Adaptive Transformation-based Learning for Improving Dictionary Tagging

Burcu Karagol-Ayan, David Doermann, and Amy Weinberg
Institute for Advanced Computer Studies (UMIACS)
University of Maryland
College Park, MD 20742
`{burcu,doermann,weinberg}@umiacs.umd.edu`

Abstract

We present an adaptive technique that enables users to produce a high quality dictionary parsed into its lexicographic components (headwords, pronunciations, parts of speech, translations, etc.) using an extremely small amount of user provided training data. We use transformation-based learning (TBL) as a postprocessor at two points in our system to improve performance. The results using two dictionaries show that the tagging accuracy increases from 83% and 91% to 93% and 94% for individual words or "tokens", and from 64% and 83% to 90% and 93% for contiguous "phrases" such as definitions or examples of usage.

1 Introduction

The availability and use of electronic resources such as electronic dictionaries has increased tremendously in recent years and their use in Natural Language Processing (NLP) systems is widespread. For languages with limited electronic resources, i.e. low-density languages, however, we cannot use automated techniques based on parallel corpora (Gale and Church, 1991; Melamed, 2000; Resnik, 1999; Utsuro et al., 2002), comparable corpora (Fung and Yee, 1998), or multilingual thesauri (Vossen, 1998). Yet for these low-density languages, printed bilingual dictionaries often offer effective mapping from the low-density language to a high-density language, such as English.

Dictionaries can have different formats and can provide a variety of information. However, they typically have a consistent layout of entries and a

1 Headword	5 Translation
2 POS	6 Example of usage
3 Sense number	7 Example of usage translation
4 Synonym	8 Subcategorization

Figure 1: Sample tagged dictionary entries. Eight tags are identified and tagged in the given entries.

consistent structure within entries. Publishers of dictionaries often use a combination of features to impose this structure including (1) changes in font style, font-size, etc. that make implicit the lexicographic information[1], such as headwords, pronunciations, parts of speech (POS), and translations, (2) keywords that provide an explicit interpretation of the lexicographic information, and (3) various separators that impose an overall structure on the entry. For example, a boldface font may indicate a headword, italics may indicate an example of usage, keywords may designate the POS, commas may separate different translations, and a numbering system may identify different senses of a word.

We developed an entry tagging system that recognizes, parses, and tags the entries of a printed dictionary to reproduce the representation electronically (Karagol-Ayan et al., 2003). The system aims to use features as described above and the consistent layout and structure of the dictio-

---

[1]For the purposes of this paper, we will refer to the lexicographic information as *tag* when necessary.

naries to capture and recover the lexicographic information in the entries. Each token[2] or group of tokens (*phrase*)[3] in an entry associates with a tag indicating its lexicographic information in the entry. Figure 1 shows sample tagged entries in which eight different types of lexicographic information are identified and marked. The system gets format and style information from a document image analyzer module (Ma and Doermann, 2003) and is retargeted at many levels with minimal human assistance.

A major requirement for a human aided dictionary tagging application is the need to minimize human generated training data.[4] This requirement limits the effectiveness of data driven methods for initial training. We chose rule-based tagging that uses the structure to analyze and tag tokens as our baseline, because it outperformed the baseline results of an HMM tagger. The approach has demonstrated promising results, but we will show its shortcomings can be improved by applying a transformation-based learning (TBL) post processing technique.

TBL (Brill, 1995) is a rule-based machine learning method with some attractive qualities that make it suitable for language related tasks. First, the resulting rules are easily reviewed and understood. Second, it is error-driven, thus directly minimizes the error rate (Florian and Ngai, 2001). Furthermore, TBL can be applied to other annotation systems' output to improve performance. Finally, it makes use of the features of the token and those in the neighborhood surrounding it.

In this paper, we describe an adaptive TBL based technique to improve the performance of the rule-based entry tagger, especially targeting certain shortcomings. We first investigate how using TBL to improve the accurate rendering of tokens' font style affects the rule-based tagging accuracy. We then apply TBL on tags of the tokens. In our experiments with two dictionaries, the range of font style accuracies is increased from 84%-94% to 97%-98%, and the range of tagging accuracies is increased from 83%-90% to 93%-94% for tokens, and from 64%-83% to 90%-93% for phrases.

Section 2 discusses the rule-based entry tagging

method. In Section 3, we briefly describe TBL, and Section 4 recounts how we apply TBL to improve the performance of the rule-based method. Section 5 explains the experiments and results, and we conclude with future work.

