Theory of Alignment Generators and Applications to Statistical Machine Translation

Raghavendra Udupa U. **Hemanta K. Maji**
IBM India Research Laboratory, New Delhi
{uraghave, hemantkm}@in.ibm.com

Abstract

Viterbi Alignment and *Decoding* are two fundamental search problems in Statistical Machine Translation. Both the problems are known to be $\mathbf{NP}$-hard and therefore, it is unlikely that there exists an optimal polynomial time algorithm for either of these search problems. In this paper we characterize exponentially large subspaces in the solution space of *Viterbi Alignment* and *Decoding*. Each of these subspaces admits polynomial time optimal search algorithms. We propose a local search heuristic using a neighbourhood relation on these subspaces. Experimental results show that our algorithms produce better solutions taking substantially less time than the previously known algorithms for these problems.

1 Introduction

Statistical Machine Translation (SMT) is a data driven approach to Machine Translation [Brown *et al.*, 1993], [Al-Onaizan *et al.*, 1999], [Berger *et al.*, 1996]. Two of the fundamental problems in SMT are [Brown *et al.*, 1993]:

$$\mathbf{a}^* = \underset{\mathbf{a}}{argmax}\, Pr\left(\mathbf{f}, \mathbf{a}|\mathbf{e}\right) \quad (\mathbf{ViterbiAlignment})$$

$$(\mathbf{e}^*, \mathbf{a}^*) = \underset{\mathbf{e},\mathbf{a}}{argmax}\, Pr\left(\mathbf{f}, \mathbf{a}|\mathbf{e}\right) Pr\left(\mathbf{e}\right) \quad (\mathbf{Decoding})$$

Viterbi Alignment has a lot of applications in Natural Language Processing [Wang and Waibel, 1998], [Marcu and Wong, 2002]. While there exist simple polynomial time algorithms for computing the *Viterbi Alignment* for IBM Models 1-2, only heuristics are known for Models 3-5. Recently, [Udupa and Maji, 2005a] showed that the computation of *Viterbi Alignment* is $\mathbf{NP}$-Hard for IBM Models 3-5.

Decoding algorithm is an essential component of all SMT systems [Wang and Waibel, 1997], [Tillman and Ney, 2000], [Och *et al.*, 2001], [Germann *et al.*, 2003], [Udupa *et al.*, 2004]. The problem is known to be $\mathbf{NP}$-Hard for IBM Models 1-5 when the language model is a bigram model [Knight, 1999].

Unless $\mathbf{P} = \mathbf{NP}$, it is unlikely that there exist polynomial time algorithms for either *Viterbi Alignment* or *Decoding*. Therefore, there is a lot of interest in finding fast heuristics to find acceptable solutions for the problems.

Previous approaches to the *Viterbi Alignment* problem have focussed on defining a graph over all possible alignments where the connectivity is based on local modifications. The best known algorithm for *Decoding* (both in terms of speed and translation quality) employs a greedy search algorithm. However, these approaches can look at only polynomial number of alignments in polynomial time.

In this paper, we characterize exponentially large subspaces of the solution space of *Viterbi Alignment* and *Decoding*. We propose polynomial time optimal dynamic programming algorithms to solve *Viterbi Alignment* and *Decoding* when the search space is restricted to a particular subspace. There are exponentially many such subspaces. So, we perform a local search on these subspaces under a suitable neighbourhood relation. Though, there is no polynomial bound on the number of iterations of the local search, experiments show that our algorithms procduce better solutions in significantly less time than the current algorithms.

We introduce the notion of an *alignment generator* in Section 2. There are $m!$ distinct alignment generators (where m is the length of the source language sentence). Each alignment generator ($\mathbf{g_i}$, where $1 \leq i \leq m!$) is associated with an exponentially large subspace of alignments ($\mathcal{A}_i$). The solution space of *Viterbi Alignment* and *Decoding* can be written as the union of these subspaces. We next consider the graph induced on the set of $\mathcal{A}_i$s by a neighbourhood relation and perform local search on this graph. The explicit mathematical definition of $\mathcal{A}_i$ and the neighbourhood relation are presented in Section 3. We present polynomial time optimal algorithms for *Viterbi Alignment* and *Decoding* over a particular $\mathcal{A}_i$ (Section 4 and Section 5 respectively). Experiments show that our search algorithms produce better solutions taking subtantially less time than current algorithms (Section 6).

2 Preliminaries

IBM Models 1-5 assume that there is a hidden *alignment* between the source and target sentences.

Definition 1 (Alignment) *An alignment is a function* $\mathbf{a}^{(\mathbf{m},\,\mathbf{l})} : \{1, \ldots, m\} \to \{0, 1, \ldots, l\}$.

$\{1, \ldots, m\}$ is the set of source positions and $\{0, 1, \ldots, l\}$ is the set of target positions. The target position 0 is known as the null position. We denote $\mathbf{a}^{(\mathbf{m},\,\mathbf{l})}(j)$ by $a_j^{(m,\,l)}$. The fertility of the target position i in alignment $\mathbf{a}^{(\mathbf{m},\,\mathbf{l})}$ is $\phi_i =$

$\sum_{j=1}^{m} \delta\left(a_j^{(m,\,l)}, i\right)$. If $\phi_i = 0$, then the target position i is an *infertile position*, otherwise it is a *fertile position*. Let $\{k, \dots, k+\gamma-1\} \subseteq \{1, \dots, l\}$ be the longest range of consecutive infertile positions, where $\gamma \geq 0$. We say that alignment $\mathbf{a}^{(\mathbf{m},\,\mathbf{l})}$ has an *infertility width* of γ.

