Learning Dependency Translation Models as Collections of Finite-State Head Transducers

Hiyan Alshawi[*]
Shannon Laboratory, AT&T Labs

Srinivas Bangalore[†]
Shannon Laboratory, AT&T Labs

Shona Douglas[‡]
Shannon Laboratory, AT&T Labs

The paper defines weighted head transducers, finite-state machines that perform middle-out string transduction. These transducers are strictly more expressive than the special case of standard left-to-right finite-state transducers. Dependency transduction models are then defined as collections of weighted head transducers that are applied hierarchically. A dynamic programming search algorithm is described for finding the optimal transduction of an input string with respect to a dependency transduction model. A method for automatically training a dependency transduction model from a set of input-output example strings is presented. The method first searches for hierarchical alignments of the training examples guided by correlation statistics, and then constructs the transitions of head transducers that are consistent with these alignments. Experimental results are given for applying the training method to translation from English to Spanish and Japanese.

1. Introduction

We will define a dependency transduction model in terms of a collection of weighted head transducers. Each head transducer is a finite-state machine that differs from "standard" finite-state transducers in that, instead of consuming the input string left to right, it consumes it "middle out" from a symbol in the string. Similarly, the output of a head transducer is built up middle out at positions relative to a symbol in the output string. The resulting finite-state machines are more expressive than standard left-to-right transducers. In particular, they allow long-distance movement with fewer states than a traditional finite-state transducer, a useful property for the translation task to which we apply them in this paper. (In fact, finite-state head transducers are capable of unbounded movement with a finite number of states.) In Section 2, we introduce head transducers and explain how input-output positions on state transitions result in middle-out transduction.

When applied to the problem of translation, the head transducers forming the dependency transduction model operate on input and output strings that are sequences of dependents of corresponding headwords in the source and target languages. The dependency transduction model produces synchronized dependency trees in which each local tree is produced by a head transducer. In other words, the dependency

[*] 180 Park Avenue, Florham Park, NJ 07932
[†] 180 Park Avenue, Florham Park, NJ 07932
[‡] 180 Park Avenue, Florham Park, NJ 07932

model applies the head transducers recursively, imposing a recursive decomposition of the source and target strings. A dynamic programming search algorithm finds optimal (lowest total weight) derivations of target strings from input strings or word lattices produced by a speech recognizer. Section 3 defines dependency transduction models and describes the search algorithm.

We construct the dependency transduction models for translation automatically from a set of unannotated examples, each example comprising a source string and a corresponding target string. The recursive decomposition of the training examples results from an algorithm for computing hierarchical alignments of the examples, described in Section 4.2. This alignment algorithm uses dynamic programming search guided by source-target word correlation statistics as described in Section 4.1.

Having constructed a hierarchical alignment for the training examples, a set of head transducer transitions are constructed from each example as described in Section 4.3. Finally, the dependency transduction model is constructed by aggregating the resulting head transducers and assigning transition weights, which are log probabilities computed from the training counts by simple maximum likelihood estimation.

We have applied this method of training statistical dependency transduction models in experiments on English-to-Spanish and English-to-Japanese translations of transcribed spoken utterances. The results of these experiments are described in Section 5; our concluding remarks are in Section 6.

2. Head Transducers

2.1 Weighted Finite-State Head Transducers

In this section we describe the basic structure and operation of a weighted head transducer. In some respects, this description is simpler than earlier presentations (e.g., Alshawi 1996); for example, here final states are simply a subset of the transducer states whereas in other work we have described the more general case in which final states are specified by a probability distribution. The simplified description is adequate for the purposes of this paper.

Formally, a weighted head transducer is a 5-tuple: an alphabet W of input symbols; an alphabet V of output symbols; a finite set Q of states $q_0, \ldots, q_s$; a set of final states $F \subseteq Q$; and a finite set T of state transitions. A transition from state q to state q' has the form

$$\langle q, q', w, v, \alpha, \beta, c \rangle$$

where w is a member of W or is the empty string ϵ; v is a member of V or ϵ; the integer α is the **input position**; the integer β is the **output position**; and the real number c is the weight or **cost** of the transition. A transition in which $\alpha = 0$ and $\beta = 0$ is called a **head transition**.

The interpretation of q, q', w, and v in transitions is similar to left-to-right transducers, i.e., in transitioning from state q to state q', the transducer "reads" input symbol w and "writes" output symbol v, and as usual if w (or v) is ϵ then no read (respectively write) takes place for the transition. The difference lies in the interpretation of the read position α and the write position β. To interpret the transition positions as transducer actions, we consider notional input and output tapes divided into squares. On such a tape, one square is numbered 0, and the other squares are numbered $1, 2, \ldots$ rightwards from square 0, and $-1, -2, \ldots$ leftwards from square 0 (Figure 1).

