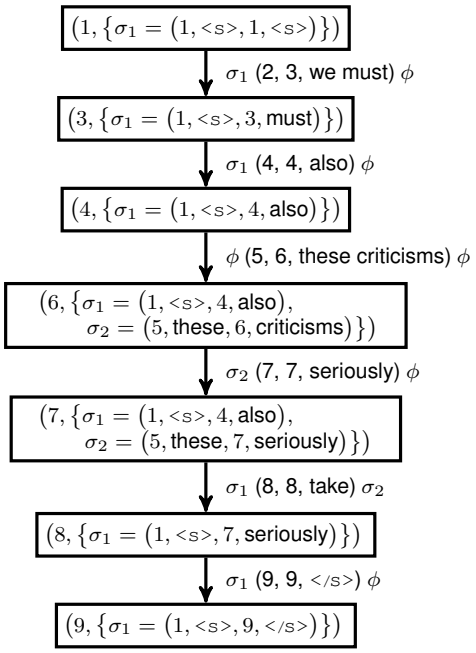

Figure 3: Dynamic programming states and the transitions from one state to another, using the same example as in Figure 1. Note that $\sigma_i = \sigma(\pi_i)$ for all $\pi_i \in H_j$.

The transition $\sigma_1(8, 8, \text{take})\, \sigma_2$ from this state leads to a new state,

$$T' = \big(8, \big\{\sigma_1 = (1, \text{<s>}, 7, \text{seriously})\big\}\big)$$

4.3 Definition of $\texttt{score}(\Delta)$

Figure 4 gives the definition of $\texttt{score}(\Delta)$, which incorporates the language model, phrase scores, and distortion penalty implied by the transition Δ.

4.4 Definition of $\texttt{valid}(T)$

Figure 5 gives the definition of $\texttt{valid}(T)$. This function checks that the start and end points of each signature are in the set of allowed start and end points given in Lemma 2.

4.5 A Bound on the Runtime of the Algorithm

We now give a bound on the algorithm's run time. This will be the product of terms N and M, where N is an upper bound on the number of states in the dynamic program, and M is an upper bound on the number of outgoing transitions from any state.

For any $j \in \{1 \dots n\}$, define $\texttt{first}(j)$ to be the set of target-language words that can begin at position j and $\texttt{last}(j)$ to be the set of target-language

Δ	Resulting phrase sequence	$\texttt{score}(\Delta)$
$\phi\, p\, \phi$	(s, e_1, t, e_m)	$\hat{w}(p)$
$\sigma_i\, p\, \phi$	$(s(\sigma_i), w_s(\sigma_i), t, e_m)$	$\hat{w}(p) + \lambda(e_1 \mid w_t(\sigma_i)) + \eta \times \lvert t(\sigma_i) + 1 - s \rvert$
$\phi\, p\, \sigma_i$	$(s, e_1, t(\sigma_i), w_t(\sigma_i))$	$\hat{w}(p) + \lambda(w_s(\sigma_i) \mid e_m) + \eta \times \lvert t + 1 - s(\sigma_i) \rvert$
$\sigma_i\, p\, \sigma_{i'}$	$(s(\sigma_i), w_s(\sigma_i), t(\sigma_{i'}), w_t(\sigma_{i'}))$	$\hat{w}(p) + \lambda(e_1 \mid w_t(\sigma_i)) + \eta \times \lvert t(\sigma_i) + 1 - s \rvert + \lambda(w_s(\sigma_{i'}) \mid e_m) + \eta \times \lvert t + 1 - s(\sigma_{i'}) \rvert$

Figure 4: Four operations that can extend a state $T = (j, \{\sigma_1 \dots \sigma_r\})$ by a phrase $p = (s, t, e_1 \dots e_m)$, and the scores incurred. We define $\hat{w}(p) = \kappa(p) + \sum_{i=2}^{m} \lambda(e_i(p) \mid e_{i-1}(p))$. The function $\hat{w}(p)$ includes the phrase translation model κ and the language model scores that can be computed using p alone. The weight η is the distortion penalty.

Function $\texttt{valid}(T)$
  Input: State $T = (j, \{\sigma_1 \dots \sigma_r\})$
  for $i = 1 \dots r$
    if $s(\sigma_i) < j - d + 2$ and $s(\sigma_i) \neq 1$
      return FALSE
    if $t(\sigma_i) < j - d$
      return FALSE
  return TRUE

Figure 5: The `valid` function.

words that can end at position j.

$$\texttt{first}(j) = \{w : \exists p = (s, t, e) \text{ s.t. } s = j, e_1 = w\}$$
$$\texttt{last}(j) = \{w : \exists p = (s, t, e) \text{ s.t. } t = j, e_m = w\}$$

In addition, define $\texttt{singles}(j)$ to be the set of phrases that translate the single word at position j:

$$\texttt{singles}(j) = \{p : s(p) = j \text{ and } t(p) = j\}$$

Next, define h to be the smallest integer such that for all j, $|\texttt{first}(j)| \leq h$, $|\texttt{last}(j)| \leq h$, and $|\texttt{singles}(j)| \leq h$. Thus h is a measure of the maximal ambiguity of any word x_j in the input.