2 A Rule-based Dictionary Entry Tagger

The rule-based entry tagger (Karagol-Ayan et al., 2003) utilizes the repeating structure of the dictionaries to identify and tag the linguistic role of tokens or sets of tokens. Rule-based tagging uses three different types of clues—font style, keywords and separators—to tag the entries in a systematic way. The method accommodates noise introduced by the document analyzer by allowing for a relaxed matching of OCRed output to tags. For each dictionary, a human operator must specify the lexicographic information used in that particular dictionary, along with the clues for each tag. This process can be performed in a few hours. The rule-based method alone achieved token accuracy between 73%-87% and phrase accuracy between 75%-89% in experiments conducted using three different dictionaries[5].

The rule-based method has demonstrated promising results, but has two shortcomings. First, the method does not consider the relations between different tags in the entries. While not a problem for some dictionaries, for others ordering the relations between tags may be the only information that will tag a token correctly. Consider the dictionary entries in Figure 1. In this dictionary, the word "a" represents POS when in italic font, and part of a translation if in normal font. However if the font is incorrect (font errors are more likely to happen with short tokens), the only way to mark correctly the tag involves checking the neighboring tokens and tags to determine its relative position within the entry. When the token has an incorrect font or OCR errors exist, and the other clues are ambiguous or inconclusive, the rule-based method may yield incorrect results.

Second, the rule-based method can produce incorrect splitting and/or merging of phrases. An erroneous merge of two tokens as a phrase may take place either because of a font error in one of the tokens or the lack of a separator, such as a punctuation mark. A phrase may split erroneously either

---

[2]*Token* is a set of glyphs (i.e., a visual representation of a set of characters) in the OCRed output. Each punctuation is counted as a token as well.

[3]In Figure 1, *not on time* is a phrase consisting of 3 tokens.

[4]For our experiments we required hand tagging of no more than eight pages that took around three hours of human effort.

[5]Using HMMs for entry tagging on the same set of dictionaries produced slightly lower performance, resulting in token accuracy between 73%-88% and phrase accuracy between 57%-85%.

as a result of a font error or an ambiguous separator. For instance, a comma may be used after an example of usage to separate it from its translation or within it as a normal punctuation mark.

3 TBL

TBL (Brill, 1995), a rule-based machine learning algorithm, has been applied to various NLP tasks.

TBL starts with an initial state, and it requires a correctly annotated training corpus, or *truth*, for the learning (or training) process. The iterative learning process acquires an ordered list of rules or transformations that correct the errors in this initial state. At each iteration, the transformation which achieved the largest benefit during application is selected. During the learning process, the templates of allowable transformations limit the search space for possible transformation rules. The proposed transformations are formed by instantiation of the transformation templates in the context of erroneous tags. The learning algorithm stops when no improvement can be made to the current state of the training data or when a pre-specified threshold is reached.

A transformation modifies a tag when its context (such as neighboring tags or tokens) matches the context described by the transformation. Two parts comprise a transformation: a rewrite rule— what to replace— and a triggering environment— when to replace. A typical rewrite rule is: *Change the annotation from a_a to a_b*, and a typical triggering environment is: *The preceding word is w_a*. The system's output is the final state of this data after applying all transformations in the order they are produced.

To overcome the lengthy training time associated with this approach, we used *fnTBL*, a fast version of TBL that preserves the performance of the algorithm (Ngai and Florian, 2001). Our research contribution shows this method is effective when applied to a miniscule set of training data.

4 Application of TBL to Entry Tagging

In this section, we describe how we used TBL in the context of tagging dictionary entries.

We apply TBL at two points: to render correctly the font style of the tokens and to label correctly the tags of the tokens[6]. Although our ultimate goal

---

[6]In reality, TBL improves the accuracy of tags and phrase boundary flags. In this paper, whenever we say "application of TBL to tagging", we mean tags and phrase boundary flags
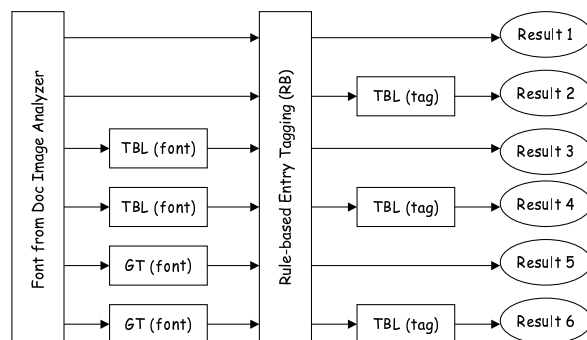