Representation of Alignments: Since all source positions that are not connected to any of the target positions $1, \dots, l$ are assumed to be connected to the null position (0), we can represent an alignment, without loss of generality, as a set of tuples $\{(j, a_j)\,|a_j \neq 0\}$. Let $m' = m - \phi_0$. Thus, a sequence representation of an alignment has m' tuples. By sorting the tuples in ascending order using the second field as the key, $\mathbf{a}^{(\mathbf{m},\,\mathbf{l})}$ can now be represented as a sequence $\left(j_1, a_{j_1}^{(m,\,l)}\right) \dots \left(j_{m'}, a_{j_{m'}}^{(m,\,l)}\right)$ where the $a_{j_k}^{(m,\,l)}$s are non-decreasing with increasing k. Note that there are $(l+1)^m$ possible alignments for a given $m, l > 0$. Let $\mathbf{a}^{(\mathbf{m},\,\mathbf{l})}\big|_{(v,u)}$ be the alignment $\{1, \dots, v\} \to \{0, \dots, u\}$ where all the tuples $\left(j, a_j^{(m,\,l)}\right)$ such that $j > v$ or $a_j^{(m,\,l)} > u$ have been removed.

Definition 2 (Generator) *A generator is a bijection* $\mathbf{g}^{(\mathbf{m})} : \{1, \dots, m\} \to \{1, \dots, m\}$.

Representation of Generators: A generator $\mathbf{g}^{(\mathbf{m})}$ can be represented as a sequence of tuples $(j_1, 1) \dots (j_m, m)$. Note that there are $m!$ different possible generators for a given $m > 0$. The identity function $\mathbf{g}^{(\mathbf{m})}(j) = j$ is denoted by $\mathbf{g}_\mathbf{0}^{(\mathbf{m})}$.

Definition 3 ($\mathbf{g}^{(\mathbf{m})} \longrightarrow \mathbf{a}^{(\mathbf{m},\,\mathbf{l})}$) *An alignment* $\mathbf{a}^{(\mathbf{m},\,\mathbf{l})}$ *(with tuple sequence* $\left(j_1, a_{j_1}^{(m,\,l)}\right) \dots \left(j_{m'}, a_{j_{m'}}^{(m,\,l)}\right)$*) is said to be generated by a generator* $\mathbf{g}^{(\mathbf{m})}$ *(with tuple sequence* $(k_1, 1) \dots (k_m, m)$*) if* $j_1 \dots j_{m'}$ *is a subsequence of* $k_1 \dots k_m$.

Note that the above definition does not take into account the *infertility* width of the alignment. Hence, we refine the above definition.

Definition 4 ($\mathbf{g}^{(\mathbf{m})} \xrightarrow{\gamma} \mathbf{a}^{(\mathbf{m},\,\mathbf{l})}$) *An alignment* $\mathbf{a}^{(\mathbf{m},\,\mathbf{l})}$ *is said to be* γ*-generated by* $\mathbf{g}^{(\mathbf{m})}$*, if* $\mathbf{g}^{(\mathbf{m})} \longrightarrow \mathbf{a}^{(\mathbf{m},\,\mathbf{l})}$ *and the infertility width of* $\mathbf{a}^{(\mathbf{m},\,\mathbf{l})}$ *is at most* γ.

Definition 5 ($\mathcal{A}_{\gamma,\,l}^{\mathbf{g}^{(\mathbf{m})}}$)

$$\mathcal{A}_{\gamma,\,l}^{\mathbf{g}^{(\mathbf{m})}} = \left\{\mathbf{a}^{(\mathbf{m},\,\mathbf{l})} \mid \mathbf{g}^{(\mathbf{m})} \xrightarrow{\gamma} \mathbf{a}^{(\mathbf{m},\,\mathbf{l})}\right\}$$

Thus, for every generator $\mathbf{g}^{(\mathbf{m})}$ and integer $\gamma \geq 0$ there is an associated family of alignments $\mathcal{A}_{\gamma,\,l}^{\mathbf{g}^{(\mathbf{m})}}$ for a given $l \in \mathbb{N}$. Further, $\mathcal{A}_{\gamma,\star}^{\mathbf{g}^{(\mathbf{m})}} = \cup_l \mathcal{A}_{\gamma,\,l}^{\mathbf{g}^{(\mathbf{m})}}$.

Swap Operation: Let $\mathbf{g}^{(\mathbf{m})}$ be a generator and $(j_1, 1) \dots (j_k, k) \dots (j_{k'}, k') \dots (j_m, m)$ be its tuple sequence. The result of a SWAP operation on the kth and k'th tuples in the sequence is another tuple sequence $(j_1, 1) \dots (j_{k'}, k) \dots (j_k, k') \dots (j_m, m)$. The new tuple sequence defines a generator $\mathbf{g}'^{(\mathbf{m})}$. The relationship between any two generators is given by:

Lemma 1 *If* $\mathbf{g}^{(\mathbf{m})}$ *and* $\mathbf{g}'^{(\mathbf{m})}$ *are two generators then* $\mathbf{g}'^{(\mathbf{m})}$ *can be obtained from* $\mathbf{g}^{(\mathbf{m})}$ *by a series of swap operations.*

Proof: Follows from the properties of permutations. ■

We can apply a sequence of $Swap$ operators to modify any given generator $\mathbf{g}^{(\mathbf{m})}$ to $\mathbf{g}_\mathbf{0}^{(\mathbf{m})}$. The source position r_v $(1 \leq v \leq m)$ in $\mathbf{g}^{(\mathbf{m})}$ corresponds to the source position v in $\mathbf{g}_\mathbf{0}^{(\mathbf{m})}$. For brevity, we denote a sequence $v_i\ v_{i+1} \dots v_j$ by $\mathbf{v}_\mathbf{i}^\mathbf{j}$. The source sentence is represented by $\mathbf{f}_\mathbf{1}^\mathbf{m}$ and the target sentence is represented by $\mathbf{e}_\mathbf{1}^\mathbf{l}$.