A transition with input position α and output position β is interpreted as reading w from square α on the input tape and writing v to square β of the output tape; if square β is already occupied, then v is written to the next empty square to the left of

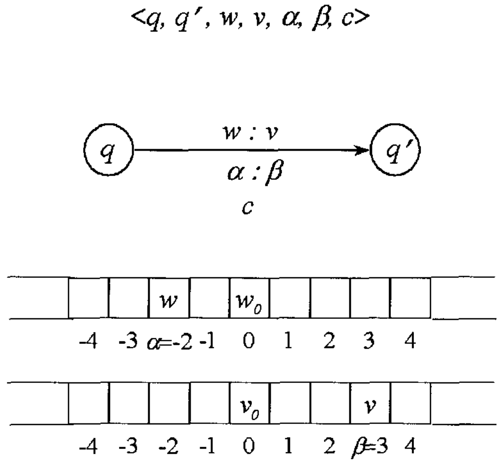

Figure 1
Transition symbols and positions.

β if $\beta < 0$, or to the right of β if $\beta \geq 0$, and similarly, if input was already read from position α, w is taken from the next unread square to the left of α if $\alpha < 0$ or to the right of α if $\alpha \geq 0$.

The operation of a head transducer is nondeterministic. It starts by taking a head transition

$$\langle q, q', w_0, v_0, 0, 0, c\rangle$$

where w_0 is one of the symbols (not necessarily the leftmost) in the input string. (The valid initial states are therefore implicitly defined as those with an outgoing head transition.) w_0 is considered to be at square 0 of the input tape and v_0 is output at square 0 of the output tape. Further state transitions may then be taken until a final state in F is reached. For a derivation to be valid, it must read each symbol in the input string exactly once. At the end of a derivation, the output string is formed by taking the sequence of symbols on the target tape, ignoring any empty squares on this tape.

The cost of a derivation of an input string to an output string by a weighted head transducer is the sum of the costs of transitions taken in the derivation. We can now define the string-to-string transduction function for a head transducer to be the function that maps an input string to the output string produced by the lowest-cost valid derivation taken over all initial states and initial symbols. (Formally, the function is partial in that it is not defined on an input when there are no derivations or when there are multiple outputs with the same minimal cost.)

In the transducers produced by the training method described in this paper, the source and target positions are in the set $\{-1, 0, 1\}$, though we have also used hand-coded transducers (Alshawi and Xia 1997) and automatically trained transducers (Alshawi and Douglas 2000) with a larger range of positions.

2.2 Relationship to Standard FSTs

The operation of a traditional left-to-right transducer can be simulated by a head transducer by starting at the leftmost input symbol and setting the positions of the first transition taken to $\alpha = 0$ and $\beta = 0$, and the positions for subsequent transitions to $\alpha = 1$ and $\beta = 1$. However, we can illustrate the fact that head transducers are more

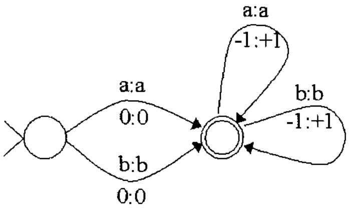

Figure 2
Head transducer to reverse an input string of arbitrary length in the alphabet $\{a, b\}$.

expressive than left-to-right transducers by the case of a finite-state head transducer that reverses a string of arbitrary length. (This cannot be performed by a traditional transducer with a finite number of states.)

For example, the head transducer described below (and shown in Figure 2) with input alphabet $\{a, b\}$ will reverse an input string of arbitrary length in that alphabet. The states of the example transducer are $Q = \{q_1, q_2\}$ and $F = \{q_2\}$, and it has the following transitions (costs are ignored here):

$$\langle q_1, q_2, a, a, 0, 0\rangle$$
$$\langle q_1, q_2, b, b, 0, 0\rangle$$
$$\langle q_2, q_2, a, a, -1, 1\rangle$$
$$\langle q_2, q_2, b, b, -1, 1\rangle$$

The only possible complete derivations of the transducer read the input string right to left, but write it left to right, thus reversing the string.

Another similar example is using a finite-state head transducer to convert a palindrome of arbitrary length into one of its component halves. This clearly requires the use of an empty string on some of the output transitions.

3. Dependency Transduction Models

3.1 Dependency Transduction using Head Transducers

In this section we describe dependency transduction models, which can be used for machine translation and other transduction tasks. These models consist of a collection of head transducers that are applied hierarchically. Applying the machines hierarchically means that a nonhead transition is interpreted not simply as reading an input-output pair (w, v), but instead as reading and writing a pair of strings headed by (w, v) according to the derivation of a subnetwork.

For example, the head transducer shown in Figure 3 can be applied recursively in order to convert an arithmetic expression from infix to prefix (Polish) notation (as noted by Lewis and Stearns [1968], this transduction cannot be performed by a pushdown transducer).

In the case of machine translation, the transducers derive pairs of dependency trees, a source language dependency tree and a target dependency tree. A dependency tree for a sentence, in the sense of dependency grammar (for example Hays [1964] and Hudson [1984]), is a tree in which the words of the sentence appear as nodes (we do not have terminal symbols of the kind used in phrase structure grammar). In such a tree, the parent of a node is its head and the child of a node is the node's dependent.

