A Computational Approach to the Discovery and Representation of Lexical Chunks

David Wible, Chin-Hwa Kuo, Meng-Chang Chen, Nai-Lung Tsao, Tsung-Fu Hung

A Computational Approach to the Discovery and Representation of Lexical Chunks

David Wible[1], Chin-Hwa Kuo[2], Meng-Chang Chen[3], Nai-Lung Tsao[3], Tsung-Fu Hung[2]

[1] Department of English, Tamkang University, Taipei
wible45@yahoo.com

[2] Computer and Network Lab, Tamkang University, Taipei

[3] Institute of Information Science, Academia Sinica, Taipei

Résumé

La connaissance des « chunks » (tronçons) lexicaux est maintenant reconnue comme une compétence essentielle pour l'apprentissage d'une seconde langue. Nous étudions deux des principaux problèmes que les « chunks » posent en lexicographie et nous présentons des méthodes de résolution informatiques. Le premier problème est celui de l'apprentissage de connaissances lexicales, c'est-à dire la nécessité de définir quelles suites de mots constituent des « chunks » utiles à l'apprenant. Le deuxième problème est celui de la représentation, c'est-à-dire comment mettre cette connaissance à la disposition de l'apprenant. Pour résoudre le premier problème, nous proposons un algorithme glouton exécuté sur un corpus de 20 millions de mots du BNC qui reproduit des mesures d'associations de mot sur des n-grams de plus en plus longs. Cette approche donne la priorité à un rappel élevé et tente d'isoler les faux positifs à l'aide de mécanismes de tri. Pour résoudre le problème de la représentation, nous nous proposons d'associer cet algorithme à un navigateur en tant qu'extension de notre outil de détection de collocations.

Mots-clés : « chunks » (tronçons) lexicaux, lexicographie computationnelle, association de mots, apprentissage de langues étrangères.

Abstract

Lexical chunks have in recent years become widely recognized as a crucial aspect of second language competence. We address two major sorts of challenge that chunks pose for lexicography and describe computational approaches to addressing these challenges. The first challenge is lexical knowledge discovery, that is, the need to uncover which strings of words constitute chunks worthy of learners' attention. The second challenge is the problem of representation, that is, how such knowledge can be made accessible to learners. To address the first challenge, we propose a greedy algorithm run on 20-million words of BNC that iterates applications of word association measures on increasingly longer n-grams. This approach places priority on high recall and then attempts to isolate false positives by sorting mechanisms. To address the challenge of representation we propose embedding the algorithm in a browser-based agent as an extension of our current browser-based collocation detection tool.

Keywords: lexical chunks, computational lexicography, word association, foreign language learning.

1. Introduction

There has been a recent growing trend in foreign language education research to recognize words as inextricably entwined with a range of syntagmatic contexts and contextual patterns as opposed to viewing them as discrete units that can be mastered in isolation. With this has come the recognition that multiword expressions are a central rather than peripheral aspect of lexical knowledge (See Wray 2002 for an extensive review). Among the range of multiword phenomena, certain types, such as phrasal verbs and collocations, have enjoyed a long tradition of attention in lexicography and language pedagogy. Other sorts of multiword

expressions such as lexical chunks and formulaic sequences, however, have only recently begun to attract this sort of attention. Knowledge resources concerning these expressions are thus correspondingly scarce. The work reported here is part of a larger project aimed at filling this gap in resources for the learning of lexical chunks.

2. Lexical Chunks and the Two Challenges to Lexicography

Lexicography traditionally has faced two kinds of challenges: lexical knowledge discovery and lexical knowledge representation. We want to suggest that lexical chunks introduce novel requirements to both sorts of challenge. In what follows we consider these two in turn and present our approach to each as we go. The literature on chunks includes a variety of criteria for defining what constitutes a chunk (see Weinert 1995 for an overview) from semantic and syntactic criteria such as (non-)compositionality and (non-)productivity to processing criteria, the most commonly cited of the latter being that users store and retrieve chunks as single multiword units rather than by rule-governed composition in real time. Implementing any of these criteria computationally for real world applications involves serious problems of scalability. The most promising trait of chunks in this respect is frequency of occurrence, yet the implementation of frequency must be appropriately nuanced. Taking inspiration from the achievements of statistical approaches to collocation extraction (Church et al 1991; Smajda 1993, inter alia), we choose as a practical starting point to operationalize the notion of lexical chunk statistically. Our specific implementation of a statistical approach is elucidated in what follows as part of our approach to lexical chunk discovery.

