A Weighted O*NET Keyword Search (WWS)

Prepared for: National Center for O*NET Development

Author: Jeremiah Morris
jm@whpress.com

Date: August 22, 2013

Updated February 16, 2021

History of the keyword search

A keyword search is an important option for customers engaged in career exploration. Access to a search allows them to use titles or activities to generate a targeted list of occupations to potentially explore. Many general-purpose algorithms exist for searching large text documents; however, the smaller, highly structured text items included in the O*NET database call for different search techniques.

Several approaches have been used to produce a keyword search for finding O*NET occupations; this document outlines the past approaches and presents the latest algorithm used within the My Next Move, O*NET OnLine, and O*NET Code Connector applications. As with all O*NET products, the keyword search is subject to a continuous improvement philosophy, with search results and customer feedback used to drive potential future enhancements.

Thematic search

The initial "thematic search" was developed under a separate grant for the U. S. Department of Labor (DOL). The "thematic search" served as the basis for keyword search implementations in O*NET's internet applications, O*NET OnLine and O*NET Code Connector, as well as DOL's Career InfoNet. It separated the text items associated with an occupation into thematic rings, and assigned a score to each 5 concentric rings. The occupational title occupied the center ring of the thematic search and received the highest score, followed by O*NET alternate titles, occupational description, task statements, and detailed work activities (formerly labor exchange skills). The count of items matching was multiplied by the ring score, and this sum became the raw score for each occupation.

User input was treated as a Boolean query; operators such as AND, OR, and NOT were accepted, and double-quoted strings were used to form exact multi-word terms. By default, terms were joined with AND, and an implicit wildcard was added to each term. The query had to exactly match each item in order to count toward the results: the search "heart surgeon" would not match the title "Surgeons" or the task "Operates on a patient's heart."

One drawback of this approach quickly became obvious soon after the initial release: O*NET OnLine's top unmatched term was "docter". Besides the inability to handle misspellings, the thematic search could not cope with variants of a word; "secretary" returned significantly different results versus "secretaries". The alternate titles database was expanded to include common errors and variants, but this took significant effort and aided only a fraction of the queries.

Spell-check improvement

In the O*NET Center's initial attempt to address the spelling issue, the user input string was corrected by a spell checker prior to running the search. The dictionary used by the spell checker was comprised only of words from the database; this allowed it to cope with a few word variants. For example, if the word "endocrinology" was not in the database, the spell checker could suggest the word “endocrinologist” instead, allowing a match without the user needing to try variations themselves.

This alleviated problems such as the “docter” query, but it introduced a new problem for the advanced users. Professionals had come to rely on the wildcard function to input partial words, but the spell checker turned “info tech” into “into teach”. The spell check was moved, so that the corrected query would be used only if the user’s original query returned no results.

The competing interests of the casual, misspelling OnLine user, and the more search-savvy Code Connector user prompted much discussion. Checking the search logs, neither group made use of the Boolean queries which could have clarified the user’s intent and allowed greater flexibility of results. A menu with query options was proposed, but eventually a new solution for interpreting the user input was developed.

Tiered search

In the next iteration, the core of the thematic search was kept, but the input was interpreted in four different ways, forming four “tiers” of possible meaning:

- The first tier treated the input as a complete phrase. The query “medical secretary” would not match “Secretary, Medical” or “medical secretaries”.
- The second tier allowed the words to occur in any order, and also matched words with a similar stem (using the Porter stemming algorithm). The query “medical secretary” would be similar to the Boolean query “(medical OR medic OR medicine) AND (secretary OR secretaries OR secretarial)”.
- The third tier was similar to the second, but it added spell-checking suggestions for each term, and a wildcard for partial word searches. A match on all of the terms was still required.
- The fourth tier was identical to the third, but it allowed a match on any of the words, as if they were joined by a Boolean OR instead of AND.

For each of these tiers, a thematic search was performed, scoring the results according to the thematic rings. Each tier had its own score, which was multiplied with the thematic score for that tier. Results from all tiers were added together to arrive at the final relevance scores.

This search was a major step forward, providing results in many cases where the thematic search would fail. However, the tiered search had its own flaws. The spell-checking method, inherited from the earlier search, would give confusing results on words unrelated to any in the O*NET database; the suggestion for “aardvark” was “artwork”, for example. A large class of multi-word queries would perform poorly, given the single tier of any-word searching; a query like “garbage men” would not match on any tier but the fourth, where a “garbage collector” match on the first term would be drowned by wildcard results for “mental health professional” and other matches on the second term.

