Proceedings of the Workshop on Speech-to-Speech Translation:
Algorithms and Systems, Philadelphia, July 2002, pp. 31-38.
Association for Computational Linguistics.

# Speech Translation Performance of Statistical Dependency Transduction and Semantic Similarity Transduction

**Hiyan Alshawi** and **Shona Douglas**
AT&T Labs - Research
Florham Park, NJ 07932, USA
{hiyan,shona}@research.att.com

## Abstract

In this paper we compare the performance of two methods for speech translation. One is a statistical dependency transduction model using head transducers, the other a case-based transduction model involving a lexical similarity measure. Examples of translated utterance transcriptions are used in training both models, though the case-based model also uses semantic labels classifying the source utterances. The main conclusion is that while the two methods provide similar translation accuracy under the experimental conditions and accuracy metric used, the statistical dependency transduction method is significantly faster at computing translations.

## 1 Introduction

Machine translation, natural language processing, and more generally other computational problems that are not amenable to closed form solutions, have typically been tackled by one of three broad approaches: rule-based systems, statistical models (including generative models), and case-based systems. Hybrid solutions combining these approaches have also been used in language processing generally (Klavans and Resnik, 1996) and more specifically in machine translation (for example Frederking et al. (1994)).

In this paper we compare the performance of two methods for speech translation. One is the statistical dependency transduction model (Alshawi and Douglas, 2000; Alshawi et al., 2000b), a trainable generative statistical translation model using head transducers (Alshawi, 1996). The other is a case-based transduction model which makes use of a semantic similarity measure between words. Both models are trained automatically using examples of translated utterances (the transcription of a spoken utterance and a translation of that transcription). The case-based model makes use of additional information in the form of labels associated with source language utterances, typically one or two labels per utterance. This additional information, which was originally provided for a separate monolingual task, is used to construct the lexical similarity measure.

In training these translation methods, as well as their runtime application, no pre-existing bilingual lexicon is needed. Instead, in both cases, the initial phase of training from the translation data is a statistical hierarchical alignment search applied to the set of bilingual examples. This training phase produces a bilingual lexicon, used by both methods, as well as synchronized hierarchical alignments used to build the dependency transduction model.

In the experiments comparing the performance of the models we look at accuracy as well as the time taken to translate sentences from English to Japanese. The source language inputs used in these experiments are naturally spoken utterances from large numbers of real customers calling telephone operator services.

In section 2 we describe the hierarchical alignment algorithm followed by descriptions of the translation methods in sections 3 and 4. We present the experiments in section 5 and provide concluding remarks in section 6.

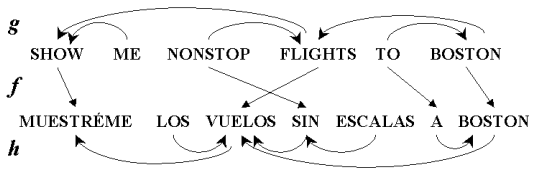


Figure 1: Alignment mapping $f$, source head-map $g$, and target head-map $h$

## 2 Hierarchical alignments

Both the translation systems described in this paper make use of automatically created hierarchical alignments of the source and target strings of the training corpus bitexts. As will be described in section 3, we estimate the parameters of a dependency transduction model from such alignments. In the case-based method described in section 4, the alignments are the basis for the translation lexicon used to compute substitutions and word-for-word translations.

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 word $\epsilon$), 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 $\epsilon$; it coincides with $f$ for pairs without $\epsilon$.) 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 head words $h(v)$ in the target string. An example hierarchical alignment is shown in Figure 1.

A hierarchical alignment is **synchronized** (i.e. corresponds to synchronized dependency trees) if, roughly speaking, $f$ induces an isomorphism between the dependency functions $g$ and $h$ (see Alshawi and Douglas (2000) for a more formal definition). The hierarchical alignment in Figure 1 is synchronized.

In some previous work (Alshawi et al., 1998; Alshawi et al., 2000a; Alshawi et al., 2000b) the training method constructs synchronized alignments in which each head word has at most two dependent phrases. Here we use the technique described by Alshawi and Douglas (2000) where the models have greater freedom to vary the granularity of phrase locality.

Constructing synchronized hierarchical alignments for a corpus has two stages: (a) computing co-occurrence statistics from the training data; (b) searching for an optimal synchronized hierarchical alignment for each bitext.

### 2.1 Word correlation statistics

For each source word in the dataset, a **translation pairing cost** $r(w, v, b)$ is assigned for all possible translations in the context of a bitext $b$. Here $w$ and $v$ are usually words, but may also be the empty word $\epsilon$ or compounds formed from contiguous words; here we restrict compounds to a maximum length of two words.