3 Framework

We now describe the common framework for solving *Viterbi Alignment* and *Decoding*. The solution space for *Viterbi Alignment* can be written as $\cup_{\mathbf{g}^{(\mathbf{m})}} \mathcal{A}_{l,\,l}^{\mathbf{g}^{(\mathbf{m})}}$, and the solution space of *Decoding* can be approximated by $\cup_{\mathbf{g}^{(\mathbf{m})}} \mathcal{A}_{\gamma,\star}^{\mathbf{g}^{(\mathbf{m})}}$ for a large enough γ. The key idea is to find the optimal solution in an exponentially large subspace $\mathcal{A}$ corresponding to a generator ($\mathcal{A} = \mathcal{A}_{l,\,l}^{\mathbf{g}^{(\mathbf{m})}}$ and $\mathcal{A} = \mathcal{A}_{\gamma,\star}^{\mathbf{g}^{(\mathbf{m})}}$ for *Viterbi Alignment* and *Decoding* respectively). Given an optimal search algorithm for $\mathcal{A}$, we use Lemma 1 to augment the search space. Our framework is as follows

1. Choose any generator $\mathbf{g}^{(\mathbf{m})}$ as the initial generator.
2. Find the best solution for the problem in $\mathcal{A}$.
3. Pick a better generator $\mathbf{g}'^{(\mathbf{m})}$ via a swap operation on $\mathbf{g}^{(\mathbf{m})}$.
4. Repeat steps 2 and 3 until the solution cannot be improved further.

3.1 Algorithms

Viterbi Alignment

The algorithm for finding a good approximate solution to *Viterbi Alignment* employs a subroutine `Viterbi_For_Generator` to find the best solution in the exponentially large subspace defined by $\mathcal{A}_{\gamma,\,l}^{\mathbf{g}^{(\mathbf{m})}}$. The implementation of this subroutine is discussed in Section 4.

Algorithm 1 Viterbi Alignment

```
1:  $\mathbf{g}^{(\mathbf{m})} \leftarrow \mathbf{g}_\mathbf{0}^{(\mathbf{m})}$.
2:  while (true) do
3:      $\mathbf{a}^{*(\mathbf{m},\,\mathbf{l})}$ = Viterbi_For_Generator $\left(\mathbf{g}^{(\mathbf{m})}, \mathbf{f}_\mathbf{1}^\mathbf{m}, \mathbf{e}_\mathbf{1}^\mathbf{l}\right)$.
4:      Find an alignment $\mathbf{a}'^{(\mathbf{m},\,\mathbf{l})}$ with higher score by using
        swap operations on $\mathbf{a}^{*(\mathbf{m},\,\mathbf{l})}$.
5:      if $\left(\mathbf{a}'^{(\mathbf{m},\,\mathbf{l})} = \mathrm{NIL}\right)$ then
6:          break.
7:      end if
8:      $\mathbf{g}^{(\mathbf{m})} \leftarrow$ A generator of $\mathbf{a}'^{(\mathbf{m},\,\mathbf{l})}$.
9:  end while
10: Output $\mathbf{a}^{*(\mathbf{m},\,\mathbf{l})}$
```

Decoding
Our algorithm for *Decoding* employs a subroutine `Decode_For_Generator` to find the best solution in the exponentially large subspace defined by $\mathcal{A}^{\mathbf{g^{(m)}}}_{\gamma,\star}$. The implementation of this subroutine is discussed in Section 5.

Algorithm 2 Decoding

```
1: $\mathbf{g^{(m)}} \leftarrow \mathbf{g_0^{(m)}}$.
2: while (true) do
3:   $\left\langle \mathbf{e_1^{*l}}, \mathbf{a}^{*(\mathbf{m},\,\mathbf{l})} \right\rangle = \texttt{Decode\_For\_Generator}\left(\mathbf{g^{(m)}}, \mathbf{f_1^m}\right)$.
4:   Find an alignment $\mathbf{a}'^{(\mathbf{m},\,\mathbf{l})}$ with higher score by using
     swap operations on $\mathbf{a}^{*(\mathbf{m},\,\mathbf{l})}$ and $\mathbf{e_1^{*l}}$.
5:   if $\left(\mathbf{a}'^{(\mathbf{m},\,\mathbf{l})} = \texttt{NIL}\right)$ then
6:     break.
7:   end if
8:   $\mathbf{g^{(m)}} \leftarrow$ A generator of $\mathbf{a}'^{(\mathbf{m},\,\mathbf{l})}$.
9: end while
10: Output $\mathbf{e_1^{*l}}$.
```

3.2 Theory of Generators

In this section, we prove some important results on generators.

Structure Transformation Operations
We define a set of structure transformation operations on generators. The application of each of these operations on a generator modifies the structure of a generator and produces an alignment. Without loss of generality [1], we assume that the generator is the identity generator $\mathbf{g_0^{(m)}}$ and define the structural transformation operations on it. The tuple sequence for $\mathbf{g_0^{(m)}}$ is $(1,1)\ldots(m,m)$. Given an alignment $\mathbf{a^{(j-1,i)}} \in \mathcal{A}^{\mathbf{g_0^{(j-1)}}}_{\gamma,i}$ with $\phi_i > 0$, we explain the modifications introduced by each of these operators to the jth tuple.

