Unsupervised acquisition of comprehensive multiword lexicons using competition in an n-gram lattice

Anonymous EACL submission

Abstract

1 Introduction

2 Background and Related work

In this paper we are attempting to identify *formulaic sequences*, following the terminology of Wray (2002; 2008), who defines a formulaic sequence (FS) as "a sequence, continuous or discontinuous, of words or other elements, which is, or appears to be, prefabricated: that is, stored and retrieved whole from memory at the time of use, rather than being subject to generation or analysis by the language grammar." In other words, a formulaic sequence shows signs of being lexicalized. In computational linguistics, the most common term used to describe multiword lexical units is *multiword expression* or MWE (Baldwin and Kim, 2010), but here we wish to make a principled distinction between at least somewhat non-compositional, strongly lexicalized MWEs and FS, a superset which includes MWEs but also (apparently) compositional linguistic formulas. This distinction is not a new one, for instance it exists to some extent in the MWE annotation of Schneider (Schneider et al., 2014b), who distinguishes between weak (collocational) and strong (non-compositional) MWEs, but it is our contention is that separate, precise terminology is useful for research targeted at either class: we need not strain the concept of MWE to include items which do not require special semantics, nor are we included to disregard or minimize the larger formulaticity of language simply because it is not the dominant focus of MWE research. Many MWE researchers would defensibly balk at including in their MWE lexicons and corpus annotations (English) formulaic sequences such as *there is something going on*, *it is more important than ever to*, *not know what it is like to...*,*it would be great if * could*, *there is no shortage of...*, *the rise and fall of...*, *now is not the time to...*, as well as tens of thousands of other such phrases which, along with less compositional, clearly MWE formulas like *be worth * weight in gold*, are identified by the FS-extraction system presented here.

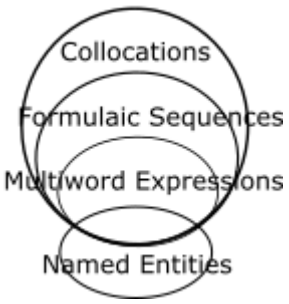

Figure 1: Multiword Terminology

Figure 1 shows our conception of the terminology for multiword relationships, including FS, MWE, as well as collocations and (multiword) named entities. We use the term *collocation* to refer to statistical co-occurrence without necessarily any lexicalization of the sort implied by our definition of FS. MWEs, again, are a non-compositional subset of FS, and they overlap with named entities. In this regard, we disagree with the MWE annotation schema of Schneider (Schneider et al., 2014b) which simply includes any named entity as an MWE, which, in the context of product reviews from the Web Treebank, means that that named entities form nearly 50% of the MWEs types attested in that corpus, with many being phrases such as *Tintman Nationwide Tints Ltd* or *Hickory Furniture Mart* which are easily identifiable as named entities based on their form, but would not be considered part of the mental lexicon of even a tiny fraction of English speakers (not even, we might argue, most of these people who work for these

businesses, they likely refer to them using shorter, informal names). The upshot of this choice that a significant portion of "MWE" identification in the dataset is in fact named entity recognition. Some well-known named entities are obviously good MWEs (e.g., *George Washington*, *the United States*), but not the names of the vast majority of people, businesses, or products which have real-word referents; they do not belong in a lexicon of formulaic sequences which is intended to reflect the internal lexicon of (at least) a significant generalizable subset of English speakers, putting them outside the scope of this work.

Regardless of the terminology used to describe them, the starting point for multiword lexicon creation has been typically lexical association measures (Church and Hanks, 1990; Schone and Jurafsky, 2001; Evert, 2004; Pecina, 2010; Araujo et al., 2011; Kulkarni and Finlayson, 2011); when these methods are used to build a lexicon, particular syntactic patterns and thresholds for the metrics are typically chosen. Only some of these measures generalize beyond two words, for example PMI (Church and Hanks, 1990), the log ratio of the joint probability to the product of the marginal probabilities. Other measures specifically designed to address sequences of larger than two words include the c-value (Frantzi et al., 2000), a metric designed for term extraction which weights term frequency by the log length of the n-gram while penalizing n-grams that appear in frequent larger ones, and mutual expectation (Dias et al., 1999), which produces a normalized statistic that reflects how much a candidate phrase resists the omission of any particular word. The lexical predictability ratio (LPR) of Brooke et al. (Brooke et al., 2015) is a association measure intended for any possible syntactic pattern which is calculated by discounting syntactic predictability from the overall conditional probability for each word given the other words in the phrase. Though most association measures involve only usage statistics of the phrase and its subparts, the DRUID measure is an exception which uses distributional semantics around the phrase to identify how easily n-gram could be replaced by a single word (Riedl and Biemann, 2015).