The assignment of these lexical translation pairing costs may be done using various statistical measures. The main component of $r$ is the so-called $\phi$ correlation measure (see Gale and Church (1991)) normalized to the range $[0, 1]$ with 0 indicating perfect correlation. In the experiments described in this paper, the cost function $r$ relating a source word (or compound) $w$ in a bitext with a target word (or compound) $v$ is

$$r(w, v, b) = \phi(w, v) + d(w, v, b)$$

where $d(w, v, b)$ is a length-normalized measure of the apparent distortion in the positions of $w$ and $v$ in the source and target strings of $b$. For example, if $w$ appears at the middle of the source string and $v$ appears at the middle of the target string, then the distortion is 0. We have found that, at least for our data, this pairing cost leads to better performance than the use of log probabilities of target words given source words (cf. Brown et al. (1993)).

The value used for $\phi(w, v)$ is first computed from counts of the number of bitexts in the training set in which $w$ and $v$ co-occur, in which $w$ only appears, in which $v$ only appears, and in which neither of them appear. In other words, we first treat any word in the target string to be a possible translation of any word in the source string. This value is then refined by re-estimation during the alignment optimization process.

### 2.2 Optimal hierarchical alignments

We wish to find a hierarchical 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 the cost of a hierarchical subalignment to be the sum of the costs $r(w, v, b)$ of each pairing $(w, v) \in f$, where $f$ is the (sub)alignment mapping function.

The complete hierarchical alignment which minimizes this cost function is computed using a dynamic programming procedure. This procedure works bottom-up, starting with all possible subalignments with at most one source word (or compound) and one target word (or compound). Adjacent source substrings are then combined to determine the lowest cost subalignments for successively larger substrings of the bitext satisfying the constraints for synchronized alignments stated above. The successively larger substrings eventually span the entire source string, yielding the optimal hierarchical alignment for the bitext.

At each combination step in the optimization procedure, one of the two source subphrases is added as a dependent of the head of the other subphrase. Since the alignment we are constructing is synchronized, this choice will force the selection of a target dependent phrase. Our current (admittedly crude) strategy for selecting the dependent subphrase is to choose the one with the highest subalignment cost, i.e. the head of the subphrase with the better subalignment becomes the head of the enlarged phrase.

Recall that the initial estimates for $\phi$ are computed from co-occurence counts for $w, v$ in bitexts. In the second and subsequent rounds of this procedure, the $\phi$ values are computed from co-occurence counts for $(w, v)$ in *pairings* in the alignments produced by the previous round. The improvement in the models resulting from this re-estimation seems to stabilize after approximately five to ten rounds.

## 3 Statistical Dependency Transduction

The dependency transduction model is an automatically trainable translation method that models cross-lingual lexical mapping, hierarchical phrase structure, and monolingual lexical dependency. It is a generative statistical model for synchronized pairs of dependency trees in which each local tree is produced by a weighted head transducer. Since this model has been presented at length elsewhere (Alshawi, 1996; Alshawi et al., 2000a; Alshawi and Douglas, 2000), the description in this paper will be relatively compact.

### 3.1 Weighted finite state head transducers

A weighted finite state 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.

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 $\epsilon$; $v$ is a member of $V$ or $\epsilon$; the integer $\alpha$ is the **input position**; the integer $\beta$ is the **output position**; and the real number $c$ is the weight of the transition. The roles of $q$, $q'$, $w$, and $v$ in transitions are similar to the roles they have in 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 $\epsilon$ then no read (respectively write) takes place for the transition.

To define the role of transition positions $\alpha$ and $\beta$, we consider notional input (source) and output (target) 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. A transition with input position $\alpha$ and output position $\beta$ is interpreted as reading $w$ from square $\alpha$ on the input tape and writing $v$ to square $\beta$ of the output tape; if square $\beta$ is already occupied then $v$ is written to the next empty square to the left of $\beta$ if $\beta < 0$, or to the right of $\beta$ if $\beta \geq 0$, and similarly if input was already read from position $\alpha$, $w$ is taken from the next unread square to the left of $\alpha$ if $\alpha < 0$ or to the right of $\alpha$ if $\alpha \geq 0$.

### 3.2 Dependency transduction models

**Dependency transduction models** are generative statistical models which derive synchronized pairs of **dependency trees**, a source language dependency tree and a target dependency tree. A dependency tree, in the sense of dependency grammar (for example Hays (1964), Hudson (1984)), is a tree in which the words of a sentence appear as nodes; the parent of a node is its **head** and the child of a node is the node's **dependent**.

