The CMU-UKA Syntax Augmented Machine Translation System for IWSLT-06

Andreas Zollmann, Ashish Venugopal, Stephan Vogel, Alex Waibel

interACT
University of Karlsruhe, Germany / Carnegie Mellon University, U.S.A.
{zollmann,ashishv,vogel,ahw}@cs.cmu.edu

Abstract

We present the CMU-UKA Syntax Augmented Machine Translation System that was used in the IWSLT-06 evaluation campaign. We participated in the C-Star data track using only the Full BTEC corpus, for Chinese-English translation, focusing on transcript translation. We applied techniques that produce true-cased, punctuated translations from non-punctuated Chinese transcripts, generating translations which score higher against the *Official* metric than against the lower-cased, punctuation removed metric. Our results demonstrate the impact of syntax and hierarchy based models for speech transcript translation.

1. Introduction

As [1] and [2] note, phrase-based models suffer from sparse data effects when required to translate conceptual elements that span or skip across several words, and distortion based re-ordering techniques tend to limit their range of operation for reasons of efficiency and model strength [3]. Syntax driven [4], [5] and hierarchical translation models [1] model structured re-ordering constraints and extend the domain of locality in the decoding process. Systems such as [6] and [7] demonstrate effective decoding using these models.

For the IWSLT-06 evaluation, we applied syntax augmented translation as per [7] to the speech translation task, using the only Full BTEC corpus to model translational equivalence. We begin by extracting lexical phrases as per [2] as the basis for our syntax augmented translation rules. We annotate and generalize these phrases by parsing the target side of the training data with the Stanford Parser [8] (trained on the Penn Treebank). Our results indicate steady improvements as we introduce hierarchy and syntax into the translation process despite the domain mismatch with the English parser.

As defined in the evaluation's *Official Specifications*, translations are evaluated considering case and punctuation markers, artifacts not typically present in Chinese ASR transcripts. We incorporate the generation of punctuation directly into the translation process, learning translation rules that represent case and punctuation decisions. Our *Official* evaluation results demonstrate the effectiveness of these approaches; our submission achieved higher performance when the system output was evaluated with punctuation and case than in the lower-cased, punctuation-free evaluation. This result is especially relevant considering the presence of multiple sentences (which should be punctuation-separated in English) within each speech transcript utterance.

We begin by summarizing the syntax augmented rule extraction process from [7] and the decoding settings used to train the model parameters to maximize performance [9] on the BLEU metric [10]. We provide a detailed description of the data-processing used to generate our evaluation submission, and demonstrate the impact of syntax for the IWSLT-06 speech translation task.

2. Syntax Augmented Translation

Traditional phrase-based translations serves as the lexical foundation for the syntactic synchronous grammar (SynCFG) presented in [7]. Syntactic, since its nonterminals are syntactic categories derived from parsing the target (English) side of the parallel training corpus, and synchronous because they define operations to derive the source and target language simultaneously. We train a phrase based translation model as in [2] on the bilingual training data, and associate the word alignment graph for each sentence pair with the highest probability parse of the target sentence. We obtain parse trees from the Stanford Parser pre-trained on the Penn Treebank [8].

Using the notation from [1], we aim to construct a synchronous grammar of the form

$$X \rightarrow \langle \lambda, \alpha, \sim \rangle$$

where $\lambda = f_1 \dots Y_i \dots f_m$ is a sequence of nonterminals Y_i and source language terminals f_k and $\alpha = e_1 \dots Y_j \dots \dots e_n$ is a sequence containing the same nonterminals Y_i as well as target language terminals e_k. The relation $\sim$ defines a one-to-one correspondence between nonterminal occurrences Y across λ and α. Target terminals are the English words of the system output, and our nonterminals are target language syntactic categories corresponding to the Penn Treebank nonterminal set along with extensions. Under this notation, phrase table entries represent purely lexical λ and α.

To produce rules of the form described above, we annotate the initial lexical rules (the phrase-table) with syntactic productions (left-hand side of the rule) based on the parse trees available for the target side of the corpus. We perform the phrase extraction from [2] for each sentence individually, identifying phrases based on the learned word alignment, and associating the *target* side of these phrases with constituent labels from the corresponding target parse tree. Details are provided below.