The source and target dependency trees derived by a dependency transduction model are ordered, i.e., there is an ordering on the nodes of each local tree. This

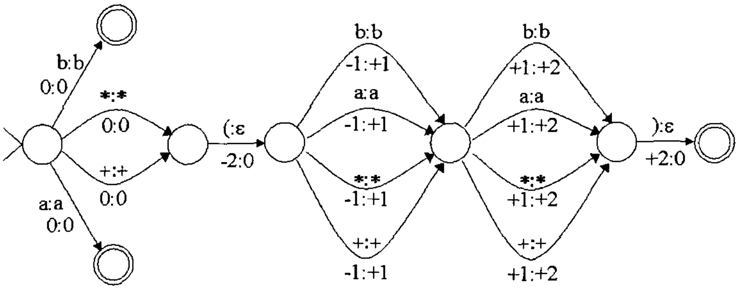

Figure 3
Dependency transduction network mapping bracketed arithmetic expressions from infix to prefix notation.

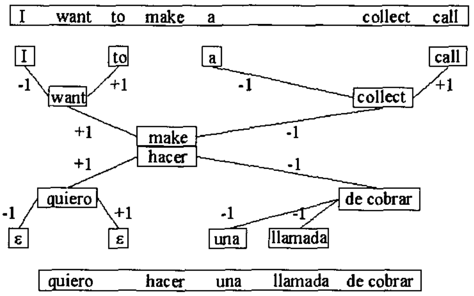

Figure 4
Synchronized dependency trees derived for transducing *I want to make a collect call* into *quiero hacer una llamada de cobrar.*

means, in particular, that the target sentence can be constructed directly by a simple recursive traversal of the target dependency tree. Each pair of source and target trees generated is synchronized in the sense to be formalized in Section 4.2. An example is given in Figure 4.

Head transducers and dependency transduction models are thus related as follows: Each pair of local trees produced by a dependency transduction derivation is the result of a head transducer derivation. Specifically, the input to such a head transducer is the string corresponding to the flattened local source dependency tree. Similarly, the output of the head transducer derivation is the string corresponding to the flattened local target dependency tree. In other words, the head transducer is used to convert a sequence consisting of a headword w and its left and right dependent words to a sequence consisting of a target word v and its left and right dependent words (Figure 5). Since the empty string may appear in a transition in place of a source or target symbol, the number of source and target dependents can be different.

The cost of a derivation produced by a dependency transduction model is the sum of all the weights of the head transducer derivations involved. When applying a dependency transduction model to language translation, we choose the target string obtained by flattening the target tree of the lowest-cost dependency derivation that also generates the source string.

We have not yet indicated what weights to use for head transducer transitions. The definition of head transducers as such does not constrain these. However, for a dependency transduction model to be a statistical model for generating pairs of strings, we assign transition weights that are derived from conditional probabilities. Several

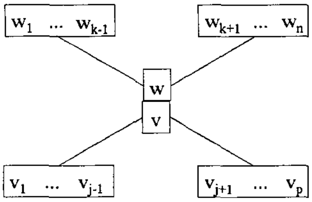

Figure 5
Head transducer converts the sequences of left and right dependents $\langle w_1 \ldots w_{k-1} \rangle$ and $\langle w_{k+1} \ldots w_n \rangle$ of w into left and right dependents $\langle v_1 \ldots v_{j-1} \rangle$ and $\langle v_{j+1} \ldots v_p \rangle$ of v.

probabilistic parameterizations can be used for this purpose including the following for a transition with headwords w and v and dependent words w' and v':

$$P(q', w', v', \alpha, \beta | w, v, q).$$

Here q and q' are the from-state and to-state for the transition and α and β are the source and target positions, as before. We also need parameters $P(q_0, q_1 | w, v)$ for the probability of choosing a head transition

$$\langle q_0, q_1, w, v, 0, 0 \rangle$$

given this pair of headwords. To start the derivation, we need parameters $P(roots(w_0, v_0))$ for the probability of choosing w_0,v_0 as the root nodes of the two trees.

These model parameters can be used to generate pairs of synchronized dependency trees starting with the topmost nodes of the two trees and proceeding recursively to the leaves. The probability of such a derivation can be expressed as:

$$P(roots(w_0, v_0))P(D_{w_0,v_0})$$

where $P(D_{w,v})$ is the probability of a subderivation headed by w and v, that is

$$P(D_{w,v}) = P(q_0, q_1 | w, v) \prod_{1 \leq i \leq n} P(q_{i+1}, w_i, v_i, \alpha_i, \beta_i | w, v, q_i) P(D_{w_i,v_i})$$

for a derivation in which the dependents of w and v are generated by n transitions.

3.2 Transduction Algorithm

To carry out translation with a dependency transduction model, we apply a dynamic programming search to find the optimal derivation. This algorithm can take as input either word strings, or word lattices produced by a speech recognizer. The algorithm is similar to those for context-free parsing such as chart parsing (Earley 1970) and the CKY algorithm (Younger 1967). Since word string input is a special case of word lattice input, we need only describe the case of lattices.

We now present a sketch of the transduction algorithm. The algorithm works bottom-up, maintaining a set of **configurations**. A configuration has the form

$$[n_1, n_2, w, v, q, c, t]$$

corresponding to a bottom-up partial derivation currently in state q covering an input sequence between nodes n_1 and n_2 of the input lattice. w and v are the topmost nodes in the source and target derivation trees. Only the target tree t is stored in the configuration.