In a dependency transduction model, each synchronized local subtree corresponds to a head transducer derivation: the head transducer is used to convert a sequence consisting of a head word $w$ and its immediate left and right dependent words to a sequence consisting of a target word $v$ and $its$ immediate left and right dependent words. (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.) When applying a dependency transduction model to translation, we choose the target string obtained by flattening the target tree of the lowest cost recursive dependency derivation that also yields the source string.

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 probabilistic parameterizations can be used for this purpose including the following for a transition with head words $w$ 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 $\alpha$ and $\beta$ are the source and target positions, as before. We also need parameters $p(q_0|w, v)$ for the probability of choosing an initial head transducer state $q_0$ given a pair of words $(w, v)$ heading a synchronized pair of subtrees. To start the derivation, we need parameters $p(R(w_0, v_0))$ 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:

$$p(R(w_0, v_0))p(D_{w_0,v_0})$$

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

$$p(D_{w,v}) = p(q_0|w, v) \prod_{0 \le i \le n} p(q_{i+1}, w_i, v_i, \alpha_i, \beta_i | w, v, q_i) p(D_{w_i,v_i})$$

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

The parameters of this probabilistic synchronized tree derivation model are estimated from the results of running the hierarchical alignment algorithm described in section 2 on the sentence pairs in the training corpus. For this purpose, each synchronized tree resulting from the alignment process is assumed to be derived from a dependency transduction model, so transition counts for the model are tallied from the set of synchronized trees. (For further details, see Alshawi and Douglas (2000).)

To carry out translation with a dependency transduction model, we apply a "middle-out" 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). It is described in Alshawi et al. (2000b).

## 4 Similarity Cased-Based Transduction

### 4.1 Training the transduction parameters

Our semantic similarity transduction method is a case-based (or example-based) method for transducing source strings to target strings that makes use of two different kinds of training data:

- A set of source-string, target-string pairs that are instances of the transduction mapping. Specifically, transcriptions of spoken utterances in the source language and their translation into the target language. This is the same data used for training the dependency transduction model. It is used in this transduction

method to construct a probabilistic bilingual lexicon, while the source side is used as the set of examples for matching.

- A mapping between the source strings and subsets of a (relatively small) set of classes, or labels. The idea is that the labels give a broad classification of the meaning of the source strings, so we will refer to them informally as "semantic" labels. In our experiments, these classes correspond to 15 call routing destinations associated with the transcribed utterances. For the purposes of the case-based method, this data is used to construct a similarity measure between words of the source language.

As noted earlier, the alignment algorithm described in section 2 is applied to the translation pairs to yield a set of synchronized dependency trees. Using the resulting trees, the probabilities of a bilingual lexicon, i.e.

$$p(v|w)$$

where $w$ is a source language word, and $v$ is a target language word, are estimated from the counts of synchronized lexical nodes. (Since the synchronized trees are dependency trees, both paired fringe nodes and interior nodes are included in the counts.) In this probabilistic lexicon, $v$ may be $\epsilon$, the empty symbol, so source words may have different probabilities of being deleted. However, for insertion probabilities, we assume that $P(\epsilon|\epsilon) = 1$, to avoid problems with spurious insertions of target words.

The labels associated with the source strings were originally assigned by manual annotation for the purposes of a different research project, specifically for training an automatic call routing system, using the methods described by Gorin et al. (1997). (Many of the training sentences are assigned multiple labels.)

For the translation task, the labels are used to compute a similarity measure $m(w_1, w_2)$ as a divergence between a probability distribution conditional on source word $w_1$ and a corresponding distribution conditional on another source word $w_2$. The distributions involved, $p(L|w_1)$ and $p(L|w_2)$, are those for the probability $p(l|w)$ that a source string which includes word $w$ has been assigned label $l$. The similarity measure $m(w_1, w_2)$ is computed from the relative entropy $D$ (Kullback Leibler distance (Kullback and Leibler, 1951)) between these distributions. To make the similarity measure symmetrical, i.e. $m(w_1, w_2) = m(w_2, w_1)$, we take the average of two relative entropy quantities:

$$\begin{aligned} m(w_1, w_2) = \quad & 1/2 \quad (D(p(L|w_1)||p(L|w_2)) + \\ & \qquad D(p(L|w_2)||p(L|w_1))) \end{aligned}$$

Of course, this is one of many different possible similarity measures which could have been used (cf Pereira et al. (1993)), including ones that do not depend on additional labels. However, since semantic labels had already been assigned to our training data, the distributions seemed like a convenient rough proxy for the semantic similarity of words in this limited domain.

