Title	A monotonic statistical machine translation approach to speaking style transformation
Author(s)	Neubig, Graham; Akita, Yuya; Mori, Shinsuke; Kawahara, Tatsuya
Citation	Computer Speech & Language (2012), 26(5): 349-370
Issue Date	2012-10
URL	http://hdl.handle.net/2433/157359
Right	© 2012 Elsevier Ltd.
Type	Journal Article
Textversion	author

A Monotonic Statistical Machine Translation Approach to Speaking Style Transformation

Graham Neubig[a,**], Yuya Akita[a], Shinsuke Mori[a], Tatsuya Kawahara[a,*]

[a] *Graduate School of Informatics, Kyoto University, Yoshida Honmachi, Sakyo-ku, Kyoto, Japan*

Abstract

This paper presents a method for automatically transforming faithful transcripts or ASR results into clean transcripts for human consumption using a framework we label *speaking style transformation* (SST). We perform a detailed analysis of the types of corrections performed by human stenographers when creating clean transcripts, and propose a model that is able to handle the majority of the most common corrections. In particular, the proposed model uses a framework of monotonic statistical machine translation to perform not only the deletion of disfluencies and insertion of punctuation, but also correction of colloquial expressions, insertions of omitted words, and other transformations. We provide a detailed description of the model implementation in the weighted finite state transducer (WFST) framework. An evaluation of the proposed model on both faithful transcripts and speech recognition results of parliamentary and lecture speech demonstrates the effectiveness of the proposed model in performing the wide variety of corrections necessary for creating clean transcripts.

Keywords: rich transcription, speaking style transformation, disfluency detection, weighted finite state transducers, monotonic machine translation

*Corresponding author

**Principal corresponding author

Email addresses: neubig@ar.media.kyoto-u.ac.jp (Graham Neubig), akita@ar.media.kyoto-u.ac.jp (Yuya Akita), forest@i.kyoto-u.ac.jp (Shinsuke Mori), kawahara@i.kyoto-u.ac.jp (Tatsuya Kawahara)

1. Introduction

One of the major goals of automatic speech recognition (ASR) is to be used in the automatic or semi-automatic creation of transcripts for human consumption. In order to achieve this goal, ASR-based transcription systems must be able to create transcripts that are similar to those created by humans assigned to do the same task. While conventional ASR systems are generally designed to faithfully reproduce utterances word-for-word, faithful transcripts are generally not suitable for human consumption. It has been shown that significant editing of faithful transcripts is necessary to create text that is aesthetically pleasing and easy to read [1].

This editing is necessary due to the spontaneous nature of speech. As speakers are generally planning what they say next as they speak, in many cases they will use fillers to buy time to think about what they want to say next, or go back to repeat or correct previously spoken information. Stenographers consistently remove these *disfluencies* [2] from final transcripts[1], and there has been significant amounts of research into systems for disfluency detection and removal.

However, even in apparently fluent speech, there is still a large disconnect between spoken and written language. The most obvious difference is the lack of punctuation in speech, which must be replaced to allow for readable written text [1]. In addition, there is often a large amount of redundancy in speech, which is added by speakers to attract listeners' attention and allow them to keep up with the speakers pace more easily. For example, the phrase "this, you know, is serious business" would be edited into "this is serious business" in a parliamentary record to remove the contentless discourse marker "you know" that is simply added for emphasis. This redundant information can be removed in transcripts, as readers may read transcripts at their own pace.

Additionally, there are significant stylistic differences between speech and written text that are corrected in transcripts. For example, colloquial expressions are transformed into written-style expressions, dialects are normalized into standard text, and omitted function words are re-inserted.

This paper first investigates the types of editing necessary to create natural written transcripts from faithful transcripts or ASR results, and then

[1] In fact, many stenographers say they do not even "hear" disfluencies when making transcriptions.

proposes a system that is able to automatically perform these edits. In particular, using corpora of Japanese parliamentary speeches, we examine the type of transformations necessary to create formal transcripts, focusing on not only disfluencies, but also on stylistic transformations. Based on the results of this study, we present an approach to *speaking style transformation* (SST), a general framework to automatically convert faithful text or ASR results into transcript-style text, for use in automatic transcription systems.

We adopt techniques from statistical machine translation (SMT), reformulating the model to match the requirements of the SST task. Using this re-formulated model, we introduce a number of improvements over traditional noisy-channel models such as log-linear weighting, context-sensitive translation modeling, and an efficient implementation based on weighted finite state transducers (SMT).

We evaluate the model on a large corpus of official transcripts from the Japanese Diet (national parliament), using both faithful transcripts and ASR results as input. In this evaluation, we find that the proposed SMT-based method is able to accurately conduct both disfluency correction and stylistic transformation. We also find that the proposed modeling techniques result in significant improvements over a conventional noisy-channel model.

The rest of this paper is organized as follows: In Section 2, we describe the task of speaking style transformation in detail. We define the type of edits we need to handle in order to create transcript-style text from spoken language, and provide examples and a detailed tabulation of different correction types for a corpus of the Japanese parliament. In Section 3, we provide a survey of previous work on the topic, describing to what extent each work covers the edits necessary to create clean transcripts.

In Section 4, we describe the framework of monotonic statistical machine translation (MSMT) that we adopt for this task. We formalize this model using weighted finite state transducers (WFSTs), which allows for efficient processing, as well as the coupling of SST with WFST-based ASR systems. In Section 5, we describe an extension of the traditional noisy channel model, allowing the model to handle context. By properly handling context, we expect to perform significantly better on context-sensitive edits such as the deletion of discourse markers, or the insertion of dropped words. In Section 6, we describe a number of additional features tailored to the SST task that were incorporated in a log-linear model.

In Section 7, we present results for two evaluation tasks on transcript creation for spontaneous Japanese in a semi-spontaneous, formal context.

We investigate a number of different statistical models, and examine the effect of data size on the effectiveness of each model.

Finally, in Section 8 we conclude the paper with a retrospective on what the proposed model was able (and not able) to do, and provide a look at possible future directions.

2. Speaking Style Transformation

In speaking style transformation (SST), the input to a system is faithful transcripts or ASR results that lack punctuation and contain spoken language phenomena rendering them inappropriate for formal transcripts. The desired output of the system is clean text, punctuated and free of extraneous or colloquial expressions, that is appropriate for a written record. This clean text output is desirable for both human readability [1] or later machine processing such as machine translation [3, 4].

2.1. Types of Transformations Performed by Stenographers

As our goal is to create transcripts that are of similar quality to those created by human stenographers, it is useful to examine the kind of edits performed by human stenographers in the creation of transcripts. For this purpose, we created the following categorization, which is roughly based on transcription guidelines drafted by stenographers working for local governments in Japan.

In general, the edits performed by stenographers can be split into three phases, simple editing, intermediate editing, and semantic checking, with several separate types of edits being performed at each phase.

1. **Simple editing** includes the deletion of disfluencies, insertion of dropped function words, correction of colloquial and dialectal expressions, and other relatively simple edits.
   (a) *Removal of fillers:* Removal of fillers that are used exclusively to buy time while the speaker plans his/her next utterance. Fillers include words such as "um" in English or "*e-tto*" in Japanese.
   (b) *Removal of discourse markers:* Discourse markers include words such as "well" in English or "*desune*" in Japanese, which are used to buy time in some contexts, but also are parts of actual fluent speech in other contexts, and thus cannot be deleted 100% of the time.

(c) *Insertion of dropped words:* In colloquial speech, function words such as postpositions in Japanese or neutral verbs in English are often dropped [5]. These words must be recovered to create grammatical transcripts.

(d) *Correction of colloquial or dialectal expressions:* There are many expressions that are generally acceptable in speech, but are changed to more appropriate written language in transcripts. For example, "she was like" is often changed into "she said" in English [5]. In addition, in many languages, a particularly notable case of which being Arabic [6], dialectal speech is generally converted into a standardized form in transcriptions.

(e) *Removal of extraneous expressions:* In many cases, a speaker will start speaking with the words "Mr. Chairman" or ask questions such as "how much time do I have left?" These contentless expressions are removed in the process of creating cleaned transcripts.

(f) *Removal of repeats and repairs:* Words are often repeated ("the Federal, Federal reserve board"), or repaired ("the FR..., Federal Reserve Board") by speakers. In transcripts, the earlier part (*reparandum*) of the repeat or repair is deleted or merged with the later part. It should be noted that repeats or repairs also often contain word fragments, as in the second example.

(g) *Insertion of punctuation or line breaks:* Punctuation and line breaks improve readability by allowing for the division of text into coherent units. As punctuation marks and line breaks are not explicitly present in speech, they must be inserted during the transcript creation process.

2. **Intermediate editing:** considers the grammar and sentence structure of the text.

(a) *Correction of out-of-order words or phrases:* In spontaneous speech, speakers will often forget to say something at the beginning of a sentence, and add it on later as an afterthought. These words or phrases are re-ordered into a more natural position, where they would occur if the speaker had the ability to edit his or her utterance. For example, "he will be going tonight, probably" becomes "he will probably be going tonight."

(b) *Correction of run-on sentences:* In spontaneous speech, particularly formal spontaneous speech, many speakers will use long sentences with no clear break point. These run-on sentences can

Table 1: The size of the Japanese parliament corpus used for analysis of correction types.

Words in Faithful Transcript		418k
Words in Clean Transcript		379k
Percentage of Words Corrected		12.87%
Punctuation	Periods	7917
	Commas	22833

be split into multiple sentences for improved readability. This often includes both the insertion of punctuation and the deletion of a conjunction such as "and" or "but."

(c) *Syntactic check:* Correction of mistaken grammar such as incorrect tense, plural forms, or substituted function words such as articles, prepositions, or particles.

3. **Semantic check** considers the actual meaning of the text. This mainly consists of the correction of factual errors, as well as common technical terms, or quotes that are not spoken 100% accurately. Another less common example is the redacting of politically offensive words or confidential information.

2.2. An Empirical Analysis of Correction Types

We performed an analysis of the prevalence of each type of correction required using transcripts from parliamentary meetings. The reason why we chose parliamentary meetings specifically is because they represent an important target of transcription systems and require a relatively large number of stylistic transformations that cannot be classified as disfluencies. Specifically, we analyzed a corpus from the House of Representatives of the Diet (national parliament) of Japan [7].