Figure 2: Phases of TBL application

is improving tagging results, font style plays a crucial role in identifying tags. The rule-based entry tagger relies on font style, which can be also incorrect. Therefore we also investigate whether improving font style accuracy will further improve tagging results. We apply TBL in three configurations: (1) to improve font style, (2) to improve tagging and (3) to improve both, one after another. Figure 2 shows the phases of TBL application. First we have the rule-based entry tagging results with the font style assigned by document image analysis (Result1), then we apply TBL to tagging using this result (Result2). We also apply TBL to improve the font style accuracy, and we feed these changed font styles to the rule-based method (Result3). We then apply TBL to tagging using this result (Result4). Finally, in order to find the upper bound when we use the manually corrected font styles in the ground truth data, we feed correct font styles to the rule-based method (Result5), and then apply TBL to tagging using this result (Result6).

In the transformation templates, we use the tokens themselves as features, i.e. the items in the triggering environment, because the token's content is useful in indicating the role. For instance a comma and a period may have different functionalities when tagging the dictionary. However, when transformations are allowed to make reference to tokens, i.e., when *lexicalized* transformations are allowed, some relevant information may be lost because of sparsity. To overcome the data sparseness problem, we also assign a *type* to each token that classifies the token's content. We use eight types: punctuation, symbol, numeric, uppercase, capitalized, lowercase, non-Latin, and other. For TBL on font style, the transformation templates contain three features: the token, the token's type, and the token's font. For TBL on tagging, we

---

together.

use four features: the token, the token's type, the token's font style, and the token's tag.

The initial state annotations for font style are assigned by document image analysis. The rule-based entry tagging method assigns the initial state of the tokens' tags. The templates for font style accuracy improvement consist of those from studying the data and all templates using all features within a window of five tokens (i.e., two preceding tokens, the current token, and two following tokens). For tagging accuracy improvement, we prepared the transformation templates by studying dictionaries and errors in the entry tagging results. The objective function for evaluating transformations in both cases is the classification accuracy, and the objective is to minimize the number of errors.

5 Experiments

We performed our experiments on a Cebuano-English dictionary (Wolff, 1972) consisting of 1163 pages, 4 font styles, and 18 tags, and on an Iraqi Arabic-English dictionary (Woodhead and Beene, 2003) consisting of 507 pages, 3 font styles, and 26 tags. For our experiments, we used a publicly available implementation of TBL's fast version, fnTBL[7], described in Section 3.

We used eight randomly selected pages from the dictionaries to train TBL, and six additional randomly selected pages for testing. The font style and tag of each token on these pages are manually corrected from an initial run. Our goal is to measure the effect of TBL on font style and tagging that have the same noisy input. For the Cebuano dictionary, the training data contains 156 entries, 8370 tokens, and 6691 non-punctuation tokens, and the test data contains 137 entries, 6251 tokens, and 4940 non-punctuation tokens. For the Iraqi Arabic dictionary, the training data contains 232 entries, 6130 tokens, and 4621 non-punctuation tokens, and the test data contains 175 entries, 4708 tokens, 3467 non-punctuation tokens.

For evaluation, we used the percentage of accuracy for non-punctuation tokens, i.e., the number of correctly identified tags divided by total number of tokens/phrases. The learning phase of TBL took less than one minute for each run, and application of learned transformations to the whole dictionary less than two minutes.

We report how TBL affects accuracy of tagging

when applied to font styles, tags, and font styles and tags together. To find the upper bound tagging results with correct font styles, we also ran rule-based entry tagger using manually corrected font styles, and applied TBL for tagging accuracy improvement to these results. We should note that feeding the correct font to the rule-based entry tagger does not necessarily mean the data is totally correct, it may still contain noise from document image analysis or ambiguity in the entry.

We conducted three sets of experiments to observe the effects of TBL (Section 5.1), the effects of different training data (Section 5.2), and the effects of training data size (Section 5.3).

