Add scoring profiles to boost search scores

Scoring profiles are used to boost the ranking of matching documents based on criteria. In this article, learn how to specify and assign a scoring profile that boosts a search score based on parameters that you provide.

You can use scoring profiles for keyword search, vector search, and hybrid search. However, scoring profiles only apply to nonvector fields, so make sure your index has text or numeric fields that can be used in a scoring profile.

Prerequisites

- A new or existing search index with text or numeric fields.

You can specify a scoring profile in the index designer in the Azure portal or through APIs like Create or Update Index REST or equivalent APIs in any Azure SDK.

Scoring profile support for vector and hybrid search is available in 2024-05-01-preview and 2024-07-01 REST APIs and in Azure SDK packages that targeting those releases.

Key points about scoring profiles

Scoring profile parameters are either:

- Weighted fields, where a match is found in a specific string field. For example, you might want matches found in a "summary" field to be more relevant than the same match found in a "content" field.
- Functions for numeric data, including dates, ranges, and geographic coordinates. There's also a Tags function that operates on a field providing an arbitrary collection of strings. You can choose this approach over weighted fields if you want to boost a score based on whether a match is found in a tags field.

You can create multiple profiles and then modify query logic to choose which one is used.

You can have up to 100 scoring profiles within an index (see service Limits), but you can only specify one profile at time in any given query.

You can use semantic ranker with scoring profiles. When multiple ranking or relevance features are in play, semantic ranking is the last step. How search scoring works provides an illustration.

> [!NOTE] Unfamiliar with relevance concepts? Visit Relevance and scoring in Azure AI Search for background. You can also watch this video segment on YouTube for scoring profiles over BM25-ranked results.

Scoring profile definition

A scoring profile is named object defined in an index schema. A scoring profile is composed of weighted fields, functions, and parameters.

The following definition shows a simple profile named "geo". This example boosts results that have the search term in the hotelName field. It also uses the `distance` function to favor results that are within 10 kilometers of the current location. If someone searches on the term 'inn', and 'inn' happens to be part of the hotel name, documents that include hotels with 'inn' within a 10 KM radius of the current location will appear higher in the search results.

```
"scoringProfiles": [
  {
    "name":"geo",
    "text": {
      "weights": {
        "hotelName": 5
      }
    },
    "functions": [
      {
        "type": "distance",
        "boost": 5,
        "fieldName": "location",
        "interpolation": "logarithmic",
        "distance": {
          "referencePointParameter": "currentLocation",
          "boostingDistance": 10
        }
      }
    ]
  }
]
```

To use this scoring profile, your query is formulated to specify scoringProfile parameter in the request. If you're using the REST API, queries are specified through GET and POST requests. In the following example, "currentLocation" has a delimiter of a single dash (`-`). It's followed by longitude and latitude coordinates, where longitude is a negative value.

```
GET /indexes/hotels/docs?
search+inn&scoringProfile=geo&scoringParameter=currentLocation-
-122.123,44.77233&api-version=2024-07-01
```

Notice the syntax differences when using POST. In POST, "scoringParameters" is plural and it's an array.

```
POST /indexes/hotels/docs&api-version=2024-07-01
{
    "search": "inn",
    "scoringProfile": "geo",
    "scoringParameters": ["currentLocation--122.123,44.77233"]
}
```

This query searches on the term "inn" and passes in the current location. Notice that this query includes other parameters, such as scoringParameter. Query parameters, including "scoringParameter", are described in Search Documents (REST API).

See the Extended example for vector and hybrid search and Extended example for keyword search for more scenarios.

How search scoring works in Azure AI Search

Scoring profiles supplement the default scoring algorithm by boosting the scores of matches that meet the profile's criteria. Scoring functions apply to:

- Text (keyword) search
- Pure vector queries
- Hybrid queries, with text and vector subqueries execute in parallel

For standalone text queries, scoring profiles identify the maximum 1,000 matches in a BM25-ranked search, and the top 50 are returned in results.

