Abstract — We survey and analyze state-of-the-art statistical machine translation (SMT) techniques for speech translation (ST). We review key learning problems, and investigate essential model structures in SMT, taking a unified perspective to reveal both the connections and contrasts between automatic speech recognition (ASR) and SMT. We show that phrase-based SMT can be viewed as a sequence of finite-state transducer (FST) operations, similar in spirit to ASR. We further inspect the synchronous context-free grammar (SCFG) based formalism that includes hierarchical phrase-based and many linguistically syntax-based models. Decoding for ASR, FST-based and SCFG-based translation are also presented from a unified perspective as different realizations of the generic Viterbi algorithm on graphs or hypergraphs. These consolidated perspectives are helpful to catalyze tighter integrations for improved ST, and we discuss joint decoding and modeling towards coupling ASR and SMT.

Index Terms — speech translation, statistical machine translation, finite state transducer, synchronous context-free grammar, discriminative training, graph, hypergraph, Viterbi search.

I. INTRODUCTION

Automatic speech translation (ST) enables cross-lingual speech communication. As one of mankind’s long-standing dreams, ST is an interesting yet highly challenging problem that intersects many research areas.

Over the last two decades, a number of research projects have been pursued, with ever-increasing complexity and more lofty goals, to address ST challenges ranging over various language pairs, different tasks, as well as on different platforms. Table 1 summarizes some of the ST projects that have been active in the field. The early ST efforts, which were typically limited-domain tasks for feasibility studies, focused more on conveying meaning by reinterpreting the transcribed source speech. Various translation techniques have been applied for ST ever since, such as the example-based approaches employed in (Wahlster, 2000), interlingua-based systems in (Levin et al., 1998), and data-driven approaches that integrate statistical parsing and natural language generation in (Zhou et al., 2002). As the domains of interest became broader and more data became available around the turn of 21st century, the statistical approach to machine translation (Brown et al., 1993), relying on a similar source-channel model to that used in Automatic Speech Recognition (ASR), has resurged. More recent advances (e.g., Och and Ney, 2004; Koehn et al., 2003; Chiang, 2005) in this direction have enabled statistical machine translation (SMT) to become dominant in text-based MT. Accordingly, SMT is also being increasingly adopted for most recent ST tasks, ranging from large-scale cloud-based services\(^1\) to completely smartphone-hosted speech-to-speech translation (Zhou et al., 2013). Applying SMT techniques to address the ST challenge has several noticeable advantages, namely its scalability, portability (to new language pairs and domains), and robustness. As an active research area in the field of computational linguistics, textual SMT has been surveyed in (Lopez, 2008) and (Koehn, 2010). This paper is intended to survey and analyze state-of-the-art statistical machine translation techniques from the perspective of speech translation. This study is of particular interest to us in advancing ST research for the following reasons.

Firstly, while the ST problem can be practically addressed by cascading ASR and SMT technologies that have been developed in isolation, it is important for ST researchers to have a broad and coherent view of the major problems in both fields. Towards this goal, the key SMT structure and learning problems for speech are surveyed from a unified perspective, revealing both contrasts and connections between the ASR and SMT processes that are intended to motivate further research in the ST area.

Secondly, the ST process also requires special considerations that are usually not fully discussed in the textual SMT literature. For example, the common statistical framework employed by both modern ASR and SMT makes it possible to pursue ST as a unified task by coupling both processes (Ney, 1999) for improved performance. From an ST perspective, it is therefore both conceptually appealing and practically beneficial to re-interpret, and further to design SMT approaches that facilitate such a coupling.

Thirdly, the ST process must deal with automatically transcribed spoken language, which is typically not well-formed and differs considerably from textual inputs. In addition to the problem of recognition errors (word deletions, substitutions, and insertions), much useful information for text analysis such as punctuation and casing is absent in the transcription. Moreover, the style of spoken language is often less grammatical and less formal than written language. Together, these challenges impose practical constraints on which SMT methods are most suitable to address ST tasks. Our survey and analysis are structured as follows:

\(^1\) For example, http://www.google.com/mobile/translate/
As an introduction to readers who are new to ST, we describe the ST problem in Sec. II, followed by an outline of the ASR process, particularly a finite-state transducer (FST) perspective of ASR (Mohri et al., 2002). We also overview the SMT process with forward references to later discussions in greater detail, concluding with a review of evaluation criteria for SMT/ST.

Sec. III is devoted to the key learning problems of SMT within a log-linear framework, starting from unsupervised word alignment and concluding with the parameterization and discriminative training of SMT.

We analyze the structures of various SMT models for ST in Sec. IV. Taking a unified perspective reveals the contrasts and connections between ASR and SMT. Both ASR and SMT address the problem of sequential pattern recognition by transforming a sequence of input observations in the source domain into another sequence of symbols in the target domain. In this view, the key issue for both is how to model the correspondence between the source and the target sequence. We show the popular phrase-based SMT as a sequence of finite-state transducer operations, which is similar in concept to ASR, with extra reordering operations dynamically incorporated. This particular perspective also catalyzes a tighter integration of ASR and SMT for ST, which is further discussed in Sec. VI.

In Sec. IV, we further inspect the synchronous context-free grammar (SCFG) based formalism that includes both hierarchical phrase-based and many linguistically syntax-based models. SCFG is a more general formalism than FST. As we know, in their monolingual counterparts, a finite-state machine (FSM) is equivalent to a regular grammar, which is a strict subset of a context-free grammar (Hopcroft and Ullman, 1979).

We present decoding algorithms in Sec. V, again from a unified perspective, to reveal a close connection between the different realizations of Viterbi decoding for ASR, FST-based translation, or even SCFG-based translation, despite their dramatic differences of first glance. Specifically, both decoding algorithms for ASR and FST-based translation can be viewed as instances of generic Viterbi algorithm on a weighted graph, and we can further generalize this to the hypergraph (Gallo, 1993), representing the search space of SCFG-based models.

We survey how to couple ASR and SMT for improved ST in Sec. VI, including the joint decoding with tight or loose coupling, and a more recent topic on joint modeling of both processes for ST.

We conclude in Sec. VII with some relevant research topics that have the potential to advance ST, additive to the respective individual improvements in ASR and SMT.

II. OVERVIEW OF SPEECH TRANSLATION

2.1. Speech Translation

![Fig. 1: A cascade approach to statistical ST](image)

Without loss of generality, a cascaded speech translation system takes a source speech signal sequence \( x_1^T = x_1, \ldots, x_T \) as input, recognizes it as a set of source word sequences, \( \{f_1^l = f_1, \ldots, f_l\} \), which are then translated into the target language sequence of words \( e_1^l = e_1, \ldots, e_l \). The ST process thus typically includes both speech recognition and machine translation as sub-tasks, as shown in Fig. 1.

At a high level, both ASR and SMT address a problem of sequential pattern recognition, e.g., taking a sequence of input observations in the source domain (e.g., acoustic feature sequences \( x_1^T \) for ASR, \( f_1^l \) for SMT) and determining a sequence of symbols that is regarded as the optimal equivalent in the target domain (e.g., \( e_1^l \) for ASR, \( e_1^l \) for SMT). Because of this strong connection, many techniques in ASR and SMT are closely related, as we will show in the following sections.

However, there is a fundamental difference between ASR and SMT, as the latter needs to deal with a different ordering between the input and output sequences. For ASR, the order of the output words is strictly monotonic in the order of input acoustic features. Conversely, output words in SMT are usually ordered non-monotonically to their counterparts in the source language. This leads to different modeling and decoding formalisms required in SMT.

To pave the ground, we will provide an overview of ASR and SMT in this section, as well as the evaluation metrics for ST.

2.2. Automatic Speech Recognition

ASR is the process of converting a speech signal into a sequence of words (or other linguistic symbols). As shown in Fig. 2, speech production from words is modeled by a
sequences, according to a pronunciation lexicon, where each phone (with its left and right contexts) is modeled by a hidden Markov model (HMM) (Rabiner, 1989), whose hidden states are transitioned with Markov assumptions. At each state, speech is produced according to state-dependent statistical distributions. In this process, the speech waveform is represented by a sequence of spectrum vectors (i.e. observations) equally spaced in time. Each observation (i.e. frame) is assumed to represent the speech waveform over a short duration of about 10 msec, within which the speech waveform is regarded as a stationary signal. Typical parametric representation choices for observations include linear prediction coefficients, perceptual linear prediction, and Mel frequency cepstral coefficients (MFCC), added by the first and second order time derivatives. These observation vectors are usually considered as independent given a state of the HMM.

Given a sequence of observation vectors $x_t$, the word string is decoded by the recognizer. The optimal word sequence $f_1^j$ given $x_1^T$ is obtained by:

$$ f_1^j = \arg\max_{f_1^j} P(f_1^j | x_1^T) = \arg\max_{f_1^j} P(f_1^j, x_1^T) $$

$$ = \arg\max_{f_1^j} P(x_1^T | f_1^j)P(f_1^j) $$

(1)

In (1) and elsewhere, argmax denotes the search process. $P(f_1^j)$, known as the language model, is usually approximated by an n-gram model (Jelinek, 1997), i.e., a (N-1)-th order Markov chain:

$$ p(f_1^j) \approx \prod_{i} p(f_i | f_{i-N+1}^i, \ldots, f_{i-1}) $$

(2)

where $f_i$ is the i-th word in the sentence $f_1^j$. $P(x_1^T | f_1^j)$, known as the acoustic model, is represented as an HMM, i.e.

$$ P(x_1^T | f_1^j) = \sum_{q_1^T} p(q_1^T | f_1^j) \cdot p(x_1^T | q_1^T) $$

(3)

where $q_1^T$ is the hidden Markov state sequence corresponding to $f_1^j$ as shown in Fig. 2, $p(q_t | q_{t-1}) = a_{q_{t-1}, q_t}$ is the state-transition probability; $p(x_t | q_t)$ is the emitting probability that are modeled by Gaussian mixture models (GMM) as follows, where $K$ is number of mixture components, $c_{j\mu}$ is the component weight, and $D$ is the dimension of the observation:

$$ b_j(x_t) = \sum_{\mu=1}^{K} c_{j\mu} \prod_{\delta=1}^{D} \frac{1}{\sqrt{2\pi}\sigma_{\delta k\mu}} \exp \left(-\frac{1}{2} \frac{(x_{\delta d} - \mu_{\delta k\mu})^2}{\sigma_{\delta k\mu}} \right) $$

(4)

The parameters from Eq. (3) and (4) are trained from speech audio labeled with a word-level transcription. Traditionally, the parameters were optimized under the criterion of maximizing the joint likelihood $P(f_1^j, x_1^T)$, and it has been shown that significantly better results can be obtained by optimizing under discriminative criteria such as minimum phone error rate (Povey and Woodland, 2002).