2.1. Annotation

We run the freely available phrase extraction system provided by Phillip Koehn for the NAACL-2006 Workshop on Statistical Machine Translation, generating initial word alignments using the *grow-diag-final* method. In order to identify phrases with their corresponding sentences (required to associate the target side parse tree), we insert sentence separator marks between each line of the source, target and alignment file. Calling *phrase-extract* (a binary provided in the workshop tools) with these marked files produces a phrase table with separators between each set of phrases extracted from a sentence.

For the set of phrases extracted for each sentence, we consider the corresponding parse tree for the whole target sentence. For each source phrase-pair $f_1 \ldots f_m / e_1 \ldots e_n$, we annotate this phrase-pair with the constituent that spans $e_1 \ldots e_n$ in the target side parse tree.

As an extension to the nonterminal set provided by the Penn Treebank, we consider the CCG nonterminal set proposed by [11]. Under this approach, rules can be assigned partially formed categories, like $DT\backslash NP$, indicating a constituent that forms a noun-phrase, but is missing its determiner at the left. [12] demonstrates the importance of considering phrases not corresponding to pure syntactic constituents in translation, and in [13], we demonstrate the value of using extended categories in our translation system. If neither a Penn Treebank or CCG constituent can be found for $e_1 \ldots e_n$, we associate a generic nonterminal symbol _X with this rule, allowing it to still take part in hierarchically motivated synchronous derivations.

2.2. Generalization

We generalize the annotated phrases described above to include nonterminals on the target side, which can be instantiated during decoding. We adhere to [1] to recursively generalize each existing rule

$$N \rightarrow f_1 \ldots f_m / e_1 \ldots e_n$$

for which there is an initial rule

$$M \rightarrow f_i \ldots f_u / e_j \ldots e_v$$

where $1 \leq i < u \leq m$ and $1 \leq j < v \leq n$, to obtain a new rule

$$N \rightarrow f_1 \ldots f_{i-1} M_k f_{u+1} \ldots f_m / e_1 \ldots e_{j-1} M_k e_{v+1} \ldots e_n$$

where k is a new index for the nonterminal M that expresses the one-to-one correspondence between the new occurrence of M on the source side and the corresponding one on the target side.

This approach generates rules comparable to the "Model2C" phrases in [6] assuming *consistent* initial phrase pairs from [2]. The target side of each rule is flattened before relative frequencies are calculated.

2.3. Rule Preparation

The set of rules resulting from the process above is more than 10 times the size of the initial lexical rules / phrase pairs used to create them. As an initial pruning step, we remove rules whose source-conditioned relative frequency is below a specified threshold, as well as those that include source terminal elements that do not occur in the development or test data. In practice we perform this filtering online during extraction. Also, generalized rules that occur only once are immediately discarded, but lexical rules (those that contain no nonterminal symbols) are never pruned away.

To allow derivations that do not comply with the syntactic structure learned from the rule extraction phase, we use "glue" rules as in [1]. The glue rules allow any nonterminal production to participate in sequential combination with other rules, in a left-bracketing manner. Unknown words are assigned production categories based on common occurrence. Specifically, we add the following rules to the learned grammar.

- _S $\rightarrow N, N$ for all nonterminals N in the grammar
- _G $\rightarrow N, N$ for all nonterminals N in the grammar
- _S $\rightarrow$ _S _G, _S _G
- {NNP, JJ, RB, NN} $\rightarrow$ _UNK, _UNK

Glue rule counts as used in the log-linear model (Section 3.2) are incremented for the _S $\rightarrow N, N$ and _S $\rightarrow$ _S _G, _S _G rules, thereby counting the number of chunks used in sequential productions.

2.4. Extracted Rules

After filtering for development and test set, we are left with 290,325 rules in the syntactic rule set, and 38,576 rules in the hierarchical rule extraction variant. The number of actual occurring nonterminals in our filtered syntactic grammar (before language model intersection) is 1618, due to the CCG extension on top of the 75 original Penn Treebank nonterminals.

Table 1 gives statistics and examples of the types of rules occurring in our syntactic rule set.