### 4.2 Case-based transduction procedure

Basically, the transduction procedure (i) finds an instance $(s, t)$ of the translation training pairs for which the example source string $s$ provides the "best" match to the input source string $u$, and (ii) produces, as the translation output, a modified version of the example target string $t$, where the modifications reflect mismatches between $s$ and the input.

For the first step, the similarity measure between words computed in terms of the relative entropy for label distributions is used to compute a distance

$$d(s1, s2)$$

between two source strings $s_1$ and $s_2$. The (semantically influenced) string distance $d$, is a weighted edit distance (Wagner and Fischer, 1974) between the two strings in which the cost of substituting one source word $w_1$ for another $w_2$ is provided by the "semantic" similarity measure $m(w_1, w_2)$. A standard quadratic dynamic programming search algorithm is used to find the weighted edit distance between two strings. This algorithm finds a sequence of edit operations (insertions, deletions, and substitutions) that yield $s2$ from $s1$ so that $d(s1, s2)$, the sum of the costs of the edit operations, is minimal over all such edit sequences.

The weighted edit distance search is applied to $u$ and each example source string $s$ to identify the example translation pair $(s, t)$ for which $d(u, s)$ is

minimal over all example source strings. The corresponding sequence of edits for this minimal distance is used to compute a modified version $t'$ from $t$. For this purpose, the source language edits are "translated" into corresponding target language edits using the probabilistic bilingual lexicon estimated from aligning the training data. Specifically, for each substitution $w_1 \mapsto w_2$ in the edits resulting from the weighted edit distance search, a substitution $v_1 \mapsto v_2$ is applied to $t$. Here $v_i$ is chosen so that $p(v_i|w_i)$ is maximal. The translated edits are applied sequentially to $t$ to give $t'$.

The modified example target string $t'$ is used as the output of this translation method unless the minimal edit distance between $u$ and the closest example $s$ exceeds a threshold determined experimentally. (For this purpose, the edit distance is normalized by utterance length.) If the threshold is exceeded, so that no "sufficiently close" examples are available, then a word-for-word translation is used as the output by simply applying the probabilistic lexicon to each word of the input. It is perhaps worth mentioning that the statistical dependency transduction method does not need a such a fall-back to word-for-word translation: the middle-out (island parsing) search algorithm used with head transducers gracefully degrades into word-for-word translation when the training data is too sparse to cover the input string.

## 5 Experiments and results

### 5.1 Data set

The corpora for the experiments reported here consist of spoken English utterances, paired with their translations into Japanese. The English utterances were the customer side of actual AT&T customer-operator conversations. There were 12,226 training bitexts and an additional 3,253 bitexts for testing. In the text experiments, the English side of the bitext is the human transcriptions of the recorded speech; in the speech experiments, it is the output of speech recognition. The case-based model makes use of additional information in the form of labels associated with source language utterances, classifying the source utterances into 15 task related classes such as "collect-call", "directory-assistance", etc.

The translations were carried out by a commercial translation company. Since Japanese text has no word boundaries, we asked the translators to insert spaces between Japanese characters whenever they 'arose from different English words in the source'. This imposed an English-centric view of Japanese text segmentation.

### 5.2 Evaluation metrics

We use two simple string edit-distance 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 the reference human translation from held-out data. Simple accuracy (the 'word accuracy' of speech recognition research) is computed by first finding a transformation of one string into another that minimizes the total number of insertions, deletions and substitutions. Translation accuracy includes transpositions (i.e. movement) of words as well as insertions, deletions, and substitutions. We regard the latter measure as more appropriate for evaluation of translation systems because the simple metric would count a transposition as two errors: an insertion plus a deletion. If we write $i$ for the number of insertions, $d$ for deletions, $s$ for substitutions, $t$ for transpositions, and $r$ for number of words in the reference translation string, we can express the metrics as follows:

$$\text{simple accuracy} = 1 - (i + d + s)/r$$

$$\text{translation accuracy} = 1 - (i + d + s + t)/r$$

Since a transposition corresponds to an insertion and a deletion, the values of $i$ and $d$ will be different in the expressions for computing the two accuracy metrics. The units for string operations in the evaluation metrics are Japanese characters.

### 5.3 Experimental conditions and results

The following experimental systems are evaluated here:

**Word-Word** A simple word for word baseline method in which each source word is replaced with the most highly correlated target word in the training corpus.