The search problem of the source-channel model in Eq. (1) can be factorized into a sequence of weighted finite-state transducers (WFST), and thus:

$$ f_1^j = \text{best}_\text{path}(O \circ H \circ C \circ L \circ G) $$

(5)

where “o” is the composition operation, O, H, C, L, G are the WFSTs (Mohri et al., 2002) whose transition relationships are illustrated in Fig. 2. For example, L transduces context-independent phonetic sequences into words, and G is a weighted acceptor that assigns language model probabilities.

Readers are referred to (Rabiner, 1989; Jelinek, 1997; Deng and O'Shaughnessy, 2003; Baker et al., 2009) for more comprehensive reviews about statistical speech recognition.

2.3. Statistical Machine Translation

Over the past few decades, the source-channel (Eq. (1)) based statistical approach to ASR described above has proved successful in both research and commercial applications. Subsequently, its success influenced machine translation, another computer-based human language technology that can be traced back to the 1950’s (Hutchins, 1986), and helped shape statistical machine translation (SMT) research since the late 1980s. Brown et al. of IBM conducted pioneering work in introducing a statistical modeling approach to translation and in establishing a range of models --- now known as IBM Model 1 to Model 5 (Brown et al., 1993). Since then, much important progress has been accomplished in a full span of the SMT component technologies including word alignment (Och and Ney, 2003), phrase-based SMT (Och and Ney, 2004; Koehn et al., 2003), hierarchical phrase-based methods (Chiang, 2007), syntax-based SMT (e.g., Galley et al., 2006), discriminative training (e.g., Och, 2003; Liang et al., 2006; Chiang, 2009), model adaptation (e.g. Foster et al., 2010), and system combination (e.g. Rosti et al., 2012), to name just a few. SMT is today in widespread public use, e.g., Google Translate.

Fig. 3 shows the typical blocks in a modern SMT system. SMT is trained from a corpus with aligned source and target sentences, known as the parallel corpus. A word is usually considered as the minimum unit in translation. Without loss of generality, for an English-French sentence pair $e_t^1$ and $f_t^j$, word alignment training, an unsupervised process, links English

\[\text{Fig. 2: An illustration of the speech recognition process.}\]
words to their translational counterparts in French. Word alignment is an important step of most modern approaches to statistical machine translation. It provides the basic translation information on which higher level translation equivalents, such as phrase pairs or translation rules, can be constructed. Optionally, syntactic parsing can be applied on one or both sides of the parallel corpus, from which syntactic translation rules can be extracted from the aligned corpus. The decoder searches for the best target language sentence for the given input source sentence, based on all the component models. To optimize translation performance, we typically need to train the weights of different models, which are determined using a development set.

We will review how to train SMT (i.e., word alignment, phrase-pair extraction, and discriminative training) in Sec. III, and discuss the structure and decoding problem of SMT, from the perspective of ST, in Sec. IV and Sec. V, respectively.

### 2.4. Evaluation metrics for translation and ST

The automatic evaluation of translation quality is conducted by comparing the machine translation hypothesis against translation reference(s) annotated by human translators. The most commonly used evaluation metric for translation quality is the BLEU score (Papineni et al., 2002), which measures the geometrically-averaged precision of n-grams in the translation hypothesis that match the reference. The widely-adopted BLEU-4 score is computed by

\[
\text{BLEU-4} = \text{BP} \cdot \exp \left( \frac{1}{4} \sum_{n=1}^{4} \log(p_n) \right)
\]

where \( p_n \) (\( n = 1, \ldots, 4 \)) is the precision of n-gram matches, and BP is the brevity penalty that penalizes if the hypothesis is shorter than references. Note that both BP and \( p_n \) are computed for the entire evaluation corpus, and therefore BLEU is a metric of the corpus level rather than individual sentence.

The readers should be aware that there are other evaluation metrics such as Translation Edit Rate (TER) (Snover et al., 2006), and METEOR (Banerjee and Lavie 2005). Automatic metrics for the evaluation of translation quality are still an open problem (Callison-Burch et al., 2010).

Automatic evaluation metrics for MT have also been adopted for ST, adding to the word error rate (WER) that measures ASR qualities. Some subjective metrics, such as the high-level and low-level concept transfer rates (Schlenoff et al., 2009), are also used to more closely assess the success rate of a ST system accomplishes for cross-lingual communications.

### III. LEARNING OF TRANSLATION MODELS

In SMT, the optimal translation \( \hat{e}^I_i \) given the input source language sentence \( f^I_i \) is obtained via the decoding process according to

\[
\hat{e}^I_i = \arg \max_{e^I_i} P(e^I_i | f^I_i) = \arg \max_{e^I_i} \frac{1}{Z} \exp \left( \sum_{m=1}^{M} \lambda_m h_m(e^I_i, f^I_i) \right)
\]

Unlike ASR, the posterior probability of the translation \( e^I_i \) given \( f^I_i \) is modeled directly through a log-linear model (Och and Ney 2002), which can be viewed as a generalization of the source-channel model used in ASR. The normalization term \( Z = \sum_{e^I_i} \exp \left( \sum_{m=1}^{M} \lambda_m h_m(e^I_i, f^I_i) \right) \) ensures that the posterior probabilities sum to one.

The set \( \{h_m(e^I_i, f^I_i)\} \) are the feature functions, also called component models, empirically constructed from \( e^I_i \) and \( f^I_i \). Since it is impractical to train or decode with component models on the level of whole sentence pairs, we need to decompose feature \( h_m(e^I_i, f^I_i) \) into functions of smaller units that allow for tying across different samples, e.g., by dividing string sentences into phrases shared across different sentences. This enables both robust model parameterization and efficient decoding through dynamic programming. Generally speaking, the decomposition is of the following form:

\[
h_m(e^I_i, f^I_i; d) = \bigotimes_{i=1}^{N} h_m(\hat{e}_{i}, \hat{f}_{i}; d)
\]

Here \( \bigotimes \) is the abstract operator from semiring (\( \Psi, \oplus, \ominus, 0, 1 \)) (Mohri 2002), \( d \) specifies a decomposition of the sentence pair into \( N \) units, and \( (\hat{e}_{i}, \hat{f}_{i}; d) \) are units that are (probabilistic) translation equivalents modeled by \( h_m(\hat{e}_{i}, \hat{f}_{i}; d) \). The actual form of the decomposition and the specific semiring operator depend on the choice of the formalism for the translation equivalents and \( h_m(\hat{e}_{i}, \hat{f}_{i}; d) \). In practice, we use the following instead of Eq. (6):

\[
\hat{e}_{i}^{I} = \arg \max_{e^{I}_{i}} P(e^{I}_{i} | f^{I}_{i}; d) = \arg \max_{e^{I}_{i}} \left( \sum_{m=1}^{M} \lambda_m h_m(e^{I}_{i}, f^{I}_{i}; d) \right)
\]

4 In a similar spirit to that we usually model speech at the phone level instead of the word level for large vocabulary speech recognition.

5 A systematic description of the semiring framework is provided in (Mohri, 2002). Typical semirings used in speech and language processing include the so-called Viterbi \( ([0,1], \max, \times, 0, 1) \), which is usually defined over probabilities, and the Tropical \( R^{*} \cup \{+\infty\} \) with addition \( + \), minimum \( + \), and maximum \( + \).
The problem in SMT is therefore to develop appropriate models for Eq. (8), to learn their parameters in training, and to perform the search of argmax at decoding. Next, we discuss how to learn the parameters of the translation models, including the weights of the features of the log-linear model, and the parameters of component models. In this section, we use the learning of phrase-based models as an example, which is a common foundation for modern SMT models, and defer the review of learning of SCFG-based models to Sec. IV, after the discussion of structures of translation equivalents. Decoding is the topic of the Sec. V.

3.1 Word alignment model

Word alignment is the cornerstone to learn translation equivalent models in SMT. The classical approaches to word alignment, which assumed a one-to-many correspondence (i.e., one source word maps into zero or more target words), are based on IBM models 1-5 (Brown et al., 1993) and the HMM based alignment models (Vogel et al., 1996; Och and Ney, 2000), while discriminative approaches (e.g., Moore, 2006) and syntax based approaches (e.g., Zhang and Gildea, 2005) have also been studied in the literature.

3.1.1 IBM Models for word alignment

Let’s denote \( \alpha_i^l = (a_1, ..., a_l) \) as the alignment that specifies the position of the English word aligned to each French word. In addition, an empty word NULL is introduced as \( e_0 \), to generate French words that do not align to any English words. In (Brown et al., 1993), a set of statistical models is proposed, from simple to complex, to model the word alignment between the source sentence and the target sentence. Starting from IBM Model 1, the probability of \( f_i^l \) given \( e_i^l \) is defined as follows.

\[
P(f_i^l | e_i^l) = \frac{C}{(l+1)!} \sum_{t=1}^{l} P(f_i | e_t)
\]

where \( C \) is a constant irrelevant to training. An efficient Expectation-Maximization (EM) algorithm exists to find the optimal word translation probabilities of IBM Model 1 due to its convex objective function.

On top of IBM Model 1, (Brown et al., 1993) also introduced more sophisticated models, Model 2-5, that consider the relative word position and word fertility (i.e., the number of French words that an English word can align to). These models give better word alignment performance with higher computational cost and non-convex objectives, so training is typically done using simpler models to bootstrap the training of more complex models.

3.1.2 HMM for word alignment

Vogel et al. (1996) proposed using HMMs for word alignment. Here, a HMM is built at the English side, i.e., each position/word pair \( (a_j, e_j) \) is a HMM state, which emits the French word \( f_j \). In order to mitigate the data sparsity problem, it is assumed that the emission probability only depends on the English word, i.e., \( P(f_j | a_j, e_j) = P(f_j | e_j) \) and the transition probability only depends on the position of the previous state and the length of the English sentence, i.e., \( P(a_j | a_{j-1}, e_{a_{j-1}}, l) = P(a_j | a_{j-1}, l) \). Then, we have

\[
P(f_i^l | e_i^l) = \prod_{j=1}^{l} P(a_j | a_{j-1}, l) P(f_j | e_j)
\]

In (Vogel et al., 1996), it is further assumed these transition probabilities \( P(a_j = t | a_{j-1} = t', l) \) depend only on the distance between two states \( (i - i') \), i.e.

\[
P(i|t', l) = \frac{c(i - i')}{\sum_{\ell = 1}^{l} (l - \ell)}
\]

The HMM parameter set, \( \Lambda = \{ P(i|t', l), P(f_j | e_j) \} \), can be estimated through maximum likelihood training using the efficient EM algorithm (Rabiner, 1989). After training, Viterbi decoding is used to find the best alignment sequence \( \hat{a}_i^l \), i.e.

\[
\hat{a_i}^l = \arg \max_{\alpha_i^l} \prod_{j=1}^{l} P(a_j | a_{j-1}, l) P(f_j | e_j)
\]

In practice, the HMM parameter set is initialized by IBM model 1, and more complicated models such as Model 3 and 4 can also be trained following HMM iterations. The word alignment quality can be further improved using some heuristics. For example, usually both directions of word alignment, source-to-target and target-to-source, are produced and later merged according to a set of heuristic rules (Och and Ney, 2003; Koehn et al., 2003). An example of a word aligned parallel sentence is shown in Fig. 4 (a).