Other research in MWEs has tended to be rather focused on particular syntactic patterns, looking at, for instance, just verb/noun combinations (Fazly et al., 2009). The recent work of Schneider et al. (2014a) is a rare example of a comprehensive MWE identification model which distinguishes a full range of MWE sequences in the English Web Treebank, including those involving gaps, using a supervised sequence tagging model; like other models in this space, Schneider et al. make use of existing manual lexical resources and they note that an (unsupervised) automatic lexical resource could be useful addition to the model, though attempts to do so have be mixed (Riedl and Biemann, 2016). Otherwise, gaps in MWEs have generally addressed by using full syntactic representations (Seretan, 2011).

Beyond association metrics, other general unsupervised approaches to the multiword unit identification include that of Newman et al. (2012), who used a generative Dirichlet Process model which jointly creates a linear segmentation of the corpus and a multiword vocabulary. Gimpel and Smith (2011) focus specifically on deriving word sequences with gaps using a generative model, with the intent of improving machine translation. The drawback to these generative methods, relative to association metrics, is scalability and lack of determinacy. Brooke et al. (2014; 2015) developed a heuristic method intended for general FS identification in larger corpora, first using the conditional probably statistics to do an initial (single pass) coarse-grained segmentation of the corpus, followed by a pass through the resulting vocabulary, breaking larger units into smaller ones based on marginal and conditional probabilities.

3 Method

The starting assumption for this work is that given a large corpus, a fairly low n-gram frequency threshold, and an extraction technique which allows for non-contiguous n-grams, we can collect a set of initial n-grams types which include a significant portion of the formulaic sequences (FS) that we can reasonably hope to identify in such a corpus; of course, the resulting vocabulary will be specific to the language, genre and domain, and there will be insufficient statistics to identify FS that are extremely rare in the corpus. Our method for FS extraction assumes such a initial set of n-grams, and focuses on identifying the true FS, and excluding those sequences which can be explained away due to the syntax of the language and/or the existence of other FS. The former can be captured to some extent using a syntax-sensitive association

measure like LPR, but the latter requires a full, iterative model, where n-grams in essence complete with each other to form a compact, parsimonious formulaic vocabulary.

Our approach to FS identification involves optimization of the total cost of a lattice where each node corresponds to an n-gram type derived from a large corpus. The cost of the whole lattice is defined simply as a product of the costs of the individual nodes. Each node can be considered either "on" (is an FS) or "off" (is not an FS), with the base cost of being off simply a variation on the LPR association measure. Most fundamentally, n-gram nodes in the lattice are directionally connected to nodes consisting of $n+1$-grams which subsume them and $n-1$-grams which they subsume: that is, the (gapped) n-gram *keep * under wraps* would, for instance be connected "upwards" to the node *keep everything under wraps* and connected "downwards" to *under wraps*. These directional relationships allow for two basic effects between nodes in the lattice when a node is turned on: *covering*, which inhibits nodes below (subsumed by) a turned-on node (e.g., , if *keep * under wraps* is on, the model will tend not to choose *under wraps* as an FS); and *clearing*, which inhibits nodes above a turned-on node (e.g., , if *keep * under wraps* is on, the model would avoid selecting *keep everything under wraps* as an FS). A third, undirected mechanism is *overlapping*, where nodes inhibit each other due to (non-subsuming) overlaps in the corpus (e.g., having both *keep * under wraps* and *be keep * under* as FS will be avoided). Given these relationships, we minimize the cost of the entire lattice by iterating over the nodes and greedily optimizing the on/off choice relative to the local neighborhood of each node, until convergence.

3.1 Collecting statistics

The first step in the process is derive a set of n-grams and related statistics from a large untagged corpus of text. Since our primary association measure is an adaption of LPR, our approach in this subsection mostly follows Brooke et al. (2015) up until the last stage. An initial requirement of any such method is a n-gram threshold cutoff, which for us (like Brooke et al. 2015) is 1 instance per 10 million words; we have done a manual analysis using the MWE corpus of (Schneider et al., 2014b) which indicates very good (over 90%) type-level coverage excepting those highly particular proper names which would be better handled by a named entity recognition system. Note that our n-grams also include gappy versions; the definition of gaps is language specific, and we will discuss the individual languages further in section XXXXX. In general, though, putting in a restriction of this sort is important because it greatly lowers the number of gappy n-gram types. We also lower our number of types generally by excluding punctuation and by lemmatizing. Note that as long as the count threshold is signifcantly above 1, efficient extraction of all n-grams can be done iteratively (one pass through the corpus per n) for both types in combination.

