Computational Complexity of Statistical Machine Translation

Raghavendra Udupa U.
IBM India Research Lab
New Delhi
India
uraghave@in.ibm.com

Hemanta K. Maji
Dept. of Computer Science
University of Illinois at Urbana-Champaign
hemanta.maji@gmail.com

Abstract

In this paper we study a set of problems that are of considerable importance to Statistical Machine Translation (SMT) but which have not been addressed satisfactorily by the SMT research community. Over the last decade, a variety of SMT algorithms have been built and empirically tested whereas little is known about the computational complexity of some of the fundamental problems of SMT. Our work aims at providing useful insights into the the computational complexity of those problems. We prove that while IBM Models 1-2 are conceptually and computationally simple, computations involving the higher (and more useful) models are hard. Since it is unlikely that there exists a polynomial time solution for any of these hard problems (unless $\mathbf{P} = \mathbf{NP}$ and $\mathbf{P}^{\#\mathbf{P}} = \mathbf{P}$), our results highlight and justify the need for developing polynomial time approximations for these computations. We also discuss some practical ways of dealing with complexity.

1 Introduction

Statistical Machine Translation is a data driven machine translation technique which uses probabilistic models of natural language for automatic translation (Brown et al., 1993), (Al-Onaizan et al., 1999). The parameters of the models are estimated by iterative maximum-likelihood training on a large parallel corpus of natural language texts using the EM algorithm (Brown et al., 1993). The models are then used to *decode*, i.e. translate texts from the source language to the target language [1] (Tillman, 2001), (Wang, 1997), (Germann et al., 2003), (Udupa et al., 2004). The models are independent of the language pair and therefore, can be used to build a translation system for any language pair as long as a parallel corpus of texts is available for training. Increasingly, parallel corpora are becoming available for many language pairs and SMT systems have been built for French-English, German-English, Arabic-English, Chinese-English, Hindi-English and other language pairs (Brown et al., 1993), (Al-Onaizan et al., 1999), (Udupa, 2004).

In SMT, every English sentence $\mathbf{e}$ is considered as a translation of a given French sentence $\mathbf{f}$ with probability $Pr(\mathbf{f}|\mathbf{e})$. Therefore, the problem of translating $\mathbf{f}$ can be viewed as a problem of finding the most probable translation of $\mathbf{f}$:

$$\mathbf{e}^* = \underset{\mathbf{e}}{\operatorname{argmax}}\, Pr(\mathbf{e}|\mathbf{f}) = \underset{\mathbf{e}}{\operatorname{argmax}}\, Pr(\mathbf{f}|\mathbf{e})P(\mathbf{e}). \tag{1}$$

The probability distributions $Pr(\mathbf{f}|\mathbf{e})$ and $Pr(\mathbf{e})$ are known as *translation model* and *language model* respectively. In the classic work on SMT, Brown and his colleagues at IBM introduced the notion of *alignment* between a sentence $\mathbf{f}$ and its translation $\mathbf{e}$ and used it in the development of translation models (Brown et al., 1993). An alignment between $\mathbf{f} = f_1 \dots f_m$ and $\mathbf{e} = e_1 \dots e_l$ is a many-to-one mapping $\mathbf{a} : \{1, \dots, m\} \rightarrow \{0, \dots, l\}$. Thus, an alignment $\mathbf{a}$ between $\mathbf{f}$ and $\mathbf{e}$ associates the french word f_j to the English word e_{a_j} [2]. The number of words of $\mathbf{f}$ mapped to e_i by $\mathbf{a}$ is called the *fertility* of e_i and is denoted by ϕ_i. Since $Pr(\mathbf{f}|\mathbf{e}) = \sum_{\mathbf{a}} Pr(\mathbf{f}, \mathbf{a}|\mathbf{e})$, equation 1 can

[1] In this paper, we use French and English as the prototypical examples of source and target languages respectively.

[2] e_0 is a special word called the *null* word and is used to account for those words in $\mathbf{f}$ that are not connected by $\mathbf{a}$ to any of the words of $\mathbf{e}$.

be rewritten as follows:

$$\mathbf{e}^* = \underset{\mathbf{e}}{\text{argmax}} \sum_{\mathbf{a}} Pr(\mathbf{f}, \mathbf{a}|\mathbf{e}) Pr(\mathbf{e}). \quad (2)$$

Brown and his colleagues developed a series of 5 translation models which have become to be known in the field of machine translation as *IBM models*. For a detailed introduction to IBM translation models, please see (Brown et al., 1993). In practice, models 3-5 are known to give good results and models 1-2 are used to seed the EM iterations of the higher models. IBM model 3 is the prototypical translation model and it models $Pr(\mathbf{f}, \mathbf{a}|\mathbf{e})$ as follows:

$$P(\mathbf{f}, \mathbf{a}|\mathbf{e}) \equiv \mathtt{n}\left(\phi_0 | \sum_{i=1}^{l} \phi_i\right) \prod_{i=1}^{l} \mathtt{n}(\phi_i|e_i)\, \phi_i! \times \prod_{j=1}^{m} \mathtt{t}\left(f_j|e_{a_j}\right) \times \prod_{j:\, a_j \neq 0} \mathtt{d}(j|i, m, l)$$

Table 1: IBM Model 3

Here, $n(\phi|e)$ is the *fertility model*, $t(f|e)$ is the *lexicon model* and $d(j|i, m, l)$ is the *distortion model*.