3.2 From words to phrases

![Illustration of word alignment and phrase pair extraction](image_url)

**Fig. 4:** An illustration of word alignment and phrase pair extraction. In (a), the solid squares indicate the alignment links between source and target words. The boxes indicate valid source-target phrase pairs. In (b), part of the bilingual phrase table extracted from this sentence is shown.

Once the word level translation mapping is established, we can expand the translation mapping to the multi-word level, i.e. phrase level, to obtain phrase tables, the building blocks for phrase-based SMT. Note that hereafter the term phrase bears no linguistic meaning but simply indicates a consecutive sequence of words.
A phrase table is extracted from the word aligned parallel data according to phrase extraction rules (Och and Ney, 2004; Koehn et al., 2003). Fig. 4 illustrates the phrase extraction process. Starting from the word alignment, all possible phrase pairs are extracted as long as there is no alignment across the phrase boundary, and there is at least one alignment link. Several examples are shown in Fig. 4 (b). As a negative example, “初步 试验 的” $\Leftrightarrow$ “of initial” is not a valid phrase pair since the word “试验” in the source phrase is aligned to a word “experiments” which is outside the target phrase. In practice, all valid phrase pairs will be extracted until the maximum length of phrases meets a pre-determined threshold.

### 3.3. Parameterization of SMT

The free parameters of the log-linear model in Eq. (8) are composed of the parameters for features $h_m(e_l, f'_l, d)$, and the weights of these features (discussed in the next section). The number of features used in modern SMT systems ranges from a handful (Koehn et al., 2003) to thousands and even millions (Liang et al., 2006; Chiang et al., 2009).

Common features include the language models, phrase/rule and lexical translation models, reordering models, word and phrase counts etc. In addition, a large number of sparse binary features are also used. Liang et al. (2006) proposed a large set of new lexical and part-of-speech features. Chiang et al. (2009) incorporated about ten thousand syntactic features in addition to the baseline features. Among them, the phrase and lexical translation features, the important ones common to all SMT systems, are presented below. We will discuss the reordering models in Sec. 4.1.

#### 3.3.1 Phrase translation features

Once a set of phrase pairs is extracted, phrase translation probabilities are computed as relative frequencies of phrases over the training dataset (Koehn et al., 2003). The probability of translating $f$ to target phrase $\bar{e}$ is given by:

$$p(e|f) = \frac{C(e, f)}{C(f)}$$

(13)

where $C(e, f)$ are the joint counts of $\bar{e}$ and $f$, and $C(f)$ is the marginal count of $f$. The target-to-source phrase translation feature $p(f|\bar{e})$ is defined similarly$^6$.

Phrase table refinement and pruning is an active research area. For the readers who may be interested, they are referred to the literature for further study (e.g., Wuebker et al., 2010).

#### 3.3.2 Lexical translation features

Lexical translation features provide alternative ways to score translation probabilities of phrase/rule pairs. There are several variations in lexical translation features (e.g., Koehn et al., 2003, Quirk et al., 2005). Assuming an extracted phrase pair $(\bar{e}, \bar{f})$ has an alignment vector $\bar{a} = [\bar{a}]$, where each element is a possibly empty set of positions in $\bar{e}$ that the word $f_j \in \bar{f}$ is aligned to, we define:

$$i(\bar{f}|\bar{e}) = \prod_{j=1}^{|\bar{f}|} \left( \frac{1}{|\bar{a}|} \sum_{i \in \bar{a}} p(f_j|e_i) \right) \text{ if } |\bar{a}| > 0$$

$$p(f_j|\text{NULL}) \text{ else}$$

(14)

Here, $p(f_j|e_i)$ and $p(f_j|\text{NULL})$ are word pair translation probabilities that are estimated by relative frequencies similar to Eq. (13) from aligned parallel corpus, and the other direction is defined symmetrically. The same lexicon feature definition is applicable to SCFG-based SMT models.

### 3.4. Discriminative training of translation models

In discriminative training, parameters are trained by maximizing the evaluation metrics (e.g., BLEU) of the translation output on a development set. For example, we optimize the feature weights $\Lambda = \{\lambda_m\}$ in Eq. (8) by:

$$\hat{\Lambda} = \text{argmax}_\Lambda \text{BLEU}(E^*, \hat{E}(\Lambda, F))$$

(15)

where $E^*$ is the translation reference(s) of the development set $F$, and $E(\Lambda, F)$ is the set of translation outputs, which is obtained through the decoding process according to Eq. (8) given $F$ and feature weights $\Lambda$. The challenge is that the objective function of Eq. (15) is non-convex and non-smooth, which rules out gradient-based optimization methods.

When the number of features $M$ is around a dozen, $\Lambda$ can be effectively tuned by Minimum Error Rate training (MERT) (Och, 2003). Motivated by the key observation that BLEU is piece-wise constant for any given dimension $\lambda_m$, MERT optimizes each $\lambda_m$ in turn, similarly in spirit to Powell’s algorithm (Brent 1973). Other gradient-free methods such as the Downhill Simplex have also been used (Zhao and Chen, 2009).

As $M$ grows to incorporate additional useful features in the translation model, MERT and Simplex both become ineffective. The discriminative training for a large-scale $\Lambda$, or more generally, a large number of parameters in the translation models, is hence an active area of great interest. Below, we survey a few of representative works in the literature.

Watanabe et al. (2007) and Chiang et al. (2009) use a large-margin version of an online perceptron-like algorithm, known as the Margin Infused Relaxed Algorithm (MIRA), which constraints $\Lambda$ to score the correct translation $e^*$ higher than incorrect ones by a margin that is no smaller than the loss occurred in choosing the incorrect translation. In an effort to incorporate the simplicity and efficiency of MERT in the batch learning of large-scale $\Lambda$, Hopkins and May (2011) propose a tuning-as-ranking approach that optimizes $\Lambda$ by requiring that the better translation of any pair selected from the $N$-best should always be ranked higher than the inferior translation.

Rosti et al. (2011) instead modify the objective in Eq. (15) to make it differentiable, including replacing the corpus BLEU with expected BLEU and approximating the brevity penalty with a smooth polynomial function. Afterwards, effective gradient-based optimization methods, such as L-BFGS, can be directly applied to optimize $\Lambda$.

---

$^6$ Interestingly, Shen (2011) showed that these simple estimations turn out to be a theoretically justified method, coping well with the uncertainty of overlapping building blocks (phrase pairs or SCFG rules) in translation models.
It should be noted that while the log-linear model itself is discriminative, features such as the phrase translation are conventionally estimated by maximizing likelihood (Sec. 3.3), which may not correspond to the translation metric closely. Therefore, it is desirable to train these parameters to maximize an objective directly linked to translation quality. Liang et al. (2006) employ an averaged perceptron algorithm to update a large number of translation probabilities, and different global discriminative training methods are also proposed in ( Tillmann and Zhang, 2006; Blunsom et al., 2008). More recently, He and Deng (2012a) propose a discriminative training method to directly optimize the phrase and lexical translation features via maximizing the sum of expected sentence BLEU score over the entire training corpus, as another approximation to the corpus BLEU used in Eq. (15). The optimization method in their work is based on the Extended Baum-Welch (EBW) algorithm, which has been found effective in the ASR community to discriminatively train acoustic models with tens of millions of free parameters (e.g., Povey and Woodland, 2002).

IV. Translation Structures for ST

The formalism of modeling and parameterizing translation equivalents is central to SMT. The translation equivalents are usually represented by some synchronous grammar as the core of modern SMT systems. The choices of such a grammar are usually limited by two factors. Firstly, is it sufficiently expressive? That is, is it adequate to model linguistic equivalence between natural language pairs, as far as translation is concerned? Secondly, is it computationally feasible for building practical machine translation solutions?

Particularly for ST, the translation equivalents are ideally to be robust to the informal style of the spoken language, and to the common flaws and incompleteness (e.g., recognition errors and missing punctuation and casing) presented in the automatic transcription. Furthermore, the speed requirement is also critical due to the interactive nature of many ST tasks, which often require low-latency operation. Therefore, it is beneficial for ST if the translation formalism can be more effectively and efficiently integrated with ASR.

Given the above two constraints, the majority of statistical translation formalisms proposed in the literature fall into one of the following two categories: 1) the finite-state transducer based formalism, and 2) the synchronous context-free grammar based formalism. These two formalisms can be viewed as a generalization of their respective monolingual counterparts, finite-state grammar and context-free grammar, which have a close connection in formal language and automata theory (Hopcroft and Ullman, 1979).

We will review below these two formalisms of translation equivalence and discuss their expressiveness. Their computational complexities will be examined in Sec. V, together with the presentation of decoding algorithms. We note that weighted finite-state transducers (WFSTs) are well-known in the speech community, as many signal processing and pattern recognition problems that concern converting an input sequence to output, including speech recognition (Mohri et al., 2002) and translation (Bangalore and Riccardi, 2003), can be effectively represented by WFSTs. Therefore, we will start from the popular phrase-based SMT, and show that it is an instance of FST-based formalism. Next, we will show how a cascade of WFST constructs translations for inputs through WFST’s generation mechanism. This process is analogous to the WFST decomposition of modern ASR, as described in Sec. II, and therefore we can demonstrate inherent connections between ASR and SMT. We will then proceed to SCFG as a generalization of the FST-based formalism.

4.1 FST-based translation equivalence

In FST-based translation, the equivalence between the source and target language is modeled by weighted and non-nested mapping between the source and target word strings. On top of that, the mapped sequences are concatenated (with possible reordering) to obtain a full sentence.