The algorithm first initializes configurations for the input words, and then performs transitions and optimizations to develop the set of configurations bottom-up:

- Initialization: For each word edge between nodes n and n' in the lattice with source word w_0, an initial configuration is constructed for any head transition of the form
$$\langle q, q', w_0, v_0, 0, 0, c\rangle$$
Such an initial configuration has the form:
$$[n, n', w_0, v_0, q', c, v_0]$$

- Transition: We show the case of a transition in which a new configuration results from consuming a source dependent w_1 to the left of a headword w and adding the corresponding target dependent v_1 to the right of the target head v. Other cases are similar. The transition applied is:
$$\langle q, q', w_1, v_1, -1, 1, c'\rangle$$
It is applicable when there are the following head and dependent configurations:
$$[n_2, n_3, w, v, q, c, t]$$
$$[n_1, n_2, w_1, v_1, q_f, c_1, t_1]$$
where the dependent configuration is in a final state q_f. The result of applying the transition is to add the following to the set of configurations:
$$[n_1, n_3, w, v, q', c + c_1 + c', t']$$
where t' is the target dependency tree formed by adding t_1 as the rightmost dependent of t.

- Optimization: We also require a dynamic programming condition to remove suboptimal (sub)derivations. Whenever there are two configurations
$$[n, n', w, v, q, c_1, t_1]$$
$$[n, n', w, v, q, c_2, t_2]$$
and $c_2 > c_1$, the second configuration is removed from the set of configurations.

If, after all applicable transitions have been taken, there are configurations spanning the entire input lattice, then the one with the lowest cost is the optimal derivation. When there are no such configurations, we take a pragmatic approach in the translation application and simply concatenate the lowest costing of the minimal length sequences of partial derivations that span the entire lattice. A Viterbi-like search of the graph formed by configurations is used to find the optimal sequence of derivations. One of the advantages of middle-out transduction is that robustness is improved through such use of partial derivations when no complete derivations are available.

4. Training Method

Our training method for head transducer models only requires a set of training examples. Each example, or **bitext**, consists of a source language string paired with a target language string. In our experiments, the bitexts are transcriptions of spoken English utterances paired with their translations into Spanish or Japanese.

It is worth emphasizing that we do not necessarily expect the dependency representations produced by the training method to be traditional dependency structures for the two languages. Instead, the aim is to produce bilingual (i.e., synchronized, see below) dependency representations that are appropriate to performing the translation task for a specific language pair or specific bilingual corpus. For example, headwords in both languages are chosen to force a synchronized alignment (for better or worse) in order to simplify cases involving so-called head-switching. This contrasts with one of the traditional approaches (e.g., Dorr 1994; Watanabe 1995) to posing the translation problem, i.e., the approach in which translation problems are seen in terms of bridging the gap between the most natural monolingual representations underlying the sentences of each language.

The training method has four stages: (i) Compute co-occurrence statistics from the training data. (ii) Search for an optimal synchronized hierarchical alignment for each bitext. (iii) Construct a set of head transducers that can generate these alignments with transition weights derived from maximum likelihood estimation.

4.1 Computing Pairing Costs

For each source word w in the data set, assign a cost, the **translation pairing cost** $c(w, v)$ for all possible translations v into the target language. These translations of the source word may be zero, one, or several target language words (see Section 4.4 for discussion of the multiword case). The assignment of translation pairing costs (effectively a statistical bilingual dictionary) may be done using various statistical measures. For this purpose, a suitable statistical function needs to indicate the strength of co-occurrence correlation between source and target words, which we assume is indicative of carrying the same semantic content. Our preferred choice of statistical measure for assigning the costs is the ϕ correlation measure (Gale and Church 1991). We apply this statistic to co-occurrence of the source word with all its possible translations in the data set examples. We have found that, at least for our data, this measure leads to better performance than the use of the log probabilities of target words given source words (cf. Brown et al. 1993).

In addition to the correlation measure, the cost for a pairing includes a distance measure component that penalizes pairings proportionately to the difference between the (normalized) positions of the source and target words in their respective sentences.

4.2 Computing Hierarchical Alignments

As noted earlier, dependency transduction models are generative probabilistic models; each derivation generates a pair of dependency trees. Such a pair can be represented as a **synchronized hierarchical alignment** of two strings. A **hierarchical alignment** consists of four functions. The first two functions are an **alignment** mapping f from source words w to target words $f(w)$ (which may be the empty string ϵ), and an **inverse alignment** mapping from target words v to source words $f'(v)$. The inverse mapping is needed to handle mapping of target words to ϵ; it coincides with f for pairs without source ϵ. The other two functions are a **source head-map** g mapping source dependent words w to their heads $g(w)$ in the source string, and a **target head-map** h mapping target dependent words v to their headwords $h(v)$ in the target string. An

