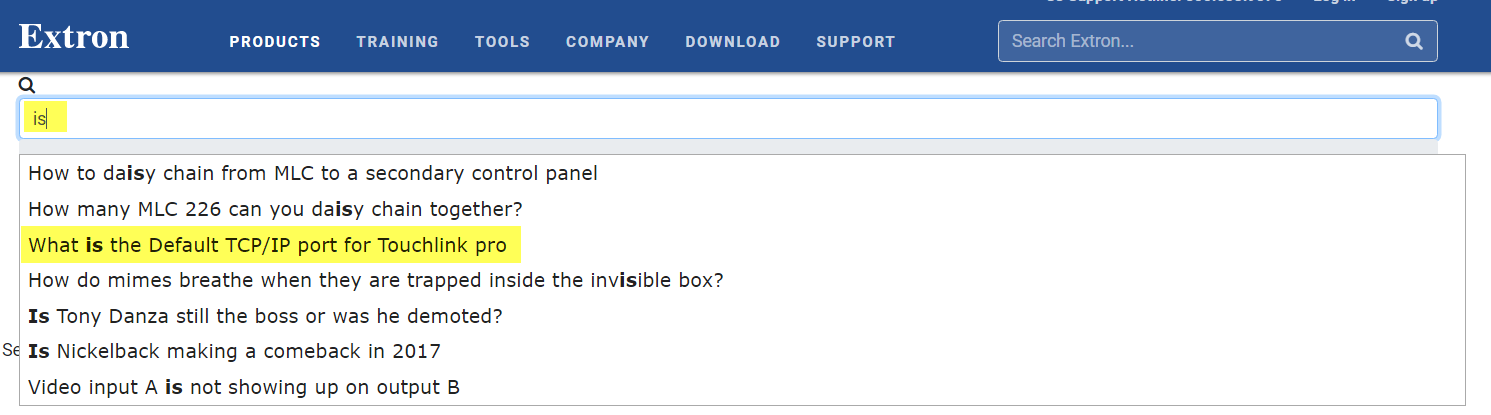
Empirical Search Suggestions

for Support Knowledgebases



A Search Autocomplete is a powerful tool which allows users to quickly find established solution routes within a knowledgebase. It also provides Stakeholders valuable information on what their userbase is searching. By storing what the user is searching, we can monitor trends on support issues and use the data to make further improvements on products. In this article, I will be describing the conceptual implementation of a simple Search Autocomplete system within the context of support knowledgebases.

Requirements & Expectations:

* Implement a system that will provide Search Suggestions to users
* Search suggestions should also be user friendly in that it can handle casual user input and finds “Most Likely” matches. It should also handle “Typos”
* Track and record every raw input that a user inputs into the Search Bar
  + Optional: Only track raw inputs that result in a Search Result “Click”
* Analyze the raw input data to compile an “Active Suggestion List” of frequently entered Search Strings
* Automate a daily service that will update the Active Suggestion List based on real-time data.
* Allow Content Managers to manually add custom “Search Suggestions” independent of the Automated Services.

Capturing a Successful Search:

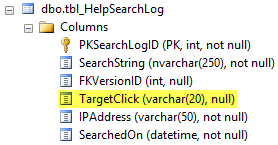
A successful search can be measured in a multitude of ways, but for the sake of rolling something out quickly (Phase 1 of the project), we can track a successful search by:

* If a user performs an initial search and clicks on a result within the first page (first page: 10 records or less)

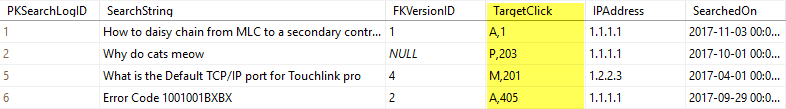
With this method, we are not focused on chasing the user all the way to the end where we can know for sure if they were able to find their answer; We are more focused with the objective of populating a list of relevant and trending Search Strings.

We can store these search strings within a SQL table via a simple clean INSERT statement without any Subqueries or in-line use of the WHERE clause. This is to ensure that search performance is not affected when the table becomes bulky.

Step 1: Create a table in your database which will store incoming search requests.



I have highlighted in yellow the “TargetClick” column which I intend to store what user clicked on within the result set. If your search system only returns Articles as results then it’s easy just to store an int ID value in this column, however for the sake of my Support Knowledgebase, We need to track what kind of link it is, as well as the ID column so I have made it a VARCHAR(20) in data type.



Above are some examples of what table data would look like. Highlighted: TargetClick data will be comma delimited. The first character represents the **Type** of link, and the second represents the **ID**.

The importance of capturing a Target Click for the scope of this article is so that A: We know what the user clicked on within results, B: We can associate similar Search Strings to a specific Target Click (This can help reduce redundancy later on).

If we have the following Search Strings below:

*“What is the default port of Touchlink Pro?”*

*“Default port of Touchlink pro”*

*“Default port of Touchlink pro v. 2016”*

And they all point to a Target Click of: **Article ID 603**. Then we can further scrutinize the 3 and find which is the most popular which will make it to the Active Suggestions List. By using an