We selected a number of committee meetings, which consist of semi-spontaneous speeches and question-and-answer sessions, purposely avoiding the plenary sessions for lack of spontaneity. Text from the official record of the Diet was used as clean transcripts, and faithful transcripts were prepared by workers contracted for the task. Details on the corpus size can be found in Table 1 and the number of each type of correction can be found in Table 2. A number of interesting observations can be gleaned from this data,

Table 2: A break-down of corrections in the Japanese parliament corpus for simple, intermediate, and syntactic editing. Categories that consist mainly of either deletions (del), insertions (ins), or substitutions (sub) are marked accordingly.

Level	Type	Instances	Instance %	Words	Word %
Simple	Filler (del)	19554	50.05%	19596	35.76%
	Discourse Marker (del)	9238	23.64%	13393	24.44%
	Repeat/Repair (del)	2342	5.99%	4501	8.21%
	Extraneous (del)	672	1.72%	1515	2.76%
	Colloquial (sub)	2465	6.31%	9297	16.97%
	Word Insertion (ins)	3024	7.74%	3098	5.65%
	Total	37295	95.46%	51400	93.80%
Inter.	Syntax Check	679	1.74%	1086	1.98%
	Reordering	765	1.96%	1632	2.98%
	Run-on Sentence	136	0.35%	190	0.35%
	Total	1580	4.04%	2908	5.31%
Semantic	**Total**	195	0.50%	492	0.90%

particularly when compared to another detailed analysis of corrections made by annotators in English telephone conversations by Fitzgerald [5][2].

First, it can be seen that the parliamentary speech we are analyzing is significantly more fluent than the English telephone speech. This is reflected by the fact that only 12.87% of all words for the parliamentary speech are corrected, as opposed to 34.4% reported for telephone speech. It is also notable that the percentage of repeats and repairs is significantly lower: only 8.21% of corrected words are attributed to repeats, repairs, and false starts, while 44.2% of corrections in telephone speech are attributed to these categories.

Another interesting aspect of the parliamentary data is the correction of simple, yet non-disfluent phenomena. The most significant examples of these corrections are the transformation of colloquial expressions (16.97%), and insertion of dropped words (5.65%). While together these result in 22.62% of corrections, they fall outside of the traditional target of previous disfluency

[2]Punctuation insertion is omitted from our analysis to allow for a direct comparison with Fitzgerald [5].

detection systems. While these also exist in English telephone conversation, they account for only 3.0% of all corrections. The first reason for this difference is that transcripts for parliamentary speech are expected to be in correct, formal style, and thus require a larger number of edits to correct colloquial or syntactically incorrect sentences. This phenomenon is also mentioned by Fitzgerald [5] with regards to a corpus of European parliament speech. In addition, in Japanese (and other languages such as Arabic [6] and Czech [8]) there is a larger disconnect between spoken and written language [9] than in English. As a result, there are a greater number of colloquial expressions to be corrected, or omitted words to be re-inserted.

Based on this analysis, we choose to develop a system that is able to handle the most common types of corrections found in parliamentary speech. Particularly, we focus on simple editing, and develop a framework that can handle all simple edits, with the exception of long-distance repairs. Even if we exclude repeats and repairs, which are the main focus of previous work in RT (described in more detail in the next section), the remaining simple edits account for 85.58% of all edited words in this task.

3. Related Work

The transformation of spoken language to written language has been the subject of a large amount of research over the past two decades. In general SST is one part of the larger task of "rich transcription" [10].

3.1. Disfluency Detection

With regards to SST in particular, the great majority of work has focused on the detection and deletion of disfluencies. These works can be classified by the features they use, and the techniques they use to model these features.

Early research includes the work by Bear et al. [11], which used lexical features in a rule-based framework, with simple prosodic features such as pauses being used to reduce the number of errors. Nakatani and Hirschberg [12] further expanded the use of prosody, integrating a number of different prosodic features in a decision-tree framework, demonstrating that a certain level of disfluency detection is possible without the use of lexical information.

Over the past ten years, there have been advances in both machine learning techniques and the features used. A number of methods using lexical features such as repeated words, deletion history, and bigram context have been introduced, in combination with machine learning techniques such as

boosting, transformation-based-learning, or noisy-channel models [13, 14, 15, 16, 17]. The work of Johnson et al. [18] is particularly notable for their use of Tree Adjoining Grammars (TAG) to find edits using a noisy-channel model, allowing for the discovery of repairs through rough matching.

In addition, many recent works integrate both prosodic and lexical features. Liu et al. [19] integrated prosodic and lexical features for disfluency detection and sentence boundary detection using hidden Markov models (HMMs), maximum entropy classifiers, and conditional random fields (CRFs). They found that CRFs were able to achieve slightly superior performance, with system combination further improving results. Yeh and Wu [20] used a language-model based approach that uses prosodic features to detect the interruption point, and an alignment model to match repair regions. Fitzgerald [5] also used CRFs to detect false-starts, incorporating both prosody and the model proposed by Johnson et al. [18], and demonstrated a two-step approach to first identify, then process erroneous sentences leads to higher accuracy.

3.2. Punctuation Insertion

In addition to disfluency detection, there has also been a large amount of research on detecting sentence boundaries and inserting punctuation. A number of works integrate lexical and syntactic features with pause and other prosodic information. These are incorporated using language models [21, 22, 23], maximum entropy models or re-rankers [24, 25], or SVMs [26]. Paulik et al. [27] use an SMT-based approach to recover punctuation by simply deleting punctuation from the original data. Gravano et al. [28] tested the effect of very large data sets on punctuation and capitalization restoration, and found that large data allowed for significant improvements in common punctuation such as periods and commas, but was less effective for question marks and dashes.

3.3. Formatting and Speaking Style Transformation

Despite the extensive research on the previously mentioned tasks, there has been relatively little research into general frameworks that are also able to perform the stylistic correction necessary to create natural transcript-style text. One example of an approach that is able to handle insertions and substitutions in speaking style transformation is Lee and Seneff [29], which describes a system for automatic correction of non-native English speakers' grammar. In addition, there have been a few works on converting Arabic

spoken dialects to Modern Standard Arabic text [30, 31], but these use rule-based approaches that are not immediately applicable to other languages. Hori et al. [32] presented a method for paraphrasing speech that uses hand-crafted rules rescored by a language model, and a similar approach is presented by Shugrina [33] in the context of formatting numbers and punctuation for voice-mail transcripts.

The first work towards full speaking style transformation with the goal of creating lecture transcripts was presented by Shitaoka et al. [34]. They used a noisy-channel model to handle filler deletion, transformation of colloquial expressions, insertion of particles, and insertion of periods, but stopped short of a general framework for handling all necessary transformations. In addition, their implementation of the naive noisy channel model was not fully probabilistic, using a number of heuristics to cope with small data sizes.

4. Monotonic Statistical Machine Translation for SST

This section describes a model for SST based on monotonic statistical machine translation (SMT). *Monotonic* SMT requires that the input and output of the system remain in fundamentally the same order. In other words, while elements can be inserted, deleted, or substituted, arbitrary reordering of elements is not allowed.

As shown in Section 2.2, only 1.96% of corrections and 2.98% of words require reordering, so this monotonicity assumption holds for the majority of corrections performed by human stenographers. By making this monotonicity assumption, it is possible to achieve effective and efficient modeling without the complex alignment and reordering models that are necessary for non-monotonic tasks such as traditional bilingual machine translation.

4.1. Modeling for Monotonic SMT

SST is performed by transforming the faithful transcript or ASR results $V = v_1^J = v_1, v_2, \ldots, v_J$, into the transcript-style text $W = w_1^I = w_1, w_2, \ldots, w_I$. SMT creates a model for the posterior probability $P(W|V)$, the probability of a particular target sequence W given a source sequence V. With this model, SMT searches for the sequence $\hat{W}$ that maximizes this probability, and returns it as the output for V

$$\hat{W} = \underset{W}{\operatorname{argmax}} P(W|V). \tag{1}$$

Bayes' law is used to decompose $P(W|V)$ into the translation model (TM) probability $P_t(V|W)$ and language model (LM) probability $P_l(W)$

$$\begin{aligned}\hat{W} &= \underset{W}{\operatorname{argmax}} P(V|W)P(W)/P(V) \\ &= \underset{W}{\operatorname{argmax}} P_t(V|W)P_l(W). \end{aligned} \tag{2}$$

In most cases, the parameters of these models are estimated from a parallel corpus. In particular, the TM must be trained on this sort of parallel corpus of faithful and clean transcripts (hereafter probabilities that require a parallel corpus for training are indicated by the subscript t). However, because the LM does not need to model V, a larger non-parallel corpus containing only clean transcripts can be used for training of the LM (indicated by subscript l).

Models decomposed in this manner are often called *noisy-channel* models, and are, as described in the previous section, the basis of many existing methods for SST. It should be noted that the main difference between the proposed monotonic SMT method and traditional bilingual SMT [35, 36] lies in the construction of the TM, while the modeling techniques used in the LM are shared between the two approaches. The following two sections provide a formal description of these two models.

4.1.1. Language Modeling

The most prevalent method for calculating the LM probability is the n-gram model. The n-gram model is based on the idea that the overall probability of a word sequence can be modeled sequentially without any loss of generality

$$P_l(W) = \prod_i P_l(w_i|w_1^{w-1}).$$

N-gram models attempt to mitigate the problem of data sparsity by limiting the history length, only conditioning the probability on the preceding $n-1$ words

$$P_l(W) = \prod_i P_l(w_i|w_{i-n+1}^{i-1}).$$

To prevent the model from assigning zero probability when the word w_i has never been observed in context w_{i-n+1}^{i-1}, models are further generalized by applying smoothing techniques [37]. In the following experiments, Kneser-Ney smoothing [38] is used unless otherwise indicated.

4.1.2. Translation Modeling

As there is no clear way to estimate the TM probability $P_t(V|W)$ for the entire sequence simultaneously, the relationship between the two sequences is modeled on a finer phrase-based level.