The unit of mapping, i.e., the \((\tilde{e}, \tilde{f})\) in Eq. (7), is a choice of the model. The pioneering SMT work (Brown et al., 1993) chose to model the mapping at the word level, largely due to computational resource limits at that time. A decade later, with the resurgent interest in SMT and the availability of more powerful computational resources, researchers soon verified that modeling the mapping with phrases led to significantly better translation performance (Och and Ney, 2004; Koehn et al., 2003). The improvement is credited to two major benefits brought by the phrase: reduced word sense ambiguities with surrounding contexts in the phrase, and appropriate local reordering encoded in the source-target phrase pair. Hence, our discussion will focus on using phrases as the basic translation unit. We denote by \(PT = \{(\tilde{e}, \tilde{f})\}\) the bilingual phrase table extracted from the training data (see Sec. 3.2).

The phrasal translation procedure can be decomposed into a sequence of generative processes. For example, one may proceed as follows (Steps 1-4 shown in Fig. 5 but not step 5, since it is a finite-state acceptor that only scores translation quality rather than generating new outputs):

1. The input sentence \(f_1^J\) is segmented into \(K\) phrases \(\tilde{f}_1, \tilde{f}_2, \ldots, \tilde{f}_K\), \(1 \leq K \leq J\)
2. Permute the source phrases in the appropriate order
3. Translate each source phrase into a target phrase
4. Concatenate target phrases to form the target sentence
5. Score the target sentence by the target language model

In terms of expressiveness, this process can generate any sequence of translation, if arbitrary reordering is permitted in step 2. However, arbitrary reordering makes translation a NP-hard problem (Knight, 1999), and thus in practice the reordering is usually limited (Zens et al., 2004).

To see why phrase-based SMT is finite-state, first notice that each step can be achieved by finite-state operations, as shown below, if we view each step as a mapping between the input and output with some probabilities. A WFST is a powerful mechanism to represent such weighted, non-nested mappings. Furthermore, the sequence of all steps is a cascade of these
WFSTs that are combined using the composition operation, which is guaranteed to produce a finite-state machine, since WFSTs are closed under such operations (Mohri et al., 2002).

![Diagram](image-url)  
(1) 初步试验的成功极大地激励了我们。
(2) 我们极大地激励了初步试验的成功。
(3) The success of initial experiments had inspired us greatly.
(4) The success of initial experiments had inspired us greatly.

**Fig. 5:** The generative process for phrase-based SMT

Realizing that phrase-based SMT is an FST-based operation is not only of theoretic convenience to conceptually connect ASR and SMT better, but is also practically useful to leverage the mature WFST optimization algorithms. Furthermore, it is also possible to have an integrated framework to address ST (Mathias and Byrne, 2006; Zhou et al., 2007; Bertoldi et al., 2008, Casacuberta et al., 2008).

Specifically, the input sentence and target language model can be represented by weighted finite-state acceptors F and G. Other models and processes such as segmentation, permutation, translation and concatenation can all be represented by WFSTs, denoted by P, R, T, W respectively. In (Kumar et al., 2005), the best translation is achieved through the following standard operations defined in a general-purpose FST toolkit:

$$E = \text{best}_\text{path} (F \circ P \circ R \circ T \circ W \circ G)$$  \hspace{1cm} (16)

That is, component transducers are composed sequentially as a binary operation left-to-right followed by a best-path search. However, SMT using a general WFST toolkit has several significant disadvantages. Namely, the large memory requirements and heavy online computation burden for each individual composition operation. Hence, the standard WFST-based system (Kumar et al., 2005) runs significantly more slowly than the multiple-stack based decoder (Koehn, 2004), which makes it impractical to deploy for interactive ST.

The above issue is addressed in the Folsom framework (Zhou et al., 2006; Zhou et al., 2008), framing the log-linear phrase-based SMT as:

$$E = \text{best}_\text{path} (F' \circ M \circ G)$$  \hspace{1cm} (17)

where F’ is a weighted finite-state acceptor encoding the uncertainty (e.g., various reordering options) of the input sentence, M is the log-linear translation model represented by a WFST that is computed offline as:

$$M = \text{Min} (\text{Min} (\text{Det}(P) \circ T) \circ W)$$  \hspace{1cm} (18)

Here Min and Det denotes the minimization and determinization operation, respectively. Therefore, the translation system is built upon three finite-state machines F’, M and G, and an efficient stand-alone decoder can solve for Eq. (17) that conducts the best-path search and the three-way composition simultaneously, using a Viterbi search algorithm (discussed in Sec. V).

Next, as concrete examples, we will describe how each of the above WFSTs is constructed to enable performing the sequential compositions offline in Eq. (18), which lead to significantly reduced runtime in translation.

**Segmentation transducer P:** It explores all “acceptable” phrase sequences for a source sentence. By “acceptable”, we mean that all phrases f_i in the output must be observed in PT, and the concatenation of the phrases produces exactly the original sentence f_i^{PT}. It is crucial that P, as the first transducer in the composition sequence, can be determinized. Not only can determinization greatly reduce the size of the FST, but it also collapses multiple edges with the same label into a single arc, vastly reducing the amount of computation during search. However, the representation of P in (Kumar et al., 2005) is non-determinizable due to the overlap between phrases such that one phrase could be a prefix of another, and thus multiple nested phrases can be produced by a single word sequence. For example, given an input sequence of “初步试验的成功”, the following overlapping phrases can be produced:

初步
初步试验
初步试验的成功

Thus, the identity of phrase sequence of a source sentence may not be uniquely determined until the entire sentence is observed, and such unbounded delays make P non-determinizable. To solve this problem, an auxiliary symbol, denoted EOP, was introduced in (Zhou et al., 2006), marking the end of each distinct phrase. By adding these artificial phrase boundary markers, the transducer becomes determinizable. Once we have determinized the FST, we can replace the EOP markers with empty strings in a later step as appropriate. As we assume a uniform distribution over segmentations, we simply set the cost associated with each arc to zero (in the Tropical semiring as described in footnote 5). It is also possible to use a non-uniform segmentation model such as in (Blackwood et al., 2008).

**Phrase translation transducer T:** The phrase translation model in step 3 is implemented by a weighted transducer that maps source phrases to target phrases. Under the assumptions of phrase translation independence, this transducer is a trivial one-state machine, with every arc corresponding to a phrase pair contained in PT. In order to be consistent with the other WFSTs, one more arc is added in this transducer to map EOP to itself with no cost.

**Target phrase-to-word transducer W:** After translation, the target phrases can be simply concatenated in step 4 to form the target translation. However, in order to constrain translations across phrases, it is necessary to incorporate the effects of a target language model in the translation system. To achieve this, the target phrases must be converted back to words. It is clear that the mapping from phrases to word sequences is deterministic. Therefore, the implementation of this transducer is straightforward. Again, we need to place the auxiliary token EOP on additional arcs to mark the ends of phrases.
**Target language model acceptor** \( G \): The target language model is represented by a weighted acceptor that assigns probabilities to a hypothesis based on a back-off n-gram language model.

**Reordered input acceptor** \( F \): Although we wish to minimize the online computation, the reordering or permutation scheme cannot be represented in general by finite-state machines, as the number of states needed to represent all possible reorderings of \( f_i \) is \( O(2^i) \), which cannot be bounded for any arbitrary inputs. In (Zhou et al., 2006), an architecture called Folsom is proposed, in which the reordering is explored by a dynamic expansion of the source-side acceptor \( F \) according to a given reordering model, leading to a reordering FSM \( F' \) (Zhou et al., 2008). Fig. 6 shows two acceptors of a 4-word input. Here, each state denotes a specific input coverage, which is represented by a bit-vector. The initial state leaves all bits zero and the final state has all bits one. Arcs linking a previous state and a next state are labeled with an input position to be covered. Arcs and their weights (omitted for simplicity) are determined by given reordering models. Every path from the initial state to the final state represents an acceptable permutation of the input. In Fig. 4(a), a linear left-to-right automaton \( F \) is constructed for the monotonic decoding; and Fig. 4(b) is reordered under the IBM constraint (Zens et al., 2004) with skips less than three.

![Diagram](image)

**Fig. 6:** Inputs (a) \( F \) and (b) \( F' \), computed on-the-fly according to a reordering model

It is worth pointing out that Folsom performs word-level reordering on inputs, followed by a lazy composition with the phrase-based transducer \( M \). It follows that the source segmentation therefore allows options with “gaps” in searching for the best-path to transduce the inputs at the phrase level. In other words, it is possible to transform the input as follows:

\[
\hat{f}_1 \hat{f}_2 \hat{f}_3 \hat{f}_4 \xrightarrow{\text{phrase segmentation}} \hat{f}_1 \hat{f}_4 \quad \hat{f}_2 \hat{f}_3
\]

Thus the decoder also explores the option to translate \( \hat{f}_1 \hat{f}_4 \) as a phrase that is otherwise impossible if not allowing two gaps skipping \( \hat{f}_2 \hat{f}_3 \), which is translated as another phrase together. Such a translation phenomenon is beneficial in many language pairs. In Chinese-English translation, for instance, it is common to translate isolated Chinese preposition or verb phrases together, e.g., to translate “在…外” as “outside” or “唱…歌” as “sing”. This adds flexibility to conventional phrase-based models (Och and Ney, 2004; Koehn et al., 2003) where phrasal segmentation forbids gaps. Studies by (Ittycheriah and Roukos, 2007; Galley and Manning 9, 2010) have confirmed the advantages of realizing a similar kind of flexibility in their respective models.

### 4.2 SCFG-based translation equivalence

Finite-state based approaches (such as phrase-based models) have been very successful and remain state-of-the-art for many tasks (Zollmann et al., 2008). However, such an approach makes no explicit use of the fact that natural language is inherently structural and hierarchical, which leads to its exponential complexity of permutation and the insufficiency to model long-distance reordering. Motivated by formal language and automata theory (Hopcroft and Ullman, 1979), on the other hand, we know that a finite-state automaton is equivalent to a regular grammar, which constitutes a strict subset of a context-free grammar (CFG). Probabilistic CFGs have been widely used in natural language processing tasks (e.g., parsing) to effectively model linguistic structures. Therefore, the synchronous version of the PCFG has been actively studied in recent years in the SMT community as an alternative to FST.