Weighted search

The weighted search was developed to perform well on the areas neglected by the tiered search, and to reduce the processing time (a tiered search took several seconds to run in many cases). The search was developed experimentally, using an evaluation process to gauge the effectiveness across a set of several dozen queries. The final algorithm treats each query term individually, adding the results at the end to achieve the total score.

The most unique feature of the weighted search is the "weight" assigned to each query term. Based on the query "garbage men", the term "garbage" receives a high multiplier as it matches only two occupations, while the term "men" receives a low multiplier as it matches over a hundred occupations. This allows the two results referencing "garbage" to be scored over the more plentiful but less relevant results for the second term. The thematic rings and input tiers survive in a modified form in this search, so the frequency weights form the third dimension of scoring a match.

A number of other improvements were added to the weighted search. The Porter stemming algorithm (http://www.tartarus.org/~martin/PorterStemmer/) was replaced by the Paice/Husk algorithm (http://www.comp.lancs.ac.uk/computing/research/stemming/) which catches more word relations such as "science" and "scientist". The custom dictionary was dropped in favor of a standard English wordlist, since the addition of stemming obviated the initial need for the custom wordlist. The thematic ring scores and limits were modified slightly; alternate titles were given increased importance, while tasks and detailed work activities were limited to avoid obscuring higher results with a large number of low-scoring matches. Stopwords (common words like "the" or "it"), which were always stripped in the tiered search, are now used in searching titles and alternate titles, so that alternate titles such as "after school caretaker" or "IT manager" will contribute to results.

Weighted search enhancement

After the initial development and release of the weighted search, an "exact match" feature was introduced to improve the results of specific queries. Queries which exactly match an occupation title or alternate title are forced to the top of the result list. This guarantees that, if an alternate title is used as a query, all occupations linked to that alternate title will be listed before other occupations.

In 2009, the "exact match" feature was extended with a database of singular "equivalent" variants of the plural O*NET-SOC titles. The occupation titles and singular equivalents are listed before alternate title exact matches. As an example, the search "carpenter" will return the O*NET-SOC title "Carpenters" as the first match, even if the alternate title "Carpenter" is attached to several occupations.

In 2017, Hot Technologies were added to the "exact match" feature. These software products and programming languages are frequently included in job postings. Searches for "SAP Crystal Reports" or "Python" return relevant occupations.

In 2021, occupation titles from previous O*NET-SOC taxonomies were added to the “exact match” feature. This ensures that a search for any historical O*NET-SOC title will always return relevant occupations in the latest O*NET-SOC taxonomy.

The weighted search implementations on the O*NET websites are kept up to date with the latest alternate titles, task statements, Detailed Work Activities, Hot Technologies and O*NET-SOC taxonomy information. While the algorithm itself changes infrequently, search results are improved through these data updates at least once per year.

The weighted search algorithm

From the initial user input, periods are removed, and any characters besides letters and numbers are treated as word separators. All searching is case insensitive, so the normalized query can be seen as a list of words containing only lowercase letters or numbers.

The searchable content of the database remains the same as in the earlier searches; each occupation is associated with a title, a description, and zero or more alternate titles, tasks, and detailed work activities (see Table 1).

For each unique word in the search query, a “word score” is calculated on a per-occupation basis as follows:

- The word is exactly matched against the content items of the database. The matches in each thematic ring for each occupation are tallied; the count of alternate title matches is limited to 1, and the number of task or DWA matches is limited to 5. If the word matches a known stopword, all matches on the description, tasks, and DWAs are discarded. The final counts are then multiplied by the ring value (Table 1) and tier value (Table 2). For exact matches, the tier value is 4.
- The word is then stemmed, using the Paice/Husk stemming algorithm, and compared to the stems of the database content. The matches are tallied and multiplied as above; the stemmed tier value is 4.
- The word is matched as a substring; any word in the database beginning with the letters of the current word is considered a match. The matches are tallied and multiplied as above; the substring tier value is 2.

After these steps are applied, the implementation has a set of matching occupations and a word score for each occupation. The number of matching occupations is used to calculate a word frequency factor; this factor varies between 64 and 1. Table 3 shows the factors used. The word score is multiplied by the frequency factor to get a weighted occupation score for that word. The final raw score of an occupation is the sum of these weighted word scores.

If any words are misspelled, the process above is repeated for each unique spelling suggestion. The spelling suggestions have their own tier values, as shown in the chart below; note that the spelling substring tier value is zero, so that step may be skipped as it does not affect the score.