1. SHRINK: Extend $\mathbf{a^{(j-1,i)}} \in \mathcal{A}^{\mathbf{g_0^{(j-1)}}}_{\gamma,i}$ to $\mathbf{a^{(j,i)}} \in \mathcal{A}^{\mathbf{g_0^{(j)}}}_{\gamma,i}$ such that $a_j^{(j,i)} = 0$.
2. MERGE: Extend $\mathbf{a^{(j-1,i)}} \in \mathcal{A}^{\mathbf{g_0^{(j-1)}}}_{\gamma,i}$ to $\mathbf{a^{(j,i)}} \in \mathcal{A}^{\mathbf{g_0^{(j)}}}_{\gamma,i}$ such that $a_j^{(j,i)} = i$.
3. (γ,k)-GROW $(0 \le k \le \gamma)$: Extend $\mathbf{a^{(j-1,i)}} \in \mathcal{A}^{\mathbf{g_0^{(j-1)}}}_{\gamma,i}$ to $\mathbf{a^{(j,i+k+1)}} \in \mathcal{A}^{\mathbf{g_0^{(j)}}}_{\gamma,i+k+1}$ such that $a_j^{(j,i+k+1)} = i+k+1$.

By removing the infertile positions at the end, we obtain the following result:

Lemma 2 $\mathbf{a^{(m,\,l)}} \in \mathcal{A}^{\mathbf{g_0^{(m)}}}_{\gamma,\,l}$ *and has at least one fertile position iff there exists s such that $0 \le s \le \gamma$, $\mathbf{a^{(m,\,l)}}\big|_{(m,\,l-s)} \in \mathcal{A}^{\mathbf{g_0^{(m)}}}_{\gamma,\,l-s}$ and $\phi_{l-s} > 0$.*

[1]by permuting the given generator $\mathbf{g^{(m)}}$ to the identity generator $\mathbf{g_0^{(m)}}$

Theorem 1 *If $\mathbf{a^{(m,l-s)}} \in \mathcal{A}^{\mathbf{g^{(m)}}}_{\gamma,\,l-s}$ such that $\phi_{l-s} > 0$ then it can be obtained from $\mathbf{g_0^{(m)}}$ by a series of* SHRINK, MERGE *and* (γ,k)-GROW *operations.*

Proof: WLOG we assume that the generator is $\mathbf{g_0^{(m)}}$. We provide a construction that has m phases. In the vth step, we construct $\mathbf{a^{(m,\,l-s)}}\big|_{(v,u)}$ from $\mathbf{g_0^{(v)}}$ in the following manner:

- We construct $\mathbf{a^{(m,\,l-s)}}\big|_{(v-1,u)}$ from $\mathbf{g_0^{(v-1)}}$ and employ a SHRINK operation on the tuple (v,v) if $a_v^{(m,\,l-s)} = 0$.
- We construct $\mathbf{a^{(m,\,l-s)}}\big|_{(v-1,u)}$ from $\mathbf{g_0^{(v-1)}}$ and employ a MERGE operation on the tuple (v,v) if $\phi_u > 1$.
- If $\phi_u = 1$: Let $\mathbf{a^{(m,\,l-s)}}\big|_{(v,u)}$ define $\phi_t = 0$ for $u-k \le t \le u-1$ and $\phi_{u-k-1} > 0$, where $0 \le k \le \gamma$. Construct $\mathbf{a^{(m,\,l-s)}}\big|_{v-1,u-k-1}$ from $\mathbf{g_0^{(v-1)}}$ and apply (γ,k)-GROW to the (v,v) tuple. (*Note*: If $(u-k-1) = 0$, then consider the alignment where $\{1,\ldots,v-1\}$ are all aligned to the null position)

As these are the only possibilities in the vth phase, at the end of the mth phase we get $\mathbf{a^{(m,l-s)}}$. ■
The dynamic programs for *Viterbi Alignment* and *Decoding* are based on ideas in Lemma 2 and Theorem 1.

Lemma 3 $\left|\mathcal{A}^{\mathbf{g^{(m)}}}_{\gamma,\,l}\right|$ *is given by the coefficient of x^l in*

$$\left(\frac{\alpha_\gamma - 2}{x}\right)\left[(\alpha_\gamma - 2)\left(\frac{\alpha_\gamma^m - 1}{\alpha_\gamma - 1}\right) + 1\right] \qquad (1)$$

, where $\alpha_\gamma = 2 + \sum_{k=0}^{\gamma} x^{k+1}$.

Proof: Refer [Udupa and Maji, 2005b] ■

Theorem 2 $\left|\mathcal{A}^{\mathbf{g^{(m)}}}_{\gamma,\star}\right| = (\gamma+1)\left(\frac{(\gamma+1)(\gamma+3)^m+1}{\gamma+2}\right)$

Proof: Substitute $x = 1$ in Equation 1 ■
`Decode_For_Generator` finds the optimal solution over all possible alignments in $\mathcal{A}^{\mathbf{g^{(m)}}}_{\gamma,\star}$ in polynomial time.

Theorem 3 *For a fixed l,* $\left|\mathcal{A}^{\mathbf{g^{(m)}}}_{\gamma,\,l}\right| = \Omega\left(2^m\right)$.

Proof: Refer [Udupa and Maji, 2005b] ■
`Viterbi_For_Generator` finds the optimal solution over all possible alignments in $\mathcal{A}^{\mathbf{g^{(m)}}}_{\gamma,\,l}$ in polynomial time.