Finally, for any position j, define $\texttt{start}(j)$ to be the set of phrases starting at position j:

$$\texttt{start}(j) = \{p : s(p) = j\}$$

and define l to be the smallest integer such that for all j, $|\texttt{start}(j)| \leq l$. Given these definitions we can state the following result:

Theorem 1. *The time complexity of the algorithm is* $O(nd!lh^{d+1})$.

To prove this we need the following definition:

Definition 4 (p-structures)**.** *For any finite set $\mathcal{A}$ of integers with $|\mathcal{A}| = k$, a p-structure is a set of r ordered pairs $\{(s_i, t_i)\}_{i=1}^{r}$ that satisfies the following properties: 1) $0 \leq r \leq k$; 2) for each $i \in \{1 \dots r\}$, $s_i \in \mathcal{A}$ and $t_i \in \mathcal{A}$ (both $s_i = t_i$ and $s_i \neq t_i$ are allowed); 3) for each $j \in \mathcal{A}$, there is at most one index $i \in \{1 \dots r\}$ such that $(s_i = j)$ or $(t_i = j)$ or $(s_i = j$ and $t_i = j)$.*

We use $g(k)$ to denote the number of unique p-structures for a set $\mathcal{A}$ with $|\mathcal{A}| = k$.

We then have the following Lemmas:

Lemma 4. *The function $g(k)$ satisfies $g(0) = 0$, $g(1) = 2$, and the following recurrence for $k \geq 2$:*

$$g(k) = 2g(k-1) + 2(n-1)g(k-2)$$

Proof. The proof is in Appendix A. □

Lemma 5. *Consider the function $h(k) = k^2 \times g(k)$. $h(k)$ is in $O((k-2)!)$.*

Proof. The proof is in Appendix B. □

We can now prove the theorem:

Proof of Theorem 1: First consider the number of states in the dynamic program. Each state is of the form $(j, \{\sigma_1 \dots \sigma_r\})$ where the set $\{(s(\sigma_i), t(\sigma_i))\}_{i=1}^{r}$ is a p-structure over the set $\{1\} \cup \{(j-d) \dots d\}$. The number of possible values for $\{(s(\sigma_i), e(\sigma_i))\}_{i=1}^{r}$ is at most $g(d+2)$. For a fixed choice of $\{(s(\sigma_i), t(\sigma_i))\}_{i=1}^{r}$ we will argue that there are at most h^{d+1} possible values for $\{(w_s(\sigma_i), w_t(\sigma_i))\}_{i=1}^{r}$. This follows because for each $k \in \{(j-d) \dots j\}$ there are at most h possible choices: if there is some i such that $s(\sigma_i) = k$, and $t(\sigma_i) \neq k$, then the associated word $w_s(\sigma_i)$ is in the set $\texttt{first}(k)$; alternatively if there is some i such that $t(\sigma_i) = k$, and $s(\sigma_i) \neq k$, then the associated word $w_t(\sigma_i)$ is in the set $\texttt{last}(k)$; alternatively if there is some i such that $s(\sigma_i) = t(\sigma_i) = k$ then the associated words $w_s(\sigma_i), w_t(\sigma_i)$ must be the first/last word of some phrase in $\texttt{singles}(k)$; alternatively there is no i such that $s(\sigma_i) = k$ or $t(\sigma_i) = k$, in which case there is no choice associated with position k in the sentence. Hence there are at most h choices associated with each position $k \in \{(j-d) \dots j\}$, giving h^{d+1} choices in total. Combining these results, and noting that there are

n choices of the variable j, implies that there are at most $ng(d+2)h^{d+1}$ states in the dynamic program.

Now consider the number of transitions from any state. A transition is of the form $\psi_1 p \psi_2$ as defined in Section 4.2.1. For a given state there are at most $(d+2)$ choices for ψ_1 and ψ_2, and l choices for p, giving at most $(d+2)^2 l$ choices in total.