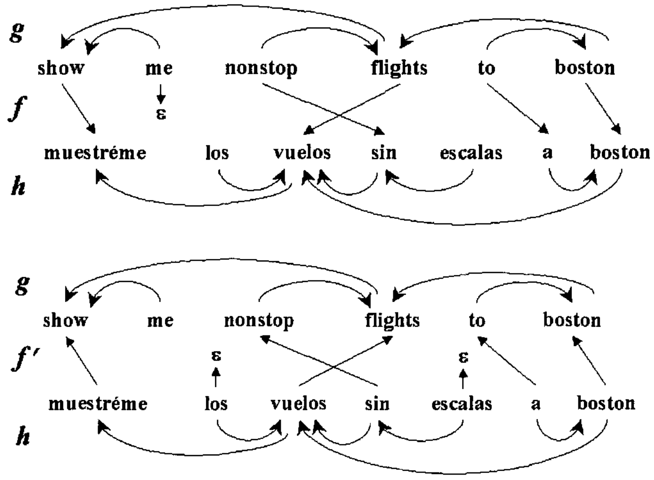

Figure 6
A hierarchical alignment: alignment mappings f and f', and head-maps g and h.

example hierarchical alignment is shown in Figure 6 (f and f' are shown separately for clarity).

A hierarchical alignment is synchronized (i.e., it corresponds to synchronized dependency trees) if these conditions hold:

Nonoverlap: If $w_1 \neq w_2$, then $f(w_1) \neq f(w_2)$, and similarly, if $v_1 \neq v_2$, then $f'(v_1) \neq f'(v_2)$.

Synchronization: if $f(w) = v$ and $v \neq \epsilon$, then $f(g(w)) = h(v)$, and $f'(v) = w$. Similarly, if $f'(v) = w$ and $w \neq \epsilon$, then $f'(h(v)) = g(w)$, and $f(w) = v$.

Phrase contiguity: The image under f of the maximal substring dominated by a headword w is a contiguous segment of the target string.

(Here w and v refer to word tokens not symbols (types). We hope that the context of discussion will make the type-token distinction clear in the rest of this article.) The hierarchical alignment in Figure 6 is synchronized.

Of course, translations of phrases are not always transparently related by a hierarchical alignment. In cases where the mapping between a source and target phrase is unclear (for example, one of the phrases might be an idiom), then the most reasonable choice of hierarchical alignment may be for f and f' to link the heads of the phrases only, all the other words being mapped to ϵ, with no constraints on the monolingual head mappings h and g. (This is the approach we take to compound lexical pairings, discussed in Section 4.4.)

In the hierarchical alignments produced by the training method described here, the source and target strings of a bitext are decomposed into three aligned regions, as shown in Figure 7: a head region consisting of headword w in the source and its corresponding target $f(w)$ in the target string, a left substring region consisting of the source substring to the left of w and its projection under f on the target string, and a right substring region consisting of the source substring to the right of w and its projection under f on the target string. The decomposition is recursive in that the left substring region is decomposed around a left headword w_l, and the right substring

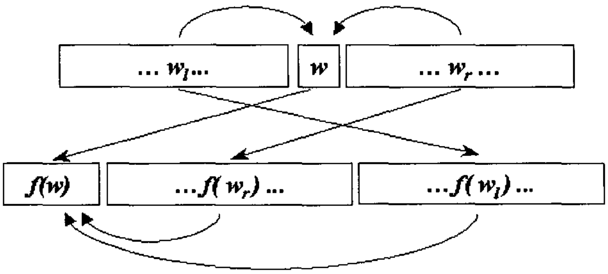

Figure 7
Decomposing source and target strings around heads w and $f(w)$.

region is decomposed around a right headword w_r. This process of decomposition continues for each left and right substring until it only contains a single word.

For each bitext there are, in general, multiple such recursive decompositions that satisfy the synchronization constraints for hierarchical alignments. We wish to find such an alignment that respects the co-occurrence statistics of bitexts as well as the phrasal structure implicit in the source and target strings. For this purpose we define a cost function on hierarchical alignments. The cost function is the sum of three terms. The first term is the total of all the translation pairing costs $c(w,f(w))$ of each source word w and its translation $f(w)$ in the alignment; the second term is proportional to the distance in the source string between dependents w_d and their heads $g(w_d)$; and the third term is proportional to the distance in the target string between target dependent words v_d and their heads $h(v_d)$.

The hierarchical alignment that minimizes this cost function is computed using a dynamic programming procedure. In this procedure, the pairing costs are first retrieved for each possible source-target pair allowed by the example. Adjacent source substrings are then combined to determine the lowest-cost subalignments for successively larger substrings of the bitext satisfying the constraints stated above. The successively larger substrings eventually span the entire source string, yielding the optimal hierarchical alignment for the bitext. This procedure has $O(n^6)$ complexity in the number of words in the source (or target) sentence. In Alshawi and Douglas (2000) we describe a version of the alignment algorithm in which heads may have an arbitrary number of dependents, and in which the hierarchical alignments for the training corpus are refined by iterative reestimation.

4.3 Constructing Transducers

Building a head transducer involves creating appropriate head transducer states and tracing hypothesized head transducer transitions between them that are consistent with the hierarchical alignment of a bitext.

The main transitions that are traced in our construction are those that map heads, w_l and w_r, of the right and left dependent phrases of w to their translations as indicated by the alignment function f in the hierarchical alignment. The positions of the dependents in the target string are computed by comparing the positions of $f(w_l)$ and $f(w_r)$ to the position of $v = f(w)$.