In our monotonic SMT framework, this is done by first creating a segmentation model (SM) $P_t(\tilde{W}|W)$ to segment word sequence W into phrase sequence $\tilde{W} = \tilde{w}_1, \tilde{w}_2, \ldots, \tilde{w}_K$. $P_t(\tilde{W}|W)$ can be estimated with a corpus of phrase-aligned word sequences. The segmentation probability is then set to be proportional to this phrase LM probability

$$P_t(\tilde{W}|W) \propto \prod_k P_t(\tilde{w}_k|\tilde{w}_{k-n+1}^{k-1}).$$

Next, a phrase-based TM $P_t(\tilde{V}|\tilde{W})$ calculates the conditional probability of a phrase sequence $\tilde{V} = \tilde{v}_1, \tilde{v}_2, \ldots, \tilde{v}_k$ in the source language given the target language phrase sequence $\tilde{W}$. The simplest way to approximate this translation probability for the full sequence is to assume that each phrase translation is independent of the surrounding translations and take the product of the phrase-to-phrase translation probabilities

$$P_t(\tilde{V}|\tilde{W}) \approx \prod_k P_t(\tilde{v}_k|\tilde{w}_k). \tag{3}$$

The phrase-to-phrase translation probabilities can be estimated using the maximum likelihood (ML) criterion over the phrase-aligned training corpus. A special empty phrase ϵ of length zero can be used to handle insertions and deletions.

As a unique V exists for each $\tilde{V}$, the full translation probability between V and W can be realized by the product of these two probabilities

$$P_t(V|W) = P_t(\tilde{V}|\tilde{W})P_t(\tilde{W}|W). \tag{4}$$

In order to obtain the phrase-aligned corpus necessary for training these models, we first performed word-based alignment to minimize the edit distance between V and W over a turn-aligned corpus. Then the expectation-maximization (EM) algorithm was used to align phrases so that the TM likelihood is maximized (Equation (3)) over the remaining unmatched sections. Once the aligned corpus has been obtained, it can be used to create an n-gram language model over the set of $\tilde{W}$ phrase sequences.

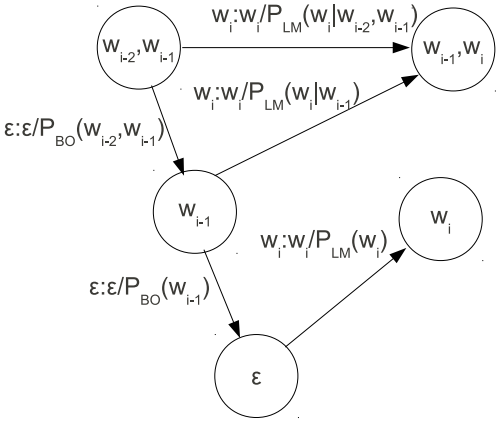

Figure 1: An example of the LM transitions that may be followed when calculating $P_{LM}(w_i|w_{i-2}^{i-1})$. P_{BO} indicates the n-gram backoff probability.

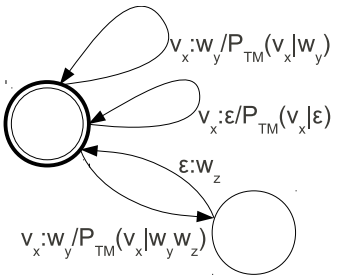

Figure 2: An example of TM transitions for one-to-one, one-to-zero, and one-to-many transformations.

4.2. Implementation of Monotonic Translation using WFSTs

While the techniques in the previous section describe a method to model the output sequence that has the largest probability of being a translation for V, we still need a mechanism to search the space of possible W for the most probable sequence $\hat{W}$. Monotonic translation lends itself to description using weighted finite state transducers (WFSTs), a framework that allows for flexible model combination and efficient optimization [39]. Once the TM, LM, and SM have been combined into one large model, a decoding algorithm is used to search for $\hat{W}$ given an input V.

4.2.1. WFSTs and Model Construction

Finite state transducers (FSTs) are an expansion of traditional finite automata with transitions labeled with both input and output symbols. WFSTs further expand FSTs by assigning a weight to transitions, allowing for the definition of weighted relations between two strings.

The LM, TM, and SM used in monotonic SMT can be represented as WFSTs by creating edges that represent a single word or translation pairs, and setting the weight of the edge to the appropriate LM, TM, or SM probability. Examples of the representations of the LM and TM as WFSTs are shown in Figures 1 and 2 respectively. A bold circle indicate the initial state of the WFST, while double circles indicate final states in Figure 2 (Figure 1 displays only part of an WFST, so the initial and final states are not marked).

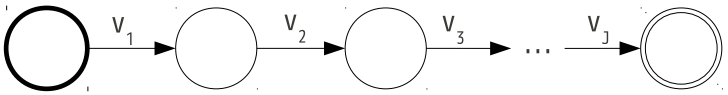

Figure 3: A finite state automaton encoding an input V.

The ϵ appearing in both figures is the aforementioned null string. ϵ transitions can always be followed regardless of the input, and thus allow for the representation of the LM smoothing operations.

After each model has been represented as an individual transducer, the models can be combined using WFST operations that are equivalent to functional composition ($X \circ Y$) or intersection ($X \cap Y$) [39]. In particular, the composition is useful, as it allows for the creation of input-output cascades. For example, if the composition operation $TM \circ SM \circ LM$ is performed, it is possible to create a single unified WFST that transforms $\tilde{V}$ directly into W and assigns a probability of

$$P_t(\tilde{V}|\tilde{W})P_t(\tilde{W}|W)P_l(W) \tag{5}$$

to each candidate W. If this is further composed with a simple transducer to enumerate all possible segmentations of V into $\tilde{V}$, a transducer that transforms directly from input V to output W with the appropriate probability can be obtained.

Another benefit of using WFSTs is that efficient algorithms for operations such as determinization and minimization exist. These algorithms can be used to automatically reduce the amount of memory space and decoding time required for the model.

4.2.2. Monotonic Translation Decoding

Once the model is represented as an optimized WFST, we can use the model to find the optimal translation for an input sequence V. This can be done in three steps:

1. Create a linear finite state automaton (FSA) to represent the input V (Figure 3).
2. Compose the input FSA and the model WFST, which creates the space that must be searched to find the appropriate $\hat{W}$ for V.
3. Use the Viterbi algorithm to find the optimal path through the search space. The output symbols along this path and the weight of the path are $\hat{W}$ and $P(\hat{W}|V)$ respectively.

This strategy is simple, and can be used to find the optimal solution for small to medium sized models. However, when searching larger models, full expansion of the search space in Step 2 and exact search in Step 3 require unrealistic amounts of memory and processing time.

In order to make Step 2 more efficient, it is possible to perform *lazy composition* [39]. Unlike regular composition, lazy composition only expands the parts of the composed WFST that are specifically referenced. This means that unnecessary expansion of highly unlikely sections of the search space will not be performed, reducing the number of operations performed in Step 2. Furthermore, search can be made more efficient by performing beam search, instead of Viterbi search. These two techniques allow for rapid search even for large models and long input sequences of up to several hundred words.

4.3. Comparison with Bilingual Machine Translation

In bilingual machine translation (MT), V and W represent sentences from two different natural languages. While the statistical models presented in the previous sections are largely similar to those applied in bilingual MT, translations between natural languages are generally non-monotonic.

Because of this, in order to create natural sentences in the target language, translations of each word in the input must be appropriately re-ordered into a natural ordering in the target language. This is achieved by replacing the SM of the previous section with an *alignment model*, which tries to model the different orderings of the words in the two languages [35, 36].

Finally, it should be noted that while WFSTs can be used for monotonic SMT, they cannot be straightforwardly applied to traditional bilingual SMT[3]. WFSTs, like all finite automata, are only able to express monotonic string relations, and have no mechanism to handle arbitrary reordering. As a result, the search problem of traditional SMT is significantly more difficult than that of monotonic SMT[4]. This is reflected in processing times of the systems, as demonstrated in Section 7.

[3]Casacuberta and Vidal [40] proposes a WFST-based model for speech translation with limited reordering. However, phrase-based SMT with more sophisticated alignment models was found to have superior performance on most translation tasks [41]. In addition, Zhou et al. [42] describe a method for using WFSTs for speech translation that requires specific reconstruction of reordering candidates for each sentence to be translated.

[4]It is NP-complete to find the optimal solution for traditional translation models with reordering [43], while the monotonic assumption makes search achievable in linear time with regards to the input length.

5. Context-Sensitive Translation Modeling for Monotonic SMT

Section 4.1.2 introduced a method for estimating translation model probabilities by assuming that each phrase TM probability is context independent (Equation (3)). However, in many cases the appropriate translation of a word is actually highly context dependent, a phenomenon that is particularly pronounced in SST. For example, whether the word "well" is used as a filler or not depends highly on context. While the LM probability helps ensure some degree of natural output, it can be expected that incorporating context information directly into the TM probability will result in improved modeling accuracy.

In Section 5.1, a previously reported method for incorporating this context information is described. In Section 5.2, we extend the method so that context information can be included in the TM for use in the noisy-channel framework.

5.1. Joint Probability Translation Model

One method for expressing context directly in the TM is through the direct modeling of the joint probability $P_t(V, W)$ [40]. By limiting the search space so that only results where V is the source sequence are returned,

$$\hat{W} = \underset{W}{\operatorname{argmax}} P(V, W)$$

is ensured to give the same result as Equation (1).

The joint probability can be modeled using the monotonic alignments described in Section 4.2. The phrase-segmented source sequence $\tilde{V}$ and target sequence $\tilde{W}$ are represented as a string of symbols $\Gamma = \gamma_1^K = \gamma_1, \ldots, \gamma_K$, where γ_k is a symbol representing $\langle \tilde{v}_k, \tilde{w}_k \rangle$. For notational convenience, we also define two functions over $\gamma = \langle \tilde{v}, \tilde{w} \rangle$:

$$\psi_v(\gamma) \equiv \tilde{v}, \psi_w(\gamma) \equiv \tilde{w}.$$

Given these monotonic alignments, it is straightforward to create a smoothed n-gram model trained over a corpus of Γ strings. $P_t(\tilde{V}, \tilde{W})$ is approximated using this model according to the standard n-gram equation:

$$P_t(\tilde{V}, \tilde{W}) = P_t(\Gamma) \approx \prod_{k=1}^{K} P_t(\gamma_k | \gamma_{k-n+1}^{k-1}). \tag{6}$$

5.2. Conditional Context-Sensitive Model

While the joint probability model provides an effective way to handle context, it must be trained on a parallel corpus. This leaves no room for use of large-scale non-parallel data through the LM probability $P_l(W)$, which can be easily incorporated using the standard noisy-channel model. This section describes a technique for approximating a context-dependent TM probability from joint probabilities. This allows for creation of a model that can both consider context when choosing translation probabilities, and use non-parallel data to compensate for sparsity in the parallel corpus.