The computational tasks involving IBM Models are the following:

- **Viterbi Alignment**
  Given the model parameters and a sentence pair $(\mathbf{f}, \mathbf{e})$, determine the most probable alignment between $\mathbf{f}$ and $\mathbf{e}$.
  $$\mathbf{a}^* = \underset{\mathbf{a}}{\text{argmax}}\ P(\mathbf{f}, \mathbf{a}|\mathbf{e})$$
- **Expectation Evaluation**
  This forms the core of model training via the EM algorithm. Please see Section 2.3 for a description of the computational task involved in the EM iterations.
- **Conditional Probability**
  Given the model parameters and a sentence pair $(\mathbf{f}, \mathbf{e})$, compute $P(\mathbf{f}|\mathbf{e})$.
  $$P(\mathbf{f}|\mathbf{e}) = \sum_{\mathbf{a}} P(\mathbf{f}, \mathbf{a}|\mathbf{e})$$
- **Exact Decoding**
  Given the model parameters and a sentence $\mathbf{f}$, determine the most probable translation of $\mathbf{f}$.
  $$\mathbf{e}^* = \underset{\mathbf{e}}{\text{argmax}} \sum_{\mathbf{a}} P(\mathbf{f}, \mathbf{a}|\mathbf{e})\ P(\mathbf{e})$$
- **Relaxed Decoding**
  Given the model parameters and a sentence $\mathbf{f}$, determine the most probable translation and alignment pair for $\mathbf{f}$.
  $$(\mathbf{e}^*, \mathbf{a}^*) = \underset{(\mathbf{e},\mathbf{a})}{\text{argmax}}\ P(\mathbf{f}, \mathbf{a}|\mathbf{e})\ P(\mathbf{e})$$

Viterbi Alignment computation finds applications not only in SMT but also in other areas of Natural Language Processing (Wang, 1998), (Marcu, 2002). *Expectation Evaluation* is the soul of parameter estimation (Brown et al., 1993), (Al-Onaizan et al., 1999). *Conditional Probability* computation is important in experimentally studying the concentration of the probability mass around the Viterbi alignment, i.e. in determining the goodness of the Viterbi alignment in comparison to the rest of the alignments. *Decoding* is an integral component of all SMT systems (Wang, 1997), (Tillman, 2000), (Och et al., 2001), (Germann et al., 2003), (Udupa et al., 2004). *Exact Decoding* is the original decoding problem as defined in (Brown et al., 1993) and *Relaxed Decoding* is the relaxation of the decoding problem typically used in practice.

While several heuristics have been developed by practitioners of SMT for the computational tasks involving IBM models, not much is known about the computational complexity of these tasks. In their seminal paper on SMT, Brown and his colleagues highlighted the problems we face as we go from IBM Models 1-2 to 3-5(Brown et al., 1993) [3]:

"As we progress from Model 1 to Model 5, *evaluating the expectations* that gives us counts becomes *increasingly difficult*. In Models 3 and 4, we must be content with approximate EM iterations because it is *not feasible* to carry out sums over all possible alignments for these models. In practice, we are *never sure* that we have found the *Viterbi alignment*".

However, neither their work nor the subsequent research in SMT studied the computational complexity of these fundamental problems with the exception of the *Decoding* problem. In (Knight, 1999) it was proved that the *Exact Decoding* problem is **NP**-Hard when the language model is a bigram model.

Our results may be summarized as follows:

[3] The emphasis is ours.

1. **Viterbi Alignment** computation is $\mathbf{NP}$-Hard for IBM Models 3, 4, and 5.

2. **Expectation Evaluation** in EM Iterations is $\#\mathbf{P}$-Complete for IBM Models 3, 4, and 5.

3. **Conditional Probability** computation is $\#\mathbf{P}$-Complete for IBM Models 3, 4, and 5.

4. **Exact Decoding** is $\#\mathbf{P}$-Hard for IBM Models 3, 4, and 5.

5. **Relaxed Decoding** is $\mathbf{NP}$-Hard for IBM Models 3, 4, and 5.

Note that our results for *decoding* are sharper than that of (Knight, 1999). Firstly, we show that *Exact Decoding* is $\#\mathbf{P}$-Hard for IBM Models 3-5 and not just $\mathbf{NP}$-Hard. Secondly, we show that *Relaxed Decoding* is $\mathbf{NP}$-Hard for Models 3-5 even when the language model is a uniform distribution.

The rest of the paper is organized as follows. We formally define all the problems discussed in the paper (Section 2). Next, we take up each of the problems discussed in this section and derive the stated result for them (Section 3). After this, we discuss the implications of our results (Section 4) and suggest future directions (Section 5).