In order to generalize from instances in the training data, some model states arising from different training instances are shared. In particular, in the construction described here, for a given pair (w, v) there is only one final state. (We have also tried using automatic word-clustering techniques to merge states further, but for the limited domain corpora we have used so far, the results are inconclusive.) To specify

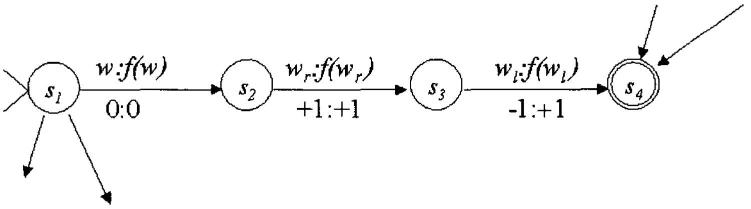

Figure 8
States and transitions constructed for the "swapping" decomposition shown in Figure 7.

the sharing of states we make use of a one-to-one state-naming function σ from sequences of strings to transducer states. The same state-naming function is used for all examples in the data set, ensuring that the transducer fragments recorded for the entire data set will form a complete collection of head transducer transition networks.

Figure 7 shows a decomposition in which w has a dependent to either side, v has both dependents to the right, and the alignment is "swapping" ($f(w_l)$ is to the right of $f(w_r)$). The construction for this decomposition case is illustrated in Figure 8 as part of a finite-state transition diagram, and described in more detail below. (The other transition arrows shown in the diagram will arise from other bitext alignments containing $(w,f(w))$ pairings.) Other cases covered by our algorithm (e.g., a single left source dependent but no right source dependent, or target dependents on either side of the target head) are simple variants.

The detailed construction is as follows:

1. Construct a transition from $s_1 = \sigma(initial)$ to $s_2 = \sigma(w,f(w),head)$ mapping the source headword w to the target head $f(w)$ at position 0 in source and target. (In our training construction there is only one initial state s_1.)

2. Since the target dependent $f(w_r)$ is to the left of target dependent $f(w_l)$ (and we are restricting positions to $\{-1,0,+1\}$) the w_r transition is constructed first in order that the target dependent nearest the head is output first.
   Construct a transition from s_2 to $s_3 = \sigma(w,f(w),swapping,w_r,f(w_r)$ mapping the source dependent w_r at position +1 to the target dependent $f(w_r)$ at position +1.

3. Construct a transition from s_3 to $s_4 = \sigma(w,f(w),final)$ mapping the source dependent w_l at position -1 to the target dependent $f(w_l)$ at position +1.

If instead the alignment had been as in Figure 9, in which the source dependents are mapped to target dependents in a parallel rather than swapping configuration (the configuration of *sin escalas* and *Boston* around *flights:los vuelos* in Figure 6), the construction is the same, except for the following differences:

1. Since the target dependent $f(w_l)$ is to the left of target dependent $f(w_r)$, the w_l transition is constructed first in order that the target dependent nearest the head is output first.

2. The source and target positions are as shown in Figure 10. Instead of state s_3, we use a different state $s_5 = \sigma(w,f(w),parallel,w_l,f(w_l))$.

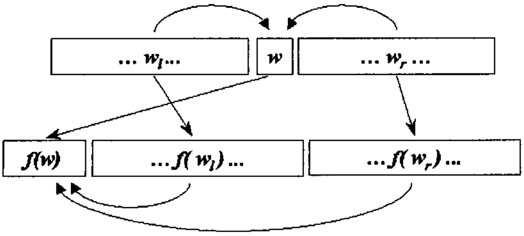

Figure 9
Decomposing source and target strings around heads w and $f(w)$—"parallel".

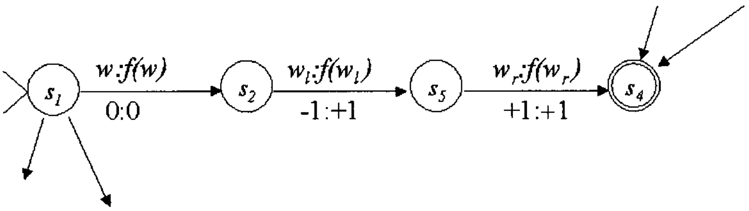

Figure 10
States and transitions constructed for the "parallel" decomposition shown in Figure 9.

Other states are the same as for the first case. The resulting states and transitions are shown in Figure 10.

After the construction described above is applied to the entire set of aligned bitexts in the training set, the counts for transitions are treated as event observation counts of a statistical dependency transduction model with the parameters described in Section 3.1. More specifically, the negated logs of these parameters are used as the weights for transducer transitions.

4.4 Multiword Pairings

In the translation application, source word w and target word v are generalized so they can be short substrings (**compounds**) of the source and target strings. Examples of such multiword pairs are *show me:muestrême* and *nonstop:sin escalas* in Figure 6. The cost for such pairings still uses the same ϕ statistic, now taking the observations to be the co-occurrences of the substrings in the training bitexts. However, in order that these costs can be comparable to the costs for simple pairings, they are multiplied by the number of words in the source substring of the pairing.