We first note that $P_t(\tilde{V}|\tilde{W})$ can be modeled sequentially:

$$\begin{aligned} P_t(\tilde{V}|\tilde{W}) &= \prod_{k=1}^{K} P_t(\tilde{v}_k|\tilde{v}_1^{k-1}, \tilde{w}_1^K) \\ &= \prod_{k=1}^{K} P_t(\tilde{v}_k|\gamma_1^{k-1}, \tilde{w}_k^K). \end{aligned}$$

This model faces the same problem of sparseness as traditional n-gram models, so an n-order Markov model is used to limit the length of the considered history

$$P_t(\tilde{V}|\tilde{W}) \approx \prod_{k=1}^{K} P_t(\tilde{v}_k|\gamma_{k-n+1}^{k-1}, \tilde{w}_k^K).$$

Further, we assume that $\tilde{v}_k$ does not depend on any target sequence symbol $\tilde{w}_h$ where $h > k$

$$P_t(\tilde{V}|\tilde{W}) \approx \prod_{k=1}^{K} P_t(\tilde{v}_k|\gamma_{k-n+1}^{k-1}, \tilde{w}_k). \tag{7}$$

This has a double-effect of helping to reduce data sparseness and facilitating a WFST implementation.

Equation (7) can be further factored as follows:

$$P_t(\tilde{V}|\tilde{W}) \approx \prod_{k=1}^{K} \frac{P_t(\gamma_k|\gamma_{k-n+1}^{k-1})}{P_t(\tilde{w}_k|\gamma_{k-n+1}^{k-1})}. \tag{8}$$

The denominator of this equation can be obtained by marginalizing over the n-gram probabilities for γ where $\psi_w(\gamma) = \tilde{w}_k$

$$P_t(\tilde{w}_k|\gamma_{k-n+1}^{k-1}) = \sum_{\gamma \in \{\gamma:\psi_w(\gamma)=\tilde{w}_k\}} P_t(\gamma|\gamma_{k-n+1}^{k-1}). \tag{9}$$

Because the numerator of Equation (8) and each element in the sum of Equation (9) have the same form as the n-gram probabilities in Equation (6), $P_t(V|W)$ can be estimated using the joint n-gram probabilities. This context-dependent model for $P_t(V|W)$ can be used along with the LM probability in the noisy-channel model as in Equation (2). Details on the implementation of this model in the WFST framework can be found in Appendix A.

5.3. Log-Linear Interpolation with Joint Probabilities

While the aforementioned conditional model has the advantage of allowing for the usage of non-parallel text, it does not consider overall translation pattern frequency. For example, if there is a pattern $\gamma_x = \langle \tilde{v}_x, \tilde{w}_x \rangle$ with counts $c_t(\gamma_x) = 100, c_t(\tilde{w}_x) = 1000$, and a pattern $\gamma_y = \langle \tilde{v}_y, \tilde{w}_y \rangle$ with counts $c_t(\gamma_y) = 1, c_t(\tilde{w}_y) = 10$, both will be given the same conditional probability

$$P_t(\tilde{v}_x|\tilde{w}_x) = P_t(\tilde{v}_y|\tilde{w}_y) = 0.1$$

even though the less frequent γ_y may simply be the result of semi-random variance in sparse training data. Infrequent patterns are particularly unreliable when dealing with the output of ASR, which is highly inconsistent.

While $P_t(\tilde{v}_x|\tilde{w}_x)$ and $P_t(\tilde{v}_y|\tilde{w}_y)$ are equal, $P_t(\gamma_x)$ is 100 times larger than $P_t(\gamma_y)$. Thus, it can be seen that the joint probability contains information about translation pattern frequency that is not included in the standard conditional TM. The log-linear model framework [44] can be used to combine the LM, TM, SM, and joint probabilities, thus capturing this frequency information

$$\log P(\tilde{W}|\tilde{V}) \propto \lambda_{lm} \log P_l(W) + \lambda_{tm} \log P_t(\tilde{V}|\tilde{W}) + \lambda_{sm} \log P_t(\tilde{W}|W) + \lambda_j \log P_t(\tilde{V}, \tilde{W}). \tag{10}$$

Here, each λ is a weight of its respective model. The weight of all four λ valuse can be normalized so that the sum of the absolute values $\sum_\lambda |\lambda|$ is equal to one with no loss of generality.

Note that while setting $\lambda_j = 0$ is an extension to the naive noisy-channel model in Equation (2), setting $\lambda_{tm} = 0$ and interpolating only the joint

and LM probabilities is neither theoretically correct nor practical. From the theoretical standpoint, without $P_t(\tilde{V}|\tilde{W})$ it is impossible to derive $P(W|V)$, the posterior function that we are trying to optimize. Practically, a model created in this way over-aggressively deletes words, resulting in accuracy no better than the standard joint model. It is for this reason that the conditional model introduced in the previous section is necessary, even when interpolating with the joint probability.

An essential element in the training of the log-linear model described in Equation (10) is the training of the weights λ. In the experiments presented in the following section, minimum error rate training (MERT) is used to optimize these weights [45]. MERT uses the following iterative process:

1. **Initialization:** Weights are initialized to an appropriate value[5].
2. **n-best decoding:** A decoder finds the n-best list given the current weights, and saves the value that each model contributed to each candidate.
3. **Weight optimization:** Weights are adjusted to minimize the error rate within the n-best list.
4. **Iterate:** A stopping criterion is checked, and if it has been met, the training stops and outputs the current weights. If the stopping criterion has not been met, the training returns to Step 2. Here, we stopped training when the absolute change in weights was 0.0001, or 10 iterations had been reached.

This is a flexible and straightforward process that is able to minimize any arbitrary error measure.

In order to keep track of these weights in a WFST implementation, it is possible to store the contribution of each component to the overall weight of the arc. When an operation such as addition is performed on the weights of the arc, an identical operation is performed on every component of the arc as well. This allows for simple bookkeeping of the contribution of each element to a path's weight, which is used in the weight optimization step.

[5]In all experiments in this paper, the weights in Equation (10) were initialized to replicate the naive noisy-channel model ($\lambda_{lm} = \lambda_{tm} = \lambda_{sm} = 1, \lambda_j = 0$).

6. Additional Features for Speaking Style Transformation

When using a log-linear model as described in Section 5.3, it is possible to introduce additional features that may be useful for the task at hand by redefining Equation (10) as a generalized log-linear model:

$$\log P(\tilde{W}|\tilde{V}) \propto \sum_{n=1}^{N} \lambda_n f_n(\tilde{W}, \tilde{V}). \tag{11}$$

The first four feature functions correspond to the probabilities in Equation (10), while the remaining feature functions represent a novel set of lexical features that we derived specifically for the SST task.

6.1. Filler Dictionary

The first additional feature is motivated by the fact that many of the deleted words were common fillers. A 23-word filler list was created, and a *fixed penalty or bonus was added every time one of the fillers was deleted* (λ_f). It should be noted that traditional rule-based methods simply delete fillers from a list, and thus adding this feature guarantees that the log-linear model can achieve performance at least equal to that of rule-based systems.

6.2. Transformation Group Penalty

Because there is a tendency for transformed areas to appear in groups (strings of deleted fillers, etc.), it makes sense to add a penalty that contributes to the cohesion of these groups. By adding a *fixed penalty for each group of words translated* (λ_g), for phrases such as "I uh don't um do," instead of creating two translation groups by deleting the underlined words in "I <u>uh</u> don't <u>um</u> do," the system will prefer to delete a single block as in "I <u>uh don't um</u> do."

6.3. Transformation Type Penalty

We observed in preliminary experiments that deletion, insertion, and substitution transformations have a different level of difficulty. Deletions tend to be easier, and substitutions and insertions are more difficult because they are less frequent in the training data and have greater variety. The transformation type penalty adds a *separate penalty or bonus for each of the transformation types* (λ_d, λ_i, λ_s), allowing each type to be modified to an appropriate level of precision or recall based on its difficulty.

6.4. On the Incorporation of Prosodic Features

It should be noted that a large amount of previous research has found it useful to incorporate prosodic features for disfluency detection and sentence boundary annotation [19]. Here, we integrate information about the existence of pauses by inserting them as tokens into the word string, but do not handle any other prosodic features.

We have two reasons for doing so. First, this eases direct integration with existing speech recognition decoders, as most WFST-based decoders are able to annotate pause information, but do not explicitly handle prosody. Second, this removes an extra acoustic processing step, expediting and simplifying the transcript generation process as a whole. Regardless, comprehensive study of the integration into the current framework is left to future work.

7. Experimental Evaluation

In order to test the effectiveness of the proposed SST model, we conducted experiments on Japanese SST using the Diet corpus described in Section 2.2 and the Corpus of Spontaneous Japanese (CSJ [46]).

7.1. Experimental Setup

We created training and testing data for the two corpora. In both tasks, either ASR results or manually created faithful transcripts were used as input, and clean transcripts were used as output. For the Diet task, the official transcripts were used. For the CSJ task, which has only faithful transcripts, we commissioned human editors to create clean transcripts according to annotation guidelines. Evaluation was performed on test sets that were held out from both corpora.

The details of the corpora are shown in Table 3. Here, ASR error rate refers to the edit distance between the ASR results and faithful transcripts, while Manual and ASR Pre-SST Error Rates refer to the edit distance between the clean transcripts and the faithful transcripts and ASR results respectively. The word error rate (WER) between the system output and clean transcripts was used as an evaluation measure.