2 Problem Definition

Consider the functions $f, g : \Sigma^* \rightarrow \{0,1\}$. We say that $g \leq_p^m f$ (g is polynomial-time many-one reducible to f), if there exists a polynomial time reduction $r(.)$ such that $g(x) = f(r(x))$ for all input instances $x \in \Sigma^*$. This means that given a machine to evaluate $f(.)$ in polynomial time, there exists a machine that can evaluate $g(.)$ in polynomial time. We say a function f is $\mathbf{NP}$-Hard, if all functions in $\mathbf{NP}$ are polynomial-time many-one reducible to f. In addition, if $f \in \mathbf{NP}$, then we say that f is $\mathbf{NP}$-Complete.

Also relevant to our work are counting functions that answer queries such as "how many computation paths exist for accepting a particular instance of input?" Let w be a witness for the acceptance of an input instance x and $\chi(x,w)$ be a polynomial time witness checking function (i.e. $\chi(x,w) \in \mathbf{P}$). The function $f : \Sigma^* \rightarrow \mathbb{N}$ such that

$$f(x) = \sum_{\substack{w \in \Sigma^* \\ |w| \leq p(|x|)}} \chi(x,w)$$

lies in the class $\#\mathbf{P}$, where $p(.)$ is a polynomial.

Given functions $f, g : \Sigma^* \rightarrow \mathbb{N}$, we say that g is polynomial-time Turing reducible to f (i.e. $g \leq_T f$) if there is a Turing machine with an oracle for f that computes g in time polynomial in the size of the input. Similarly, we say that f is $\#\mathbf{P}$-Hard, if every function in $\#\mathbf{P}$ can be polynomial time Turing reduced to f. If f is $\#\mathbf{P}$-Hard and is in $\#\mathbf{P}$, then we say that f is $\#\mathbf{P}$-Complete.

2.1 Viterbi Alignment Computation

VITERBI-3 is defined as follows. Given the parameters of IBM Model 3 and a sentence pair $(\mathbf{f}, \mathbf{e})$, compute the most probable alignment $\mathbf{a}^*$ betwen $\mathbf{f}$ and $\mathbf{e}$:

$$\mathbf{a}^* = \operatorname*{argmax}_{\mathbf{a}} \; P(\mathbf{f}, \mathbf{a}|\mathbf{e}).$$

2.2 Conditional Probability Computation

PROBABILITY-3 is defined as follows. Given the parameters of IBM Model 3, and a sentence pair $(\mathbf{f}, \mathbf{e})$, compute the probability $P(\mathbf{f}|\mathbf{e}) = \sum_{\mathbf{a}} P(\mathbf{f}, \mathbf{a}|\mathbf{e})$.

2.3 Expectation Evaluation in EM Iterations

(f,e)-COUNT-3, (ϕ,e)-COUNT-3, (j,i,m,l)-COUNT-3, 0-COUNT-3, and 1-COUNT-3 are defined respectively as follows. Given the parameters of IBM Model 3, and a sentence pair $(\mathbf{f}, \mathbf{e})$, compute the following [4]:

$$c(f|e;\mathbf{f},\mathbf{e}) = \sum_{\mathbf{a}} P(\mathbf{a}|\mathbf{f},\mathbf{e}) \sum_{j} \delta(f, f_j)\delta(e, e_{a_j}),$$

$$c(\phi|e;\mathbf{f},\mathbf{e}) = \sum_{\mathbf{a}} P(\mathbf{a}|\mathbf{f},\mathbf{e}) \sum_{i} \delta(\phi, \phi_i)\delta(e, e_i),$$

$$c(j|i,m,l;\mathbf{f},\mathbf{e}) = \sum_{\mathbf{a}} P(\mathbf{a}|\mathbf{f},\mathbf{e})\delta(i, a_j),$$

$$c(0;\mathbf{f},\mathbf{e}) = \sum_{\mathbf{a}} P(\mathbf{a}|\mathbf{f},\mathbf{e})(m - 2\phi_0), \text{ and}$$

$$c(1;\mathbf{f},\mathbf{e}) = \sum_{\mathbf{a}} P(\mathbf{a}|\mathbf{f},\mathbf{e})\phi_0.$$

2.4 Decoding

E-DECODING-3 and R-DECODING-3 are defined as follows. Given the parameters of IBM Model 3,

[4] As the counts are normalized in the EM iteration, we can replace $P(\mathbf{a}|\mathbf{f},\mathbf{e})$ by $P(\mathbf{f},\mathbf{a}|\mathbf{e})$ in the *Expectation Evaluation* tasks.

and a sentence $\mathbf{f}$, compute its most probable translation according to the following equations respectively.

$$\mathbf{e}^* = \operatorname*{argmax}_{\mathbf{e}} \sum_{\mathbf{a}} P(\mathbf{f}, \mathbf{a}|\mathbf{e})\ P(\mathbf{e})$$
$$(\mathbf{e}^*, \mathbf{a}^*) = \operatorname*{argmax}_{(\mathbf{e},\mathbf{a})} P(\mathbf{f}, \mathbf{a}|\mathbf{e})\ P(\mathbf{e}).$$

2.5 SETCOVER

Given a collection of sets $\mathcal{C} = \{\mathbf{S}_1, \ldots, \mathbf{S}_l\}$ and a set $\mathbf{X} \subseteq \cup_{i=1}^{l} \mathbf{S}_i$, find the minimum cardinality subset $\mathcal{C}'$ of $\mathcal{C}$ such that every element in $\mathbf{X}$ belongs to at least one member of $\mathcal{C}'$.