The use of compounds in pairings does not require any fundamental changes to the hierarchical alignment dynamic programming algorithm, which simply produces dependency trees with nodes that may be compounds. In the transducer construction phase of the training method, one of the words of a compound is taken to be the primary or "real" headword. (In fact, we take the least common word of a compound to be its head.) An extra chain of transitions is constructed to transduce the other words of compounds, if necessary using transitions with epsilon strings. This compilation means that the transduction algorithm is unaffected by the use of compounds when aligning training data, and there is no need for a separate compound identification phase when the transduction algorithm is applied to test data. Some results for different choices of substring lengths can be found in Alshawi, Bangalore, and Douglas (1998).

5. Experiments

5.1 Evaluation Method

In order to reduce the time required to carry out training evaluation experiments, we have chosen two simple, string-based evaluation metrics that can be calculated automatically. These metrics, **simple accuracy** and **translation accuracy**, are used to compare the target string produced by the system against a reference human translation from held-out data.

Simple accuracy is computed by first finding a transformation of one string into another that minimizes the total weight of insertions, deletions, and substitutions. (We use the same weights for these operations as in the NIST ASR evaluation software [National Institute of Standards and Technology 1997].) Translation accuracy includes transpositions (i.e., movement) of words as well as insertions, deletions, and substitutions. We regard the latter metric as more appropriate for evaluation of translation systems because the simple metric would count a transposition as two errors: an insertion plus a deletion. (This issue does not arise for speech recognizers because these systems do not normally make transposition errors.)

For the lowest edit-distance transformation between the reference translation and system output, if we write I for the number of insertions, D for deletions, S for substitutions, and R for number of words in the reference translation string, we can express simple accuracy as

$$\text{simple accuracy} = 1 - (I + D + S)/R.$$

Similarly, if T is the number of transpositions in the lowest weight transformation including transpositions, we can express translation accuracy as

$$\text{translation accuracy} = 1 - (I' + D' + S + T)/R.$$

Since a transposition corresponds to an insertion and a deletion, the values of I' and D' for translation accuracy will, in general, be different from I and D in the computation of simple accuracy. For Spanish, the units for string operations in the evaluation metrics are words, whereas for Japanese they are Japanese characters.

5.2 English-to-Spanish

The training and test data for the English-to-Spanish experiments were taken from a set of transcribed utterances from the Air Travel Information System (ATIS) corpus together with a translation of each utterance to Spanish. An utterance is typically a single sentence but is sometimes more than one sentence spoken in sequence. Alignment search and transduction training was carried out only on bitexts with sentences up to length 20, a total of 13,966 training bitexts. The test set consisted of 1,185 held-out bitexts at all lengths. Table 1 shows the word accuracy percentages (see Section 5.1) for the trained model, **e2s**, against the original held-out translations at various source sentence lengths. Scores are also given for a "word-for-word" baseline, **sww**, in which each English word is translated by the most highly correlated Spanish word.

5.3 English-to-Japanese

The training and test data for the English-to-Japanese experiments was a set of transcribed utterances of telephone service customers talking to AT&T operators. These utterances, collected from real customer-operator interactions, tend to include fragmented language, restarts, etc. Both training and test partitions were restricted to bitexts with at most 20 English words, giving 12,226 training bitexts and 3,253 held-out test bitexts. In the Japanese text, we introduce "word" boundaries that are convenient

Table 1
Simple accuracy/translation accuracy (percent) for the trained English-to-Spanish model (**e2s**) against the word-for-word baseline (**sww**).

Length	≤ 5	≤ 10	≤ 15	≤ 20	All
sww	45.1/45.8	46.7/48.6	46.5/48.2	45.5/47.1	45.2/46.9
e2s	75.4/75.8	76.3/78.0	75.4/77.0	74.4/76.0	73.3/75.0

Table 2
Simple accuracy/translation accuracy as percentages of Japanese characters, for the trained English-to-Japanese model (**e2j**) and the word-for-word baseline (**jww**).

Length	≤ 5	≤ 10	≤ 15	≤ 20	All
jww	75.8/78.0	45.2/50.4	40.0/45.4	37.2/42.8	37.2/42.8
e2j	89.2/89.7	74.0/76.6	68.6/72.2	66.4/70.1	66.4/70.1

for the training process. These word boundaries are parasitic on the word boundaries in the English transcriptions: the translators are asked to insert such a word boundary between any two Japanese characters that are taken to have arisen from the translation of distinct English words. This results in bitexts in which the number of multicharacter Japanese "words" is at most the number of English words. However, as noted above, evaluation of the Japanese output is done with Japanese characters, i.e., with the Japanese text in its natural format. Table 2 shows the Japanese character accuracy percentages for the trained English-to-Japanese model, **e2j**, and a baseline model, **jww**, which gives each English word its most highly correlated translation.

5.4 Note on Experimental Setting

The vocabularies in these English-Spanish and English-Japanese experiments are only a few thousand words; the utterances are fairly short (an average of 7.3 words per utterance) and often contain errors typical of spoken language. So while the domains may be representative of task-oriented dialogue settings, further experimentation would be needed to assess the effectiveness of our method in situations such as translating newspaper articles. In terms of the training data required, Tsukada et al. (1999) provide indirect empirical evidence suggesting accuracy can be further improved by increasing the size of our training sets, though also suggesting that the learning curve is relatively shallow beyond the current size of corpus.