Lemma 4 *Let $\mathcal{S} = \left\{\mathbf{g_1^{(m)}},\ldots,\mathbf{g_N^{(m)}}\right\}$ be a set of generators. Define span $(\mathcal{S}) = \cup_{i=1}^{N} \mathcal{A}^{\mathbf{g_i^{(m)}}}_{\gamma=l,\,l}$. Let $p(.)$ be a polynomial. If $l \ge 2$ and $N = \mathcal{O}(p(m))$, then there exists an alignment $\mathbf{a^{(m,\,l)}}$ such that $\mathbf{a^{(m,\,l)}} \notin$ span $(\mathcal{S})$.*

Proof: Refer [Udupa and Maji, 2005b] ■
This Lemma shows that by considering any polynomially large set of generators we can not obtain the optimal solutions to either *Viterbi Alignment* or *Decoding*.

4 Viterbi Alignment

We now develop the algorithm for the subroutine `Viterbi_For_Generator` for IBM Model 3. Recall that this subroutine solves the following problem:

$$\mathbf{a}^{*(\mathbf{m},\,\mathbf{l})} = \underset{\mathbf{a}^{(\mathbf{m},\,\mathbf{l})} \in \mathcal{A}^{\mathbf{g}^{(\mathbf{m})}}_{\gamma,\,l}}{argmax} \; Pr\left(\mathbf{f}_\mathbf{1}^\mathbf{m}, \mathbf{a}^{(\mathbf{m},\,\mathbf{l})} | \mathbf{e}_\mathbf{1}^\mathbf{l}\right).$$

Our algorithm has a time complexity polynomial in m and l. Without loss of generality, we assume that $\mathbf{g}^{(\mathbf{m})} = \mathbf{g}_\mathbf{0}^{(\mathbf{m})}$.
Note that $\mathbf{a}^{*(\mathbf{m},\,\mathbf{l})}\big|_{(m,\,l-s)}$ (where $0 \le s \le \gamma$ and $\phi_{l-s} > 0$) can be obtained from $\mathbf{g_0}$ using the structure transformation operations described in Section 3.2. Therefore, we build the Viterbi alignment from left to right. We scan the tuple sequence from left to right in m phases and in any phase we determine the best structure transformation operation for that phase.
We consider all possible partial alignments between $\mathbf{f}_\mathbf{1}^\mathbf{v}$ and $\mathbf{e}_\mathbf{1}^\mathbf{u}$, such that $\phi_u = \varphi > 0$ and $\phi_0 = \varphi_0$. Let $B(u, v, \varphi_0, \varphi)$ be the best partial alignment between $\mathbf{f}_\mathbf{1}^\mathbf{v}$ and $\mathbf{e}_\mathbf{1}^\mathbf{u}$ and $A(u, v, \varphi_0, \varphi)$ be its score[2]. Here, φ_0 is the number of French words aligned to e_0 in the partial alignment.
The key idea here is to compute $A(u, v, \varphi_0, \varphi)$ and $B(u, v, \varphi_0, \varphi)$ recursively using Dynamic Programming. In the vth phase (corresponding to the tuple (v, v)), we consider each of the structure transformation operation.

1. SHRINK: If $\varphi_0 > 0$, the SHRINK operation extends $B(u, v-1, \varphi_0 - 1, \phi)$ and the score of the resulting partial alignment is
$$s_1 = cA(u, v-1, \varphi_0 - 1, \varphi)$$
where $c = \mathtt{t}(f_v|\mathtt{NULL}) \frac{\mathtt{p}_1}{\mathtt{p}_0^2} \frac{(m-2\varphi_0+1)(m-2\varphi_0+2)}{(m-\varphi_0+1)\varphi_0}$.
2. MERGE: If $\varphi > 0$, the MERGE operation extends $B(u, v-1, \varphi_0, \varphi - 1)$ and the score of the resulting partial alignment is
$$s_2 = cA(u, v-1, \varphi_0, \varphi - 1)$$
where $c = \frac{\mathtt{n}(\varphi|e_u)\phi}{\mathtt{n}(\varphi-1|e_u)} \mathtt{t}(f_v|e_u)\, \mathtt{d}(r_v|u, l, m)$.
3. (γ, k)-GROW: Let
$$\varphi' = \underset{k}{argmax}\, B(u-k-1, v-1, \varphi_0, k).$$
(γ, k)-GROW extends $B(u-k-1, v-1, \varphi_0, \varphi')$ if $\varphi = 1$. If $u-k-1 > 0$, the score of the resulting partial alignment is
$$s_{3,k} = cA(u-k-1, v-1, \varphi_0, \varphi')$$
where
$$c = \mathtt{n}(1|e_u)\, \mathtt{t}(f_v|e_u)\, \mathtt{d}(r_v|u, l, m) \prod_{p=u-k}^{u-1} \mathtt{n}(0|e_p).$$
If $u-k-1 = 0$, then
$$s_{3,k} = \binom{m-\varphi_0}{\varphi_0} \mathtt{p_0}^{m-2\varphi_0} \mathtt{p_1}^{\varphi_0} \prod_{p=1}^{v-1} \mathtt{t}(f_p|e_0)$$
$$\times\, \mathtt{n}(1|e_u)\, \mathtt{t}(f_v|e_u)\, \mathtt{d}(r_v|u, l, m) \prod_{p=u-k}^{u-1} \mathtt{n}(0|e_p).$$

We choose that operation which gives the best score. As a result, $B(u, v, \varphi_0, \varphi)$ now represents the best partial alignment so far and $A(u, v, \varphi_0, \varphi)$ represents its score. We have,

$$A(u, v, \varphi_0, \varphi) = \underset{\substack{\mathbf{a}^{(\mathbf{v},\,\mathbf{u})} \in \mathcal{A}^{\mathbf{g}^{(\mathbf{v})}}_{\gamma,\,u} \\ \phi_u = \varphi,\; \phi_0 = \varphi_0}}{argmax} \; Pr\left(\mathbf{f}_\mathbf{1}^\mathbf{v}, \mathbf{a}^{(\mathbf{v},\,\mathbf{u})} | \mathbf{e}_\mathbf{1}^\mathbf{u}\right)$$