3. Decoding

3.1. Parsing Strategy

We apply the SynCFG to an input source sentence by using a bottom-up, CYK+ [14] style decoder which does not require

Rule type	N.o. rules	N.o. rule instances	Frequent example
Total	290,325	4,891,724	
Abstract (i.e., no terminals)	75,555	2,234,685	PUNCT\SQ → PRP MD VP, MD PRP VP
Lexical / initial (i.e., only terminals)	18,104	554,091	我 可以, May I
Mixed (both terminals and NTs)	196,666	2,102,948	MD+PRP → PRP 能, Could PRP
Regular syntactic (containing only regular NTs)	19,547	818,240	PRP → 我, I
Extended synt. (containing only regular or CCG NTs)	106,243	1,814,699	SQ → PUNCT\SQ 吗, ? PUNCT\SQ
_X rules (containing the _X nonterminal)	164,535	2,258,785	_X → 我 _X, . I _X
With reordering	28,840	467,028	SQ → PRP MD VP 吗, ? MD PRP VP
no substitution site (i.e., lexical)	18,104	554,091	(see lexical above)
1 substitution site pair	28,840	467,028	MD+PRP → PRP 能, can PRP
2 substitution site pairs	90,105	1,877,373	SQ → 你 MD VP 吗, ? MD you VP
3 substitution site pairs	147,499	1,606,745	_X → _X$_1$ CD _X$_2$, _X$_1$ CD _X$_2$

Table 1: *Rule statistics after filtering for dev+test set*

pre-binarization of the grammar. A target n-gram language model is intersected with the SynCFG rules during the decoding process. A probabilistic beam is applied to mitigate the combined effects of the large nonterminal set intersected with the n-gram language model. The Viterbi parsing process results in a "_S" (sentence) rooted most probable parse tree. Traversing its nodes in a left-to-right, depth-first manner yields the corresponding translation.

3.2. Log-linear model

We employ a log-linear model to select the highest probability (lowest neg-log prob) target derivation tree for the given source sentence. We model most aspects of the translation process by augmenting the SynCFG grammar with features that accumulate during decoding. The only feature explicitly introduced during decoding is the n-gram language model. Given a source sentence f to translate with our SynCFG, we model the decoding process as a search through the derivation space of f, where the lowest cost derivation encodes a target translation sequence in α. We define our translation model in log space (dealing with costs, not probabilities) as

$$\arg\min_{R_0 \circ \cdots \circ R_n} lm(tgt_{R_0 \circ \cdots \circ R_n}) + \sum_{i=1}^{m} \lambda_i \sum_{j=1}^{n} (v^j)_i \ . \qquad (1)$$

where $R_1 \circ \cdots \circ R_n$ is a derivation for f and $v^1, \ldots, v^n \in \mathbb{R}^m$ are the feature vectors of the applied rules $R_1, \ldots, R_n$. Further, $lm(tgt_{R_0 \circ \cdots \circ R_n})$ denotes the neg log probability of the target language sequence represented by the derivation $R_0 \circ \cdots \circ R_n$, and $\lambda_1, \ldots, \lambda_m$ are the parameters of the log-linear model.

Each rule in the grammar is augmented with the following features:

- Lexical weights as described in [2]
- Neg-log relative frequencies:
  - left-hand-side-conditioned
  - target-phrase-conditioned
  - source-phrase-conditioned
- Counters: n.o. rule applications, n.o. target words
- Flags:
  - IsPurelyLexical (i.e., contains only terminals)
  - IsPurelyAbstract (i.e., contains only nonterminals)
  - IsXRule (i.e., non-syntactical span),
  - IsGlueRule
- Penalties:
  - Rareness penalty: $\exp(1 - \mathit{RuleFrequency})$
  - Unbalancedness penalty: relative to ratio of source to target corpus length

Model weights for these features within the log-linear model are trained against the BLEU metric with Minimum Error Rate (MER) training as in [9], using the implementation provided from [15]. Given the large number of features represented in the model and the greedy search process used in [9] we perform feature selection during the MER training process. We generate unique target N-Best derivations for MER training using the process described in [13].

4. Data Preparation

We present our results using only the Full BTEC corpus as made available in the C-STAR evaluation track. The 163K parallel sentence pairs cover 906 thousand source words and 1.2 million target words. Both sides of the training data include punctuation, and the target side is provided with case information. The development and test corpora for the IWSLT-06 corpora contain no punctuation on the source side, and the *Official* evaluation track preserves both case and punctuation.

To account for these differences between training and test conditions, we apply a set of heuristic processing operations to learn translation models that directly generate true-cased, punctuated target output from non-punctuated source input.