For pure vectors, the query is vector-only, but if the *k*-matching documents include nonvector fields with human-readable content, a scoring profile can be applied. The scoring profile revises the result set by boosting documents that match criteria in the profile.

For text queries in a hybrid query, scoring profiles identify the maximum 1,000 matches in a BM25-ranked search. However, once those 1,000 results are identified, they're restored to their original BM25 order so that they can be rescored alongside vectors results in the final Reciprocal Ranking Function (RRF) ordering, where the scoring profile (identified as "final document boosting adjustment" in the illustration) is applied to the merged results, along with vector weighting, and semantic ranking as the last step.

Add a scoring profile to a search index

1. Start with an index definition. You can add and update scoring profiles on an existing index without having to rebuild it. Use an Create or Update Index request to post a revision.

2. Paste in the template provided in this article.

3. Provide a name that adheres to naming conventions.

4. Specify boosting criteria. A single profile can contain text weighted fields, functions, or both.

You should work iteratively, using a data set that will help you prove or disprove the efficacy of a given profile.

Scoring profiles can be defined in Azure portal as shown in the following screenshot, or programmatically through REST APIs or in Azure SDKs, such as the ScoringProfile class in the Azure SDK for .NET.

Use text-weighted fields

Use text-weighted fields when field context is important and queries include `searchable` string fields. For example, if a query includes the term "airport", you might want "airport" in the Description field to have more weight than in the HotelName.

Weighted fields are name-value pairs composed of a `searchable` field and a positive number that is used as a multiplier. If the original field score of HotelName is 3, the boosted score for that field becomes 6, contributing to a higher overall score for the parent document itself.

```
"scoringProfiles": [
    {
      "name": "boostSearchTerms",
      "text": {
        "weights": {
          "HotelName": 2,
          "Description": 5
        }
      }
    }
]
```

Use functions

Use functions when simple relative weights are insufficient or don't apply, as is the case of distance and freshness, which are calculations over numeric data. You can specify multiple functions per scoring profile. For more information about the EDM data types used in Azure AI Search, see Supported data types.

| Function | Description | Use cases | |-|-|-| | distance | Boost by proximity or geographic location. This function can only be used with `Edm.GeographyPoint` fields. | Use for "find near me" scenarios. | | freshness | Boost by values in a datetime field (`Edm.DateTimeOffset`). Set boostingDuration to specify a value representing a timespan over which boosting occurs. | Use when you want to boost by newer or older dates. Rank items like calendar events with future dates such that items closer to the present can be ranked higher than items further in the future. One end of the range is fixed to the current time. To boost a range of times in the past, use a positive boostingDuration. To boost a range of times in the future, use a negative boostingDuration. | | magnitude | Alter rankings based on the range of values for a numeric field. The value must be an integer or floating-point number. For star ratings of 1 through 4, this would be 1. For margins over 50%, this would be 50. This function can only be used with `Edm.Double` and `Edm.Int` fields. For the magnitude function, you can reverse the range, high to low, if you want the inverse pattern (for example, to boost lower-priced items more than higher-priced items). Given a range of prices from $100 to $1, you would set `boostingRangeStart` at 100 and `boostingRangeEnd` at 1 to boost the lower-priced items. | Use when you want to boost by profit margin, ratings, clickthrough counts, number of downloads, highest price, lowest price, or a count of downloads. When two items are relevant, the item with the higher rating will be displayed first. | | tag | Boost by tags that are common to both search documents and query strings. Tags are provided in a `tagsParameter`. This function can only be used with search fields of type `Edm.String` and `Collection(Edm.String)`. | Use when you have tag fields. If a given tag within the list is itself a comma-delimited list, you can use a text normalizer on the field to strip out the commas at query time (map the comma character to a space). This approach will "flatten" the list so that all terms are a single, long string of comma-delimited terms. |

Rules for using functions

- Functions can only be applied to fields that are attributed as `filterable`.