Multiplying the upper bounds on the number of states and number of transitions for each state gives an upper bound on the runtime of the algorithm as $O(ng(d+2)h^{d+1}(d+2)^2 l)$. Hence by Lemma 5 the runtime is $O(nd!lh^{d+1})$ time. □

The bound $g(d+2)$ over the number of possible values for $\{(s(\sigma_i), e(\sigma_i))\}_{i=1}^r$ is somewhat loose, as the set of p-structures over $\{1\} \cup \{(j-d) \dots d\}$ includes impossible values $\{(s_i, t_i)\}_{i=1}^r$ where for example there is no i such that $s(\sigma_i) = 1$. However the bound is tight enough to give the $O(d!)$ runtime.

5 Discussion

We conclude the paper with discussion of some issues. First we describe how the dynamic programming structures we have described can be used in conjunction with beam search. Second, we give more analysis of the complexity of the widely-used decoding algorithm of Koehn et al. (2003).

5.1 Beam Search

Beam search is widely used in phrase-based decoding; it can also be applied to our dynamic programming construction. We can replace the line

for each state $T \in \mathcal{T}_j$

in the algorithm in Figure 2 with

for each state $T \in \texttt{beam}(\mathcal{T}_j)$

where `beam` is a function that returns a subset of $\mathcal{T}_j$, most often the highest scoring elements of $\mathcal{T}_j$ under some scoring criterion. A key question concerns the choice of scoring function $\gamma(T)$ used to rank states. One proposal is to define $\gamma(T) = \alpha(T) + \beta(T)$ where $\alpha(T)$ is the score used in the dynamic program, and $\beta(T) = \sum_{i:w_s(\sigma_i) \neq \texttt{<s>}} \lambda_u(w_s(\sigma_i))$. Here $\lambda_u(w)$ is the score of word w under a unigram language model. The $\beta(T)$ scores allow different states in $\mathcal{T}_j$, which have different words $w_s(\sigma_i)$ at the start of signatures, to be comparable: for example it compensates for the case where $w_s(\sigma_i)$ is a rare word, which will incur a low probability when the bigram $\langle w \; w_s(\sigma_i) \rangle$ for some word w is constructed during search.

The $\beta(T)$ values play a similar role to "future scores" in the algorithm of Koehn et al. (2003). However in the Koehn et al. (2003) algorithm, different items in the same beam can translate different subsets of the input sentence, making future-score estimation more involved. In our case all items in $\mathcal{T}_j$ translate all words $x_1 \dots x_j$ inclusive, which may make comparison of different hypotheses more straightforward.

5.2 Complexity of Decoding with Bit-string Representations

A common method for decoding phrase-based models, as described in Koehn et al. (2003), is to use beam search in conjunction with a search algorithm that 1) creates the target language string in strictly left-to-right order; 2) uses a bit string with bits $b_i \in \{0,1\}$ for $i = 1 \dots n$ representing at each point whether word i in the input has been translated. A natural question is whether the number of possible bit strings for a model with a fixed distortion limit d can grow exponentially quickly with respect to the length of the input sentence. This section gives an example that shows that this is indeed the case.

Assume that our sentence length n is such that $(n-2)/4$ is an integer. Assume as before $x_1 = \texttt{<s>}$ and $x_n = \texttt{</s>}$. For each $k \in \{0 \dots ((n-2)/4 - 1)\}$, assume we have the following phrases for the words $x_{4k+2} \dots x_{4k+5}$:

$$(4k+2, 4k+2, u_k) \quad (4k+3, 4k+3, v_k)$$
$$(4k+4, 4k+4, w_k) \quad (4k+5, 4k+5, z_k)$$
$$(4k+4, 4k+5, y_k)$$

Note that the only source of ambiguity is for each k whether we use y_k to translate the entire phrase $x_{4k+4} x_{4k+5}$, or whether we use w_k and z_k to translate x_{4k+4} and x_{4k+5} separately.

With a distortion limit $d \geq 5$, the number of possible bit strings in this example is at least $2^{(n-2)/4}$. This follows because for any setting of the variables $b_{4k+4} \in \{0,1\}$ for $k \in \{0 \dots ((n-2)/4 - 1)\}$,

there is a valid derivation $p_1 \dots p_L$ such that the prefix $p_1 \dots p_l$ where $l = 1 + (n-2)/4$ gives this bit string. Simply choose $p_1 = (1, 1, \texttt{<s>})$ and for $l' \in \{0 \dots (n-2)/4 - 1\}$ choose $p_{l'+2} = (4l' + 4, 4l' + 5, y_i)$ if $b_{4k+4} = 1$, $p_{l'+2} = (4l' + 5, 4l' + 5, z_i)$ otherwise. It can be verified that $p_1 \dots p_l$ is a valid prefix (there is a valid way to give a complete derivation from this prefix). As one example, for $n = 10$, and $b_4 = 1$ and $b_8 = 0$, a valid derivation is

$$(1, 1, \texttt{<s>})(4, 5, y_1)(9, 9, z_2)(7, 7, v_2)(3, 3, v_1)$$
$$(2, 2, u_1)(6, 6, u_2)(8, 8, w_2)(10, 10, \texttt{</s>})$$