SETCOVER is a well-known $\mathbf{NP}$-Complete problem. If SETCOVER $\leq_p^m f$, then f is $\mathbf{NP}$-Hard.

2.6 PERMANENT

Given a matrix $\mathcal{M} = [\mathcal{M}_{j,i}]_{n\times n}$ whose entries are either 0 or 1, compute the following:
$\texttt{perm}(\mathcal{M}) = \sum_{\pi} \prod_{j=1}^{n} \mathcal{M}_{j,\pi_j}$ where π is a permutation of $1, \ldots, n$.

This problem is the same as that of counting the number of perfect matchings in a bipartite graph and is known to be $\#\mathbf{P}$-Complete (**?**). If PERMANENT $\leq_T f$, then f is $\#\mathbf{P}$-Hard.

2.7 COMPAREPERMANENTS

Given two matrices $\mathcal{A} = [\mathcal{A}_{j,i}]_{n\times n}$ and $\mathcal{B} = [\mathcal{B}_{j,i}]_{n\times n}$ whose entries are either 0 or 1, determine which of them has a larger permanent. PERMANENT is known to be Turing reducible to COMPAREPERMANENTS (Jerrum, 2005) and therefore, if COMPAREPERMANENTS $\leq_T f$, then f is $\#\mathbf{P}$-Hard.

3 Main Results

In this section, we present the main reductions for the problems with Model 3 as the translation model. Our reductions can be easily carried over to Models $4-5$ with minor modifications. In order to keep the presentation of the main ideas simple, we let the lexicon, distortion, and fertility models to be any non-negative functions and not just probability distributions in our reductions.

3.1 VITERBI-3

We show that VITERBI-3 is $\mathbf{NP}$-Hard.

Lemma 1 SETCOVER $\leq_p^m$ VITERBI-3.

Proof: We give a polynomial time many-one reduction from SETCOVER to VITERBI-3. Given a collection of sets $\mathcal{C} = \{\mathbf{S}_1, \ldots, \mathbf{S}_l\}$ and a set $\mathbf{X} \subseteq \cup_{i=1}^{l} \mathbf{S}_i$, we create an instance of VITERBI-3 as follows:
For each set $\mathbf{S}_i \in \mathcal{C}$, we create a word e_i ($1 \leq i \leq l$). Similarly, for each element $v_j \in \mathbf{X}$ we create a word f_j ($1 \leq j \leq |\mathbf{X}| = m$). We set the model parameters as follows:

$$\mathtt{t}(f_j|e_i) = \begin{cases} 1 & \text{if } v_j \in \mathbf{S}_i \\ 0 & \text{otherwise} \end{cases}$$
$$\mathtt{n}(\phi|e) = \begin{cases} \frac{1}{2\phi!} & \text{if } \phi \neq 0 \\ 1 & \text{if } \phi = 0 \end{cases}$$
$$\mathtt{d}(j|i,m,l) = 1.$$

Now consider the sentences $\mathbf{e} = e_1 \ldots e_l$ and $\mathbf{f} = f_1 \ldots f_m$.

$$P(\mathbf{f}, \mathbf{a}|\mathbf{e}) = \mathtt{n}\left(\phi_0|\sum_{i=1}^{l}\phi_i\right)\prod_{i=1}^{l}\mathtt{n}(\phi_i|e_i)\,\phi_i! \times \prod_{j=1}^{m}\mathtt{t}(f_j|e_{a_j}) \prod_{j:\,a_j\neq 0}\mathtt{d}(j|i,m,l)$$
$$= \prod_{i=1}^{l}\frac{1}{2^{1-\delta(\phi_i,0)}}$$

We can construct a cover for $\mathbf{X}$ from the output of VITERBI-3 by defining $\mathcal{C}' = \{\mathbf{S}_i|\phi_i > 0\}$. We note that $P(\mathbf{f}, \mathbf{a}|\mathbf{e}) = \prod_{i=1}^{n}\frac{1}{2^{1-\delta(\phi_i,0)}} = 2^{-|\mathcal{C}'|}$. Therefore, Viterbi alignment results in the minimum cover for $\mathbf{X}$. ■

3.2 PROBABILITY-3

We show that PROBABILITY-3 is $\#\mathbf{P}$-Complete. We begin by proving the following:

Lemma 2 PERMANENT $\leq_T$ PROBABILITY-3.

Proof: Given a $0,1$-matrix $\mathcal{M} = [\mathcal{M}_{j,\,i}]_{n\times n}$, we define $\mathbf{f} = f_1 \ldots f_n$ and $\mathbf{e} = e_1 \ldots e_n$ where each e_i and f_j is distinct and set the Model 3 parameters as follows:

$$\mathtt{t}(f_j|e_i) = \begin{cases} 1 & \text{if } \mathcal{M}_{j,i} = 1 \\ 0 & \text{otherwise} \end{cases}$$
$$\mathtt{n}(\phi|e) = \begin{cases} 1 & \text{if } \phi = 1 \\ 0 & \text{otherwise} \end{cases}$$
$$\mathtt{d}(j|i,n,n) = 1.$$