A SCFG (Lewis and Stearns, 1968) is a synchronous rewriting system generating the source and target sides simultaneously based on a PCFG. Each synchronous production rule probabilistically rewrites the non-terminal (NT) on its left-hand side (LHS) into a pair of representations on its right-hand side (RHS), \( \gamma \) and \( \alpha \), corresponding to the source and target side of the transduction, respectively, subject to the constraint that there is a one-to-one correspondence between the source and the target of every NT occurrence. For example, the following SCFG rule captures the phenomenon that VP (verb phrase) follows PP (prepositional phrase) in Chinese (source) while VP precedes PP in English (target) when rewriting a VP.

\[
VP \xrightarrow{p} < PP_1, VP_2, VP_2 PP_1 >
\]

Here the one-to-one correspondence of source and target NTs on the RHS are co-indexed by their subscripts, and \( p \) is (one of) the weight(s) associated with this rule\(^{10} \). The NTs on the RHS in the production rule can be recursively instantiated simultaneously for both the source and the target, by applying any rules in the grammar that have a matched NT on the LHS. Therefore, the SCFG has the advantage of capturing the hierarchical structure of natural language and also provides a more principled way to model reordering, e.g., the PP and VP in

---

9 Galley and Manning (2010) also introduce gaps on the target side but report that most of the benefits are due to the source gaps. In addition, they describe a neat algorithm of how to explicitly extract phrase pairs with gaps, which can also be used to add phrase pairs with source gaps in \( M \).

10 We often omit \( p \) in this paper when it is clear from contexts.
the above example will be reordered regardless of their span lengths in the sentence.

State-of-the-art SCFG-based models also fit into the log-linear model in Eq. (8), with features decomposed to the level of SCFG rules. The \( p \) shown in the above equation is an example of such features. In recent years, SCFG-based models (e.g., Chiang, 2007; Galley et al., 2004; Shen et al., 2008) have shown promising improvements on language pairs involving significant word reordering in translation. There are two major elements accounting for such an improvement: the incorporation of phrase translation structures adopted from phrase-based models into SCFG rules to handle local fluency, and the engagement of SCFG to enhance non-local reordering that is difficult to model in a finite-state framework.

In contrast to the exponential complexity of full reordering in finite-state models, the SCFG-based formalism is preferred for its polynomial complexity. For example, the complexity of parsing with a rank two SCFG (i.e., allows up to two NT’s on the RHS, a.k.a. binary SCFG) is cubic in the input length\(^{11}\). In terms of expressiveness, although there are examples (Wu 1997; Wellington et al., 2006) of reordering that cannot be covered by a binary SCFG, it is arguably sufficient in practice since these examples are rarely observed in real data.

As an active research area in SMT, there are various approaches proposed to model the transduction rules used in the SCFG family. Some approaches define \( \gamma \) and \( \alpha \) as a pair of strings that (a) are either terminals or NTs, called an inversion transduction grammar (ITG) (Wu, 1997); (b) employ a combination of terminals and NTs, known as hierarchical phrase-based models (Chiang, 2007). Alternatively, it is possible to incorporate syntactic parse (constituent or dependency) tree fragments on the source or target of the transduction. Depending on which side the tree fragment is added, the models range from source tree to target string (Quirk et al., 2005; Huang et al., 2006), source forest to target string (Mi et al., 2008), source string to target tree (Galley et al., 04; Shen et al., 08), and source tree to target tree (Eisner 2003; Cowan et al., 2006; Zhang et al., 2008; Chiang 2010). Therefore, SCFG-based translation models can be further categorized into two classes, according to their dependency on annotated data. Following Chiang (Chiang, 2007), we note the following distinction between these two classes:

1. Linguistically syntax-based: models utilize structures defined over linguistic theory and annotations (e.g., Penn Treebank), and SCFG rules are derived from the parallel corpus that is guided by explicitly parsing on at least one side of the parallel corpus. This class of models includes all of these mentioned above that utilize tree fragments directly in the SCFG rules.

2. Formally syntax-based: models are based on the hierarchical structure of natural language but synchronous grammars are automatically extracted from the parallel corpus without using any linguistic knowledge or annotations. Both ITG (Wu, 1997) and hierarchical models (Chiang, 2007) fall into this category.

By employing linguistic analysis on the source or target and capturing multi-level syntactic tree fragments in transductions, linguistically syntax-based models have the potential to incorporate more refined linguistic constraints in selecting translation equivalents and combining them together towards a full translation, which tends to be more syntactically well-formed. However, in practice and particularly for ST, such models are often less robust than hierarchical phrase-based models for the following reasons. Firstly, they inevitably suffer from noise in the automatic parsing of the corpus and the translation inputs. Secondly, some useful phrasal structures are discarded due to extra constraints in forming tree fragments (DeNeefe et al., 2007). Thirdly, more refined constraints sometimes prevent applying useful rules due to a slight mismatch. Fourthly, parsing of speech faces more obstacles. Even for well-studied languages such as English, parsing accuracy degrades significantly on spontaneous spoken inputs that tend to be informal and often ungrammatical.

In contrast, hierarchical models do not rely on the availability and parsing performance of a linguistic parser, and may fit the ST task better. Hierarchical models exploit the hierarchical structure of natural language but utilize only a unified NT symbol, \( X \), in the grammar \( \mathcal{R} \). Capitalizing on the strengths of phrase-based models, all phrase pairs are included as rules (with only terminals on the RHS), such that \( PT \subset \mathcal{R} \), which refer to \textit{phrasal rules} in the form of:

\[
X \rightarrow \text{< 初步试验, initial experiments >}
\]

The model assumes a binary SCFG, to balance the expressiveness of the grammar and computational complexity during decoding. For example,

\[
X \rightarrow < X_1, The X_2 of X_1 >
\]

Again, the co-indexed subscripts on the RHS indicate the one-to-one correspondence of source and target NTs. We refer to rules with any NT on the RHS as \textit{abstract rules}. Finally, the following two rules, known collectively as the \textit{glue rule}, are embedded in the grammar\(^{12}\) to concatenate sub-translations sequentially into the sentence start \( S \):

\[
S \rightarrow < S_1, X_2, S_1, X_2 >
\]

The parameterization of SCFG models determines which derivation \( d \) is a more desirable translation than the others. All SCFG rules in \( \mathcal{R} \) are paired with statistical parameters (i.e., the grammar is weighted), which are combined with other features \( h_m(e_i, f_i, d) \) (Sec. III) that are decomposable at the SCFG rule-level to score the translation hypothesis using the general log-linear framework of Eq. (8).

Combining the strength of FST-based phrase models (i.e., phrasal and glue rules) with hierarchical structures (i.e., abstract rules), the hierarchical string-to-string system is robust and highly competitive in translation accuracy, and thus a potentially good option for ST. In terms of runtime speed, however, phrase-based and tree-to-string (Huang and Mi, 2010) systems usually have the advantage of being faster in practice. Another noticeable weakness is the strong tendency of over-generalization of SCFG rules, since no constraints are attached to the \( X \) used throughout the grammar. Adding linguistically motivated constraints can alleviate this issue to a certain degree, and much research proposed to address this

\(^{11}\) In practice, finite-state based search still runs faster than SCFG-based models due to the usage of limited reordering windows and the beam pruning.

\(^{12}\) If the SCFG grammar has only phrasal and glue rules, this SCFG model degrades to FST-based monotonic phrasal SMT.
issue, which can be viewed as a third category: a hybrid of the two approaches mentioned above. In this family, formal syntax models are augmented with linguistic syntax in different ways, either as enriched features tied to NTs (Zhou et al., 2008b, Huang et al., 2010), or soft constraints (Marton et al., 2008), or refined non-terminals with direct linguistic annotations (Zollmann and Venugopal, 2006).

It is also possible to represent SCFG-based models using pushdown automata (PDA), a generalization of FST, and thereby leverage efficient algorithms of PDA operations to optimize the search space (Iglesias et al., 2011).

4.3 Learning of SCFG-based Models

After obtaining phrase tables from an aligned parallel corpus, we can loop through each phrase pair and generalize the sub-phrase pair it contains with co-indexed NT symbols, thereby obtaining an abstract rule. The number of NTs in each rule is limited to no more than two, thus ensuring the rank of the SCFG is two. To reduce the size of rules and spurious ambiguity, constraints described in (Chiang, 2007) are usually applied. During such a rule extraction procedure, note that there is a many-to-many mapping between phrase pairs and derived abstract rules. In other words, one original phrase pair can induce a number of different abstract rules, and the same rule can also be derived from a number of different phrase pairs. The conditional probabilities in both directions for SCFG rules are estimated, similar to what was described in Sec. 3.3.1, except that there are no real counts for these abstract rules observed in the training data. This issue was addressed by using heuristic counts. The training procedure described in (Chiang, 2007) employs heuristics to hypothesize a distribution of possible rules. A count of one is assigned to every phrase pair occurrence, equally distributed among rules that are derived from the phrase pair. Under this assumption, relative-frequency estimates are used to obtain the bi-directional conditional probabilities of SCFG rules.

One may argue, however, that these parameters are often poorly estimated due to the use of inaccurate heuristics. Huang and Zhou (2009) employ an EM procedure to directly estimate these probabilities. Their approach forces-aligns the parallel training corpus using a SCFG grammar estimated as above, storing the likely derivations in a forest, and then iteratively updates the probabilities using expected counts obtained from an inside-outside algorithm. Later the work was extended in (Cmejrek and Zhou, 2010) to propose additional SCFG rules, using so-called rule arithmetic to include rules that are not extractable following the original SCFG extraction procedure (Chiang, 2007).

To extract SCFG rules for linguistically syntax-based models, as shown in Fig. 3, we first perform syntactic parsing on at least one side of the parallel corpus. Next, the translation rules are extracted along with the parsing structures. There are algorithms (Galley et al., 2004; Galley et al., 2006) and (Shen et al., 2008) proposed to extract translation rules with constituency and dependency structures, respectively.

V. TRANSLATION DECODING

Decoding is a problem that concerns how to efficiently search for the best translation \( \hat{e}_t \) that maximizes Eq. (8) for a given foreign sentence \( f_t \). Any output sentences that can be transduced from \( f_t \) by iteratively applying the translation equivalents defined in Sec. IV are considered a valid option. Usually, there are many options that compete with each other. To make search efficient, similar to ASR decoding, dynamic programming (DP) is widely used in SMT decoding. The principle of DP is divide-and-conquer with reusable sub-solutions: break the bigger problem into sub-problems that are smaller and easier to solve, whose sub-solution can be shared between across partial solutions to the original problem, and thus only need to be solved once.