For the system using manual transcripts as input, a parallel corpus of faithful and official transcripts was used as TM training data. Likewise, when using ASR results as input, a parallel corpus of ASR results and official transcripts was used for training the TM. A system trained with manual transcripts was also tested, but it performed approximately 3% absolute

Table 3: The size of the corpora for SST.

Corpus		Diet	CSJ
LM Training		158M	181k
TM Training		2.31M	181k
Weight Training		66.3k	21.5k
Test Set		300k	11.4k
ASR Error Rate		17.10%	19.43%
Pre-SST Error Rate	Manual	18.62%	27.70%
	ASR	36.10%	36.49%

WER worse, largely because models trained on ASR results are better at inserting punctuation, as well as correcting homonyms and ASR errors. It should be noted that this result is contrary to Honal and Schultz [47], who found that noise introduced by ASR results in the training data degraded final accuracy. It is likely that the fact that we used a larger training set was able to absorb some of this noise. This result is particularly interesting, as it indicates that the system can be trained with no faithful transcripts. As clean transcripts are generally available in greater quantities than faithful transcripts, this allows the system to be trained with significantly larger amounts of data.

The ASR results used for both the training and testing of the system were created using the Julius decoder, version 4.1.2 [48]. The acoustic models used in ASR were trained specifically for each task, and are based on shared-state triphone HMMs with cepstral variance normalization, vocal tract length normalization, and minimum phoneme error training. HMMs of 5000 shared states and 32 mixture components were used for the Diet task, and HMMs of 3000 states and 16 mixture components were used for the CSJ task. For the CSJ, a Kneser-Ney smoothed 3-gram LM was trained for ASR using the faithful transcripts that are included with the corpus. For the Diet, a Witten-Bell smoothed 3-gram LM was trained using a large volume of clean transcripts, and adapted to the speaking style by adjusting n-gram frequencies [49]. The ASR error rate on the Diet task (17.10%) was slightly smaller than that on the CSJ task (19.43%).

All models for SST were represented as WFSTs, and composed, determinized, and minimized using the OpenFST toolkit [50]. Model weights were

Table 4: SST results (WER) for different TM n-gram orders

	Diet		CSJ	
TM Order	Manual	ASR	Manual	ASR
Pre-SST	18.62%	36.10%	27.70%	36.49%
1-gram	6.51%	21.84%	15.06%	25.70%
2-gram	5.33%	**20.99%**	14.71%	**25.03%**
3-gram	**5.32%**	21.09%	**14.64%**	25.26%

optimized using the MERT tools included in the Moses [51] toolkit. WFST decoding used depth-first beam search with histogram pruning, which we implemented and released as an open-source toolkit[6]. The decoder also supports the hierarchical ϕ transitions mentioned in Appendix A.

7.2. Effect of Context-Sensitive Translation Modeling

The first evaluation was performed to assess the effectiveness of incorporating context in the noisy-channel model (Equations (5) and (8)). Models were trained using TMs of orders 1-3, and the WER was compared over the Diet and CSJ tasks[7]. The results in Table 4 show that in all cases the context-sensitive 2-gram and 3-gram TMs exceeded the context-insensitive 1-gram TM in accuracy. This advantage was particularly obvious for the Diet corpus, which contains a relatively large amount of data for use in n-gram training.

In the evaluations on manual transcripts of both the Diet and CSJ corpora, the 3-gram models outperformed the 2-gram models, while the 2-grams outperformed the 3-grams when evaluated on ASR results. This is most likely due to the fact that the noise intrinsic in the ASR result training set caused the 3-gram context to be unreliable, preventing any increase in accuracy. In all the following experiments, 2-gram context will be used for the ASR transcripts, while 3-gram context will be used for manual transcripts.

[6]Available at `http://www.phontron.com/kyfd`.

[7]In preliminary experiments, 3-grams outperformed 4-grams in all cases, so the results of 4-grams are omitted.

7.3. Translation Model Type

This section compares the effectiveness of the three translation models proposed in this paper.

- **Noisy:** The context-sensitive noisy channel model of Equation (8)
- **Noisy LL:** The context-sensitive noisy channel model with log-linear weights for each element (Equation (10) with $\lambda_j = 0$)
- **Noisy+Joint:** The log-linear interpolation of the noisy channel and joint probability models of Equation (10)

In addition, the following three existing methods are also tested for comparison.

- **Baseline:** The noisy channel model using the traditional context-insensitive TM of Equation (3)
- **Joint:** The joint probability model of Equation (6)
- **Moses:** The open-source software package "Moses," an implementation of phrase-based SMT [51]. The default settings were used, with exception of word alignment, which was performed by the method proposed in this paper, as it achieved higher accuracies in preliminary experiments than Moses's standard word alignment program "GIZA++."

The WER of each system is shown in Table 5. In every test situation, the proposed **Noisy+Joint** model resulted in the highest accuracy. In all tasks, this was a statistically significant difference from **Baseline** according to the two-proportions z test (significance $p < 0.01$). It can thus be concluded that the proposed method is effective over both manual transcripts and ASR results, as well as over the different styles of speech represented by the Diet and CSJ corpora.

The **Joint** model was able to achieve a higher accuracy than the **Noisy** model in the case of manual transcripts, likely a result of its clean design and inclusion of frequency information as mentioned in Section 5.3. However, with respect to ASR results, the **Noisy** model achieved a higher accuracy, as the LM was able to help reduce the number of clearly ungrammatical outputs. Finally, the **Noisy+Joint** model was able to outperform each of the separate models in all testing situations, demonstrating that both models are able to provide complementary information when combined together.

Table 5: SST results (WER) across various model types. Italics indicate a significant difference from **Baseline**.

	Diet		CSJ	
Task	Manual	ASR	Manual	ASR
Pre-SST	18.62%	36.10%	27.70%	36.49%
Baseline	6.51%	21.84%	15.06%	25.70%
Joint	*4.59%*	22.61%	14.56%	25.08%
Moses	*5.45%*	*20.97%*	14.73%	25.62%
Noisy	*5.32%*	*20.99%*	14.64%	25.03%
Noisy LL	*5.13%*	*20.97%*	14.49%	24.65%
Noisy+Joint	***4.05%***	***20.04%***	***13.55%***	***23.39%***

The **Moses** decoder uses log-linear weight tuning, lexical-probability-based phrase smoothing, and a number of other techniques to achieve high accuracies on traditional bilingual MT tasks. However, unlike the methods presented in this paper, **Moses** does not directly condition the phrase TM probabilities on the identity of the surrounding phrases. In the evaluation, **Moses** achieved a higher accuracy than **Baseline** in all situations, but only achieved accuracy approximately equal to, or lower than **Noisy**. This can be explained by the fact that the majority of the sophisticated techniques used by **Moses** are aimed at achieving proper reordering of words between languages, and are less useful in a monotonic translation task such as SST. In addition, it should be noted that due to the monotonicity constraint, processing with the proposed system is significantly faster than Moses, with the most accurate monotonic SMT system processing text approximately 12.5 times faster (611 words/sec. and 49 words/sec. respectively).

7.4. Effect of TM Data Size

As collecting parallel data to train the TM is more difficult than gathering non-parallel data to train the LM, it is desirable to have a method that works even with a small amount of parallel data. The following experiments assess the amount of parallel data required to effectively train each model by varying the size of the data used in the TM training over the Diet corpus. For all the cases, the size of the data used in the training of the LM remains the same. The results are displayed in Figure 4.

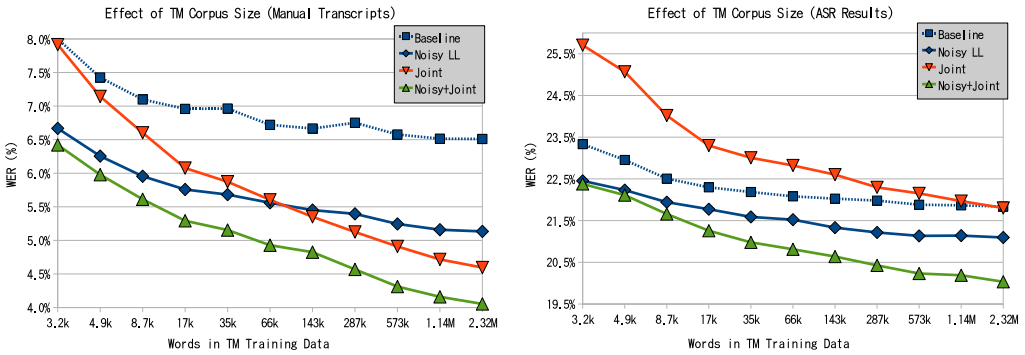

Figure 4: The effect of TM corpus data size on SST accuracy on the Diet corpus.

It can be seen that models using the noisy-channel approach are more robust in the face of small amounts of TM training data. This is a natural result as they are able to use the large amounts of LM training data to ensure that the output is somewhat natural. However, **Joint** is able to make better use of large amounts of TM data. In the manual transcript experiments of Figure 4, **Joint** surpasses **Noisy** at large data sizes. **Noisy+Joint** demonstrates both robustness to small data sizes and effective use of large amounts of data. As a result, the **Noisy+Joint** model outperforms the other models at all data sizes.

In addition, it should be noted that performance of **Joint** and **Noisy+Joint** has not saturated even when using the full 2.32M-word TM training data. This indicates that increasing the data size would further improve the accuracy of the system. This can be easily achieved in the case of ASR results, where the only resources needed are clean transcripts and speech data.

Finally, for most models there is a rapid increase in accuracy up until 17k words, after which accuracy increases at a slower pace. This may be because simple context-independent corrections such as fillers are learned in the first 17k words of TM training data, while the corrections that follow are more difficult context-dependent transformations.