Clearly, with the above parameter setting, $P(\mathbf{f}, \mathbf{a}|\mathbf{e}) = \prod_{j=1}^{n} \mathcal{M}_{j,\, a_j}$ if $\mathbf{a}$ is a permutation and 0 otherwise. Therefore,

$$P(\mathbf{f}|\mathbf{e}) = \sum_{\mathbf{a}} P(\mathbf{f}, \mathbf{a}|\mathbf{e}) = \sum_{\mathbf{a} \text{ is a permutation}} \prod_{j=1}^{n} \mathcal{M}_{j,\, a_j} = \mathtt{perm}(\mathcal{M})$$

Thus, by construction, PROBABILITY-3 computes $\mathtt{perm}(\mathcal{M})$. Besides, the construction conserves the number of witnesses. Hence, PERMANENT $\leq_T$ PROBABILITY-3. ■

We now prove that

Lemma 3 PROBABILITY-3 *is in* $\#\mathbf{P}$.

Proof: Let $(\mathbf{f}, \mathbf{e})$ be the input to PROBABILITY-3. Let m and l be the lengths of $\mathbf{f}$ and $\mathbf{e}$ respectively. With each alignment $\mathbf{a} = (a_1, a_2, \ldots, a_m)$ we associate a unique number $n_{\mathbf{a}} = a_1 a_2 \ldots a_m$ in base $l+1$. Clearly, $0 \leq n_{\mathbf{a}} \leq (l+1)^m - 1$. Let $\mathbf{w}$ be the binary encoding of $n_{\mathbf{a}}$. Conversely, with every binary string $\mathbf{w}$ we can associate an alignment $\mathbf{a}$ if the value of $\mathbf{w}$ is in the range $0, \ldots, (l+1)^m - 1$. It requires $\mathcal{O}(m \log(l+1))$ bits to encode an alignment. Thus, given an alignment we can compute its encoding and given the encoding we can compute the corresponding alignment in time polynomial in l and m. Similarly, given an encoding we can compute $P(\mathbf{f}, \mathbf{a}|\mathbf{e})$ in time polynomial in l and m. Now, if $p(.)$ is a polynomial, then function

$$f(\mathbf{f}, \mathbf{e}) = \sum_{\substack{w \in \{0,1\}^* \\ |\mathbf{w}| \leq p(|\langle \mathbf{f},\, \mathbf{e} \rangle|)}} P(\mathbf{f}, \mathbf{a}|\mathbf{e})$$

is in $\#\mathbf{P}$. Choose $p(x) = \lceil x \log_2(x+1) \rceil$. Clearly, all alignments can be encoded using at most $p(|(\mathbf{f},\, \mathbf{e})|)$ bits. Therefore, if $(\mathbf{f}, \mathbf{e})$ computes $P(\mathbf{f}|\mathbf{e})$ and hence, PROBABILITY-3 is in $\#\mathbf{P}$. ■

It follows immediately from Lemma 2 and Lemma 3 that

Theorem 1 PROBABILITY-3 *is* $\#\mathbf{P}$*-Complete.*

3.3 (f, e)-COUNT-3

Lemma 4 PERMANENT $\leq_T$ (f, e)-COUNT-3.

Proof: The proof is similar to that of Lemma 2. Let $\mathbf{f} = f_1\ f_2\ \ldots\ f_n\ \hat{f}$ and $\mathbf{e} = e_1\ e_2\ \ldots\ e_n\ \hat{e}$. We set the translation model parameters as follows:

$$\mathtt{t}(f|e) = \begin{cases} 1 & \text{if } f = f_j, e = e_i \text{ and } \mathcal{M}_{j,i} = 1 \\ 1 & \text{if } f = \hat{f} \text{ and } e = \hat{e} \\ 0 & \text{otherwise.} \end{cases}$$

The rest of the parameters are set as in Lemma 2. Let $\mathbf{A}$ be the set of alignments $\mathbf{a}$, such that $a_{n+1} = n+1$ and $\mathbf{a}_1^n$ is a permutation of $1, 2, \ldots, n$. Now,

$$\begin{aligned} c\left(\hat{f}|\hat{e}; \mathbf{f}, \mathbf{e}\right) &= \sum_{\mathbf{a}} P(\mathbf{f}, \mathbf{a}|\mathbf{e}) \sum_{j=1}^{n+1} \delta(\hat{f}, f_j)\delta(\hat{e}, e_{a_j}) \\ &= \sum_{\mathbf{a} \in \mathbf{A}} P(\mathbf{f}, \mathbf{a}|\mathbf{e}) \sum_{j=1}^{n+1} \delta(\hat{f}, f_j)\delta(\hat{e}, e_{a_j}) \\ &= \sum_{\mathbf{a} \in \mathbf{A}} P(\mathbf{f}, \mathbf{a}|\mathbf{e}) \\ &= \sum_{\mathbf{a} \in \mathbf{A}} \prod_{j=1}^{n} \mathcal{M}_{j,\, a_j} = \mathtt{perm}(\mathcal{M}). \end{aligned}$$

Therefore, PERMANENT $\leq_T$ COUNT-3. ■

Lemma 5 (f, e)-COUNT-3 *is in* $\#\mathbf{P}$.