The well-known Viterbi search algorithm in HMM-based ASR systems is a typical instance of DP. In contrast to ASR decoding where the input is scanned left-to-right and the output is also generated left-to-right, the decoding problem of SMT is further complicated by reordering operations required to produce outputs from the input. Moreover, compared to the dominance of HMM in ASR, there is a wide variety of translation models as reviewed in Sec. IV, each of which may call for its own decoding algorithm that matches its individual modeling assumption (e.g., finite-state vs. SCFG-based). Instead of describing specific decoding algorithms in detail for each type of translation model, we will present various decoding algorithms from a unified perspective, which thus reveals a close connection between Viterbi decoding\(^{15}\) for ASR, FST-based translation, or even SCFG-based translation, despite their dramatic differences upon first glance. Specifically, both decoding algorithms for ASR and FST-based translation can be viewed as instances of the Viterbi algorithm on a weighted graph with the semiring framework (Mohri, 2002), and we can further generalize it to the hypergraph decoding for SCFG-based models (Huang, 2008).

In Sec. 5.1 we will first review the graph theory and the generic Viterbi algorithm for directed acyclic graph (DAG), which includes the decoding of speech recognition as a special case, followed by the introduction of hypergraph and generic Viterbi algorithm on directed acyclic hypergraph (DAH)\(^{14}\). As concrete examples of the generic algorithms, we present decoding algorithms for both phrase-based SMT and the Folsom architecture (Zhou et al., 2006) in Sec. 5.2, followed by Viterbi decoding for SCFG models in Sec. 5.3. Pruning for all of these search procedures is discussed in Sec. 5.4.

5.1. Graph, Hypergraph, and Generalized Viterbi

The search space of any finite-state automata, such as the one used in HMM-based ASR and FST-based translation, can be represented by a weighted directed graph \( \Sigma = (V, E, \Omega) \), which includes a vertices set \( V \) and edges set \( E \), with a weight mapping function \( \Omega: E \rightarrow \Psi \) that assigns to each edge a weight from \( \Psi \), a set defined in a semiring \( ((\Psi, \otimes, 0, 1)) \). Here we

\(^{15}\) We focus on 1-best decoding in this paper though k-best decoding can be generalized with our abstract representation here. Further information about k-best decoding can be found in (Jelineck, 1999; Huang and Chiang, 2005).

\(^{14}\) (Huang, 2008) is a source of further reading on this topic.
assume that there is a single source vertex \( s \in V \) in the graph. A path \( \pi \) in a graph \( G \) is a sequence of consecutive edges \( \pi = e_1e_2\cdots e_l \) such that the end vertex of one edge is the start vertex of the subsequent edge in the sequence. The weight of a path is obtained by \( \Omega(\pi) = \otimes_{i=1}^{l} \Omega(e_i) \), and the shortest distance (i.e., lowest weight) from \( s \) to a vertex \( q \), \( \delta(q) \), is the “\( \oplus \)-sum” of the weights of all paths from \( s \) to \( q \), and we define \( \delta(s) = \mathbb{I} \).

Viewed from the angle of weighted directed graph, many search problems can be converted to the classical shortest distance problem in the DAG (Cormen et al., 2001). For example, an HMM-based ASR decoder can be viewed as a distance problem in the DAG (Cormen et al., 2001). For integrated transducer (Mohri et al., 2002), as shown in Sec. 2.2, search problems can be converted to the classical shortest path problem in a graph.

The introduction of the hypergraph is not only of theoretic interest, but also provides a unification of the concept of “path” in a hypergraph and the “derivation” of an HMM. The search space expanded over CFG models can be abstracted by a hypergraph with arity two.

Similarly to the concept of using a graph to represent the relation between vertices, the concept of a hypergraph can be defined in a semiring \( (\Psi,\otimes,\emptyset,\mathbb{I}) \). Each hyperedge \( e \in E \) is also a triplet \( e = (T(e), h(e), f_e) \), where \( h(e) \in V \) is the head vertex of \( e \), \( T(e) \in V^* \) is an ordered list of tail vertices \(^{15}\), and \( f_e \) is a mapping \(^{16}\), \( f_e: \Psi^{\{T(e)\}} \to \Psi \).

Here we require \( f_e \) to be monotonic on each of its \( |T(e)| \) arguments (to ensure that DP is possible). We call \( |T(e)| \) the arity of hyperedge \( e \), and the arity of \( H \) is the maximum arity of all hyperedges.

The definition suggests that a regular graph can be viewed as a special class of hypergraph with arity one. The search space that a decoder explores using a binary SCFG-based translation model amounts to a hypergraph of arity two.

A generalized concept of “path” in a hypergraph is known as a derivation \( d \) of a vertex \( q \), which is a sequence of consecutive hyperedges connecting source vertices to \( q \). The weight \( \Omega(d) \) of the derivation \( d \) is recursively computed by applying their weight functions to each of its hyperedges. As a special case, a weight function may be of the following form:

\[
\Omega(d) = f_e(\Omega(d_1), \ldots, \Omega(d_{|T(e)|}))
\]

(19)

where \( \Omega(e) \) is the weight of the hyperedge, \( d_1 \) is a derivation of the 1st tail vertex that constitutes a portion of \( d \), and \( \Omega(d_i) \) is its weight. Similarly, the “best” weight of a vertex \( q \) is the “\( \oplus \)-sum” of the weights of all of its derivations. That is:

\[
\delta(q) = \begin{cases} 
\mathbb{I} & \text{if } q \text{ is a source vertex} \\
\oplus_{d} \Omega(d) & \text{otherwise}
\end{cases}
\]

The best derivation is a target vertex with the best weight. To search for the best derivation, we can use the generic Viterbi search shown in Fig. 8 that is almost identical to Algorithm 1, assuming the hypergraph is acyclic.

One of the differences, compared to Algorithm 1, is that we now have multiple source vertices \( s_1^1 \), and another noticeable difference lies in line 4 and 5, considering that each hyperedge may have multiple tail vertices. Rather than following each edge from start to end as in Algorithm 1, Algorithm 2 locates the head vertex first and then follows the hyperedge in reverse order to identify its tail vertices. The reason why this is possible is that while \( H \) is usually unknown and constructed on-the-fly during search, the topological order is typically predefined. Such an order ensures that all tail vertices are visited earlier than the head vertex. The complexity of Algorithm 2 is also \( O(|V| + |E|) \).

15 A head vertex of a hyperedge with an empty tail list is a source vertex.

16 Note that in principle hypergraph has the flexibility to define a distinct mapping function for each hyperedge as long as it satisfies certain requirements (e.g., being monotonic and superior).

Algorithm 1: Generic Viterbi Search of DAG

<table>
<thead>
<tr>
<th>Procedure Viterbi ((\Sigma, s))</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Initialize ( \Sigma, s )</td>
</tr>
<tr>
<td>2. Topological sort vertices of ( \Sigma )</td>
</tr>
<tr>
<td>3. for each vertex ( p \in \Sigma ) in topological order do</td>
</tr>
<tr>
<td>4. for each edge ( e ) such that ( \text{start}(e) = p ) do</td>
</tr>
<tr>
<td>5. ( q = \text{end}(e) )</td>
</tr>
<tr>
<td>6. ( \delta(q) \oplus = \delta(p) \otimes w(e) )</td>
</tr>
</tbody>
</table>

Fig. 7: Pseudo code of Viterbi algorithm of DAG

Algorithm 2: Generic Viterbi Search of DAH

<table>
<thead>
<tr>
<th>Procedure Viterbi ((H, s_1^1))</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Initialize ( H, s_1^1 )</td>
</tr>
<tr>
<td>2. Topological sort the vertices of ( H )</td>
</tr>
<tr>
<td>3. for each vertex ( q \in H ) in topological order do</td>
</tr>
<tr>
<td>4. for each hyperedge ( e ) such that ( \text{head}(e) = q ) do</td>
</tr>
<tr>
<td>5. ( {p_1, \ldots, p_{</td>
</tr>
<tr>
<td>6. ( \delta(q) \oplus = f_e(\delta(p_1), \ldots, \delta(p_{</td>
</tr>
</tbody>
</table>

Fig. 8: Pseudo code for Viterbi Search of DAH

Equipped with the concept of hypergraphs, widely-used DP algorithms for CFG parsing, such as the Cocke-Kasami-Younger (CKY) algorithm operating on Chomsky Normal Form (CNF), can be viewed as an instance of the generic Viterbi search in a hypergraph of arity two.
Different types of CKY (e.g., bottom-up vs. top-down) result from different realizations of topological order sorting of line 3 in Algorithm 2.

5.2. Viterbi decoding for FST-based MT

Many FST-based MT decoders described in the literature can be viewed as instances of generic Algorithm 1. Below we review two decoders proposed for phrasal SMT.

For the first example, Moses (Koehn et al., 2007) is a Viterbi search with beam-pruning on the following graph $\Sigma$. The (single) source vertex of $\Sigma$ represents the state where none of the source words has been translated and the translation hypothesis is empty. Each edge leaves its start vertex to the end vertex by choosing a consecutive set of source words, which have not been covered yet, to apply one of the source-matched phrase pairs; the target side of the phrase pair is appended to obtain a new hypothesis. The weight of each edge is determined by the model, as discussed in Sec. IV, usually a combination of weights from the phrase-based translation model, a distortion model and language models etc. Such edges are expanded consecutively until the target vertex $t$ arrives, which is the state where all the source words have been covered. The best translation is thus retrieved from the best path to $t$ with the lowest weight $\delta(t)$.

With the help of the concept of a graph, it is straightforward to understand an important process in the search known as hypothesis recombination. In phrase-based SMT (Koehn et al., 2007), the key idea is that at any instance during the search, among all the partial hypotheses that share the same signature (i.e., having the same set of source words being covered, with the same n-1 last words to use an n-gram LM in the current hypothesis), we only need to keep the one with the lowest weights and all others can be discarded. This is because from a scoring perspective, only the partial hypothesis with the lowest cost (a sum of negative log probabilities) has a possibility to become the best hypothesis in the future. Hence, hypothesis recombination loses no information for the Viterbi (1-best) search. In the graph that represents the search space, it is clear that all the partial hypotheses with the same signature arrive at the same vertex, and all future paths leaving this vertex are indistinguishable afterwards. Thus, with the “ $\bigoplus$ -sum” operation (i.e., the min in the Tropical semiring), only the partial hypothesis with the lowest cost is kept.

The search algorithm (Zhou et al., 2007, Zhou et al., 2008) of Folsom is another instance of Algorithm 1. The search space is expanded by an on-the-fly three-way composition of the three-layers of WFSTs described in Sec. 4.1, as depicted in Fig. 9 (for simplicity, the weights associated with each arc are not shown). At initialization (line 2), only the source vertex, corresponding to a composition of the start state from each WFST $(F_0, M_0, G_0)$, is active with zero accumulated weight. Following line 4 and 5 in Algorithm 1, subsequent vertices are visited following current and active vertices in topological order, by traversing each edge as follows.