7.5. Correction Success by Transformation Type

We also tabulated the success of each system by transformation type, with Figure 5 showing results for faithful transcript input, and Figure 6 showing results for ASR input. The graphs show instance-based correction recall, the number of correctly transformed instances divided by the total number of

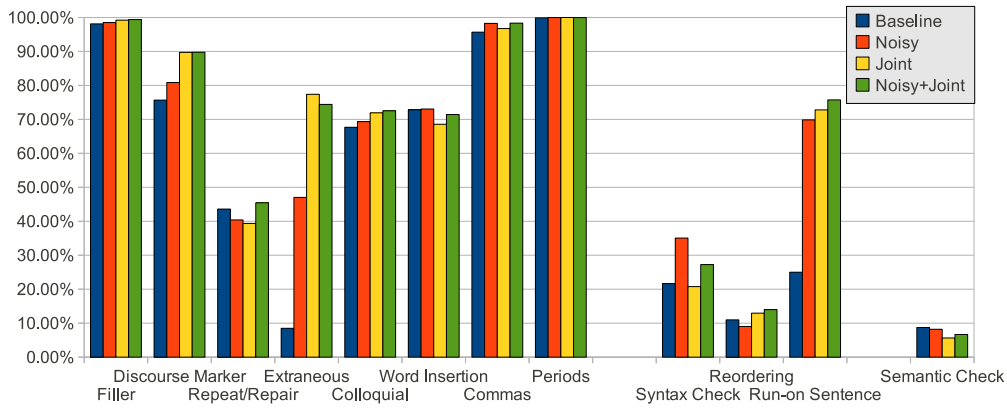

Figure 5: The number of transformations correctly performed by type on faithful transcripts (recall).

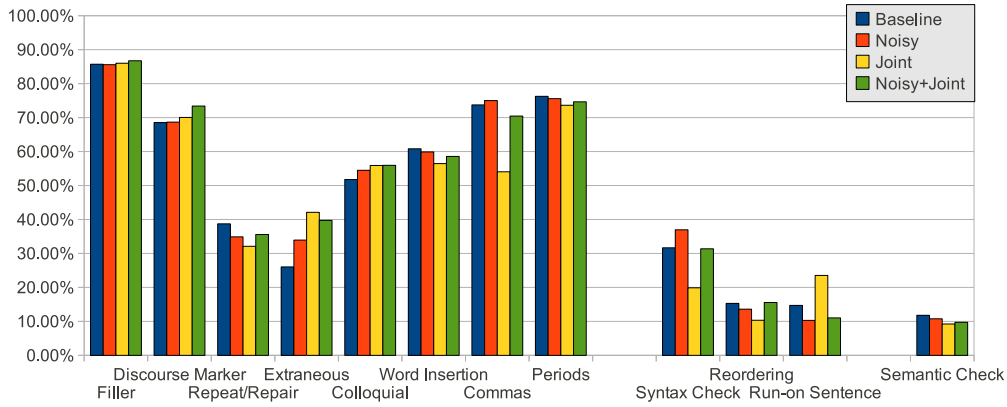

Figure 6: The number of transformations correctly performed by type on ASR resuls (recall).

instances requiring transformation[8]. It should be noted that in the faithful transcript results, the punctuation annotated in the input follows the gold-standard, and thus punctuation insertion and run-on sentence detection is largely trivial.

As an overall trend, it can be seen that the proposed context-sensitive features are generally helpful, with **Noisy+Joint** meeting or exceeding the baseline in most categories. Also, it can be seen that the **Noisy** and **Joint** both have strong and weak points. **Noisy** tends to outperform **Joint** on word, comma, and period insertion. This is because insertions tend to be more context dependent than deletions and thus rely on large amounts of LM training data to resolve sparsity issues. On the other hand, **Joint** is much stronger at deletion of multi-word phrases such as discourse markers and multi-word phrases, a result of the cleaner model that is able to more directly model deleted multi-word phrases. **Noisy+Joint** allows for combination of these two sources of information, and achieves accuracy similar to the better of the two models in most categories.

When comparing results using fathful and ASR-produced transcripts, it can be seen that recall drops across the board when using ASR, as would be expected. Ignoring the drops in punctuation insertion and sentence boundary accuracy, which are a by-product of the switch from gold-standard punctuation annotation to pause-based information, it can be seen that most simple editing categories see a drop between 10-20%, which can be expected given the WER of 17.10%. The one exception to this is extraneous expressions, which are often not included in the ASR language model, and thus tend to be mis-recognized more often than other words.

It can be seen that the system is largely successful in correcting all of its major targets. For simple edits other than repeats and repairs, it was able to achieve a recall between 71-99% across all categories on clean transcripts, and between 56-87% on ASR results.

7.6. Effect of Additional Features

Experiments were also conducted to assess the effect of adding the features mentioned in Section 6, the results of which can be found in Figure 7. It can be seen that adding additional features is useful when using the **Baseline**

[8]As our system does not explicitly determine transformation types when making corrections, it is difficult to accurately determine type-by-type precision. Increases in overall precision are roughly reflected in the WER results reported in the previous section.

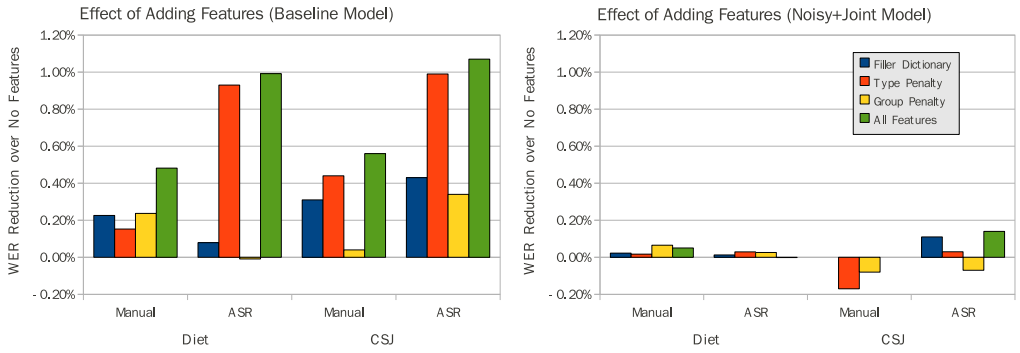

Figure 7: The increase in accuracy provided by each feature over the model with no additional features.

TM, resulting in an over 1% increase in accuracy on ASR results. However, when the best **Noisy+Joint** TM is used, the additional features results in very little change [9].

From this result, it may be concluded that the lexical features introduced in this paper are redundant with the more advanced translation model. However, features representing prosody [52] might provide information orthogonal to that provided by the proposed method, and further increase accuracy.

8. Conclusion

This paper has described a novel framework for creating clean transcripts from faithful transcripts or ASR results. We proposed an SMT-based model that combines joint probability, context-sensitive conditional probability, and language models in a log-linear framework. This was implemented using WFSTs, which allow for integration with existing ASR systems, and simple addition of task-specific features. Experiments on both faithful transcripts and ASR results showed that the proposed method was effective in not only deletions of redundant words and insertion of punctuation, but also insertion of dropped words and correction of colloquial expressions, which were not handled by previous models.

[9]The actual learned feature weights for **Noisy+Joint** were $\lambda_{lm} = .153$, $\lambda_{tm} = .143$, $\lambda_{sm} = .139$, $\lambda_j = .330$, $\lambda_i = -.020$, $\lambda_s = -.007$, $\lambda_d = -.012$, $\lambda_g = .051$, and $\lambda_f = -.140$. It can be seen that the majority of weight is put on the LM, TM, SM, joint probability, and filler deletion bonus.

Future research directions include the incorporation of richer information such as prosody in the transformation process, which has the potential to further improve the handling of disfluent phenomena. In addition, we also plan to examine the use of part of speech information to improve robustness to sparsity, or structural and semantic information to improve robustness to repairs. Finally, we also plan on examining the tight coupling of the proposed system with a WFST-based speech recognition decoder.

References

[1] D. Jones, F. Wolf, E. Gibson, E. Williams, E. Fedorenko, D. Reynolds, M. Zissman, Measuring the Readability of Automatic Speech-to-Text Transcripts, in: Proceedings of the 8th European Conference on Speech Communication and Technology (EuroSpeech), 1585–1588, 2003.

[2] E. Shriberg, Preliminaries to a Theory of Speech Disfluencies, Ph.D. thesis, University of California at Berkeley, 1994.

[3] S. Rao, I. Lane, T. Schultz, Improving Spoken Language Translation by Automatic Disfluency Removal: Evidence from Conversational Speech Transcripts, in: Machine Translation Summit XI, 177–180, 2007.

[4] B. Favre, R. Grishman, D. Hillard, H. Ji, D. Hakkani-Tür, M. Ostendorf, Punctuating speech for information extraction, in: Proceedings of the International Conference on Acoustics, Speech, and Signal Processing (ICASSP), 2008.

[5] E. Fitzgerald, Reconstructing Spontaneous Speech, Ph.D. thesis, Johns Hopkins University, 2009.

[6] D. Chiang, M. Diab, N. Habash, O. Rambow, S. Shareef, Parsing Arabic Dialects, in: Proceedings of the 11th European Chapter of the Association for Computational Linguistics, 2006.

[7] Y. Akita, M. Mimura, T. Kawahara, Automatic Transcription System for Meetings of the Japanese National Congress, in: Proceedings of the 10th Annual Conference of the International Speech Communication Association (InterSpeech), 84–87, 2009.

[8] J. Kolář, Design, creation, and analysis of Czech corpora for structural metadata extraction from speech, Language Resources and Evaluation (2010) 1–24.

[9] P. M. Clancy, Spoken and Written Language: Exploring Orality and Literacy, chap. Written and Spoken Style in Japanese Narratives, ABLEX, 55–76, 1982.

[10] Y. Liu, E. Shriberg, A. Stolcke, B. Peskin, J. Ang, D. Hillard, M. Ostendorf, M. Tomalin, P. Woodland, M. Harper, Structural Metadata Research in the EARS Program, in: Proceedings of the International Conference on Acoustics, Speech, and Signal Processing (ICASSP), 957–960, 2005.

[11] J. Bear, J. Dowding, E. Shriberg, Integrating multiple knowledge sources for detection and correction of repairs in human-computer dialog, in: Proceedings of the 30th Annual Meeting of the Association for Computational Linguistics, 56–63, 1992.

[12] C. Nakatani, J. Hirschberg, A speech-first model for repair detection and correction, in: Proceedings of the 31st Annual Meeting of the Association for Computational Linguistics, 46–53, 1993.

[13] E. Charniak, M. Johnson, Edit detection and parsing for transcribed speech, in: Proceedings of the Second Meeting of the North American Chapter of the Association for Computational Linguistics, 9, 2001.