5.1 TBL on Font Styles and Tags

	Cebuano	Iraqi Arabic
Original	84.43	94.15
TBL(font)	97.07	98.13

Table 1: Font style accuracy results for non-punctuation tokens

We report the accuracy of font styles on the test data before and after applying TBL to the font style of the non-punctuation tokens in Table 1. The initial font style accuracy of Cebuano dictionary was much less than the Iraqi Arabic dictionary, but applying TBL resulted in similar font style accuracy for both dictionaries (97% and 98%).

	Cebuano		Iraqi Arabic	
	Token	Phrase	Token	Phrase
RB	83.25	64.08	90.89	82.72
RB+TBL(tag)	91.44	87.37	94.05	92.33
TBL(font)+RB	87.99	72.44	91.46	83.48
TBL(font)+RB+TBL(tag)	93.06	90.19	94.30	92.58
GT(font)+RB	90.76	74.71	91.74	83.90
GT(font)+RB+TBL(tag)	95.74	92.29	94.54	93.11

Table 2: Tagging accuracy results for non-punctuation tokens and phrases for two dictionaries

The results of tagging accuracy experiments are presented in Table 2. In the tables, RB is rule-based method, TBL(tag) is the TBL run on tags, TBL(font) is the TBL run on font style, and GT(font) is the ground truth font style. In each case, we begin with font style information provided by document image analysis. We tabulate percentages of tagging accuracy of individual non-punctuation tokens and phrases[8]. The results for

---

[7]http://nlp.cs.jhu.edu/rflorian/fntbl

[8]In phrase accuracy, if a group of consequent tokens is assigned one tag as a phrase in the ground truth, the tagging of the phrase is considered correct only if the same group of

token and phrase accuracy are presented for three different sets: The entry tagger using the font style (1) provided by document image analysis, (2) after TBL is applied to font style, and (3) corrected manually, i.e. the ground truth. All results reported, except the token accuracies for two cases for the Iraqi Arabic dictionary, namely using TBL(font) vs. GT(font) and using TBL(font) and TBL(tag) together vs. using GT(font) and TBL(tag), are statistically significant within the 95% confidence interval with two-tailed paired t-tests[9].

Using TBL(font) instead of initial font styles improved initial accuracy as much as 4.74% for tokens, and 8.36% for phrases in the Cebuano dictionary which has a much lower initial font style accuracy than the Iraqi Arabic dictionary. Using the GT(font) further increased the tagging accuracy by 2.77% for tokens and 2.27% for phrases for the Cebuano dictionary. As for the Iraqi Arabic dictionary, using TBL(font) and GT(font) resulted in an improvement of 0.57% and 0.85% for tokens and 0.74% and 1.18% for phrases respectively. The improvements in these two dictionaries differ because the initial font style accuracy for the Iraqi Arabic dictionary is very high while for the Cebuano dictionary potentially very useful font style information (namely, the font style for POS tokens) is often incorrect in the initial run.

Using TBL(tag) alone improved rule-based method results by 8.19% and 3.16% for tokens and by 23.25% and 9.61% for phrases in Cebuano and Iraqi Arabic dictionaries respectively. The last two rows in Table 2 show the upper bound. For the two dictionaries, our results using TBL(font) and TBL(tag) together is 2.68% and 0.24% for token accuracy and 2.10% and 0.53% for phrase accuracy less than the upper bound of using the GT(font) and TBL(tag) together.

Applying TBL to font styles resulted in a higher accuracy than applying TBL to tagging. Since the number of tag types (18 and 26) is much larger than that of font style types (4 and 3), TBL application on tags requires more training data than the font style to perform as well as TBL application on font style.

In summary, applying TBL using the same templates to two different dictionaries using very limited training data resulted in performance increase,

and the greatest increases we observed are in phrase accuracy. Applying TBL to font style first increased the accuracy even further.

5.2 Effect of Training Data

We conducted experiments to measure the robustness of our method with different training data. For this purpose, we trained TBL on eight pages randomly selected from the 14 pages for which we have ground truth, for each dictionary. We used the remaining six pages for testing. We did this ten times, and calculated the average accuracy and the standard deviation. Table 3 presents the average accuracy and standard deviation. The accuracy results are consistent with the results we presented in Table 2, and the standard deviation is between 0.56-2.28. These results suggest that using different training data does not affect the performance dramatically.