While the development and test data are provided with source language word segmentation, the training data is not pre-segmented. We apply the Stanford Chinese segmenter on the source training data as well as the development and test data to maintain consistent segmentation.

We also note the presence of a large number of numeric or ordinal terms in the development data, which typically lead to unreliable language model estimates (due to the sparsity in encountered numeric forms). We handle source and target number forms as described below.

4.1. Case Information

Word alignment and subsequent phrase and rule extraction rely on well estimated lexical probabilities; a requirement that is often strained in limited data scenarios with true-cased data. Common solutions include training translation and language models on lower-cased text, generating system output in lower-case and then applying a true-case system like [16], or training all models in true-case.

We take a intermediate approach, based on the premise that the real source of sparsity in the direct true-case approach comes from the first word of the sentence being upper-cased due to its position. Upper-case words used within the sentence (rather than in the first position) tend to be consistently cased across the corpus (a notable exception being “May” vs “may”).

For each first word of the target side of the corpus we estimate its most common case in the corpus (ignoring first word occurrences). If the most common case is lower, we lower case the word, otherwise we leave it as upper case. Models are trained on this mixed corpus, and we expect that phrase pairs with upper-cased target word are well estimated. Our algorithm’s casing decisions for the 15 most frequent upper-case words in the training corpus are given in Table 2.

Our approach avoids maintaining mixed case versions of frequent words that occasionally occupy the first word position (words like “The” or “Is”). As a post processing step we simply upper-case the first words of each sentence, accounting for multiple-sentence utterances using the punctuation approach described in the next subsection.

For an upper-case word to be incorrectly output in lower-case during translation, that word instance would have to occur as first word of the training sentence from which the corresponding rule / phrase-pair was extracted, *and* the occurrence frequency of the upper-case word in the training corpus would have to be lower than the frequency of the lower-case variant, *and* the word would have to occur as a *non-first* word in the translation output sentence (because first-words are upper-cased during post-processing anyhow).

Similarly, for a lower-case word to be incorrectly output in upper-case, the word would need to come from a first-word in the training corpus, its upper-case instances would need to dominate the lower-case instances, and it could not be the first word of the output sentence (otherwise the upper-case spelling would be correct).

English word	Frequency	Fr. of lower-case variant
I	21611	10
Japan	1939	0
Japanese	1816	0
Tokyo	813	0
Hotel	703	1166
Mr.	691	0
New	487	315
York	438	0
English	372	0
Boston	318	0
Park	295	147
A	275	31786
Chicago	263	0
Street	233	506
Airlines	232	29

Table 2: *Most frequent upper-case words in the training corpus (disregarding sentences’ first-words). Words for which the lower-case variant is preferred are bold-faced.*

4.2. Punctuation Generation

Speech transcript translation typically encounters punctuation-free source sentences, while target output is preferred in punctuated form. Following the observation that punctuation generation is typically a function of target word context, we leave punctuation marks in the target side of the training corpus, while removing all punctuation marks from the source side of the data. The resulting models effectively generate punctuation across languages, as necessary for the development and test data conditions.

We also note the dependency between sentence end punctuation (period, question mark) with the first word of the sentence. To model this dependency during training, we shift sentence final punctuation marks to before the first word of the sentence. As a post-processing step we shift the sentence leading punctuation marks back to the end of each sentence, where we consider also mid-utterance end-punctuation marks as sentence boundaries.

4.3. Number Handling

We pre-process the source and target training data using number processing tools developed for the GALE-Rosetta consortium. Chinese numbers are identified by considering character context within the source sentence since numeric characters in Chinese are often combined with other characters to produce new (non-numeric) words. Words that contain at least one Chinese character are split into their component characters. The surrounding characters typically provide valuable information as to the meaning of the surface word form. The presence of quantifier characters typically defines a word as a numeric entity and warrants splitting the word into its numeric and quantifier component. We do not tag the Chinese character for *one*, as it is heavily used as determiner.

We paid special attention to timing expressions. We used the Chinese 点 (literally "point") character to split words into multiple fragments. If the fragments surrounding the point can be analyzed as numbers using the method above then we confirm the split. We find that several unknown surface word forms in the development data can be broken down into known numeric forms using this approach. We also utilize the *point* marker to influence translation decisions, converting numbers in 24-hr notation to 12 hour notation. We also converted numeric digits to words to match the development data references.