3. Lexical Chunks and Knowledge Discovery

A fundamental task of the lexicographer is to uncover facts concerning the behavior and meaning of words. These facts constitute the substance of any dictionary. One of the most important developments in this work over the past few decades has been the rise of machine-readable corpora and computational tools that aid in their analysis. Learner dictionaries in particular have benefited from the lexicographers' extended capacity to distill patterns and nuances governing the use of specific words made possible by access to massive collections of texts and tools for mining them (Sinclair 1987, inter alia). Certain multiword expressions have proven particularly susceptible to discovery with such tools. Inspired by the work of Church and his colleagues in the late 1980's (Church *et al.*, 1991), researchers have found, for example, that collocations are detectable by well-understood statistical measures of word association strength such as mutual information (MI). There are certain characteristics of collocations and, say, phrasal verbs as well, that render them especially vulnerable to detection by these statistical tools. Word association measures can detect the association strength between two event types. Construing a pair of lexemes occurring in close proximity to each other to be a pair of events in running text, lexicographers can use these word association measures to determine the strength of association between these two events and thus detect which word pairs have sufficiently strong association to be collocations.

Lexical chunks as a class of multiword expression, however, are much less homogeneous in this respect. A chunk could consist of two words (*e.g.*, *in fact*) or five words (*as a matter of fact*). In fact, there would seem to be no principled limit to the size of a chunk. What we want to suggest is, first, that this property of chunks creates a non-trivial challenge to computational lexicography and, second, that it is tractable.

We have developed a computational approach to the extraction of lexical chunks from very large corpora. Our algorithm takes as its kernel a simple word association measure (any such

measure can serve as this kernel, for example, mutual information or simple conditional probability, inter alia) and iterates its application. Unlike traditional word association measures, then, our algorithm is greedy. As a consequence, our approach is insensitive to chunk size. The same algorithm that detects *in fact* also detects *as a matter of fact*.

The current interface is designed for lexicographers, not for learners (the design of a representation for learners is the topic of the next section). The algorithm takes as its input a single word/POS pairing. A sample input would be *fact/N* (*i.e.*, the noun *fact)*. The current version of the algorithm runs both a conditional probability measure and MI for the target word paired with each possible candidate collocate occurring within a five-word window of the target word in a 20-million-word portion of BNC (See Wible *et al.*, 2004a for details). The user selects a minimum association score to set as a threshold for each of the two measures; a default threshold is used if the user makes no selection of score threshold. Word association scores are then tabulated for every word that has tokens which occur within the five-word span of any token of the target word (*e.g.*, *fact*). These pairings are sensitive to linear order; for example, tokens of the pair *in…fact* and *fact…in* are scored separately. Those pairings with association scores that satisfy the threshold are considered hits and are highlighted in the results display.[1] To this point, this process resembles closely the extraction of collocations. Since chunks are not limited to word pairs, however, we iterate this process, and all (ordered) pairings with the target word (*e.g.*, *X fact* or *fact X*) serve as the input to the next iteration. We therefore refer to these inputs as bigrams. Word association measures look at the strength of association between two events x and y. In traditional collocation detection, as in our first iteration, x and y are individual words. At the second iteration, however, we use the same measures but change the definition of the events x and y. Here x is a word pairing (or bigram) from the first iteration (rather than just a word) and y is a word. This iteration then is measuring the association between each bigram (or a word pair) on the one hand and each word appearing within the five-word span of that bigram on the other. The output of this next iteration would consist of a set of ordered trigrams (potentially discontinuous due to our five-word window). Those trigrams that achieve the threshold word association score at this iteration are considered hits and highlighted in the results representation. All trigrams from this iteration in turn serve as input to the next iteration, and so on, in greedy fashion until no candidate strings meet the threshold association score. A hit for this algorithm, then, is a string of any number of words that results from satisfying the threshold score at the last iteration of any number of iterations. The system then creates a link from each such string to a display of all of the BNC sentences that instantiate that string.

Our searches provide two levels of results, which we will refer to as string types and string tokens. An example of a string type would be the string *point of view* whereas string tokens would be specific attested instances in the corpus that instantiate this string type. One reason this distinction becomes important is that our algorithm allows a five-word span for co-occurrence at any iteration. Thus, the string type *point of view* includes not only tokens where the three words are contiguous (*e.g.*, *According to her point of view…*), but also tokens where the same three words are non-contiguous (*e.g.*, *The point of mentioning this view is to…*).