We determine $A(u, v, \varphi_0, \varphi)$ and $B(u, v, \varphi_0, \varphi)$, for all $0 \le u \le l$, $0 \le v \le m$, $0 < \varphi \le \varphi_{\max}$ and $0 \le \varphi_0 \le \frac{m}{2}$. To complete the scores of the alignments, we multiply each $A(u, m, \varphi_0, \varphi)$ by $\prod_{p=u+1}^{l} \mathtt{n}(0|e_p)$ for $l - \gamma \le u \le l$.
Let

$$(\hat{u}, \hat{\varphi}_0, \hat{\varphi}) = \underset{\substack{l-\gamma \le u \le l \\ 0 \le \varphi_0 \le \frac{m}{2} \\ 0 < \varphi \le \varphi_{\max}}}{argmax} A(u, m, \varphi_0, \varphi).$$

The Viterbi Alignment is, therefore, $B(\hat{u}, m, \hat{\varphi}_0, \hat{\varphi})$.

4.1 Time Complexity

The algorithm computes $B(u, v, \varphi_0, \varphi)$ and $A(u, v, \varphi_0, \varphi)$ and therefore, there are $\mathcal{O}\left(lm^2\varphi_{\max}\right)$ entries that need to be computed. Note that each of these entries is computed incrementally by a structure transformation operation.

- SHRINK operation requires $\mathcal{O}(1)$ time.
- MERGE operation requires $\mathcal{O}(1)$ time.
- (γ, k)-GROW $(0 \le k \le \gamma)$ operation requires $\mathcal{O}(\varphi_{\max})$ time for finding the best alignment and another $\mathcal{O}(1)$ time to update the score[3].

Each iteration takes $\mathcal{O}(\gamma\varphi_{\max})$ time. Computation of the tables A and B takes $\mathcal{O}\left(lm^2\gamma\varphi_{\max}^2\right)$ time. The final step of finding the Viterbi alignment from the table entries takes $\mathcal{O}(m\gamma\varphi_{\max})$ time[4]. Thus, the algorithm takes $\mathcal{O}\left(lm^2\gamma\varphi_{\max}\right)$ time. In practice, γ and $\varphi_{\max}$ are constants. The time complexity of our algorithm is $\mathcal{O}\left(lm^2\right)$.

4.2 Space Complexity

There are $\mathcal{O}\left(lm^2\varphi_{\max}\right)$ entries in each of the two tables. Assuming γ and $\varphi_{\max}$ to be constants, the space complexity of `Viterbi_For_Generator` is $\mathcal{O}\left(lm^2\right)$.

IBM Models 4 and 5 Extending this procedure for IBM Models 4 and 5 is easy and we leave it to the reader to figure out the modifications.

[2] A naive implementation of the table $B(*,*,*,*)$ takes $\mathcal{O}(m)$ size for each element. It can instead be implemented as a table of decisions and a pointer to the alignment that was extended to obtain it. This makes the size of each entry $\mathcal{O}(1)$.

[3] Note: The product $\prod_{p=u-k}^{u-1} \mathtt{n}(0|e_p)$ can be calculated incrementally for each k, so that at each iteration only one multiplication needs to be done. Similarly, the score $\prod_{p=1}^{v-1} \mathtt{t}(f_p|\mathtt{NULL})$ can be calculated incrementally over the loop on v.

[4] The factor $\prod_{u+1}^{l} \mathtt{n}(0|e_p)$ can be calculated incrementally over the loop of u. So, it takes $\mathcal{O}(1)$ time in each iteration.

5 Decoding

The algorithm for the subroutine `Decode_For_Generator` is similar in spirit to the algorithm for `Viterbi_For_Generator`. We provide the details for IBM Model 4 as it is the most popular choice for *Decoding*. We assume that the language model is a trigram model. $\mathcal{V}_E$ is the target language vocabulary and $\mathcal{I}_E \subset \mathcal{V}_E$ is the set of infertile words. Our algorithm employs Dynamic Programming and computes the following:

$$\left(\mathbf{e}_1^{*\mathbf{L}}, \mathbf{a}^{*(\mathbf{m},\mathbf{L})}\right) = \underset{\mathbf{a}^{(\mathbf{m},\mathbf{L})} \in \mathcal{A}_{\gamma,\star}^{\mathbf{g}^{(\mathbf{m})}},\ \mathbf{e}_1^{\mathbf{L}}}{\text{argmax}} \quad Pr\left(\mathbf{f}_1^{\mathbf{m}}, \mathbf{a}^{(\mathbf{m},\mathbf{L})} | \mathbf{e}_1^{\mathbf{L}}\right)$$

Let $\mathbf{H}_1^{\mathbf{v}}(e'', e', \varphi_0, \varphi, \rho)$ be the set of all partial hypotheses which are translations of $\mathbf{f}_1^{\mathbf{v}}$ and have $e''e'$ as their last two words (ρ is the center of cept associated with e' and $\varphi > 0$ is the fertility of e'). Observe that the scores of all partial hypothesis in $\mathbf{H}_1^{\mathbf{v}}(e'', e', \varphi_0, \varphi, \rho)$ are incremented by the same amount thereafter if the partial hypotheses are extended with the same sequence of operations. Therefore, for every $\mathbf{H}_1^{\mathbf{v}}(v, e'', e', \varphi_0, \varphi, \rho)$ it suffices to work only on the best partial hypothesis in it.