6. Concluding Remarks

Formalisms for finite-state and context-free transduction have a long history (e.g., Lewis and Stearns 1968; Aho and Ullman 1972), and such formalisms have been applied to the machine translation problem, both in the finite-state case (e.g., Vilar et al. 1996) and the context-free case (e.g., Wu 1997). In this paper we have added to this line of research by providing a method for automatically constructing fully lexicalized statistical dependency transduction models from training examples.

Automatically training a translation system brings important benefits in terms of maintainability, robustness, and reducing expert coding effort as compared with traditional rule-based translation systems (a number of which are described in Hutchins and Somers [1992]). The reduction of effort results, in large part, from being able to do without artificial intermediate representations of meaning; we do not require the development of semantic mapping rules (or indeed any rules) or the creation of a corpus including semantic annotations. Compared with left-to-right transduction, middle-out transduction also aids robustness because, when complete derivations are not available, partial derivations tend to have meaningful headwords.

At the same time, we believe our method has advantages over the approach developed initially at IBM (Brown et al. 1990; Brown et al. 1993) for training translation systems automatically. One advantage is that our method attempts to model the natural decomposition of sentences into phrases. Another is that the compilation of this decomposition into lexically anchored finite-state head transducers produces implementations that are much more efficient than those for the IBM model. In particular, our search algorithm finds optimal transductions of test sentences in less than "real time" on a 300MHz processor, that is, the time to translate an utterance is less than the time taken to speak it, an important consideration for our speech translation application.

References

Aho, Alfred V. and Jeffrey D. Ullman. 1972. *The Theory of Parsing, Translation, and Compiling*. Prentice-Hall, Englewood Cliffs, NJ.

Alshawi, H. 1996. Head automata for speech translation. In *Proceedings of the International Conference on Spoken Language Processing*, pages 2360–2364, Philadelphia, PA.

Alshawi, H., S. Bangalore, and S. Douglas. 1998. Learning phrase-based head transduction models for translation of spoken utterances. In *Proceedings of the International Conference on Spoken Language Processing*, pages 2767–2770, Sydney, Australia.

Alshawi, H. and S. Douglas. 2000. Learning dependency transduction models from unannotated examples. *Philosophical Transactions of the Royal Society (Series A: Mathematical, Physical and Engineering Sciences)*. To appear.

Alshawi, Hiyan and Fei Xia. 1997. English-to-Mandarin speech translation with head transducers. In *Proceedings of the Workshop on Spoken Language Translation*, Madrid, Spain.

Brown, P. J., J. Cocke, S. A. Della Pietra, V. J. Della Pietra, J. Lafferty, R. L. Mercer, and P. Rossin. 1990. A statistical approach to machine translation. *Computational Linguistics*, 16(2):79–85.

Brown, P. J., S. A. Della Pietra, V. J. Della Pietra, and R. L. Mercer. 1993. The mathematics of machine translation: Parameter estimation. *Computational Linguistics*, 16(2):263–312.

Dorr, B. J. 1994. Machine translation divergences: A formal description and proposed solution. *Computational Linguistics*, 20(4):597–634.

Earley, J. 1970. An efficient context-free parsing algorithm. *Communications of the ACM*, 13(2):94–102.

Gale, W. A. and K. W. Church. 1991. Identifying word correspondences in parallel texts. In *Proceedings of the Fourth DARPA Speech and Natural Language Processing Workshop*, pages 152–157, Pacific Grove, CA.

Hays, D. G. 1964. Dependency theory: A formalism and some observations. *Language*, 40:511–525.

Hudson, R. A. 1984. *Word Grammar*. Blackwell, Oxford.

Hutchins, W. J. and H. L. Somers. 1992. *An Introduction to Machine Translation*. Academic Press, New York.

Lewis, P. M. and R. E. Stearns. 1968. Syntax-directed transduction. *Journal of the Association for Computing Machinery*, 15(3):465–488.

National Institute of Standards and Technology. 1997. Spoken Natural Language Processing Group Web page. http://www.itl.nist.gov/div894.

Tsukada, Hajime, Hiyan Alshawi, Shona Douglas, and Srinivas Bangalore. 1999. Evaluation of machine translation system based on a statistical method by using spontaneous speech transcription. In *Proceedings of the Fall Meeting of the Acoustical Society of Japan*, pages 115–116, September.

Vilar, J. M., V. M. Jiménez, J. C. Amengual, A. Castellanos, D. Llorens, and E. Vidal. 1996. Text and speech translation by means of subsequential transducers. *Natural Language Engineering*, 2(4):351–354.

Watanabe, Hideo. 1995. A model of a bi-directional transfer mechanism using rule combination. *Machine Translation*, 10(4):269–291.

Wu, Dekai.1997. Stochastic inversion transduction grammars and bilingual parsing of parallel corpora. *Computational Linguistics*, 23(3):377–404.

Younger, D. 1967. Recognition and Parsing of Context-Free Languages in Time n^3. *Information and Control*, 10:189–208.