For each active vertex $p$ with the composite state of $(F_p, M_p, G_p)$, we first advance $F_p$ in the transducer $F$ following an arc leaving $F_p$ with label $f$. Then, look at all outgoing arcs of $M_p$ labeled with the current input word $f$ on the input, and traverse one of these arcs, advancing $M_p$. Then, given the output label $w$ of this arc, look at all outgoing arcs of $G_p$ with $w$ as its input\(^1\), and traverse one of these arcs, advancing $M_L$.

The red states in Fig. 9, for example, construct such an active vertex in the search graph. The set of all states $\{(F_p, M_p, G_p)\}$ that are reachable in this way is then added to the set of current vertices replacing the $(F_p, M_p, G_p)$. The weight of the edge $e$ connecting $(F_p, M_p, G_p)$ to $(F_q, M_q, G_q)$ is the $\bigoplus$-product of the arc weights in each WFST that are associated with the transition:

$$\Omega(e) = \bigoplus_{m \in \{F, M, G\}} \lambda_m \Omega(e_m)$$

where $\lambda_m$ is the feature weight as in Eq. (8). In practice, $\Omega$ is further refined with other desirable features such as word and phrase counts etc.

Any active vertices are merged whenever they have identical $(F_p, M_p, G_p)$, and the remaining component states are inherited from the ones with lower overall cost. This is free from search errors in finding the 1-best path, and equivalent to hypothesis recombination described above. Still, the number of edges that need to be explored can conceivably grow to be quite large. Further pruning is needed to speed up search, as discussed in Sec. 5.4.

The target vertices corresponding to complete translations are those from which each component state is a final state in each individual WFST (indicated as bold circles in Fig. 9). Thus, any path in the graph that connects the source to one of the target vertices corresponds to a complete translation which is generated by collecting the output label of each arc along the path. The best translation is the one with the lowest accumulated weight.

Fig.9: An instance of generic Viterbi search on multi-level graph for WFST-based translation

From the viewpoint of standard FST operations, the decoding algorithm above can be viewed as an optimized version of on-demand three-way composition combined with an on-demand minimization, and followed by best-path search.

\(^1\) During these match processes in machine $M$ and $G$, any $\epsilon$-transitions must also be handled correctly.
One potential disadvantage of this decoding algorithm, compared to more conventional ones (Koehn et al., 2007), lies in the increased difficulty of parallelizing the search or storage when both $M$ and $G$ are too big to fit into one computing node for web-scale tasks.

5.3. Viterbi decoding for SCFG models

The decoding of SCFG-based translation models largely follows the chart parsing process (e.g., CKY). For the particular hierarchical translation model with a binary SCFG, Chiang (2007) described a generalized CKY algorithm that does not require converting the SCFG into CNF. We present the equivalent decoding process below as the Viterbi traversal on a hypergraph. This unified view is not only of theoretic importance, but also helps in implementing a decoder that is more flexible to be extended, packs more translation options more effectively, and improves its pruning process (Huang and Chiang, 2007).

![Fig. 10: Hypergraph decoding without LM](image)

Using a hypergraph for SCFG-based translation, we need to label each hyperedge with a transduction from source (input) to target (output) language with an equal number of co-indexed NTs. Fig. 10 shows an example of such a hypergraph.

Decoding starts with a construction of source vertices, by applying the phrasal rules (i.e., no NTs on the RHS) that match any consecutive sequence of input words. To simplify our discussion, we first do not consider incorporating language models on the output side, as shown in Fig. 10. In this case, the vertices are labeled with the NT of the LHS of the applied rule, added by the span of what has been covered so far on the input, i.e., in the form of $[X,i,j]$, where $i$ is the start position and $j$ is the end position. Next, each hyperedge connecting tail(s) to a head is explored following Algorithm 2. This is achieved by applying one of the matching rules in the SCFG grammar whose arity has to equal to the number of co-indexed NTs on the rule’s RHS. For the example shown in Fig. 10, two intermediate vertices, labeled $[X,0,2]$ and $[X,3,4]$, can be used together to match the rules with “$X_1$ of $X_2$” on the input; the two different translation options of “$X_1$ of $X_2$” on the output lead to two distinctive hyperedges being applied to reach the vertex of $[X,0,4]$. The best derivation weight $\delta(q)$ at this head vertex is updated according to line 7 of Algorithm 2. The topological sort of line 3 ensures that vertices with shorter spans $j - i$ are visited earlier than those with longer spans, so the above process can be repeated until the target vertex $[S,0,J]$ is reached. The best derivation can be found by back-tracing from the target vertex, and the output of each hyperedge is collected as the best translation hypothesis.

The Viterbi search on a hypergraph to find the 1-best translation without a LM is thus quite efficient. The number of hyperedges in the hypergraph of a binary SCFG is $O(|R|^3)$, and so is the decoding complexity since it is linear to $O(|E|)$.

One difference from the search procedure of FST-based translation models is that the translation hypotheses of SCFG models are not guaranteed to generate the output in left-to-right order. For the example shown in Fig. 10, a partial hypothesis that starts with “initial experiments” later grows into “the success of initial experiments”. Therefore, adding an n-gram LM into the search is intuitively more challenging. Since there are NTs on the output side, we can no longer apply n-gram LM scores on each edge as we did in Sec. 5.2, since the required n-1 word history maybe undetermined, unless the contextual information is encoded at each tail vertex to keep track of possible boundary words within each NT, when following a hyperedge.

Note that decoding SCFG models with a LM essentially requires the intersection of the target side of a SCFG grammar with a FSM, which is guaranteed to be a CFG (Hopcroft and Ullman, 1979). Therefore, the search space of the decoder can still be represented by a hypergraph.

![Fig.11: Hypergraph decoding with bigram LM](image)
Even though we are dealing with a much larger hypergraph now, the generic decoding Algorithm 2 can still be applied here, with LM scores factored in to the update in line 7. However, the worst-case time complexity is multiplied by \( O(|T|^4(n-1)) \), where \(|T|\) is the target vocabulary size, compared to the search in Figure 10. In general, such complexity is too expensive and therefore pruning is essential.

5.4. Pruning

For the search procedures discussed in Sec. 5.2 and 5.3, the number of paths that can be expanded in the graph or hypergraph, even with the help of DP, is enormous (being exponential for the unconstrained reordering in the graph and a high-order of polynomial in the hypergraph, on the input size \(|J|\)). To reduce the size of the search space, it is necessary to prune paths/derivations that are unlikely to become the best one in the future.

The concepts of graph and hypergraph can also help understand the pruning process. To prune a graph, we can (a) prune some of the edges leaving a start vertex (line 5 of Fig. 7), or (b) discard some end vertices (line 4 of Fig. 7). The first pruning strategy can be achieved by applying certain reordering constraints (Zens and Ney, 2004) or by discarding some higher-cost phrase-based translation options. This is known as the beam pruning commonly used in both phrase-based SMT (Koehn et al., 2007) and ASR decoding. The second pruning strategy is done by first sorting the vertices in a group sharing the same coverage vector based on \( \delta(q) \) and then skipping certain vertices that are either outside a beam or outside the top \( k \) list in the group (where \( k \) is the preset histogram pruning threshold). We should note that these kinds of pruning, unlike hypothesis recombination, may lead to search errors.

As a concrete example, for the Viterbi search of Fig. 9, beam or histogram pruning can be applied at either the individual layer or during composition. In addition to the usual reordering constraints, in order to practically limit the size of reordering layer, we also expand the reordering graph in a lazy fashion with beam pruning. That is, the reordering states are composed with translation and language model states dynamically during the search of Eq. (18), and only those reordering states that have promising joint scores will be kept. Specifically, some of the arcs in \( F \) become inactive if the accumulated weight along the path exceeds a certain threshold. This allows us to perform an effective reordering pruning using the joint scores from all models at an early stage to reduce the search space dramatically. In addition, beam and histogram pruning are also performed to check all current active vertices in the set \( \{F_q, M_q, G_q\} \), and any vertex \( q \) is removed from the list when \( \delta(q) \) is outside of beam or histogram threshold. The same beam or histogram pruning can also be applied to a hypergraph.

The histogram pruning can be made much more efficient with a lazy pruning technique, generally called cube pruning, proposed in (Huang and Chiang, 2005 and 2007).

The insight of lazy pruning is as follows. Assume that we need to compute a list of weighted items, and we know that only the top \( k \) of these items will be kept afterwards. In this case, to speed up the computation, we should avoid calculating these items if we know that they fall outside of the top \( k \). This is possible if the item weights are computed from a function that is monotonic in each of its arguments, and if the input list for each argument is already sorted. Chiang (2007) employs this idea for SCFG translation models, and Huang and Chiang (2007) apply it to phrase-based models, achieving significant decoding speed improvements for both models.

Specifically, for the hypergraph-based Viterbi decoding, cube pruning is applied to Eq. (19), to determine what hyperedges can be discarded without expanding them. Suppose there is a hyperedge group \( \{e_j\}\) (i.e., every \( e_j \) in the group share the same source side and identical tail vertices), called the hyperedge bundle and we visit (line 5 in Algorithm 2) each hyperedge \( e \) in this group in a particular order. We push \( e \) into a priority queue that is sorted by the weight assigned by \( f_e \), \( \Psi|\{e\}| \rightarrow \Psi \). The hyperedge \( e \) popped from the priority queue is explored. This process stops when the top \( k \) hyperedges are popped from the priority queue, and whatever hyperedges remain in the bundle are discarded without any computation.

The name cube pruning comes from the fact that the weight mapping function shown in Eq. (19) has three arguments, two corresponding to the two tail vertices (i.e., \( T(e) = 2 \)) used in binary SCFG models, and the other for the hyperedge weight \( \Omega(e) \). The general idea actually can be applied to situations with any number of arguments. However, it should be noted that cube-pruning is not free of search errors, since the weight mapping function in Eq. (19) is no longer strictly monotonic when the language model score is incorporated. To reduce the need for more aggressive pruning, Iglesias et al. (2009) compactly encode the partial hypotheses as WFSTs during search.