The initial partial hypothesis is $h_0(.,.,0,0,0)$ with a score of $\mathtt{p_0}^m$. Initially, the hypothesis set $\mathcal{H}$ has only one hypothesis, $h_0(.,.,0,0,0)$. We scan the tuple sequence of $\mathbf{g}_0^{(\mathbf{m})}$ left to right in m phases and build the translation left to right. In the vth phase, we extend each partial hypothesis in $\mathcal{H}$ by employing the structure transformation operations. Let $h_1^{v-1}(e'', e', \varphi_0, \varphi, \rho)$ be the partial hypothesis which is being extended.

- SHRINK: We create a new partial hypothesis $h_1^v(e'', e', \varphi_0 + 1, \varphi, \rho)$ whose score is the score of $h_1^{v-1}(e'', e', \varphi_0, \varphi, \rho)$ times

  $$\mathtt{t}(f_v|\mathtt{NULL}) \frac{\mathtt{p_1}}{\mathtt{p_0}^2} \frac{(m - 2\varphi_0 - 1)(m - 2\varphi_0)}{(m - \varphi_0)(\varphi_0 + 1)}.$$

  We put the new hypothesis in to $\mathbf{H}_1^{\mathbf{v}}(e'', e', \varphi_0 + 1, \varphi, \rho)$.

- MERGE: We create a new partial hypothesis $h_1^v(e'', e', \varphi_0, \varphi + 1, \rho)$ whose score is the score of $h_1^{v-1}(e'', e', \varphi_0, \varphi, \rho)$ times

  $$\frac{\mathtt{n}(\varphi + 1|e')}{\mathtt{n}(\varphi|e')} \mathtt{t}(f_v|e') \frac{\mathtt{d}_{\text{new}}}{\mathtt{d}_{\text{old}}}.$$

  Here, $\mathtt{d}_{\text{old}}$ is the distortion probability score of the last tableau before expansion, $\mathtt{d}_{\text{new}}$ is the distortion probability of the expanded hypothesis after r_v is inserted into the tableau and ρ' is the new center of the cept associated with e'. We put the new hypothesis into $\mathbf{H}_1^{\mathbf{v}}(e'', e', \varphi_0, \varphi + 1, \rho')$.

- (γ, k)-GROW: We choose k words $\left(e^{(1)}, \ldots, e^{(k)}\right)$ from $\mathcal{I}_E$ and one word $\left(e^{(k+1)}\right)$ from $\mathcal{V}_E$. We create a new partial hypothesis $h_1^v\left(e', e^{(1)}, \varphi_0, 1, r_v\right)$ if $k = 0$, $h_1^v\left(e^{(k)}, e^{(k+1)}, \varphi_0, 1, r_v\right)$ otherwise. The score of this hypothesis is the score of $h_1^{v-1}(e'', e', \varphi_0, \varphi, \rho)$ times

  $$\mathtt{n}(1|e_{k+1})\, \mathtt{t}(f_v|e_{k+1}) \mathtt{d}(r_v - \rho|\mathcal{A}(e'), \mathcal{B}(f_v)) \prod_{p=1}^{k} \mathtt{n}(0|e_p)$$

  We put the new hypothesis in to $\mathbf{H}_1^{\mathbf{v}}\left(e^{k+1}, e^k, \varphi_0, 1, r_v\right)$.

At the end of the phase, we retain only the best hypothesis for each $\mathbf{H}_1^{\mathbf{v}}(e'', e', \varphi_0, \varphi, \rho)$. After m phases, we append at most γ infertile words to all hypotheses in $\mathcal{H}$. The hypothesis with the best score in $\mathcal{H}$ is the output of the algorithm.

5.1 Time Complexity

At the beginning of vth phase, there are at most $\mathcal{O}\left(|\mathcal{V}_E|^2 v^2 \varphi_{\max}\right)$ distinct partial hypotheses. It takes $\mathcal{O}(1) + \mathcal{O}(\varphi_{\max}) + \mathcal{O}(|\mathcal{I}_E|^\gamma |\mathcal{V}_E| \varphi_{\max})$ time to extend a hypothesis. Therefore, the algorithm takes totally $\mathcal{O}\left(|\mathcal{V}_E|^3 |\mathcal{I}_E|^\gamma m^3 \varphi_{\max}^2\right)$ time. Now, since $|\mathcal{V}_E|$, $|\mathcal{I}_E|$, $\varphi_{\max}$ and γ are assumed to be constant, the time complexity of the algorithm is $\mathcal{O}\left(m^3\right)$.

5.2 Space Complexity

In each phase, we need $\mathcal{O}\left(|\mathcal{V}_E|^2 m^2 \varphi_{\max}\right)$ space. Assuming $|\mathcal{V}_E|$ and $\varphi_{\max}$ to be constant, the space complexity is $\mathcal{O}\left(m^2\right)$.

Extending this procedure for IBM Models 3 and 5 is easy.

6 Experiments and Results

6.1 Decoding

We trained the models using the GIZA++ tool [Och, 2001] for the translation direction French to English. For the current set of experiments, we set $\gamma = 1$ and $\varphi_{\max} = 10$. To compare our results with those of a state of the art decoder, we used the Greedy decoder [Marcu and Och, 2004]. Our test data consisted of a corpus of French sentences with sentence length in the range $06 - 10,\ 11 - 15, \ldots, 56 - 60$. Each sentence class had 100 French sentences. In Table 1, we compare the mean logscores (negative logarithm of the probability) of each of the length classes. Lower logscore indicates better probability score. We observe that the scores for our algorithm are (20%) better than the scores for Greedy algorithm. In the same table, we report the average time required for decoding for each sentence class. Our algorithm is much faster than the Greedy algorithm for most length classes. This demonstrates the power of our algorithm to search an exponentially large subspace in polynomial time.