Proof: The proof is essentially the same as that of Lemma 3. Note that given an encoding $\mathbf{w}$, $P(\mathbf{f}, \mathbf{a}|\mathbf{e}) \sum_{j=1}^{m} \delta(f_j, f)\, \delta\left(e_{a_j}, e\right)$ can be evaluated in time polynomial in $|(\mathbf{f}, \mathbf{e})|$. ■

Hence, from Lemma 4 and Lemma 5, it follows that

Theorem 2 (f, e)-COUNT-3 *is* $\#\mathbf{P}$*-Complete.*

3.4 (j, i, m, l)-COUNT-3

Lemma 6 PERMANENT $\leq_T$ (j, i, m, l)-COUNT-3.

Proof: We proceed as in the proof of Lemma 4 with some modifications. Let $\mathbf{e} = e_1 \ldots e_{i-1} \hat{e} e_i \ldots e_n$ and $\mathbf{f} = f_1 \ldots f_{j-1} \hat{f} f_j \ldots f_n$. The parameters are set as in Lemma 4. Let $\mathbf{A}$ be the set of alignments, $\mathbf{a}$, such that $\mathbf{a}$ is a permutation of $1, 2, \ldots, (n+1)$ and $a_j = i$. Observe that $P(\mathbf{f}, \mathbf{a}|\mathbf{e})$ is non-zero only for the alignments in $\mathbf{A}$. It follows immediately that with these parameter settings, $c(j|i, n, n; \mathbf{f}, \mathbf{e}) = \mathtt{perm}(\mathcal{M})$. ■

Lemma 7 (j, i, m, l)-COUNT-3 *is in* $\#\mathbf{P}$.

Proof: Similar to the proof of Lemma 5. ■

Theorem 3 (j, i, m, l)-COUNT-3 *is* $\#\mathbf{P}$*-Complete.*

3.5 (ϕ, e)-COUNT-3

Lemma 8 PERMANENT $\leq_T$ (ϕ, e)-COUNT-3.

Proof: Let $\mathbf{e} = e_1 \dots e_n \hat{e}$ and $\mathbf{f} = f_1 \dots f_n \overbrace{\hat{f} \dots \hat{f}}^{k}$. Let $\mathbf{A}$ be the set of alignments for which $\mathbf{a}_1^n$ is a permutation of $1, 2, \dots, n$ and $\mathbf{a}_{n+1}^{n+k} = \overbrace{(n+1) \dots (n+1)}^{k}$. We set

$$\mathtt{n}(\phi|e) = \begin{cases} 1 & \text{if } \phi = 1 \text{ and } e \neq \hat{e} \\ 1 & \text{if } \phi = k \text{ and } e = \hat{e} \\ 0 & \text{otherwise.} \end{cases}$$

The rest of the parameters are set as in Lemma 4. Note that $P(\mathbf{f}, \mathbf{a}|\mathbf{e})$ is non-zero only for the alignments in $\mathbf{A}$. It follows immediately that with these parameter settings, $c(k|\hat{e}; \mathbf{f}, \mathbf{e}) = \mathtt{perm}(\mathcal{M})$. ■

Lemma 9 (ϕ, e)-COUNT-3 *is in* $\#\mathbf{P}$.

Proof: Similar to the proof of Lemma 5. ■

Theorem 4 (ϕ, e)-COUNT-3 *is* $\#\mathbf{P}$*-Complete.*

3.6 0-COUNT-3

Lemma 10 PERMANENT $\leq_T$ 0-COUNT-3.

Proof: Let $\mathbf{e} = e_1 \dots e_n$ and $\mathbf{f} = f_1 \dots f_n \hat{f}$. Let $\mathbf{A}$ be the set of alignments, $\mathbf{a}$, such that $\mathbf{a}_1^n$ is a permutation of $1, \dots, n$ and $a_{n+1} = 0$. We set

$$\mathtt{t}(f|e) = \begin{cases} 1 & \text{if } f = f_j, e = e_i \text{ and } \mathcal{M}_{j,\,i} = 1 \\ 1 & \text{if } f = \hat{f} \text{ and } e = \mathtt{NULL} \\ 0 & \text{otherwise.} \end{cases}$$

The rest of the parameters are set as in Lemma 4. It is easy to see that with these settings, $\frac{c(0;\mathbf{f},\mathbf{e})}{(n-2)} = \mathtt{perm}(\mathcal{M})$. ■

Lemma 11 0-COUNT-3 *is in* $\#\mathbf{P}$.

Proof: Similar to the proof of Lemma 5. ■

Theorem 5 0-COUNT-3 *is* $\#\mathbf{P}$*-Complete.*

3.7 1-COUNT-3

Lemma 12 PERMANENT $\leq_T$ 1-COUNT-3.

Proof: We set the parameters as in Lemma 10. It follows immediately that $c(1; \mathbf{f}, \mathbf{e}) = \mathtt{perm}(\mathcal{M})$. ■

Lemma 13 1-COUNT-3 *is in* $\#\mathbf{P}$.