There is no way of us predicting a priori which of these patterns within the tokens of one string type constitutes a true positive, so we cast our net wide in this way for the sake of recall at the cost of introducing noise. In the case of *point of view*, for example, it is the completely

[1] The results display distinguishes three conditions: word pairs that meet the conditional probability threshold only are highlighted in red, those meeting the MI threshold only are displayed in blue, and those meeting both in purple. This makes it easy to compare the performance of the two measures or the value of using both.

contiguous string tokens that are the true positives while the non-contiguous example (*the point of mentioning this view…*) illustrates a false positive. There are cases, however, where the reverse holds, that is, where discontinuity is a true property of a particular chunk. For example, our algorithm detects the longer string type *from point of view* as well. Notice, however, that for this case, in the true positives, *from* and *point* are non-adjacent (*from a logical point of view*; *from their point of view*, etc).

Thus, while our use of a five-word window at each iteration is motivated, it also introduces substantial noise in the results. To address the noise, we apply to these results some sorting mechanisms intended to help distinguish noise from the true positives. While this sorting falls short of actually affecting the precision of the results, it is aimed at organizing the results into groups of patterns for the hand inspection of the lexicographers at the user end.

We illustrate our approach to the sorting with a specific string type: *tell the time*. This string token is extracted by taking the noun *time* as the query target. At the second iteration, the trigram *tell the time* is detected as satisfying both the conditional probability and the MI thresholds. Moreover, 54 tokens of this type are found in the 20-million-word portion of BNC that we use. While the string type *tell the time* would appear at first glance to be a true positive, a look at the 54 tokens shows the noise. Our original version of the system left these 54 tokens unsorted and displayed them in the order they were detected in BNC. A sample of these 54 tokens is provided here:

She told herself sternly that the time had passed when sympathy
You tell me lies all the time
and one half could not tell the time or correctly select a medicine bottle
He found messages telling him that the time was not ripe
observation of the sun was a useful way of telling the time
I'm telling you for the last time, Harvey
I was told at the time that this system had been adopted because…
No mechanical indicator can tell you the right time to strike
…activities of daily living (such as counting money, telling the time, reading…)
observation of the sun was a useful way of telling the time

Our current version of the algorithm adds two stages of sorting to these 54 examples. The first step sorts the examples according to the patterns of contiguity of the string members. Thus, at this stage, all tokens where the string members (*tell*, *the*, *time*) are contiguous (*tell the time*) are grouped together, then all tokens where the first and second word (*tell* and *the*) are separated by one word are grouped together (*tell him the time*; *tell of the time*), and those with two words intervening there (tell him that the time), and so on. This stage of sorting, thus, is sensitive to the existence and location of any gaps separating the words in the string type and to how many words appear in those intervening gaps. These patterns are displayed in order of frequency. Thus, in the case of *tell the time*, the most frequent pattern is the one with no intervening words, such as *So what if you can't tell the time?* These comprise 16 of the 54 tokens. The second most common is the pattern where three words separate *tell* from *the time*, as in *She told herself sternly that the time has passed*. There are 8 tokens of this pattern. A look at these 8 tokens, however, will illustrate the motivation for adding a second stage to this sorting. Specifically, there is no coherent pattern shared by these 8 tokens. The fact that *tell* and *the time* are separated by three words in all eight examples follows from nothing interesting.

1. The court was told that teenagers were made to suck dummies and wear nappies , were bathed like babies and told to regress to the time they were last happy.

2. But , I told myself , by the time you are standing at the airport terminal (not the train or the bus station) , you have burned off the top ones and , come on , lad , you can afford to relax a little.
3. She told herself sternly that the time had passed when sympathy , hope , tender care , even love could have anything to do with the figure on the bed.
4. You tell me lies all the time !
5. Once the commotion had died down , he told them to break the time pencils and get to work.
6. I told her I thought The Times would probably have a man on the spot and it was late , and I prised my Toshiba away from her grasping hands.
7. Junior accountants at Price Waterhouse , one of the big six firms , have been told that now is the time to take their once-in-a-lifetime world tour or perhaps a summer stint as a yacht deckhand and that applications for extended unpaid leave bquo will be looked on favourably.
8. Carter had told the police at the time they had a row over money and his wife had walked out , never to return.