In this case the prefix $(1, 1, \texttt{<s>})(4, 5, y_1)(9, 9, z_2)$ gives $b_4 = 1$ and $b_8 = 0$. Other values for b_4 and b_8 can be given by using $(5, 5, z_1)$ in place of $(4, 5, y_1)$, and $(8, 9, y_2)$ in place of $(9, 9, z_2)$, with the following phrases modified appropriately.

6 Conclusion

We have given a polynomial-time dynamic programming algorithm for phrase-based decoding with a fixed distortion limit. The algorithm uses a quite different representation of states from previous decoding algorithms, is easily amenable to beam search, and leads to a new perspective on phrase-based decoding. Future work should investigate the effectiveness of the algorithm in practice.

A Proof of Lemma 4

Without loss of generality assume $\mathcal{A} = \{1, 2, 3, \dots k\}$. We have $g(1) = 2$, because in this case the valid p-structures are $\{(1, 1)\}$ and $\emptyset$. To calculate $g(k)$ we can sum over four possibilities:

Case 1: There are $g(k-1)$ p-structures with $s_i = t_i = 1$ for some $i \in \{1 \dots r\}$. This follows because once $s_i = t_i = 1$ for some i, there are $g(k-1)$ possible p-structures for the integers $\{2, 3, 4 \dots k\}$.

Case 2: There are $g(k-1)$ p-structures such that $s_i \neq 1$ and $t_i \neq 1$ for all $i \in \{1 \dots r\}$. This follows because once $s_i \neq 1$ and $t_i \neq 1$ for all i, there are $g(k-1)$ possible p-structures for the integers $\{2, 3, 4 \dots k\}$.

Case 3: There are $(k-1) \times g(k-2)$ p-structures such that there is some $i \in \{1 \dots r\}$ with $s_i = 1$ and $t_i \neq 1$. This follows because for the i such that $s_i = 1$, there are $(k-1)$ choices for the value for t_i, and there are then $g(k-2)$ possible p-structures for the remaining integers in the set $\{1 \dots k\}/\{1, t_i\}$.

Case 4: There are $(k-1) \times g(k-2)$ p-structures such that there is some $i \in \{1 \dots r\}$ with $t_i = 1$ and $s_i \neq 1$. This follows because for the i such that $t_i = 1$, there are $(k-1)$ choices for the value for s_i, and there are then $g(k-2)$ possible p-structures for the remaining integers in the set $\{1 \dots k\}/\{1, s_i\}$.

Summing over these possibilities gives the following recurrence:

$$g(k) = 2g(k-1) + 2(k-1) \times g(k-2) \qquad \square$$

B Proof of Lemma 5

Recall that $h(k) = f(k) \times g(k)$ where $f(k) = k^2$.

Define k_0 to be the smallest integer such that for all $k \geq k_0$,

$$\frac{2f(k)}{f(k-1)} + \frac{2f(k)}{f(k-2)} \cdot \frac{k-1}{k-3} \leq k - 2 \qquad (4)$$

For $f(k) = k^2$ we have $k_0 = 9$.

Now choose a constant c such that for all $k \in \{1 \dots (k_0 - 1)\}$, $h(k) \leq c \times (k-2)!$. We will prove by induction that under these definitions of k_0 and c we have $h(k) \leq c(k-2)!$ for all integers k, hence $h(k)$ is in $O((k-2)!)$.

For values $k \geq k_0$, we have

$$\begin{aligned}
h(k) &= f(k)g(k) \\
&= 2f(k)g(k-1) + 2f(k)(k-1)g(k-2) \qquad (5) \\
&= \frac{2f(k)}{f(k-1)} h(k-1) + \frac{2f(k)}{f(k-2)}(k-1)h(k-2) \\
&\leq \left(\frac{2cf(k)}{f(k-1)} + \frac{2cf(k)}{f(k-2)} \cdot \frac{k-1}{k-3} \right) (k-3)! \qquad (6) \\
&\leq c(k-2)! \qquad (7)
\end{aligned}$$

Eq. 5 follows from $g(k) = 2g(k-1) + 2(k-1)g(k-2)$. Eq. 6 follows by the inductive hypothesis that $h(k-1) \leq c(k-3)!$ and $h(k-2) \leq c(k-4)!$. Eq 7 follows because Eq. 4 holds for all $k \geq k_0$. $\square$

References

Wilker Aziz, Marc Dymetman, and Lucia Specia. 2014. Exact decoding for phrase-based statistical machine translation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics.

