A Method for Unsupervised Broad-Coverage Lexical Error Detection and Correction

Nai-Lung Tsao
Graduate Institute of Learning and Instruction
National Central University
Jhongli City, Taoyuan County 32001, Taiwan
beaktsao@gmail.com

David Wible
Graduate Institute of Learning and Instruction
National Central University
Jhongli City, Taoyuan County 32001, Taiwan
Wible45@yahoo.com

Abstract

We describe and motivate an unsupervised lexical error detection and correction algorithm and its application in a tool called Lexbar appearing as a query box on the Web browser toolbar or as a search engine interface. Lexbar accepts as user input candidate strings of English to be checked for acceptability and, where errors are detected, offers corrections. We introduce the notion of hybrid n-gram and extract these from BNC as the knowledgebase against which to compare user input. An extended notion of edit distance is used to identify most likely candidates for correcting detected errors. Results are illustrated with four types of errors.

1 Introduction

We describe and motivate an unsupervised lexical error detection and correction algorithm and its application in a tool called Lexbar appearing as a query box in a web-based corpus search engine or on the Web browser toolbar. The tool is intended as a proxy for search engines in the common practice where users put search engines to use as error checkers. A problem with this use of search engines like Google is that such searches commonly provide false positives, hits for strings that contain errors. Lexbar accepts as user input candidate strings of English to be checked for acceptability and, where errors are detected, offers corrections.

2 Related Work

Among the many works on error detection, recently unsupervised error detection approaches have been proposed, such as [Chodorow and Leacock, 2000] and [Quixal and Badia 2008]. These use contextual features and statistical word association measurement to decide if the detected bigram or trigram is an error or not. To our knowledge, such unsupervised methods have not been applied in error correction. [Gamon et al 2008] and [Felice and Pulman 2008] propose unsupervised approaches to build a probabilistic model for detecting errors (prepositions and articles) and providing correct answers. They also typically focus on a particular type of error, usually limited to a specific word class such as preposition errors, often in a pre-determined paradigmatic slot. Our approach reported here is unsupervised in both detection and correction and is not tailored to a specific target error subtype or targeted to a specific position in a string. More generally the family of error types suitable for this approach are lexical or lexico-grammatical errors since detection and correction are based on patterns of word use detected statistically. At the core of our approach is a bank of what we call "hybrid n-grams" extracted from BNC to serve as the target knowledge against which learner input is compared for detection and correction. We illustrate the single algorithm with results on four different categories of errors.

3 Overview of the Algorithm

The Lexbar application consists of two main components: (1) the target language knowledgebase of hybrid n-grams that serves as the standard against which learner production is examined for errors, and (2) the error detection and correction algorithm that uses this knowledgebase to evalu-

ate learner production through matching and edit distance. Relatively broad coverage is achieved from one algorithm since no specific error type is targeted but violations of word behaviors patterns.

Typically, n-grams are contiguous sequences of lemmas or specific word forms. Using traditional n-grams and string matching against them as a means of error detection leads to weak precision since the absence of a specific n-gram in a standard corpus does not render it an error. To address this limitation, we extend the notion of n-gram to include in the string not only lemmas or word forms but parts-of-speech as well. For example, the chunk *point of view* can be part of a longer string *from my point of view*. Here, the preposition *from* is non-substitutable whereas the possessive pronoun *my* can be replaced by others of the same POS (*his/her/your/*etc.). Hence, replacing the one results in an error (**in my point of view*[1]) while replacing the other is fine (*from her/his/their/our point of view*). The purpose of hybrid n-grams is to introduce the flexibility to capture the appropriate level of abstraction for each slot in a lexical chunk. Hybrid n-grams permit any combination of word forms, lemmas, POSs in a string (see details below). Thus the hybrid n-gram for *from my point of view* is *from [dps] point of view*[2].

For a string of input submitted for error checking, the algorithm first does a matching operation between the input string and the hybrid n-gram bank. The second step for input is finding hybrid n-grams which nearly match the input, using edit distance to measure nearness or similarity. Hybrid n-grams with a distance of 1 or less from the input string are candidates as correction suggestions and are ranked, least distant from the input string ranked as top correction suggestion.

4 The Knowledgebase: Hybrid N-grams

As mentioned in Section 3, a hybrid n-gram bank will be needed. In our model, each slot has four levels of representation to choose from: word form (*enjoys* but not *enjoy or enjoying, etc*); lemma (representing all word forms of that lexeme, e.g., *enjoy, enjoys, and enjoyed, etc*); detailed POS (CLAWS5 with 46 different POSs); rough POS (9 different POSs)[3]. The main challenge is to extract hybrid n-grams which are the optimum combination of representations for each slot to represent a lexical chunk or pattern. One key to this is a pruning method (described below). Clearly, compared with traditional n-gram extraction, the size of our hybrid n-gram bank size will be extremely large if we save all the combinations that can be generated for each n-gram. Considering the example *from my point of view* and setting *point* as the target word, if we only extract hybrid 5-gram strings for it, we will get $2*4^4=512$ (two forms of noun *point* and four forms of others) different hybrid 5-grams. This entails many disadvantages, for example in storage space and processing time. Therefore, we apply several pruning approaches to keep only useful hybrid n-grams in the bank. Another motivation for pruning the bank is to reach optimum recall and precision. The choice of which hybrid n-grams to retain in or discard from the bank directly determines which input strings would be judged as errors and what candidate corrections would be generated for errors. We illustrate the effects of pruning below.