While this group is the second most common pattern for *tell the time* string type, it is a false pattern, one that represents no interesting regularity. In this case, then, the first stage of sorting does not contribute directly to chunk detection. Part of the motivation of the second stage is to bring us closer to detecting such results as noise automatically, eventually enabling us to not only sort examples but also filter out noise. To help achieve this, the second stage examines the POS of the intervening words in the string tokens within a pattern. The POS tags are from the CLAWS tagset used to tag BNC[2]. Listed here are the eight different POS trigrams representing the three words separating *tell* from *the time* in this group.

1.	TO0_VVI_PRP	3. PNX_AV0_CJT	5. PNP_TO0_VVI	7. CJT-DT0_AV0_VBZ
2.	PNX_PUN_PRP	4. PNP_VVZ_DT0	6. PNP_PNP_VVD	8. AT0_NN2_PRP

The fact that the eight sentences, while all sharing a 3-word gap between *tell* and *the time*, each has a different POS trigram in that gap can serve as readily detectable indication that this group of eight tokens does not reflect an interesting regularity concerning *tell the time*. Eventually our goal is to represent results to learners and teachers, not only to lexicographers. For this reason, such strategies for automatically detecting false positives and increasing precision will be important to ensure such examples are not presented to learners as instances of the chunk tell the time. In what follows, we describe our approach to representing our lexical chunk knowledge for learners and teachers.

4. Lexical Chunks and Knowledge Representation

A second challenge for lexicography concerning chunks is how to represent them to learners. We propose embedding the representation of chunk knowledge within the contexts where learners encounter chunks. As learners encounter the target language in contexts of authentic use, they are exposed to chunks (whether they recognize them as such or not). Our aim is to identify these chunks in real time directly within digital contexts. We illustrate how we have implemented this approach with collocations and describe the challenges in extending this approach to chunks. Our collocation tool (called Collocator) is a browser-based tool that can

[2] http://www.natcorp.ox.ac.uk/what/garside_allc.html

detect collocations in real time within the web pages that the user browses (Wible *et al.*, 2004a).

The core component of Collocator that detects collocations in web pages in real time works much like the first iteration of our chunk detecting algorithm. The collocation-extracting scheme is part-of-speech sensitive, which means we have to detect the part-of-speech information of each word in browsed web pages in real time. We train a Markov Model-based POS tagger (Brants, 2000) and use British National Corpus (BNC)[3] as our training data. The internal evaluation shows this tagger has 93 % precision including identifying unknown words. After part-of-speech tagging, the agent uses the following equation from (Wible *et al.*, 2004b) to measure the word association score for all candidate word pairs:

$$normMI(x, y) = \log_2 \frac{P(x, y)}{\left(P(x)/sn(x)\right) \cdot \left(P(y)/sn(y)\right)}$$

where x, y mean the word with specific part-of-speech and *sn* means the number of distinct senses for that word listed in WordNet. This adaptation of traditional MI takes into account the polysemy of the words x and y by normalizing for the number of senses of x and y, helping overcome traditional MI's under-extraction of collocations that contain high frequency words. For example, traditional MI does not detect *take* as a collocate of the noun *temperature* (*The nurse took the patient's temperature*), but our normalized MI does detect *take* in this case.

Similar to the first iteration in our chunk detector, Collocator takes as collocation candidates all possible pairings of POS-specific words (x and y above) in which the two words appear within a five-word window of each other in our 20 million-words of running text of the BNC. Using the above measure, each x,y ordered pair yields a word association score. Collocations are word pairs that show a sufficiently strong word association between the two words in the pair. Thus, a minimum score threshold is used to select which of the candidate word pairs constitute collocations. This threshold can be lowered or raised to adjust the agent's precision and recall. The collocation knowledge thus extracted from our POS-tagged BNC feeds our browser-based Collocator, enabling it to detect and highlight collocations that appear in the web pages that the user browses (See figure 1).

Figure 1. Toolbar's dropdown menu with detected collocations and link to examples

[3] http://www.natcorp.ox.ac.uk/

Using Collocator as our reference point, we can now describe how our lexical chunk extraction is to be applied to learners. In brief, our purpose is to enrich Collocator so that it detects and represents not only collocations (typically, word pairs) but longer strings, that is, chunks. This tool will then detect and highlight chunks in real time within the web pages that the user browses. The approach to lexical chunk discovery described above is intended, then, as the knowledge source supporting this tool. The fundamental challenge this application poses is that of precision.