Chinese	Literal	Translation
十四 点	14 point	two
两点三十	two point 30	two thirty
二十四点	24	midnight
百分之十	-	10 %
十几	-	more than ten

Table 3: *Examples of split and tagged expressions.*

Source and target numerical expressions are tagged with a special left-hand-side production marker "CD", producing a number tagged bilingual corpus. Discarding all sentence pairs where there is a mismatch in the number of tags, we add the tagged corpus to the original untagged corpus, and build our translation and n-gram language models on this combined data-set. This combination mitigates the impact of tagging errors, while the n-gram language model should give preference towards tagged sequences. Our decoder is able to recognize tagged entities and ensures that the language model is applied over the tag when a derivation uses a tag-based rule.

4.4. Tokenization

In order to generate target translations compatible with the tokenization (use of apostrophes to conjoin words like "I'm") present in the development data, we tokenize (split) target surface forms that include apostrophes during training. As a post-processing step we conjoin words that have been split across the apostrophe. We sometimes encounter words in the translation output that have been generated without the subject in the apostrophe. In these cases, we simply add the most commonly occurring subject form back into the translation.

5. Empirical Evaluation

We evaluate our work on the IWSLT-06 test data track with the goal of judging improvements in translation quality due to our data processing approaches as well as syntax over hierarchy and the purely phrase based approach.

5.1. Decoder Settings

We perform our experiments using the decoder implementation described in [7] and use Pharaoh from [2] as the baseline phrase based implementation. Pharaoh parameters are trained following the instructions for baseline generation in the NAACL-MT06 Shared Task.

To speed up the decoding process, we limit the source span of non-glue based rule applications to maximally 8 words, and discard chart items that differ from the shortest derivation chart item in the same cell by more than 1 rule. We prune items within the same production category, propagating a maximum of 200 chart items falling within a probabilistic beam of 0.8. We do not prune across production nonterminal categories.

5.2. System Runtime

Translating the 489 sentence development set or the 500 sentence test set on a 3GHz processor (producing a 1000-best list of unique translations per sentence) takes approximately 6 minutes for the hierarchical version and 50 minutes for the syntax-augmented version of our system. Training initial lexical phrases takes approximately 30 minutes, and rule extraction takes another 30 minutes.

5.3. Processing Evaluation

Processing	Dev IBM-BLEU	Test IBM-BLEU
TrainLowerCase	22.04 (23.91)	19.16 (21.55)
TrainTrueCase	22.47 (24.05)	19.23 (20.48)
SmartCase	23.50 (25.17)	20.04 (21.76)

Table 4: *Comparison of different case-handling methods using the syntax-augmented translation system evaluated on the official case- and punctuation-sensitive IBM-BLEU metric. The numbers in parentheses indicate the IBM-BLEU score when case (but not punctuation) is ignored.*

We compared our case handling method "SmartCase" against the commonly used methods of "TrainLowerCase"; training and translating in lower-case and employing a true-caser in a post-processing step (using interACT's internal true-caser)—and "TrainTrueCase"; keeping all training data in its original case and upper-casing translation output sentences' first-words if they are not upper-cased already. Table 4 gives the experimental results on the 'devset4' development set, on which MER training was performed, as well as the IWSLT-06 test set.

While "TrainTrueCase" slightly outperforms "TrainLowerCase", our "SmartCase" approach significantly outperforms both of these traditional approaches. Interestingly, even when ignoring case during the evaluation, "SmartCase" outperforms “TrainLowerCase”, suggesting that segregating upper and lower-case words and the resulting more refined translation model overcomes the impact of mis-casing beginning-of-sentence words in the training data.

5.4. Baseline Comparison and Impact of Syntax

Rules	Dev IBM-BLEU	Test IBM-BLEU
Pharaoh	23.2	19.3
Chiang-sim	21.25	18.08
SAMT	23.50	20.04

Table 5: *Comparison of translation-models system using “SmartCase”, evaluated on the official case and punctuation sensitive IBM-BLEU metric.*

In table 5, we compare “Pharaoh”, a freely available state-of-the art phrase-based system trained on the same data with the same pre- and post-processing applied as for our system; “Chiang-sim”, our decoder using a [1] style purely-hierarchical rule set; and finally “SAMT”, our decoder using the fully-syntactic rule set. While our system’s hierarchical variant performs poorly w.r.t. the external phrase-based baseline, lacking nearly two BLEU points on the development and 0.7 BLEU points on the actual test data, our syntax-augmented system outperforms the baseline by 0.3 BLEU points on the development and 0.7 BLEU points on the test set. Also, one can clearly see the impact of using syntax vs. hierarchical rules within our decoding framework.