Yin-Wen Chang and Michael Collins. 2011. Exact decoding of phrase-based translation models through Lagrangian relaxation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 26–37. Association for Computational Linguistics.

Colin Cherry, Robert C Moore, and Chris Quirk. 2012. On hierarchical re-ordering and permutation parsing for phrase-based decoding. In *Proceedings of the Seventh Workshop on Statistical Machine Translation*, pages 200–209. Association for Computational Linguistics.

John DeNero and Dan Klein. 2008. The complexity of phrase alignment problems. In *Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics on Human Language Technologies: Short Papers*, pages 25–28. Association for Computational Linguistics.

Yang Feng, Haitao Mi, Yang Liu, and Qun Liu. 2010. An efficient shift-reduce decoding algorithm for phrased-based machine translation. In *Proceedings of the 23rd International Conference on Computational Linguistics: Posters*, pages 285–293. Association for Computational Linguistics.

Michel Galley and Christopher D Manning. 2008. A simple and effective hierarchical phrase reordering model. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 848–856. Association for Computational Linguistics.

Michel Galley and Christopher D Manning. 2010. Accurate non-hierarchical phrase-based translation. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 966–974. Association for Computational Linguistics.

Kevin Knight. 1999. Decoding complexity in word-replacement translation models. *Computational Linguistics*, 25(4).

Philipp Koehn, Franz Josef Och, and Daniel Marcu. 2003. Statistical phrase-based translation. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology-Volume 1*, pages 48–54. Association for Computational Linguistics.

Philipp Koehn, Amittai Axelrod, Chris Callison-Burch, Miles Osborne, and David Talbot. 2007a. Edinburgh system description for the 2005 IWSLT speech translation evaluation. In *Proceedings of the Second Workshop on Statistical Machine Translation*, StatMT '07, pages 224–227, Stroudsburg, PA, USA. Association for Computational Linguistics.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondřej Bojar, Alexandra Constantin, and Evan Herbst. 2007b. Moses: Open source toolkit for statistical machine translation. In *Proceedings of the 45th annual meeting of the ACL on interactive poster and demonstration sessions*, pages 177–180. Association for Computational Linguistics.

Shankar Kumar and William Byrne. 2005. Local phrase reordering models for statistical machine translation. In *Proceedings of the conference on Human Language Technology and Empirical Methods in Natural Language Processing*, pages 161–168. Association for Computational Linguistics.

Eugene Leighton Lawler, Jan Karel Lenstra, Alexander Hendrik George Rinnooy Kan, and David Bernard Shmoys. 1985. *The Traveling Salesman Problem*. John Wiley & Sons Ltd.

Adam Lopez. 2009. Translation as weighted deduction. In *Proceedings of the 12th Conference of the European Chapter of the Association for Computational Linguistics*, pages 532–540. Association for Computational Linguistics.

Burkhard Monien and Ivan Hal Sudborough. 1981. Bandwidth constrained NP-complete problems. In *Proceedings of the thirteenth annual ACM symposium on Theory of computing*, pages 207–217. ACM.

Franz Josef Och and Hermann Ney. 2004. The alignment template approach to statistical machine translation. *Computational linguistics*, 30(4):417–449.

H Donald Ratliff and Arnon S Rosenthal. 1983. Order-picking in a rectangular warehouse: a solvable case of the traveling salesman problem. *Operations Research*, 31(3):507–521.

Christoph Tillmann. 2004. A unigram orientation model for statistical machine translation. In *Proceedings of HLT-NAACL 2004: Short Papers*, pages 101–104. Association for Computational Linguistics.

Dekai Wu. 1997. Stochastic inversion transduction grammars and bilingual parsing of parallel corpora. *Computational linguistics*, 23(3):377–403.

Mikhail Zaslavskiy, Marc Dymetman, and Nicola Cancedda. 2009. Phrase-based statistical machine translation as a traveling salesman problem. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 1-Volume 1*, pages 333–341. Association for Computational Linguistics.

Richard Zens and Hermann Ney. 2003. A comparative study on reordering constraints in statistical machine translation. In *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics-Volume 1*, pages 144–151. Association for Computational Linguistics.

Richard Zens, Hermann Ney, Taro Watanabe, and Eiichiro Sumita. 2004. Reordering constraints for phrase-based statistical machine translation. In *Proceedings of the 20th international conference on Computational Linguistics*, page 205. Association for Computational Linguistics.