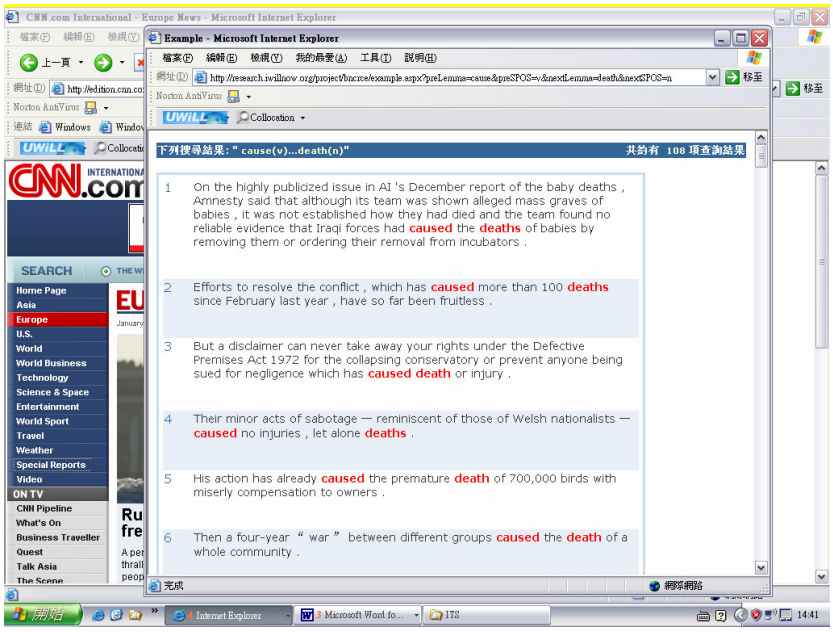

Figure 2. Example sentences

As shown above, each iteration of the chunking algorithm introduces noise and leads to more false positives with each iteration. A ubiquitous browser-based version of the chunker would detect strings in the web page that match any string types that have achieved a threshold word association measure at our 'discovery' stage. Recalling the example of *tell the time*, this means that a browser-based chunk detector would treat as a match not only the true positive *He hasn't learned to tell the time*, but also false positives like *Tell him that unfortunately the time has not arrived*. These failures in precision, while forgivable for a tool intended for lexicographers, are unacceptable for a tool intended directly for learners as they browse the Web. We choose to retain the higher recall created by our 5-word window and focus on increasing precision by adding a second stage that sorts these results into patterns, as described above. While our current sorting method does make it much easier to filter out false positives by hand, this is not improvement enough for a browser-based version for learners. The sorting strategy that we mentioned early of exploiting POS patterns, though requiring refinements, does hold promise. For example, it would enable the chunker to discard false positives like *Tell him that unfortunately the time has not arrived* since, as we have seen, it can discover that cases like this with a 3-word gap between *tell* and *the time* exhibit no regularities in the POS sequences appearing in that gap. However, the implementation of this strategy must be much more fully articulated and tested. Recently, we have added entropy measures to detect how stable the POS patterns are that occur within the strings. Preliminary results suggest high entropy as a promising indicator of false positives, allowing for automatic filtering for increased precision of our results.

References

BRANTS T. (2000). "TnT-A statistical part-of-speech tagger". In *Processings of ANLP-2000*. Seattle, Washington.

CHURCH K, GALE W., HANKS P., HINDLE D. (1991). "Using statistics in lexical analysis". In U. Zernik (ed.), *Lexical Acquisition: Exploiting Online Resources to Build a Lexicon*. Lawrence Erlbaum Associates, Hillsdale: 115-164.

SINCLAIR J. (ed.) (1987). *Looking UP: An Account of the COBUILD Project in Lexical Computing*. Collins, London.

SMADJA F. (1993). "Retriving Collocations from Text: Xtract". In C*omputational Linguistics* 19: 143-177.

WIBLE D., KUO C.-H., TSAO N.-L. (2004a). "Contextualizing Language Learning in the Digital Wild: Tools and a Framework". In *Proceedings of IEEE International Conference on Advanced Learning Technologies (ICALT)*. Joensuu.

WIBLE D., KUO C.-H., TSAO N.-L. (2004b). "Improving the Extraction of Collocations with High Frequency Words". In *Proceedings of International Conference on Language Resources and Evaluation (LREC)*. Lisbon.

WEINERT R. (1995). "The Role of Formulaic Language in Second Language Acquisition: A Review". In *Applied Linguistics* 16: 180-205.

WRAY A. (2002). *Formulaic Language in the Lexicon*. Cambridge University Press, Cambridge.