**Stat-Dep** The statistical dependency transduction method as described in section 3.

| | Simple accuracy | Translation accuracy |
|---|---|---|
| Word-Word | 37.2 | 42.8 |
| Stat-Dep | 69.3 | 72.9 |
| Sim-Case | 70.6 | 71.5 |

Table 1: Accuracy for text (%)

| | Simple accuracy | Translation accuracy |
|---|---|---|
| Word-Word | 29.2 | 33.7 |
| Stat-Dep | 57.4 | 59.7 |
| Sim-Case | 59.4 | 60.2 |

Table 2: Accuracy for speech (%)

**Sim-Case** The semantic similarity case-based method described in section 4.

Table 1 shows the results on human transcriptions of the set of test utterances.

Table 2 shows the test set results of translating automatic speech recognition output. The speech recognizer used a speaker-independent telephony acoustic model and a statisical trigram language model.

Table 3 shows the speed of loading (once per test set) and the average run time per utterance translation for the dependency transduction and case-based systems.

## 6 Concluding Remarks

In this paper we have compared the accuracy and speed of two translation methods, statistical dependency transduction and semantic similarity cased-based transduction. The statistical transduction model is trainable from unannotated examples of sentence translations, while the case-based method additionally makes use of a modest amount of annotation to learn a lexical semantic similarity function, a factor in favor of the dependency transduction method.

In the experiments we presented, the transduction methods were applied to translating automatic speech recognition output for English utterances into Japanese in a limited domain. The evaluation metric used to compare translation accuracy was an automatic string comparison function applied to the output produced by both methods. The basic result

| | Load time | Run time/ translation |
|---|---|---|
| *text* | | |
| Stat-Dep | 7176 | 53 |
| Sim-Case | 3856 | 2220 |
| *speech* | | |
| Stat-Dep | 7447 | 66 |
| Sim-Case | 5925 | 2333 |

Table 3: Translation time (ms)

was that translation accuracy was very similar for both models, while the statistical dependency transduction method was significantly faster at producing translations at run time. Since training time for both methods is dominated by the alignment training phase they share, training time issues do not favor one method over the other.

These results need to be interpreted in the rather narrow experimental setting used here: the amount of training data used, the specific language pair (English to Japanese), the evaluation metric, and the uncertainty in the input strings (speech recognition output) to which the methods were applied. Further research varying these experimental conditions is needed to provide a fuller comparison of the relative performance of the methods. However, it should be possible to develop algorithmic improvements to increase the computational efficiency of similarity cased-based transduction to make it more competitive with statistical dependency transduction at runtime.

## References

H. Alshawi and S. Douglas. 2000. Learning dependency transduction models from unannotated examples. *Philosophical Transactions of the Royal Society (Series A: Mathematical, Physical and Engineering Sciences)*, 358:1357–1372, April.

H. Alshawi, 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.

H. Alshawi, S. Bangalore, and S. Douglas. 2000a. Head transducer models for speech translation and their automatic acquisition from bilingual data. *Machine Translation*, 15(1/2):105–124.

H. Alshawi, S. Bangalore, and S. Douglas. 2000b. Learning dependency translation models as collections of finite state head transducers. *Computational Linguistics*, 26(1), January.

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

P.J. Brown, 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.

J. Earley. 1970. An Efficient Context-Free Parsing Algorithm. *Communications of the ACM*, 13(2):94–102.

R. Frederking, S. Nirenburg, D. Farwell, S. Helmreich, E. Hovy, K. Knight, S. Beale, C. Domashnev, D. Attardo, D. Grannes, and R. Brown. 1994. Integrating translations from multiple sources within the pangloss mark iii machine translation. In *Proceedings of the first conference of the Association for Machine Translation in the Americas (AMTA-94)*, Maryland.

W.A. Gale 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, California.

A.L. Gorin, G. Riccardi, and J.H. Wright. 1997. How may I help you? *Speech Communication*, 23(1-2):113–127.

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

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

Judith L. Klavans and Philip Resnik, editors. 1996. *The Balancing Act: combining Symbolic and Statistical Approaches to Language*. The MIT Press.

S. Kullback and R. A. Leibler. 1951. On information and sufficiency. *Annals of Mathematical Statistics*, 22:76–86.

F. Pereira, N. Tishby, and L. Lee. 1993. Distributional clustering of English words. In *Proceedings of the 31st meeting of the Association for Computational Linguistics*, pages 183–190.

Robert A. Wagner and Michael J. Fischer. 1974. The string-to-string correction problem. *Journal of the Association for Computing Machinery*, 21(1):168–173, January.

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