- Function type ("freshness", "magnitude", "distance", "tag") must be lower case.
- Functions can't include null or empty values.
- Functions can only have a single field per function definition. To use magnitude twice in the same profile, provide two definitions magnitude, one for each field.

Template

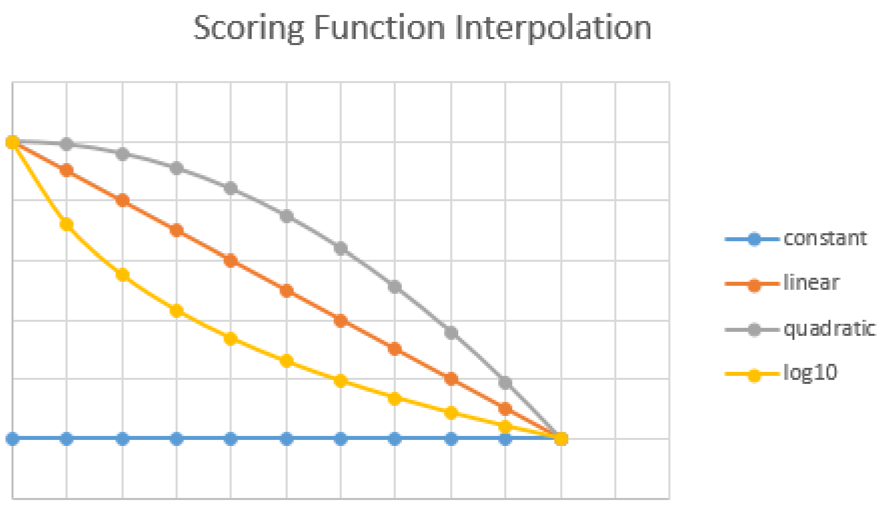
This section shows the syntax and template for scoring profiles. For a description of properties, see the REST API reference.

```
"scoringProfiles": [
  {
    "name": "name of scoring profile",
    "text": (optional, only applies to searchable fields) {
      "weights": {
        "searchable_field_name": relative_weight_value (positive #'s),
        ...
      }
    },
    "functions": (optional) [
      {
        "type": "magnitude | freshness | distance | tag",
        "boost": # (positive number used as multiplier for raw score != 1),
        "fieldName": "(...)",
        "interpolation": "constant | linear (default) | quadratic | logarithmic",

        "magnitude": {
          "boostingRangeStart": #,
          "boostingRangeEnd": #,
          "constantBoostBeyondRange": true | false (default)
        }

        // ( - or -)

        "freshness": {
          "boostingDuration": "..." (value representing timespan over which
boosting occurs)
        }

        // ( - or -)

        "distance": {
          "referencePointParameter": "...", (parameter to be passed in queries to
use as reference location)
          "boostingDistance": # (the distance in kilometers from the reference
location where the boosting range ends)
        }

        // ( - or -)

        "tag": {
          "tagsParameter":  "..."(parameter to be passed in queries to specify a
list of tags to compare against target field)
        }
      }
    ],
    "functionAggregation": (optional, applies only when functions are specified)
"sum (default) | average | minimum | maximum | firstMatching"
  }
],
"defaultScoringProfile": (optional) "...",
```

Set interpolations

Interpolations set the shape of the slope used for scoring. Because scoring is high to low, the slope is always decreasing, but the interpolation determines the curve of the downward slope. The following interpolations can be used:

Interpolation	Description
`linear`	For items that are within the max and min range, boosting is applied in a constantly decreasing amount. Linear is the default interpolation for a scoring profile.
`constant`	For items that are within the start and ending range, a constant boost is applied to the rank results.
`quadratic`	In comparison to a linear interpolation that has a constantly decreasing boost, Quadratic initially decreases at smaller pace and then as it approaches the end range, it decreases at a much higher interval. This interpolation option isn't allowed in tag scoring functions.
`logarithmic`	In comparison to a linear interpolation that has a constantly decreasing boost, logarithmic initially decreases at higher pace and then as it approaches the end range, it decreases at a much smaller interval. This interpolation option isn't allowed in tag scoring functions.