5.3 Effect of Training Data Size

The problem to which we apply TBL has one important challenge and differs from other tasks in which TBL has been applied. Each dictionary has a different structure and different noise patterns, hence, TBL must be trained for each dictionary. This requires preparing ground truth manually for each dictionary before applying TBL. Moreover, although each dictionary has hundreds of pages, it is not feasible to use a significant portion of the dictionary for training. Therefore the training data should be small enough for someone to annotate ground truth in a short amount of time. One of our goals is to calculate the quantity of training data necessary for a reasonable improvement in tagging accuracy. For this purpose, we investigated the effect of the training data size by increasing the training data size for TBL one entry at a time. The entries are added in the order of the number of errors they contain, starting with the entry with maximum errors. We then tested the system trained with these entries on two test pages[10].

Figure 3 shows the number of font style and tagging errors for non-punctuation tokens on two test pages as a function of the number of entries in the training data. The tagging results are presented when using font style from document image analysis and font style after TBL. In these graphs, the

---

tokens was assigned the same tag as a phrase in the result.

[9]We did the t-tests on the results of individual entries.

[10]We used two test data pages because if such a method will determine the minimum training data required to obtain a reasonable performance, the test data should be extremely limited to reduce human provided data.

	Cebuano		Iraqi Arabic	
	Token	**Phrase**	**Token**	**Phrase**
RB	81.46±1.14	62.38±1.09	92.10±0.69	85.05±1.64
RB + TBL(tag)	89.34±0.96	85.17±1.55	94.94±0.56	93.25±0.87
TBL(font) + RB	87.40±1.69	71.97±1.26	93.20±1.02	85.49±1.13
TBL(font) + RB + TBL(tag)	93.13±1.58	90.48±0.80	94.88±0.56	93.03±0.70
GT(font) + RB	89.25±1.57	73.13±1.02	93.02±0.58	85.03±2.28
GT(font) + RB + TBL(tag)	95.31±1.43	91.89±1.80	95.32±0.65	93.36±0.81

Table 3: Average tagging accuracy results with standard deviation for ten runs using different eight pages for training, and six pages for testing
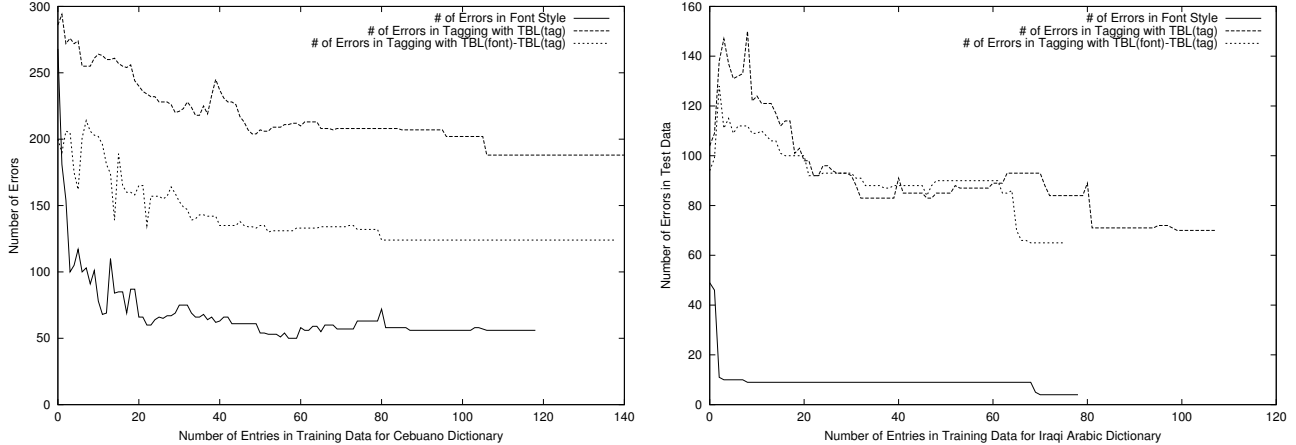

Figure 3: The number of errors in two test pages as a function of the number of entries in the training data for two dictionaries

number of errors declines dramatically with the addition of the first entries. For the tags, the decline is not as steep as the decline in font style. The main reason involves the number of tags (18 and 26), which are more than the number of font styles (4 and 3). The method of adding entries to training data one by one, and finding the point when the number of errors on selected entries stabilizes, can determine minimum training data size to get a reasonable performance increase. lexicalized

5.4 Example Results

Table 4 presents some learned transformations for Cebuano dictionary. Table 5 shows how these transformations change the font style and tags of tokens from Figure 4. The first column gives the tagging results before applying TBL. The consecutive columns shows how different TBL runs changes these results. The tags with * indicate incorrect tags, the tags with + indicate corrected tags, and the tags with - indicate introduced errors. The font style of tokens is also represented. The **No** column in Tables 4 and 5 gives the applied transformation number.