Proof: Similar to the proof of Lemma 5. ■

Theorem 6 1-COUNT-3 *is* $\#\mathbf{P}$*-Complete.*

3.8 E-DECODING-3

Lemma 14 COMPAREPERMANENTS $\leq_T$ E-DECODING-3

Proof: Let $\mathcal{M}$ and $\mathcal{N}$ be the two 0-1 matrices. Let $\mathbf{f} = f_1 f_2 \dots f_n$, $\mathbf{e}^{(1)} = e_1^{(1)} e_2^{(1)} \dots e_n^{(1)}$ and $\mathbf{e}^{(2)} = e_1^{(2)} e_2^{(2)} \dots e_n^{(2)}$. Further, let $\mathbf{e}^{(1)}$ and $\mathbf{e}^{(2)}$ have no words in common and each word appears exactly once. By setting the bigram language model probabilities of the bigrams that occur in $\mathbf{e}^{(1)}$ and $\mathbf{e}^{(2)}$ to 1 and all other bigram probabilities to 0, we can ensure that the only translations considered by E-DECODING-3 are indeed $\mathbf{e}^{(1)}$ and $\mathbf{e}^{(2)}$ and $P\left(\mathbf{e}^{(1)}\right) = P\left(\mathbf{e}^{(2)}\right) = 1$. We then set

$$\mathtt{t}(f|e) = \begin{cases} 1 & \text{if } f = f_j, e = e_i^{(1)} \text{ and } \mathcal{M}_{j,i} = 1 \\ 1 & \text{if } f = f_j, e = e_i^{(2)} \text{ and } \mathcal{N}_{j,i} = 1 \\ 0 & \text{otherwise} \end{cases}$$

$$\mathtt{n}(\phi|e) = \begin{cases} 1 & \phi = 1 \\ 0 & \text{otherwise} \end{cases}$$

$\mathtt{d}(j|i, n, n) = 1$.

Now, $P\left(\mathbf{f}|\mathbf{e}^{(1)}\right) = \mathtt{perm}(\mathcal{M})$, and $P\left(\mathbf{f}|\mathbf{e}^{(2)}\right) = \mathtt{perm}(\mathcal{N})$. Therefore, given the output of E-DECODING-3 we can find out which of $\mathcal{M}$ and $\mathcal{N}$ has a larger permanent. ■

Hence E-DECODING-3 is $\#\mathbf{P} - Hard$.

3.9 R-DECODING-3

Lemma 15 SETCOVER $\leq_p^m$ R-DECODING-3

Proof: Given an instance of SETCOVER, we set the parameters as in the proof of Lemma 1 with the following modification:

$$\mathtt{n}(\phi|e) = \begin{cases} \frac{1}{2\phi!} & \text{if } \phi > 0 \\ 0 & \text{otherwise.} \end{cases}$$

Let $\mathbf{e}$ be the optimal translation obtained by solving R-DECODING-3. As the language model is uniform, the exact order of the words in $\mathbf{e}$ is not important. Now, we observe that:

- $\mathbf{e}$ contains words only from the set $\{e_1, e_2, \dots, e_l\}$. This is because, there cannot be any zero fertility word as $\mathtt{n}(0|e) = 0$ and the only words that can have a non-zero fertility are from $\{e_1, e_2, \dots, e_l\}$ due to the way we have set the lexicon parameters.
- No word occurs more than once in $\mathbf{e}$. Assume on the contrary that the word e_i occurs $k > 1$

times in $\mathbf{e}$. Replace these k occurrences by only one occurrence of e_i and connect all the words connected to them to this word. This would increase the score of $\mathbf{e}$ by a factor of $2^{k-1} > 1$ contradicting the assumption on the optimality of $\mathbf{e}$.

As a result, the only candidates for $\mathbf{e}$ are subsets of $\{e_1, e_2, \ldots, e_l\}$ in any order. It is now straight forward to verify that a minimum set cover can be recovered from $\mathbf{e}$ as shown in the proof of Lemma 1. ■

3.10 IBM Models 4 and 5

The reductions are for Model 3 can be easily extended to Models 4 and 5. Thus, we have the following:

Theorem 7 *Viterbi Alignment computation is* $\mathbf{NP}$*-Hard for IBM Models* $3-5$.

Theorem 8 *Expectation Evaluation in the EM Steps is* $\#\mathbf{P}$*-Complete for IBM Models* $3-5$.

Theorem 9 *Conditional Probability computation is* $\#\mathbf{P}$*-Complete for IBM Models* $3-5$.

Theorem 10 *Exact Decoding is* $\#\mathbf{P}$*-Hard for IBM Models* $3-5$.

Theorem 11 *Relaxed Decoding is* $\mathbf{NP}$*-Hard for IBM Models* $3-5$ *even when the language model is a uniform distribution.*

4 Discussion

Our results answer several open questions on the computation of *Viterbi Alignment* and *Expectation Evaluation*. Unless $\mathbf{P} = \mathbf{NP}$ and $\mathbf{P}^{\#\mathbf{P}} = \mathbf{P}$, there can be no polynomial time algorithms for either of these problems. The evaluation of expectations becomes increasingly difficult as we go from IBM Models 1-2 to Models 3-5 exactly because the problem is $\#\mathbf{P}$-Complete for the latter models. There cannot be any trick for IBM Models 3-5 that would help us carry out the sums over all possible alignments exactly. There cannot exist a closed form expression (whose representation is polynomial in the size of the input) for $P(\mathbf{f}|\mathbf{e})$ and the counts in the EM iterations for Models 3-5.