[14] J. Kim, S. Schwarm, M. Ostendorf, Detecting structural metadata with decision trees and transformation-based learning, in: Proceedings of the Human Language Technology Conference / North American Chapter of the Association for Computational Linguistics Meeting (HLT/NAACL), 137–144, 2004.

[15] M. Snover, B. Dorr, R. Schwartz, A lexically-driven algorithm for disfluency detection, in: Proceedings of the Human Language Technology Conference / North American Chapter of the Association for Computational Linguistics Meeting (HLT/NAACL), 157–160, 2004.

[16] M. Honal, T. Schultz, Correction of disfluencies in spontaneous speech using a noisy-channel approach, in: Proceedings of the 8th European

Conference on Speech Communication and Technology (EuroSpeech), 2781–2784, 2003.

[17] S. Maskey, B. Zhou, Y. Gao, A phrase-level machine translation approach for disfluency detection using weighted finite state transducers, in: Proceedings of the 9th International Conference on Spoken Language Processing (InterSpeech 2006 - ICSLP), 749–752, 2006.

[18] M. Johnson, E. Charniak, M. Lease, An improved model for recognizing disfluencies in conversational speech, in: Proceedings of the Rich Transcription Workshop, 2004.

[19] Y. Liu, E. Shriberg, A. Stolcke, D. Hillard, M. Ostendorf, M. Harper, Enriching Speech Recognition with Automatic Detection of Sentence Boundaries and Disfluencies, IEEE Transactions on Audio, Speech, and Language Processing 14 (5) (2006) 1526–1540.

[20] J.-F. Yeh, C.-H. Wu, Edit Disfluency Detection and Correction Using a Cleanup Language Model and an Alignment Model, IEEE Transactions on Audio, Speech, and Language Processing 14 (5) (2006) 1574–1583.

[21] Y. Gotoh, S. Renals, Sentence boundary detection in broadcast speech transcripts, in: ASR2000-Automatic Speech Recognition: Challenges for the new Millennium ISCA Tutorial and Research Workshop (ITRW), 2000.

[22] H. Christensen, Y. Gotoh, S. Renals, Punctuation annotation using statistical prosody models, in: ISCA Tutorial and Research Workshop (ITRW) on Prosody in Speech Recognition and Understanding, 2001.

[23] J. Kim, P. Woodland, The use of prosody in a combined system for punctuation generation and speech recognition, in: Proceedings of the 7th European Conference on Speech Communication and Technology (EuroSpeech), 2001.

[24] J. Huang, G. Zweig, Maximum entropy model for punctuation annotation from speech, in: Proceedings of the 7th International Conference on Speech and Language Processing (ICSLP), 2002.

[25] B. Roark, Y. Liu, M. Harper, R. Stewart, M. Lease, M. Snover, I. Shafran, B. Dorr, J. Hale, A. Krasnyanskaya, et al., Reranking for

sentence boundary detection in conversational speech, in: Proceedings of the International Conference on Acoustics, Speech, and Signal Processing (ICASSP), 57–60, 2006.

[26] K. Shitaoka, K. Uchimoto, T. Kawahara, H. Isahara, Dependency structure analysis and sentence boundary detection in spontaneous Japanese, in: Proceedings of the 20th International Conference on Computational Linguistics, 1107, 2004.

[27] M. Paulik, S. Rao, I. Lane, S. Vogel, T. Schultz, Sentence segmentation and punctuation recovery for spoken language translation, in: Proceedings of the International Conference on Acoustics, Speech, and Signal Processing (ICASSP), 2008.

[28] A. Gravano, M. Jansche, M. Bacchiani, Restoring punctuation and capitalization in transcribed speech, in: Proceedings of the International Conference on Acoustics, Speech, and Signal Processing (ICASSP), 4741–4744, 2009.

[29] J. Lee, S. Seneff, Automatic Grammar Correction for Second-Language Learners, in: Proceedings of the 9th International Conference on Spoken Language Processing (InterSpeech 2006 - ICSLP), 1978–1981, 2006.

[30] H. Bakr, K. Shaalan, I. Ziedan, A Hybrid Approach for Converting Written Egyptian Colloquial Dialect into Diacritized Arabic, in: The 6th International Conference on Informatics and Systems (INFOS 2008), 2008.

[31] G. Al-Gaphari, M. Al-Yadoumi, A Method to Convert Sana'ani Accent to Modern Standard Arabic, International Journal of Information, Science, and Technology 8 (1) (2010) 39–49.

[32] T. Hori, D. Willett, Y. Minami, Paraphrasing spontaneous speech using weighted finite-state transducers, in: ISCA & IEEE Workshop on Spontaneous Speech Processing and Recognition, 2003.

[33] M. Shugrina, Formatting Time-Aligned ASR Transcripts for Readability, in: Proceedings of the Human Language Technology: The 11th Annual Conference of the North American Chapter of the Association for Computational Linguistics, 2010.

[34] K. Shitaoka, H. Nanjo, T. Kawahara, Automatic Transformation of Lecture Transcription into Document Style using Statistical Framework, in: Proceedings of the 8th International Conference on Spoken Language Processing (InterSpeech 2004 - ICSLP), 2169–2172, 2004.

[35] P. F. Brown, V. J. Pietra, S. A. D. Pietra, R. L. Mercer, The Mathematics of Statistical Machine Translation: Parameter Estimation, Computational Linguistics 19 (1993) 263–311.

[36] F. J. Och, H. Ney, The Alignment Template Approach to Statistical Machine Translation, vol. 30, MIT Press, ISSN 0891-2017, 417–449, 2004.

[37] S. F. Chen, J. Goodman, An Empirical Study of Smoothing Techniques for Language Modeling, in: Proceedings of the 34th Annual Meeting of the Association for Computational Linguistics, 310–318, 1996.

[38] R. Kneser, H. Ney, Improved backing-off for M-gram language modeling, Proceedings of the International Conference on Acoustics, Speech, and Signal Processing (ICASSP) 1 (1995) 181–184.

[39] M. Mohri, Finite-state transducers in language and speech processing, Computational Linguistics 23 (2) (1997) 269–311, ISSN 0891-2017.

[40] F. Casacuberta, E. Vidal, Machine Translation with Inferred Stochastic Finite-State Transducers, Computational Linguistics 30 (2) (2004) 205–225, ISSN 0891-2017.

[41] F. Casacuberta, M. Federico, H. Ney, E. Vidal, Recent efforts in spoken language translation, IEEE Signal Processing Magazine 25 (3) (2008) 80–88.

[42] B. Zhou, S. Chen, Y. Gao, Constrained phrase-based translation using weighted finite-state transducers, in: Proceedings of the International Conference on Acoustics, Speech, and Signal Processing (ICASSP), 2005.

[43] K. Knight, Decoding complexity in word-replacement translation models, Computational Linguistics 25 (4) (1999) 607–615.

[44] F. J. Och, H. Ney, Discriminative training and maximum entropy models for statistical machine translation, in: Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics, 295–302, 2002.

[45] F. J. Och, Minimum Error Rate Training in Statistical Machine Translation, in: Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics, 160–167, 2003.

[46] K. Maekawa, Corpus of Spontaneous Japanese: Its design and evaluation, in: Proceedings of the ISCA/IEEE Workshop on Spontaneous Speech, 2003.

[47] M. Honal, T. Schultz, Automatic Disfluency Removal on Recognized Spontaneous Speech-Rapid Adaptation to Speaker-Dependent Disfluencies, in: Proceedings of the International Conference on Acoustics, Speech, and Signal Processing (ICASSP), vol. 2005, Citeseer, 969–972, 2005.

[48] A. Lee, T. Kawahara, Recent Development of Open-Source Speech Recognition Engine Julius, in: Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA ASC), 2009.

[49] Y. Akita, T. Kawahara, Statistical transformation of language and pronunciation models for spontaneous speech recognition, IEEE Transactions on Audio, Speech, and Language Processing 18 (6) (2010) 1539–1549.

[50] C. Allauzen, M. Riley, J. Schalkwyk, W. Skut, M. Mohri, OpenFst: a general and efficient weighted finite-state transducer library, in: Proceedings of the CIAA '07, 11–23, 2007.

[51] P. Koehn, et al., Moses: Open Source Toolkit for Statistical Machine Translation, in: Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics, 2007.

[52] E. Shriberg, R. Bates, A. Stolcke, A prosody-only decision-tree model for disfluency detection, in: Proceedings of the 5th European Conference on Speech Communication and Technology (EuroSpeech), 1997.

[53] C. Allauzen, M. Mohri, B. Roark, Generalized algorithms for constructing statistical language models, in: Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics, 40–47, 2003.

Appendix A. WFST Implementation of the Context Sensitive TM

In order to implement a context-sensitive translation model in the WFST framework, it is necessary to create a WFST to represent the TM probability. The naive representation would be to create a single edge from every node γ_{k-n+1}^{k-1} to every node γ_{k-1}^{k} weighted with the probability $P_{TM}(\tilde{v}_k|\gamma_{k-n+1}^{k-1}, \tilde{w}_k)$. However, for a vocabulary of size M and n-gram length n, this creates a WFST with M^n edges, which requires excessive amounts of memory for large M or n. This problem can be ameliorated through a representation similar to that of the LM in Section 4.2, which only requires that specifically observed histories be explicitly expressed in the WFST.