VI. COUPLING ASR AND SMT FOR ST

From a Bayesian perspective, the speech translation task is to find the best translation \( \hat{e}^*_1 \) as follows (Ney, 1999):

\[
\hat{e}^*_1 = \arg\max_{e_1} P(e_1^*|x_T^1) = \arg\max_{e_1} \left\{ \sum_{f_1^1} P(e_1^1|f_1^1)P(x_T^1|f_1^1)P(f_1^1) \right\} \approx \arg\max_{e_1} \left\{ \max_{f_1^1} P(e_1^1|f_1^1)P(x_T^1|f_1^1)P(f_1^1) \right\} \approx \arg\max_{e_1} \left\{ P[e_1^1]|\max_{f_1^1} P(x_T^1|f_1^1)P(f_1^1) \right\}
\]

The first approximation in Eq. (20) replaces the sum with the max, which is a common practical choice in many search problems. The second approximation further assumes that \( f_1^1 \) is determined only by \( x_T^1 \), and can therefore be solved as an isolated problem first. This is the foundation of the cascaded approach discussed above. Simple and practically effective, the cascaded approach however is impaired by the compounding of errors propagated from ASR to SMT, which degrades overall speech translation performance. Conversely, speech translation as a whole could be improved if the interactions between ASR...
and SMT are taken into account, with better coupled components. This coupling can be achieved via joint decoding and more coherent modeling across components, as described below.

6.1. Joint decoding

Joint ST decoding can be either tight, i.e., a fully integrated search over all possible $e_i^l$ and $f_i^l$, or loose, i.e., by approximating the full search space of $f_i^l$ with a promising subset. The first approximation in Eq. (20) suggests that a fully integrated search of $e_i^l$ and $f_i^l$ maximizes the product of a direct translation model $P(e_i^l|f_i^l)$, an acoustic model $P(x_i^T|f_i^l)$ and a source language model $P(f_i^l)$, which gives rise to a highly complex search problem. With a unified WFST approach to both ASR and SMT, fully integrated search is practically feasible for some simplified translation models. Matusov et al. (2006) constructed a WFST-based word-level quasi-monotonic joint translation model $P(e_i^l,f_i^l)$ and substituted it for the usual source language model used in the ASR WFSTs in Eq. (5) to produce speech translation in the target language. Monotonic joint translation models are explored at the phrase-level in (Casacuberta et al., 2008).

Fully integrated search using a more powerful phrase-based log-linear translation model $P(e_i^l|f_i^l)$ can also be achieved. For example, within the Folsom architecture (Sec. 4.1), fully integrated search is to extend the ASR WFSTs in Eq. (5) by further composing the $M$ and $G$ from Eq. (17) in an online fashion. The limitation, however, is that the reranking can only occur within a phrase.

Considering the enormous size of the fully integrated search space, loose coupling is more often used in practice. The promising subsets have been defined at different levels, including N-best ASR hypotheses (Zhang et al., 2004), word lattices produced by ASR recognizers (Matusov et al., 2006; Mathias and Byrne, 2006; Zhou et al., 2007), and confusion networks i.e. CN (Bertoldi et al., 2008). A word lattice is a DAG that can be represented by a WFSA. Note that the N-best (a union of N linear left-to-right graphs) and the CN (a graph that requires every valid path traverses all nodes in the graph) are both simplified special cases of a word lattice.

The N-best approach is a straightforward extension to the SMT of 1-best ASR but it is not very effective, since SMT decoding complexity grows linearly with $N$ and the quality of the ASR promising set only increases slowly with $N$. In contrast, word lattices efficiently encode large numbers of competing ASR hypotheses, and there are efficient algorithms (Zhou et al., 2007) to translate word lattices. The decoding algorithm is similar to the one illustrated in Fig. 9, except replacing the dynamically constructed reordering WFST with a static word lattice. However, reranking is a challenge to model and search effectively in word lattices, given that various paths between two nodes may traverse a vastly different number of intermediate nodes leading to divergent reranking costs. The CN-based approach (Bertoldi et al., 2008) is a more compelling approach to address the reordering issue for phrase-based ST, since any path connecting any given two nodes in CN traversed the same set of intermediate consecutive nodes. The CN-based decoding algorithm is extended from (Koehn et al., 2007) such that a hypothesis is expanded by translating paths from yet uncovered consecutive nodes (as opposed to consecutive source words as in the 1-best case) in the CN. Note that the number of paths inside the consecutive nodes is exponential in the number of nodes spanned. The efficiency of retrieving phrasal translations for all paths is addressed by representing phrase tables in prefix trees and using incremental pre-fetching (Bertoldi et al., 2008).

SCFG-based models can also be effectively applied for joint ST decoding with lattice inputs, based on a generalized CKY algorithm for translating lattices (Dyer et al., 2008). Having the nodes in the input lattice appropriately numbered such that the end node is always numbered higher than the start node for any edge, the vertex in the hypergraph is now labeled with $[X,i,j]$: here $i < j$ are the node numbers in the input lattice (as opposed to source word positions as in the 1-best case) that are spanned by $X$. Indeed, if greater reranking flexibility is critical for the joint ST decoding of lattice inputs, it is more convenient to employ the SCFG-based models in SMT, since reranking is modeled by the NT transduction of SCFG rules, and thus avoids traversing the lattice to compute the distortion costs required by FST-based models.

6.2. Joint modeling with end-to-end ST criteria

To design integrated ST, in addition to coupling ASR and SMT during decoding, one may desire to further optimize all component models involved in Eq. (20) using a combined ST evaluation criterion (e.g., BLEU), i.e., joint modeling.

Joint modeling for ST is theoretically appealing but also a lofty task. It has only recently started attracting more attention (Zhang et al., 2011; He and Deng, 2012), although there are empirical studies in earlier work showing potential benefits of heuristic approaches motivated along this line. For example, Zhou et al. (2007) interpolate the ASR LM $P(f_i^l)$ with a LM trained using every source phrase in the phrase table weighted by the phrasal translation probabilities corresponding to each target phrase. The recognizer was thus fine-tuned to favor hypotheses that the SMT has higher confidence to translate, and it was found to improve ST performance though not necessarily the WER. With a similar motivation to factor in the interactions between ASR and SMT, Zhang et al. (2011) more formally formulate the estimation of ASR LM probabilities (as well as the SMT LM and phrase table probabilities). Their models are trained to minimize the classification errors of assigning a pair $(e_i^l, f_i^l)$ to a given source speech $x_i^T$, and requires the $e_i^l$ in the correct classification pair must have the highest utterance-level BLEU score among all competitors being considered. He and Deng (2012) continued to sketch a theoretical framework that learns phrase translation parameters by optimizing the sum of expected utterance-level BLEU for ST using the speech translation posterior $P(e_i^l, f_i^l |x_i^T)$. Their work improves over (Zhang et al., 2011) by using an objective (BLEU) that is more relevant to ST’s evaluation criteria, and an optimization method based on EBW that has faster
convergence than the steepest gradient method used in (Zhang et al., 2011).

It is worth noting that one of the challenges of joint modeling is scalability to large tasks, hindered by the need for training data $X_l^T$ labeled with both $f_l^j$ and $e_l^j$, as well as the high computational costs to train models from large-scale data.

VII. SUMMARY AND DISCUSSION

This paper surveyed and analyzed state-of-the-art statistical machine translation techniques for speech translation. Taking a unified perspective that draws the connections and contrasts between ASR and SMT, we showed that phrase-based SMT can be viewed a sequence of finite-state transducer operations, similar in concept to ASR, with reordering operations that can be dynamically incorporated. We further analyzed the structures of the synchronous context-free grammar (SCFG) based formalism that includes hierarchical phrase-based and many linguistically syntax-based models for SMT. Decoding for ASR, FST-based and SCFG-based translation were also presented from a unified perspective as different realizations of the generic Viterbi algorithm on a weighted directed acyclic graph or hypergraph. These particular perspectives are helpful to catalyze a tighter integration of ASR and SMT for improved ST, for which we discussed both joint decoding and modeling.

Ongoing and future advances in ASR and SMT will certainly continue improving ST. Additionally, various other research directions may lead to usability or accuracy improvements for real world ST applications, which are often orthogonal to isolated progress in ASR and SMT. In concluding this paper, we list some of these research directions as examples:

- Overcome the low-resource problem: The main challenge includes how to rapidly develop ST systems with a limited amount of training data (e.g., Zhou et al., 2013); how to adapt (e.g., Foster et al., 2010) from one task to another that differs in domain or applications, and how to adapt SMT systems for ST.

- Confidence measurement: It is critical to reliably measure the confidence of speech translation outputs such that interactive ST applications can avoid misleading dialogue turns and lower the risk of miscommunication. This task is greatly complicated by the combination of two sources of uncertainty in ASR and SMT.

- Active/passive ST: As an agent to facilitate cross-lingual communications, a ST system needs to become more active when it is appropriate, e.g., to actively clarify with the user or warn the user when low confidence is detected. This requires the development of suitable dialogue strategies for ST.

- Context-aware ST: The deployment of ST technologies on smart phones is an opportunity for more research in context-aware (e.g., location, conversation history) ST.

VIII. ACKNOWLEDGEMENT

The author would like to thank Xiaodong He for his contributions to an early version of Sec. II and III, and to thank Graeme Blackwood, Stanley Chen, Kenneth Church, Kevin Knight, Haitao Mi, Michael Riley, and Libin Shen for their insightful comments. I am also grateful to the anonymous reviewers and the editor Haizhou Li for their thorough review and constructive comments.

IX. REFERENCES


Biography: Bowen Zhou (Senior Member, IEEE) received the B.E. degree from the University of Science and Technology of China, Hefei, Anhui, China, in 1996, the M.E. degree from Chinese Academy of Sciences, Beijing, China, in 1999, and the Ph.D. degree from the University of Colorado at Boulder, Boulder, CO, USA, in 2003, all in Electrical Engineering. He is a Research Manager and Research Staff Member with the IBM Thomas J. Watson Research Center, Yorktown Heights, NY, USA, where he has been heading the Department of Machine Translation and Learning, and previously, the Department of Speech-to-Speech Translation. He has also served as a Principal Investigator of the DARPA TRANSTAC and Transformative Apps Programs. In these roles, he has been responsible to advance end-to-end speech-to-speech translation technologies spanning areas of speech recognition, natural language processing, machine translation, etc. He is currently leading a research team to collaborate and compete in machine translation with top researchers around the world under the DARPA BOLT Program. Adding to his research efforts, he was also involved in developing and deploying speech and machine translation systems for various real-world applications and platforms, ranging from smartphones to clouds. Among other things, he and his team created the MASTOR, a multilingual two-way real-time speech-to-speech translator that is completely hosted on smartphones. He has published over 60 papers in top conferences and journals. He has a broad interest in statistical models, machine learning, and human language technologies, including both speech and text.

Dr. Zhou is a member of the Association for Computational Linguistics (ACL), and he is an elected member of the IEEE Speech and Language Technical Committee, serving as an Area Chair for ICASSP since 2012. He also served as an Area Co-Chair of Machine Translation of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL/HLT), and an invited panelist at the 2010 NAACL/HLT. He has also served as committee members and session chairs for major speech and language conferences, professional meetings, and workshops.