Set boostingDuration for freshness function

`boostingDuration` is an attribute of the `freshness` function. You use it to set an expiration period after which boosting will stop for a particular document. For example, to boost a product line or brand for a 10-day promotional period, you would specify the 10-day period as "P10D" for those documents.

`boostingDuration` must be formatted as an XSD "dayTimeDuration" value (a restricted subset of an ISO 8601 duration value). The pattern for this is: "P[nD][T[nH][nM][nS]]".

The following table provides several examples.

Duration	boostingDuration
1 day	"P1D"
2 days and 12 hours	"P2DT12H"
15 minutes	"PT15M"
30 days, 5 hours, 10 minutes, and 6.334 seconds	"P30DT5H10M6.334S"

For more examples, see XML Schema: Datatypes (W3.org web site).

Extended example for vector and hybrid search

See this blog post and notebook for a demonstration of using scoring profiles and document boosting in vector and generative AI scenarios.

Extended example for keyword search

The following example shows the schema of an index with two scoring profiles (`boostGenre`, `newAndHighlyRated`). Any query against this index that includes either profile as a query parameter will use the profile to score the result set.

The `boostGenre` profile uses weighted text fields, boosting matches found in albumTitle, genre, and artistName fields. The fields are boosted 1.5, 5, and 2 respectively. Why is genre boosted so much higher than the others? If search is conducted over data that is somewhat homogenous (as is the case with 'genre' in the musicstoreindex), you might need a larger variance in the relative weights. For example, in the musicstoreindex, 'rock' appears as both a genre and in identically phrased genre descriptions. If you want genre to outweigh genre description, the genre field will need a much higher relative weight.

```
{
  "name": "musicstoreindex",
  "fields": [
    { "name": "key", "type": "Edm.String", "key": true },
    { "name": "albumTitle", "type": "Edm.String" },
    { "name": "albumUrl", "type": "Edm.String", "filterable": false },
    { "name": "genre", "type": "Edm.String" },
    { "name": "genreDescription", "type": "Edm.String", "filterable": false },
    { "name": "artistName", "type": "Edm.String" },
    { "name": "orderableOnline", "type": "Edm.Boolean" },
    { "name": "rating", "type": "Edm.Int32" },
    { "name": "tags", "type": "Collection(Edm.String)" },
    { "name": "price", "type": "Edm.Double", "filterable": false },
    { "name": "margin", "type": "Edm.Int32", "retrievable": false },
    { "name": "inventory", "type": "Edm.Int32" },
    { "name": "lastUpdated", "type": "Edm.DateTimeOffset" }
  ],
  "scoringProfiles": [
    {
      "name": "boostGenre",
      "text": {
        "weights": {
          "albumTitle": 1.5,
          "genre": 5,
          "artistName": 2
        }
      }
    },
    {
      "name": "newAndHighlyRated",
      "functions": [
        {
          "type": "freshness",
          "fieldName": "lastUpdated",
          "boost": 10,
          "interpolation": "quadratic",
          "freshness": {
            "boostingDuration": "P365D"
          }
        },
        {
          "type": "magnitude",
          "fieldName": "rating",
          "boost": 10,
          "interpolation": "linear",
          "magnitude": {
            "boostingRangeStart": 1,
            "boostingRangeEnd": 5,
            "constantBoostBeyondRange": false
          }
        }
      ]
    }
  ],
  "suggesters": [
    {
      "name": "sg",
      "searchMode": "analyzingInfixMatching",
      "sourceFields": [ "albumTitle", "artistName" ]
    }
  ]
}
```

See also

- Relevance and scoring in Azure AI Search
- REST API Reference

- Create Index API
- Azure AI Search .NET SDK
- What Are Scoring Profiles?