In order to create this WFST representation, we first note that all n-gram TM probabilities $P_{TM}(\tilde{v}_k|\gamma_{k-n+1}^{k-1}, \tilde{w}_k)$ for which $P_{ML}(\gamma_k|\gamma_{k-n+1}^{k-1}) = 0$ are a constant multiple of the $(n-1)$-gram TM probability $P_{TM}(\tilde{v}_k|\gamma_{k-n+2}^{k-1}, \tilde{w}_k)$. We show this by first noting that the language model probabilities that we use in construction of the context-sensitive translation model probabilities take the standard format of maximum likelihood probability P_{ML} and a backoff probability P_{BO}.

$$P_{LM}(w_i|w_{i-n+1}^{i-1}) = \begin{cases} \phi(w_{i-n+1}^{i})P_{ML}(w_i|w_{i-n+1}^{i-1}) & \text{if } P_{ML}(w_i|w_{i-n+1}^{i-1}) > 0, \\ P_{BO}(w_{i-n+1}^{i-1})P_{LM}(w_i|w_{i-n+2}^{i-1}) & \text{otherwise.} \end{cases} \tag{A.1}$$

Next, we substitute in the LM probability for unobserved instances into Equation (7) and perform some simple algebra:

$$\begin{aligned} P_{TM}(\tilde{v}_k|\gamma_{k-n+1}^{k-1}, \tilde{w}_k) &= \frac{P_{TM}(\gamma_k|\gamma_{k-n+1}^{k-1})}{P_{TM}(\tilde{w}_k|\gamma_{k-n+1}^{k-1})} \\ &= \frac{P_{TM}(\gamma_k|\gamma_{k-n+2}^{k-1})P_{BO}(\gamma_{k-n+1}^{k-1})}{P_{TM}(\tilde{w}_k|\gamma_{k-n+1}^{k-1})} \\ &= P_{TM}(\tilde{v}_k|\gamma_{k-n+2}^{k-1}, \tilde{w}_k)\frac{P_{TM}(\tilde{w}_k|\gamma_{k-n+2}^{k-1})P_{BO}(\gamma_{k-n+1}^{k-1})}{P_{TM}(\tilde{w}_k|\gamma_{k-n+1}^{k-1})} \\ &= P_{TM}(\tilde{v}_k|\gamma_{k-n+2}^{k-1}, \tilde{w}_k) * \text{adj}(\gamma_{k-n+1}^{k-1}, \tilde{w}_k) \end{aligned} \tag{A.2}$$

where

$$\text{adj}(\gamma_{k-n+1}^{k-1}, \tilde{w}_k) \equiv \frac{P_{TM}(\tilde{w}_k|\gamma_{k-n+2}^{k-1})P_{BO}(\gamma_{k-n+1}^{k-1})}{P_{TM}(\tilde{w}_k|\gamma_{k-n+1}^{k-1})}. \tag{A.3}$$

Because of this, $\text{adj}(\gamma_{k-n+1}^{k-1}, \tilde{w}_k)$ can be used similarly to the backoff probability in the WFST implementation of a LM in Figure 1.

In addition, it should be noted that in the case where

$$\forall_{\gamma \in \{\gamma:\psi_w(\gamma)=\tilde{w}_k\}} P_{ML}(\psi_v(\gamma)|\gamma_{k-n+1}^{k-1}, \psi_w(\gamma)) = 0$$

it can be shown that

$$\text{adj}(\gamma_{k-n+1}^{k-1}, \tilde{w}_k) = 1.$$

This can be done by substituting the sum from Equation (9) into Equation (A.3), and further substituting in the LM probability for unobserved events from Equation (A.1)

$$\begin{aligned}
\text{adj}(\gamma_{k-n+1}^{k-1}, \tilde{w}_k) &= \frac{\sum_{\gamma \in \{\gamma:\psi_w(\gamma)=\tilde{w}_k\}} P_t(\gamma|\gamma_{k-n+2}^{k-1})P_{BO}(\gamma_{k-n+1}^{k-1})}{\sum_{\gamma \in \{\gamma:\psi_w(\gamma)=\tilde{w}_k\}} P_t(\gamma|\gamma_{k-n+1}^{k-1})} \\
&= \frac{\sum_{\gamma \in \{\gamma:\psi_w(\gamma)=\tilde{w}_k\}} P_t(\gamma|\gamma_{k-n+2}^{k-1})P_{BO}(\gamma_{k-n+1}^{k-1})}{\sum_{\gamma \in \{\gamma:\psi_w(\gamma)=\tilde{w}_k\}} P_t(\gamma|\gamma_{k-n+2}^{k-1})P_{BO}(\gamma_{k-n+1}^{k-1})} \\
&= 1.
\end{aligned} \tag{A.4}$$

Thus, while a WFST edge must be created to express all $\text{adj}(\gamma_{k-n+1}^{k-1}, \tilde{w}_k)$ where there is at least one observed instance of $\gamma_{k-n+1}^{k-1}, \gamma$ where $\psi_w(\gamma) = \tilde{w}_k$, all other $\text{adj}(\cdot)$ can be expressed with a single edge with a weight equal to 1.

Appendix A.1. ϕ Transitions

However, these separate backoff weights for each $\tilde{w}_k$ cannot be expressed with the ϵ transitions that are traditionally used to express non-determinism in WFSTs. This can be best illustrated by an example. Suppose there are four γ in our vocabulary, $\Gamma = \{c = \langle \tilde{v}_c, \tilde{w}_a \rangle, d = \langle \tilde{v}_d, \tilde{w}_b \rangle, e = \langle \tilde{v}_e, \tilde{w}_b \rangle, f = \langle \tilde{v}_f, \tilde{w}_b \rangle\}$, and that only d has been observed after c

$$P_{ML}(\gamma_k = d|\gamma_{k-1} = c) = 1$$

By simply converting the LM transducer in figure 1, this can be expressed as a transducer in the form of Figure A.8-a. However, this transducer is not guaranteed to assign the correct probability to an input/output pair.

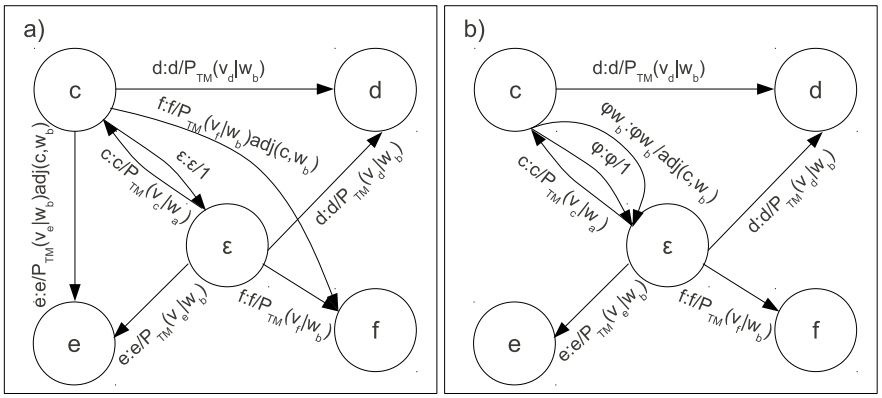

Figure A.8: The context-sensitive TM using ϵ transitions and hierarchical ϕ transitions

This is because, as shown in Equation (A.4), the adj$(\cdot)$ for non-observed sequences will be 1, which is generally larger than adj$(\cdot)$ for the observed sequences. As the Viterbi search employed by the decoder will prefer larger probabilities, it will follow the ϵ transition, even for γ for which $\gamma_{k-n+1}^{k-1}, \tilde{\gamma}$ where $\psi_w(\tilde{\gamma}) = \psi_w(\gamma)$ has been observed, resulting in an incorrectly high probability.

As a solution to this problem, ϕ transitions (failure transitions) can be substituted for ϵ transitions [53]. While ϵ transitions may be followed under any circumstances, ϕ transitions can only be followed if *there is no other matching edge* outgoing from the node. By expanding all edges for which $\text{adj}(\gamma_{k-n+1}^{k-1}, \tilde{w}_k) \neq 1$ and ensuring that the backoff transition will only be followed when no other edge is matched, it is possible to ensure that $\text{adj}(\gamma_{k-n+1}^{k-1}, \tilde{w}_k) = 1$ will only be assigned to the appropriate sequences.

Appendix A.2. Hierarchical ϕ Transitions

Even when using ϕ transitions, the WFST is still unnecessarily large. This is because all edges for which $\text{adj}(\gamma_{k-n+1}^{k-1}, \tilde{w}_k) \neq 1$ must be expanded, even though all γ for which $\psi_w(\gamma) = \tilde{w}_k$ share a single backoff weight (e.g. the edges from c to e and c to f in Figure A.8). In SST this is a particular problem, as there are often hundreds or thousands of separate words that may be deleted, and all share the single backoff weight $\text{adj}(\gamma_{k-n+1}^{k-1}, \tilde{w}_k = \epsilon)$.

In order to resolve this problem, we first notice that the rule for which

Table A.6: Size of Context-Sensitive TM for Different Construction Approaches

Type	1-gram	2-gram	3-gram	4-gram
ϵ Transitions (Full Expansion)	53.0k	1.28G	17.4G	66.1G
ϕ Transitions	53.0k	18.7M	145M	436M
Hierarchical ϕ Transitions	53.0k	382k	1.61M	4.04M

path to follow according to failure transitions is:

$$\text{follow arc labeled } \begin{cases} \gamma & \text{if adj}(\gamma_{i-n+1}^{i-1}, \psi_w(\gamma)) \neq 1, \\ \phi & \text{otherwise.} \end{cases} \tag{A.5}$$

Instead we are interested in a logic where a single symbol can be used to represent a backoff for all symbols for which the adj$(\gamma_{i-n+1}^{i-1}, \psi_w(\gamma))$ is equal

$$\text{follow arc labeled } \begin{cases} \gamma & \text{if } P_{ML}(\psi_v(\gamma)|\gamma_{i-n+1}^{i-1}, \psi_w(\gamma)) > 0, \\ \phi_{\psi_w(\gamma)} & \text{else if adj}(\gamma_{i-n+1}^{i-1}, \psi_w(\gamma)) \neq 1, \\ \phi & \text{otherwise.} \end{cases} \tag{A.6}$$

Using this logic, we define hierarchical ϕ transitions, where every symbol has a parent symbol. All γ symbols have a parent of $\phi_{\psi_w(\gamma)}$ and all ϕ_x symbols have ϕ as their parent. When searching for a transition for a particular symbol in the WFST, first the symbol itself is searched, and if it is not found, its parents are recursively searched until a match is made.

An example of hierarchical ϕ transitions can be found in Figure A.8-b. Compared with Figure A.8-a, the edges between c and e or f have been replaced by a hierarchical ϕ transition between the c and ϵ nodes.

Table A.6 shows the actual reduction in the number of arcs necessary to implement the n-gram TMs using the Diet corpus described in section 7.1. It can be seen that the hierarchical ϕ transitions allow for an approximately 100-fold decrease in WFST size over regular ϕ transitions.