For these entries, using TBL on font styles and tagging together gives correct results in all cases.

Using TBL only on tagging gives the correct tagging only for the last entry.

TBL introduces new errors in some cases. One error we observed occurs when an example of usage translation is assigned a tag before any example of usage tag in an entry. This case is illustrated when applying transformation 9 to the token *Abaa* because of a misrecognized comma before the token.

6 Conclusion

In this paper, we introduced a new dictionary entry tagging system in which TBL improves tagging accuracy. TBL is applied at two points, – on font style and tagging– and yields high performance even with limited user provided training data. For two different dictionaries, we achieved an increase from 84% and 94% to 97% and 98% in font style accuracy, from 83% and 91% to 93% and 94% in tagging accuracy of tokens, and from 64% and 83% to 90% and 93% in tagging accuracy of phrases. If the initial font style is not accurate, first improving font style with TBL further assisted the tagging accuracy as much as 2.62% for tokens and 2.82% for phrases compared to using TBL only for tagging. This result cannot be

No	Triggering Environment	Change To
10	$type_{n-2}$ = lowercase and $type_{n-1}$ = punctuation and $type_n$ = capitalized and $font_{n+1}$ = normal and $font_{n+2}$ = normal	normal
15	$font_{n-1}$ = italic and $type_n$ = lowercase and $type_{n+1}$ = lowercase and $font_{n+2}$ = italic	italic
18	$token_n$ = the first token in the entry	bold
1	$token_n$ = a and tag_{n-1} = translation and tag_{n+1} = translation	translation
4	$tag_{[n-7,n-1]}$ = example and $token_{n-1}$ = , and $font_n$ = bold	example translation
2	$type_n$ = lowercase and $font_n$ = normal and tag_{n-1} = translation and $font_{n-1}$ = normal	translation
9	$token_{n-1}$ = , and $font_n$ = italic and $type_n$ = capitalized	example translation
8	tag_{n-2} = example translation and tag_{n-1} = separator and tag_n = example translation and $type_n$ = capitalized	continuation of a phrase
11	tag_{n-2} = example and tag_{n-1} = separator and tag_n = example and $type_n$ = capitalized	continuation of a phrase

Table 4: Some sample transformations used for Cebuano dictionary entries in Figure 4. Here, *continuation of a phrase* indicates this token merges with the previous one to form a phrase.

attributed to a low rule-based baseline as a similar, even a slightly lower baseline is obtained from an HMM trained system. Results came from a method used to compensate for extremely limited training data. The similarity of performance across two different dictionaries shows the method as adaptive and able to be applied genericly.

In the future, we plan to investigate the sources of errors introduced by TBL and whether these can be avoided by post-processing TBL results using heuristics. We will also examine the effects of using TBL to increase the training data size in a bootstrapped manner. We will apply TBL to a few pages, then correct these and use them as new training data in another run. Since TBL improves accuracy, manually preparing training data will take less time.

Acknowledgements

References

Eric Brill. 1995. Transformation-based error-driven learning and natural language processing: A case study in part of speech tagging. *Computational Linguistics*, 4(21):543–565.

Radu Florian and Grace Ngai. 2001. Multidimensional transformational-based learning. *Proceedings of the 5th Conference on Computational Natural Language Learning, CoNLL 2001*, pages 1–8, July.

Pascale Fung and Lo Yuen Yee. 1998. An IR approach for translating new words from nonparallel, comparable texts. In Christian Boitet and Pete Whitelock, editors, *Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics*, pages 414–420, San Francisco, California. Morgan Kaufmann Publishers.

William A. Gale and Kenneth W. Church. 1991. A program for aligning sentences in bilingual corpora. In *Proceedings of the 29th Annual Meeting of the ACL*, pages 177–184, Berkeley, California, June.

Burcu Karagol-Ayan, David Doermann, and Bonnie Dorr. 2003. Acquisition of bilingual MT lexicons from OCRed dictionaries. In *Proceedings of the 9th MT Summit*, pages 208–215, New Orleans, LA, September.

Huanfeng Ma and David Doermann. 2003. Bootstrapping structured page segmentation. In *Proceedings of SPIE Conference Document Recognition and Retrieval*, Santa Clara, CA, January.