We compare the NIST and BLEU scores of our solutions with the Greedy solutions. Our NIST scores are 14% better while our BLEU scores are 16% better than those of the Greedy solutions.

6.2 Viterbi Alignment

We compare our results with those of local search algorithm of GIZA++ [Och, 2001]. We set $\gamma = 4$ and $\varphi_{\max} = 10$. The initial generator for our algorithm is obtained from the solution of the local search algorithm.

In Table 3, we report the mean logscores of the alignments (for each length class) found by our algorithm and the local search algorithm. Our scores are about 5% better than those of the local search algorithm.

	Time (sec.)		Score	
Class	Our	Greedy	Our	Greedy
06-10	000.18	000.08	025.31	028.09
11-15	000.89	000.40	036.35	041.36
16-20	001.86	001.47	048.49	054.52
21-25	003.74	003.82	061.90	072.29
26-30	005.41	008.21	073.62	086.04
31-35	007.70	016.62	087.40	101.98
36-40	011.23	029.09	098.84	116.62
41-45	013.29	048.19	112.06	133.10
46-50	016.83	074.03	122.88	146.28
51-55	020.22	112.82	135.31	160.75
56-60	024.21	168.64	146.21	175.53

Table 1: Mean Time and Mean logscores

	BLEU		NIST	
Class	Our	Greedy	Our	Greedy
06-10	0.3285	0.2696	4.6922	4.2705
11-15	0.2942	0.2701	5.0521	4.7236
16-20	0.3335	0.2867	5.6887	5.1301
21-25	0.2816	0.2328	5.4845	4.8847
26-30	0.3261	0.3014	5.6388	5.0480
31-35	0.2666	0.2330	5.2857	4.6997
36-40	0.3347	0.2988	5.8144	5.1785
41-45	0.3303	0.2812	6.0273	5.1311
46-50	0.3620	0.3092	6.2733	5.2489
51-55	0.3686	0.3032	6.5108	5.4288
56-60	0.3627	0.3030	6.4538	5.4089

Table 2: BLEU and NIST Scores

7 Conclusions

It would be interesting to see if the theory of alignment generators can be applied successfully for other problems in NLP.

References

[Al-Onaizan *et al.*, 1999] Y. Al-Onaizan, J. Curin, M. Jahr, K. Knight, J. Lafferty, D. Melamed, F. Och, D. Purdy, N. Smith, and D. Yarowsky. Statistical machine translation: Final reprot. *JHU Workshop*, 1999.

[Berger *et al.*, 1996] A. Berger, P. Brown, S. Della Pietra, V. Della Pietra, A. Kehler, and R. Mercer. Language translation apparatus and method using context-based translation models. *United States Patent 5,510,981*, 1996.

[Brown *et al.*, 1993] P. Brown, S. Della Pietra, V. Della Pietra, and R. Mercer. The mathematics of machine translation: Parameter estimation. *Computational Linguistics*, 19(2):263–311, 1993.

[Germann *et al.*, 2003] U. Germann, M. Jahr, D. Marcu, and K. Yamada. Fast decoding and optimal decoding for machine translation. *Artificial Intelligence*, 2003.

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

Class	Our	Greedy
06-10	017.51	017.76
11-15	026.77	027.49
16-20	037.63	038.98
21-25	046.96	048.61
26-30	057.31	059.74
31-35	068.46	071.46
36-40	079.83	083.59
41-45	090.04	094.24
46-50	102.24	107.74
51-55	112.65	119.40
56-60	122.15	129.46

Table 3: Viterbi Alignment Logscores

[Marcu and Och, 2004] D. Marcu and F. Och. Greedy decoder for statistical machine translation. `http://www.isi.edu/licensed-sw/rewrite-decoder`, 2004.

[Marcu and Wong, 2002] D. Marcu and W. Wong. A phrase-based, joint probability model for statistical machine translation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 133–139. Philadelphia, 2002.

[Och *et al.*, 2001] F. Och, N. Ueffing, and H. Ney. An efficient a* search algorithm for statistical machine translation. In *Proceedings of the ACL 2001 Workshop on Data-Driven Methods in Machine Translation*, pages 55–62. Toulouse, France, 2001.

[Och, 2001] F. Och. Giza++. `http://www-i6.informatik.rwth-aachen.de/Colleagues/och/software/GIZA++.html`, 2001.

[Tillman and Ney, 2000] C. Tillman and H. Ney. Word reordering and dp-based search in statistical machine translation. In *Proceedings of the 18th COLING*, pages 850–856. Saarbrucken, Germany, 2000.

[Udupa and Maji, 2005a] R. Udupa and H. Maji. On the computational complexity of statistical machine translation. 2005.

[Udupa and Maji, 2005b] R. Udupa and H. Maji. Theory of alignment generators (extended version). `http://www.geocities.com/hemanta_maji/algo.pdf`, 2005.

[Udupa *et al.*, 2004] R. Udupa, T. Faruquie, and H. Maji. An algorithmic framework for the decoding problem in statistical machine translation. In *Proceedings of COLING*. Geneva, Switzerland, 2004.

[Wang and Waibel, 1997] Y. Wang and A. Waibel. Decoding algorithm in statistical machine translation. In *Proceedings of the 35th ACL*, pages 366–372. Madrid, Spain, 1997.

[Wang and Waibel, 1998] Y. Wang and A. Waibel. Modeling with structures in statistical machine translation. In *Proceedings of the 36th ACL*. Montreal, Canada, 1998.