Appendix 3: Computational Complexity of Concept Matching

Two steps tend to be common to all concept-matching approaches. First, one generates combinations of subsets of words in a block (i.e., sentence or phrase) of text. Next, for each combination, one searches the thesaurus for terms containing this subset of words. We now analyze the complexity of each step.

Generating Combinations

For a text block with N non-stop-words, one first tries to locate the concept/s containing all N words. It is desirable to locate the most specific matches, if they exist, e.g., given the phrase high blood pressure , one should attempt to match hypertension rather than high , blood and pressure separately.

If no matches are found, then one looks for concepts containing all combinations of N-1 words, and so on, finally looking for concepts matching individual words. The number of combinations for each possibility is the coefficient of the standard binomial expansion to the Nth power (26), and the sum of these combinations is equal to 2^N— 1. That is, in the worst case, this step is exponentially complex.

In our test medical narrative data, the median number of non-stop-words per sentence was six. However, 10% of the sentences had 13 or more non-stop-words, 5% had 16 or more, and 1% had 25 or more. It is clear that if we process all the words in a sentence together, time requirements might be prohibitive, because a few pathologically long sentences will dominate processing time. More important, much of the processing time will have been wasted: for instance, one is hardly likely to find a concept that contains all of 13 or more non-stop words, because a single sentence will typically contain multiple

independent concepts. To make the problem tractable, we have to break the sentences into smaller units using some form of syntactic analysis, as phrase-based approaches do.

Thesaurus Search

A thesaurus such as the UMLS is large enough to require management through a relational database management system (RDBMS). In analyzing the complexity of thesaurus search, we make the simplifying assumption that the number of concepts containing a single word is roughly proportional to the number of thesaurus entries. This assumption, while obviously false for uncommon words such as systematic chemical names that match only a single concept, is possibly true for most words encountered in medical narrative. The 1999 UMLS contains around 700,000 concepts, a million terms and three million unique words. While the average number of concepts per word is 14.7, the median is one, the upper quartile three, and the upper decile ten. The reason for the skew is that certain common words occur many times. Thus, spinal occurs in 5,000 concepts, and penicillin in 300 concepts. Stop-words are far more frequent than this, but obviously we do not use them.

To find all concepts that contain all of N non-stop-words, the RDBMS must find the intersection of the individual sets of concepts that match each word. Intersections are typically performed through join operations, and for N words, N-1 consecutive joins are needed. RDBMSs can choose between several join algorithms based on the query and data distribution. The complexity of these algorithms is described in (27). If two individual sets return r and *s* concepts respectively, then a sophisticated algorithm like merge join , takes time proportional to $(r+s)* log (r+s)$, while a simpler algorithm like nested-loops join takes time proportional to $r * s$. If both r and s are proportional to T,

the total number of terms in the thesaurus, the time complexity for all N-1 joins ranges from $N*T*\log(T)$ to $N*T^2$.

From the above analysis, it pays to restrict both N (through a phrase-based approach) and T (by reducing the number of concepts if possible). Greatly reduced subsets of the UMLS have been successfully used to identify trauma conditions (28) and dosage information (22). Indeed, such subsets may fit entirely in RAM, allowing set intersection through computational strategies that are dramatically more efficient than disk-based joins. For example, one may employ bit-vectors, as used by Miller and Masarie in QMR (29).