I. Dan Melamed. 2000. Models of translational equivalence among words. *Computational Linguistics*, 26(2):221–249, June.

Grace Ngai and Radu Florian. 2001. Transformation-based learning in the fast lane. In *Proceedings of the 2nd Conference of the North American Chapter of the Association for Computational Linguistics (NAACL)*, pages 40–47, Pittsburgh, PA, June.

Philip Resnik. 1999. Mining the web for bilingual text. In *Proceedings of the 37th Annual Meeting of the Association for Computational Linguistics (ACL'99)*, University of Maryland, College Park, Maryland, June.

Takehito Utsuro, Takashi Horiuchi, Yasunobu Chiba, and Takeshi Hamamoto. 2002. Semi-automatic compilation of bilingual lexicon entries from cross-lingually relevant news articles on WWW news sites. In *Fifth Conference of the Association for Machine Translation in the Americas, AMTA-2002*, pages 165–176, Tiburon, California.

Piek Vossen. 1998. *EuroWordNet: A Multilingual Database with Lexical Semantic Networks*. Kluwer Academic Publishers, Dordrecht.

John U. Wolff. 1972. *A Dictionary of Cebuano Visaya*. Southeast Asia Program, Cornell University. Ithaca, New York.

D. R. Woodhead and Wayne Beene, editors. 2003. *A Dictionary of Iraqi Arabic: Arabic–English Dictionary*. Georgetown University Press, Washington D.C.

abaa particle indicating disapproval (literary). *Abaa! Mahúg ka gánì dihà!* Stop that! You might fall!

aguntù *v* [A] emit a short grunt when hit in the pit of the stomach or when exerting an effort. *Miaguntù siya dihang naigù sa kutukutu,* He groaned when he was hit in the pit of the stomach. *n* short grunt.

aspaltu *n* asphalt. *v* [A; a12] apply asphalt. *Aspaltúhun ang baskitbulan,* The basketball court will be asphalted. **aspaltádu** *a* paved with asphalt.

Figure 4: Cebuano-English dictionary entry samples

RB			RB + TBL (tag)			TBL (font) + RB			TBL (font) + RB + TBL (tag)	
Tag	Token(s)	No	Tag	Token(s)	No	Tag	Token(s)	No	Tag	Token(s)
*hw	abaa particle indicating disapproval		*hw	abaa particle indicating disapproval	18	+hw +tr	**abaa** particle indicating disapproval		hw tr	**abaa** particle indicating disapproval
*ex *ex	*Abaa!* *Mahúg ka gani dihà!*	9	-ex-tr *ex	*Abaa!* *Mahúg ka gani dihà!*		*ex *ex	*Abaa!* *Mahúg ka gani dihà!*	11	+ex	*Abaa!* *Mahúg ka gani dihà!*
*ex-tr *ex-tr	Stop that! You might fall!	8	+ex-tr	Stop that! You might fall!		*ex-tr *ex-tr	Stop that! You might fall!	8	+ex-tr	Stop that! You might fall!
*tr *pos *tr	emit a short grunt when hit *in* the pit of the stomach or when exerting an effort	1	+tr	emit a short grunt when hit *in* the pit of the stomach or when exerting an effort		*tr *pos *tr	emit a short grunt when hit in the pit of the stomach or when exerting an effort	2	+tr	emit a short grunt when hit in the pit of the stomach or when exerting an effort
*ex *al-sp *ex	*Miaguntú* **siya** *dihang naig sa kutukutu,*		*ex *al-sp *ex	*Miaguntú* **siya** *dihang naig sa kutukutu,*	15	+ex	*Miaguntú* *siya* *dihang naig* *sa kutukutu,*		ex	*Miaguntú* *siya* *dihang naig* *sa kutukutu,*
*al-sp *ex-tr	**The** basketball court will be asphalted.	4	+ex-tr	**The** basketball court will be asphalted.	10	 +ex-tr	The basketball court will be asphalted.		 ex-tr	The basketball court will be asphalted.
hw: headword; tr: translation; al-sp: alternative spelling of headword; pos: POS; ex: example of usage; ex-tr: example of usage translation										

Table 5: Illustration of TBL application to the incorrect tags in the sample entries shown in Figure 4.
* indicates incorrect tags, + indicates corrected tags, and - indicates introduced errors.