Once a set of relevant n-grams is identified (and counted), other statistics required to calculate *Lexical Predictability Ratios* (LPR) for each word in the n-gram are collected. Again, LPR measures how statistically unlikely a particular word is relative to how predictable the word is given the same amount of syntactic context. Formally, the LPR for some word w_i in the context of a sequence of words of length n which have a corresponding set of sequence of POS tags t, is given by:

$$LPR(w_i, w_{0,n}) = \max_{0 \leq j < k < n} \frac{p(w_i|w_{j,k})}{p(w_i|t_{j,k})}$$

These conditional probabilities can be derived in the standard way from n-gram and POS n-gram counts. We can use the same equation for gap n-grams, with the caveat that quantities involving sequences which include the location where the gap occurs are derived from special gap n-gram statistics, not regular n-gram statistics.

In Brooke et al. (2015), LPR for an entire span is calculated as a product of the individual LPRs, but this is not an appropriate choice here: instead, the base "off" cost for a node in our lattice is the minimum LPR across the words in the sequence (minLPR):

$$minLPR(w_{0,n}) = \min_{0 \leq i < n} LPR(w_i, w_{0,n})$$

The use of minimum here should be understood in the context of competition between nodes in the lattice: minLPR for a particular n-gram does not reflect the *overall* degree to which an n-gram holds together, instead it focuses on the word which is its weakest link. For example, in the case we are evaluating the quality of *be keep * under*

as an FS, a general statistical metric might give it a high score due the strong statistical relationship between *keep* and *under*, but the minLPR is focused on the weaker relationship between *be* and *keep * under*.

3.2 Node cost functions

For each node, there are two cost functions, one which is used if the node on (C_1), and the other if the node is off (C_0). The base value of C_0 is the minLPR metric discussed above. The base value of C_1 is a parameter to the model which, for nodes not influenced by one of the other factors discussed below, is equivalent to an initial threshold for building a vocabulary using the minLPR association measure. The lower bound on both costs is 1. There is no necessary upper bound on C_1, but we note here that there is a fairly small range of good choices for C_1; we have found that setting it too high (> 5, corresponding to being five times more likely than predicted by POS context alone) will restrict the model primarily to sequences with only open-class words, while setting it too low (< 3) will allow in too much random noise.

In general, we will be expanding these cost functions by adding exponential terms, reflecting the fact we do not want to override the association measure, but rather modulate it. Most of these modulations we use here are related to the node interactions discussed below, but we begin by introducing a simple example of a modulation which uses the n-gram count. First, lets define C_0 as the minLPR based taken to the power of some as yet undefined function d:

$$C_0(t) = minLPR(t)^{d_0(t)}$$

LPR is a association measure based on conditional probability, and therefore tends to underestimate the importance of marginal probability, missing common expression formed using high frequency words. As such, it is common for more complex association measures to include multiplication by the count, or logarithm of the count. In the context of this model, multiplying our cost by even the logarithm of the count would be far too extreme, but we can give apply a more restricted increase to C_0, forcing the model to give more weight to relatively common expressions, as follows: Let $c()$ be the count function, and $c(min)$ and $c(max)$ refer to the lowest and highest count n-gram in the lattice (not including unigrams).

$$d_0(t) = 1 + \frac{\log \frac{c(t)}{c(min)}}{\log \frac{c(max)}{c(min)}}$$

This logarithmically scales the count into the range of 0 and 1; the C_0 of n-grams at the cutoff will be unaffected ($mod(t) = 1$) , while the maximum value of d results in a squaring of the initial LPR-ratio.