Once all words and spelling suggestions have been processed, the “exact match” phases are processed. In each of the three phases, matching occupations are moved to the top of the results:

1. Hot Technologies and variants: Programming languages such as “Ruby”, or specific software products such as “Microsoft Project”, are matched to relevant occupations. The technologies included in the search are frequently included in employer job postings.
2. Alternate titles: Exact matches on a title in our alternate titles database will always be listed in the results before partial matches using the same words.
3. Occupation titles and variants: This phase comes last, so that an occupation title exact match will be shown ahead of an alternate title exact match.

In each phase, the entire normalized query is checked against a normalized set of titles. If the query exactly matches, the score for the matching occupation(s) is adjusted as follows: the occupation’s raw score is divided by 10, and the maximum raw score from the previous steps is added. Thus, each exact match’s raw score is slightly higher than any previous score. When the scores are sorted, the exact matches will rank above all non-exact matches.

Finally, the scores can be normalized and sorted for display to the user. The raw scores are divided by the largest raw score and then multiplied by 100 to obtain the normalized score. The normalized scores are shown in O*NET OnLine and O*NET Code Connector. My Next Move does not display the scores, but uses them for sorting the search results.

My Next Move is also unique in that it only presents data-level O*NET-SOC occupations. The search itself is unchanged; occupations that aren’t part of the O*NET data collection plan are filtered out of the displayed result set.

Table 1 - Ring weights and filters

Thematic ring	Ring weight	Maximum count	Stopwords allowed
Occupation title	16	1	yes
Alternate titles	16	1	yes
Description	8	1	no
Tasks	2	5	no
DWAs	1	5	no

Table 2 - Tier weights

Search tier	Tier weight
exact word	4
stemmed word	4
substring	2
exact word (spelling suggestion)	2
stemmed word (spelling suggestion)	2
substring (spelling suggestion)	0

Table 3 - Word frequency factors

Minimum number of matching occupations	Maximum number of matching occupations	Frequency factor
1	4	64
5	9	32
10	24	16
25	49	8
50	99	4
100	399	2
400	n/a	1

Performance optimizations

To provide our audience with keyword search results in a timely fashion, an efficient implementation of this algorithm is important. The O*NET Center's approach uses a relational database for storage and lookups, combined with a scripting language for input normalization and coordination of the database lookups. Improvements to this implementation include:

- **User query caching:** For our traffic patterns, the "top queries" change frequently, but any particular query has a significant change of being requested again within a short period. A classroom setting may potentially lead to the same search request from multiple students' computers, for example. We employ least-recently-used (LRU) caching to retain the full results of recent searches, skipping the algorithm calculations entirely when a repeat query is seen.

- **Word score caching:** Within the "word score" phase, the list of occupations and raw scores for a given word is independent of the rest of the query. This means commonly used words can be pre-calculated, and the scores can be stored for later use. We pre-calculate scores for each of the 1.2 million normalized words in our occupation data, which covers a large percentage of words seen in user queries.

- **Database structure and indexing:** As with any application relying on a relational database management system (RDBMS), table and index creation is very important. We have made optimizations by following general guidelines for RDBMS optimization:
  - Creating table structures to minimize the number of JOIN operations needed
  - Designing queries to return only data necessary for later steps
  - Combining and calculating data within the database, instead of retrieving and calculating in code
  - Adding indexes to cover all queries made by the application

For optimal performance, the most important lessons we have learned are to search for performance bottlenecks at all levels of your implementation, and to carefully consider the queries and patterns you see from your particular audience.

Evaluation of the weighted search

The accuracy of the weighted search was evaluated using a procedure developed by the O*NET Center, based on the position and score of a target "best" occupation for a given query. Using this procedure on a set of sample O*NET application queries, the weighted search scored 95.9% versus the previous tiered search score of 67.5%. The weighted search did not match the tiered search performance for every query, but generally the regressions were minor (the target dropped from first to third) while the advances were significant (the target rose from fortieth to first).

The search was also evaluated on a set of analyst-coded job orders, with 529 job titles tested. The thematic search scored 12.4%, the tiered search scored 58.8%, and the weighted search scored 67.3%. It is important to note that the analysts used job descriptions (which include valuable information) in addition to job titles, accounting for some of the differences between the three search scores and the analysts coded jobs. (Scores were lower as the coding took job descriptions into account, which often included vital information not available in the title.)

The weighted search was added to O*NET OnLine and O*NET Code Connector in September 2005. In 2009, the "exact match" phases were added, which improve many specific queries. The search has been used in My Next Move since its launch in 2011. While the Center is always looking for further improvements, the weighted search is the best tool yet developed for providing O*NET occupational keyword results.