6. Discussion

Our work demonstrates the potential for syntax augmented models in the speech transcription translation task. While our decoder still makes search errors as evidenced by the results of the hierarchical model, the introduction of syntax modeling does improve system performance. The IWSLT-06 task provides a rapid development environment for syntax based systems, allowing us to take advantage of decoder constraints that can cover whole source sentences while still generating translations efficiently.

We showed how to take advantage of simple, but effective case handling, in order to build low perplexity translation models, yet still generating accurate case markings in translation output whose scores improve in case and punctuation sensitive evaluation.

Our translation system is available open-source under the GNU General Public License at: `www.cs.cmu.edu/~zollmann/samt`

7. Acknowledgments

This work was partly funded by the *European Union* (EU) under the integrated project TC-Star (Grant number IST-506738), and the interACT exchange fellowship.

8. References

[1] D. Chiang, “A hierarchical phrase-based model for statistical machine translation,” in *Proc. of the Association for Computational Linguistics*, 2005.

[2] P. Koehn, F. J. Och, and D. Marcu, “Statistical phrase-based translation,” in *Proc. of HLT/NAACL*, Edomonton, Canada, May 27-June 1 2003.

[3] F. Och and H. Ney, “The alignment template approach to statistical machine translation,” *Computational Linguistics*, 2004.

[4] M. Galley, M. Hopkins, K. Knight, and D. Marcu, “Scalable inferences and training of context-rich syntax translation models,” in *Proceedings of NAACL-HLT*, 2006.

[5] I. D. Melamed, “Statistical machine translation by parsing.” in *ACL*, 2004, pp. 653–660.

[6] D. Marcu, W. Wang, A. Echihabi, and K. Knight, “SPMT: Statistical Machine Translation with Syntactified Target Language Phrases,” in *Proceedings of EMNLP*, Sydney, 2006.

[7] A. Zollmann and A. Venugopal, “Syntax augmented machine translation via chart parsing,” in *NAACL 2006 - Workshop on statistical machine translation*, New York, 2006.

[8] C. M. D. Klein, “Accurate unlexicalized parsing,” in *Proceedings of ACL*, 2003.

[9] F. J. Och, “Minimum error rate training in statistical machine translation,” in *Proc. of ACL*, Sapporo, Japan, July 6-7 2003.

[10] K. Papineni, S. Roukos, T. Ward, and W.-J. Zhu, “Bleu: a method for automatic evaluation of machine translation.” in *Proc. of ACL*, 2002, pp. 311–318.

[11] M. Steedman, “Alternative quantifier scope in ccg,” in *Proceedings of ACL*, 1999.

[12] P. Koehn, F. J. Och, and D. Marcu, “Pharaoh: A beam search decoder for phrase-base statistical machine translation models,” in *Proceedings of the Sixth Confernence of the Association for Machine Translation in the Americas*, Edomonton, Canada, May 27-June 1 2004.

[13] A. Zollmann and A. Venugopal, “Syntax augmented machine translation via chart parsing with integrated language modelling,” in *Technical report, www.cs.cmu.edu/ zollmann/publications.html*, 2006.

[14] J.-C. Chappelier and M. Rajman, “A generalized CYK algorithm for parsing stochastic CFG,” in *Proceedings of Tabulation in Parsing and Deduction (TAPD’98)*,

Paris, 1998, pp. 133–137. [Online]. Available: citeseer.ist.psu.edu/chappelier98generalized.html

[15] A. Venugopal and S. Vogel, "Considerations in MCE and MMI training for statistical machine translation," in *Proceedings of the Tenth Conference of the European Association for Machine Translation (EAMT-05)*. Budapest, Hungary: The European Association for Machine Translation, May 2005.

[16] D. M. W. Wang, K. Knight, "Capitalizing machine translation," in *Proceedings of the North American Association for Computational Linguistics (HLT/NAACL)*, 2006.