[!tb]

3.3 Node interactions

The most important node interaction is *covering*, which corresponds to discounting or entirely excluding a node based on the presence of a node higher in the lattice. Our model includes two types of covering: hard and soft. Hard covering is based on the idea that, due to very similar counts, we can reasonably conclude that the presence of n-gram in our statistics is a direct result of the other: if we have 143 counts of *keep * under wraps* and 152 counts of *under wraps*, the presence of *keep * under wraps* almost completely explains *under wraps*, and it is better if we collapse these two decisions into one. We do this by permanently disabling any covered node, and giving the covering node the maximum minLPR among all the nodes it covers; this means that longer n-grams with function words (which often have lower minLPR) can benefit from the strong statistical relationships between open-class lexical features in n-grams that they cover. This is done as a preprocessing step, and besides being effective it also greatly improves the tractability of the iterative optimization of the lattice. Of course, a threshold for hard covering must be chosen: empirically, we have found that a ratio of $2/3$ (corresponding to a significant but not overwhelming majority of the counts of a lower node corresponding to the higher node) works well. Finally, we also use hard covering to partially address the rather challenging problem of pronouns, which, due to pragmatic biases in particular corpora, often have higher than desirable LPR ratios, but cannot reasonably be excluded either since many common FS contain them. In our model, n-grams with pronouns are considered covered (inactive) unless they cover at least one other node which does not have a pronoun.

Soft covering is used in cases when a single n-gram does not entirely account for another, but a turned-on n-gram to some extent may explain

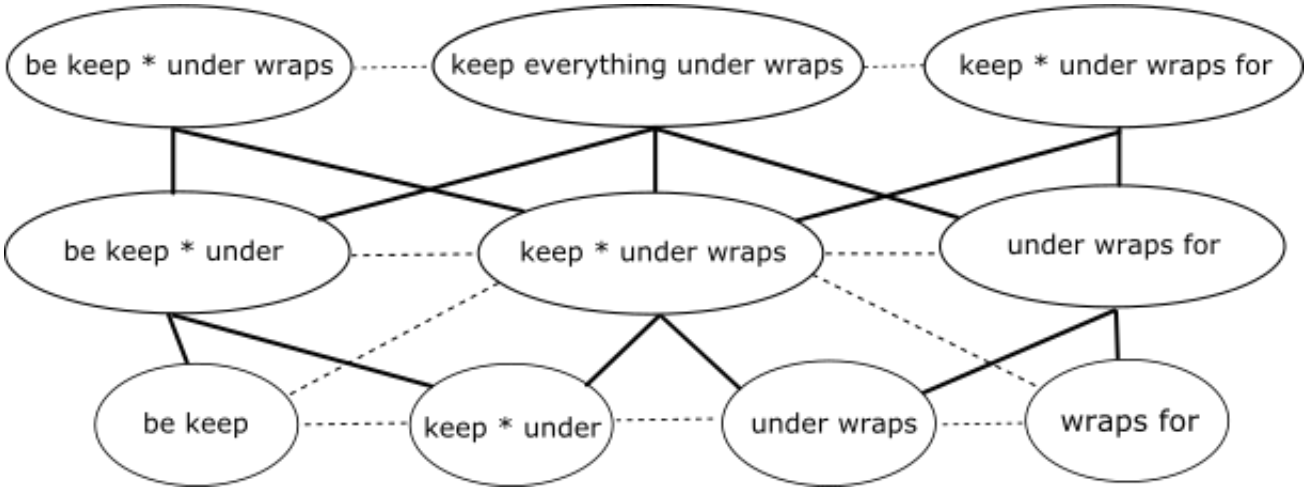

Figure 2: A portion of an n-gram lattice. Solid lines indicate subsumption, dotted lines overlaps

some of the statistical irregularity of one lower in the lattice. For instance, *keep * under* is not hard-covered by *keep * under wraps* (since there are FS such as *keep * under surveillance*, *keep it under your hat*,), but if *keep * under wraps* is tagged as an FS, we nevertheless want to discount the portion of the *keep * under* counts that correspond to *keep * under wraps*. We accomplish this by lowering the turned-off cost function of keep * under (and thus making turning on less desirable) in the following manner: let t be the target node, and $s_1 \ldots_m$ be any turned-on nodes which are above t in the lattice. Then, then the new d_0, d'_0 for a node t is:

$$d'_0(t) = \frac{c(t) - \sum c(s_1) \ldots c(s_m)}{c(t))} d_0(t)$$

This equation is intended as a simple, quick-to-calculate approximation of the result of recalculating minLPR with the counts corresponding to the covering nodes actually removed.