The first criterion for pruning is frequency. Only hybrid n-grams with a frequency greater than the threshold are saved. The second criterion is called **subset pruning**. There will be overlap among different hybrid n-grams. For example, the chunk *from my point of view* could be represented by dozens of hybrid n-grams. Two of them are: (1) *from [dps] point of view*, and (2) *from my point of view*. Notice an input string *from her point of view* would match (1) but not (2). Here the optimum n-gram is (1) because it includes all cases covered by (2) but other acceptable ones as well. Crucially, it is not the case that the more general hybrid n-gram will always yield the more optimum results, however. This must be determined case by case. Consider the first slot in the same chunk *from my point of view*. The following two versions could represent that chunk: (3) *from [dps] point of view* and (4) *[prp] [dps] point of view*[4]. Notice here, however, that it will be the more specific rather than the more inclusive version that is to be preferred. (3) specifies the exact preposition for the chunk whereas (4) would accept any preposition

[1] We use * to represent the error part in n-gram string.
[2] We use [] to represent POS categories. *[dps]* is the CLAWS5 tag for possessive pronoun.
[3] Rough POS includes verb, noun, adj, adv, conj, interj, prep, pron, vm0.
[4] *[prp]* is the CLAWS5 tag for preposition.

(or *[prp]*) occurring in the first slot. But indeed *from* is not freely substitutable in this chunk (cf **in my point of view*). Thus in each slot in each chunk, pruning checks each potential hybrid n-gram against the target corpus to determine statistically the n-grams that capture the optimum degree of substitutability or frozenness for each slot.

This creates an extremely flexible means of representing the knowledgebase. Consider verb complement selection. In examples such as *They enjoy swimming*, the level of generalization is different for the governing verb slot (*enjoy*) on the one hand and the complement (*swimming*) on the other. The right generalization for the complement is a specific verb form but not specific to any one verb. This slot is captured under the CLAWS5 POS *[vvg]* [5], thus permitting *enjoy swimming/reading/sleeping*, but not *enjoy to swim/swam* and so on. Unlike the complement, the governing verb slot here is a specific lexeme (*enjoy swimming* but not *hope swimming; cf hope to swim*) and moreover, it permits that lexeme in any of its word forms (*enjoy/enjoying/enjoyed swimming*). A hybrid n-gram representation has the power to capture these different levels of generalization and restriction in one representation.

Here is how pruning is done. First, we set a **filter factor** ε, where $0<\varepsilon<1$. Assume x and y are two hybrid n-grams and len(x)=len(y). If $x \subset y$ and $|x|/|y| \geq \varepsilon$[6], we will eliminate y from bank. For example, for the two 5-grams *x=from [dps] point of view* and *y=[prp] [dps] point of view*, obviously $x \subset y$ because *from* is a kind of *[prp]* (preposition). If we set the filter factor ε=80% and $|x|/|y|>\varepsilon$, y will be not included in the hybrid n-gram bank. For example from 100M-word BNC, before pruning, there are 110K hybrid n-grams containing target lemma *point*. After pruning, there are only 5K useful hybrid n-grams left.

5 The Edit Distance Algorithm for Error Detection and Correction

5.1 Error Detection

We apply a simple edit distance for error detection by comparing user input n-grams and standard hybrid n-gram in the bank. The approaches are briefly summarized and short examples given in the following:

Step 1: POS tag the user input string and get all hybrid n-grams that can represent that string. For example, a user inputs *in my point of view* and then *[prp] my point of view*, *[prp] [dps] point of view*, *in [dps] point of view*, *in my point of [nn1]*… etc. will be generated. Let *C* denote the entire set of hybrid n-grams generated from an instance of user input.

Step 2: Search all hybrid n-grams in the target knowledgebase containing *point* or *view*, which are the content words in user input. Let *S* denote all of the target hybrid n-grams containing *point* or *view*.

Step 3: Compute the edit distance *d* between every element in *C* and *S*. If $\exists\, d=0$ in (*C*, *S*), we assume the user input n-gram is correct. If $\forall\, d>1$ in (*C*, *S*), our system will ignore this case and provide nothing. If $\exists\, d=1$, we assume the user input might be wrong and the system will enter the error correction procedure.