It should be noted that the computation of *Viterbi Alignment* and *Expectation Evaluation* is easy for Models 1-2. What makes these computations hard for Models 3-5? To answer this question, we observe that Models 1-2 lack explicit fertility model unlike Models 3-5. In the former models, fertility probabilities are determined by the lexicon and alignment models. Whereas, in Models 3-5, the fertility model is independent of the lexicon and alignment models. It is precisely this freedom that makes computations on Models 3-5 harder than the computations on Models 1-2.

There are three different ways of dealing with the computational barrier posed by our problems. The first of these is to develop a restricted fertility model that permits polynomial time computations. It remains to be found what kind of parameterized distributions are suitable for this purpose. The second approach is to develop provably good approximation algorithms for these problems as is done with many $\mathbf{NP}$-Hard and $\#\mathbf{P}$-Hard problems. Provably good approximation algorithms exist for several covering problems including *Set Cover* and *Vertex Cover*. *Viterbi Alignment* is itself a special type of covering problem and it remains to be seen whether some of the techniques developed for covering algorithms are useful for finding good approximations to *Viterbi Alignment*. Similarly, there exist several techniques for approximating the permanent of a matrix. It needs to be explored if some of these ideas can be adapted for *Expectation Evaluation*.

As the third approach to deal with complexity, we can approximate the space of all possible $(l+1)^m$ alignments by an exponentially large subspace. To be useful such large subspaces should also admit optimal polynomial time algorithms for the problems we have discussed in this paper. This is exactly the approach taken by (Udupa, 2005) for solving the decoding and Viterbi alignment problems. They show that very efficient polynomial time algorithms can be developed for both *Decoding* and *Viterbi Alignment* problems. Not only the algorithms are provably superior in a computational complexity sense, (Udupa, 2005) are also able to get substantial improvements in BLEU and NIST scores over the *Greedy* decoder.

5 Conclusions

IBM models 3-5 are widely used in SMT. The computational tasks discussed in this work form the backbone of all SMT systems that use IBM models. We believe that our results on the computational complexity of the tasks in SMT will result in a better understanding of these tasks from a theoretical perspective. We also believe that our results may help in the design of effective heuristics

for some of these tasks. A theoretical analysis of the commonly employed heuristics will also be of interest.

An open question in SMT is whether there exists closed form expressions (whose representation is polynomial in the size of the input) for $P(\mathbf{f}|\mathbf{e})$ and the counts in the EM iterations for models 3-5 (Brown et al., 1993). For models 1-2, closed form expressions exist for $P(\mathbf{f}|\mathbf{e})$ and the counts in the EM iterations for models 3-5. Our results show that there cannot exist a closed form expression (whose representation is polynomial in the size of the input) for $P(\mathbf{f}|\mathbf{e})$ and the counts in the EM iterations for Models 3-5 unless $\mathbf{P} = \mathbf{NP}$.

References

K. Knight. 1999. Decoding Complexity in Word-Replacement Translation Models. *Computational Linguistics*.

Brown, P. et al: 1993. The Mathematics of Machine Translation: Parameter Estimation. *Computational Linguistics*, 2(19):263–311.

Al-Onaizan, Y. et al. 1999. Statistical Machine Translation: Final Report. *JHU Workshop Final Report*.

R. Udupa, and T. Faruquie. 2004. An English-Hindi Statistical Machine Translation System. *Proceedings of the 1st IJCNLP*.

Y. Wang, and A. Waibel. 1998. Modeling with Structures in Statistical Machine Translation. *Proceedings of the 36th ACL.*

D. Marcu and W. Wong. 2002. A Phrase-Based, Joint Probability Model for Statistical Machine Translation. *Proceedings of the EMNLP*.

L. Valiant. 1979. The complexity of computing the permanent. *Theoretical Computer Science*, 8:189–201.

M. Jerrum. 2005. Personal communication.

C. Tillman. 2001. Word Re-ordering and Dynamic Programming based Search Algorithm for Statistical Machine Translation. *Ph.D. Thesis, University of Technology Aachen* 42–45.

Y. Wang and A. Waibel. 2001. Decoding algorithm in statistical machine translation. *Proceedings of the 35th ACL* 366–372.

C. Tillman and H. Ney. 2000. Word reordering and DP-based search in statistical machine translation. *Proceedings of the 18th COLING* 850–856.

F. Och, N. Ueffing, and H. Ney. 2000. An efficient A* search algorithm for statistical machine translation. *Proceedings of the ACL 2001 Workshop on Data-Driven Methods in Machine Translation* 55–62.

U. Germann et al. 2003. Fast Decoding and Optimal Decoding for Machine Translation. *Artificial Intelligence*.

R. Udupa, H. Maji, and T. Faruquie. 2004. An Algorithmic Framework for the Decoding Problem in Statistical Machine Translation. *Proceedings of the 20th COLING*.

R. Udupa and H. Maji. 2005. Theory of Alignment Generators and Applications to Statistical Machine Translation. *Proceedings of the 19th IJCAI*.