In general, covering prefers turning on longer, covering n-grams since doing so lowers the turned-off cost function of nodes lower in the lattice. Not surprisingly, it is important to have a mechanism working in opposition, i.e., one which generally prefers shorter n-grams. *Clearing* does this by changing the C_0 of nodes higher in the lattice when a lower node is turned-on. The basic mechanism is the same as covering, except that we cannot make direct make use of counts in the same way; whereas it makes sense to discount the turned-off cost of covered nodes in proportion to the (lower) relative counts of their covering nodes, in the reverse direction this logic entirely fails. A simple but effective solution is to use the minLPR value: a turned-on node will clear any node higher in the lattice which has a minLPR that is lower than it. We refer to this mechanism as clearing because it tends to clear away a variety of trivial uses of common FS which many have higher LPR due to the lexical and syntactic specificity of the FS.

To give an example of its effect, if the node *keep * under wraps* is turned on, then, assuming that their minLPR is lower, the C_0 of nodes such as *be keep * under wraps* and *keep * under wraps for* will be lowered to zero. This dissuades (but does not prevent) the model from turning on these nodes. Note that the effect of both covering and clearing is to lower the cost function for covered/clears nodes, which encourages the model to turn *keep * under wraps* on, even potentially in cases where its minLPR is relatively low.

The third mechanism of node interaction involves n-grams which overlap in the corpus. Though there are some token-level exceptions (e.g., coordinated compound nouns), it is generally the case that independent formulaic sequences do not overlap. For example, given that *be keep * under* and *keep * under wraps* often appear together, overlapping on the tokens *keep * under*, we do not want both being selected as an FS, even in the case where both had high minLPR. To address this problem, we use a mechanism somewhat different than above: rather than lowering the off-cost, we raise the on-cost. Let $oc(x, y)$ refer to number of times nodes x and y overlapped in the corpus, and $o_1 \ldots o_m$ refer to *turned-on* nodes which overlap with our target node t. With analogy to m_0 above, we define an m_1 exponential

term for our target node t as:

$$m_1(t) = \frac{c(t)}{c(t) - \sum oc(o_1) \dots oc(o_m)}$$

The effect of overlaps on the cost is hyperbolic: small amounts of overlapping have little effect, but as the amount of overlapping approaches the total count of the node, the cost becomes prohibitive, effectively forcing one of nodes to turn off. The idea behind lowering the cost when covering and clearing is to encourage the model to turn on nodes which effectively explain the presence of nodes above and/or below it in the lattice, but we do not want to encourage nodes which overlap at all, so instead we apply a penalty which increases the cost of turning on pairs of nodes which overlap.

3.4 Optimization

The complex interaction between nodes discussed in the preceding section prohibit a global optimization of the lattice. Fortunately, though a majority of nodes in the lattice are connected to each other though some path, most of the effects of nodes on each other are relatively local, and effective local optimizations can be made tractable by applying some simple restrictions. The main optimization loop consists of iterations over the lattice until complete convergence (no changes in the final iteration). For each iteration over the main loop, each potentially active node is examined in order to evaluate whether its current status is optimal given the current state of the lattice. The order that we do this has a significant effect on the result: among the obvious options, we have found that we get the best results by ordering our nodes by frequency, which gives an implicit advantage to relatively common n-grams, since they will tend to be turned on first.

Given the relationships between nodes that we have defined above, it is obviously not sufficient to consider switching only the present node; If, for instance, one or more of *be keep * under wraps*, *under wraps*, or *be keep * under* has been turned on, the covering, blocking, or overlapping effects of these other nodes will likely prevent a competing node like *keep * under wraps* from being correctly activated. Instead, the algorithm identifies a small set of nodes which are the most relevant to status of the node under consideration whose status should be considered simultaneously. Since turned-off nodes have no direct effect on each other, only turned-on nodes above, below, or overlapping with the current node need be considered. Once the relevant nodes have been identified, all (including turned-off nodes) nodes whose cost function is affected by one or more of the relevant nodes are identified, and then a greedy search is carried out for the optimal configuration of the relevant nodes starting from an 'all-on' state.

In practice, we apply the following restrictions which significantly reduce the runtime of the algorithm without sacrificing the quality of the output lexicon:

- The complexity of the algorithm increases exponentially with the number of "relevant" nodes, so we limit the total number to 5. When there are more than 5 nodes turned on the in the vicinity of the target node, they are selected by ranking in terms of the relative influence on each other given by the relative change in cost functions as defined in preceding section.
- The total number of pairwise n-gram overlaps in a large corpus is too big too effectively store or search, and so we exclude overlaps with a count of less than 5 from our lattice.
- Many nodes have a minLPR which is a little larger than 1. There is very little chance these nodes will be activated by the algorithm, and so to save time we simply do not consider activating nodes with a minLPR of less than 2.