For efficiency's sake in Step 2, the hybrid n-grams are indexed by content words. We use Levenshtein's edit distance algorithm [Levenshtein 1996] in Step 3. It indicates the difference between user input and standard n-grams in three ways: "substitute relation," i.e., two n-grams are the same length and identical except for one slot. "Delete relation" and "insert relation" hold between two different length n-grams. In this paper we consider only the "substitute relation," such as *in my point of view* and *from my point of view*. This limits edit distance computing to pairs of n-grams of the same length (e.g. 5-gram to 5-gram).

5.2 Error Correction

The system identifies correction candidates from *S* as those with edit distance *d*=1 from some member(s) in *C*. Once the system gets several correction candidates for an input string whose edit distances from user input are 1, we have to decide the ranking of the correct candidates by a value called **weighted edit distance**. Weighted edit distance can identify more appropriate correct n-grams for the user. Imagine a case where an n-gram from *C* and an n-gram from *S* show a substitution relation. Assume *u* is the differing element in the C n-gram and *v* is its counterpart in the S n-

[5] *[vvg]* is the CLAWS5 tag for gerund.

[6] $|x|$ means the frequency of x in BNC.

gram. Weighted edit distance between these two is computed by the following rules:

Rule 1: If *u* and *v* are both word-forms and are different word-forms of the same lemma (for example *enjoyed* and *enjoying*), given distance α.

Rule 2: If *u* and *v* are both members of CLAWS5 POS and their rough POS are the same, given distance β[7].

Rule 3: If *u* and *v* are both function words, give distance γ.

Rule 4: If *u* and *v* are both content word, give distance δ.

We set $\alpha<\beta$ and $\gamma<\delta$. Correct candidate with lower weighted distance makes itself more appropriate for suggestion. For example, before weighting, the error string *pay attention on* gets two distance 1 correct candidates *pay attention to* and *focus attention on*. Weighting will give *pay attention to* a lower weighted distance because *on* and *to* are function words whereas *focus* and *pay* are content words.

6 Experimental Result

Four types of errors shown in Table 1 are examined for our detection and correction algorithm.

Error string	Algorithm result	Correction suggested to user
Preposition		
have a look *of	have a look at	have a look at
I am interested *of	[pnp] **be** interested in	I am interested in
*in my point of view	from [dps] point of view	from my point of view
pay attention *on	pay attention to **pay** attention to	pay attention to
We can discuss *about.	we [vm0] discuss it we [vm0] discuss [noun] we [vm0] discuss [av0]	we can discuss it we can discuss [noun] we can discuss {adv}
Adjectival participles		
He is *confusing with	[pnp] **be** confused [prp]	He is confused with
I am *interesting in	[pnp] **be** interested in	I am interested in
I am *exciting about	[pnp] **be** excited [prp]	I am excited about
Verb form		
He wants *reading.	he wants [vvt] he **want** [vvt]	He wants to read
I enjoy *to read.	i enjoy [vvg] i **enjoy** [vvg]	I enjoy reading
let them *to stay.	let them [vvi] **let** them [vvi]	let them stay
make him *to leave	make him [vvi] **make** him [vvi]	make him leave
must let them *to stay	[vm0] let them [vvi]	must let them stay
spend time to understand	**spend** time [vvg]	spend time understanding
will make him *to leave	will make [pnp] [vvi]	will make him leave
Missing be		
I* afraid of	**be** afraid of [adv] afraid of [av0] afraid of	be afraid of [adv]afraid of [adj] afraid of
They* aware of	**be** aware of [av0] aware of	be aware of [adv]aware of

Table 1: Four error types and their examples with correct suggestions.

[7] Recall we use two levels of POS tagging in our hybrid n-grams: 1. The detailed one is CLAWS5 with 46 tags. 2. The rough or simple tag set of 9 tags.

7 Conclusion

We propose an algorithm for unsupervised lexical error detection and correction and apply it to a user tool called Lexbar. This is a work-in-progress report, and we have not yet run full testing with a large data set, such as a learner corpus. However the early stage experimental results show promise, especially its broad coverage over different error types compared to error-specific approaches.

Acknowledgments

The work described in this paper was partially supported by the grants from the National Science Council, Taiwan (Project Nos. 96-2524-S-008-003- and 98-2511-S-008-002-MY2)

Reference

Martin Chodorow and Claudia Leacock 2000. An unsupervised method for detecting grammatical errors. *Proceedings of the 1st conference of NAACL*, pages 140–147.

Rachele De Felice and Stephen G. Pulman 2008. Automatic detection of preposition errors in learner writing. *CALICO AALL Workshop*.

M. Gamon, J. Gao, C. Brockett, A. Klementiev, W. B. Dolan, D. Belenko, and L. Vanderwende 2008. Using Contextual Speller Techniques and Language Modeling for ESL Error Correction. *Proceedings of IJCNLP*.

V. I. Levenshtein 1966. Binary codes capable of correcting deletions, insertions, and reversals. *Soviet Physics Doklady*, 10:707–710.

Martí Quixal and Toni Badia 2008. Exploiting unsupervised techniques to predict EFL learner errors. *CALICO AALL Workshop*.