Azure AI Search でのセマンティック ランク付け

[アーティクル] • 2024/02/08

Azure AI Search のセマンティック ランク付けでは、検索結果を再ランク付けするために、言語理解を利用して検索結果の関連性をある程度高めます。 この記事では、概要について説明します。 最後のセクションでは、販売状況と価格について説明します。

セマンティック ランカーはプレミアム機能であり、使用量に基づいて課金されます。 最初にこの記事を読んで基礎知識を得ることをお勧めしますが、それはかまわないのですぐに使いたいという方は、これらの手順を実行してください。

- ✓ リージョン別の提供状況を確認する
- ✓ Azure portal にサインインして、検索サービスが Basic 以上であることを確認する
- ✓ セマンティック ランク付けを有効にして価格プランを選択する
- ✓ 検索インデックスでセマンティック構成を設定する
- ✓ セマンティック キャプションとハイライトを返すようにクエリを設定する
- ✓ 必要に応じて、セマンティック回答を返す

> ⓘ **注意**
>
> セマンティック ランク付けでは、生成 AI やベクトルは使用されません。 ベクトルのサポートと類似性検索をお探しの場合、 「**Azure AI Search でのベクトル検索**」で詳細をご確認いただけます。

セマンティック ランク付けとは

セマンティック ランカーは、テキストベースのクエリに対する、BM25 でランク付けされたあるいは RRF でランク付けされた最初の検索結果の品質を高めるクエリ関連機能のコレクションです。 これを検索サービスで有効にすると、セマンティック ランク付けによってクエリの実行パイプラインに 2 つの機能が追加されます。

- 1 つ目として、BM25 または RRF を使用してスコア付けされた、最初の結果セットに対する二次ランク付けが追加されます。 この二次ランク付けでは多言語の、Microsoft Bing から変化したディープ ラーニング モデルが使用されて、セマンティック的に最も関連性の高い結果が奨励されます。

- 2 つ目として、キャプションと回答が抽出されて応答で返されます。これを検索ページにレンダリングして、ユーザーの検索エクスペリエンスを向上させることができます。

セマンティック リランカーの機能を次に示します。

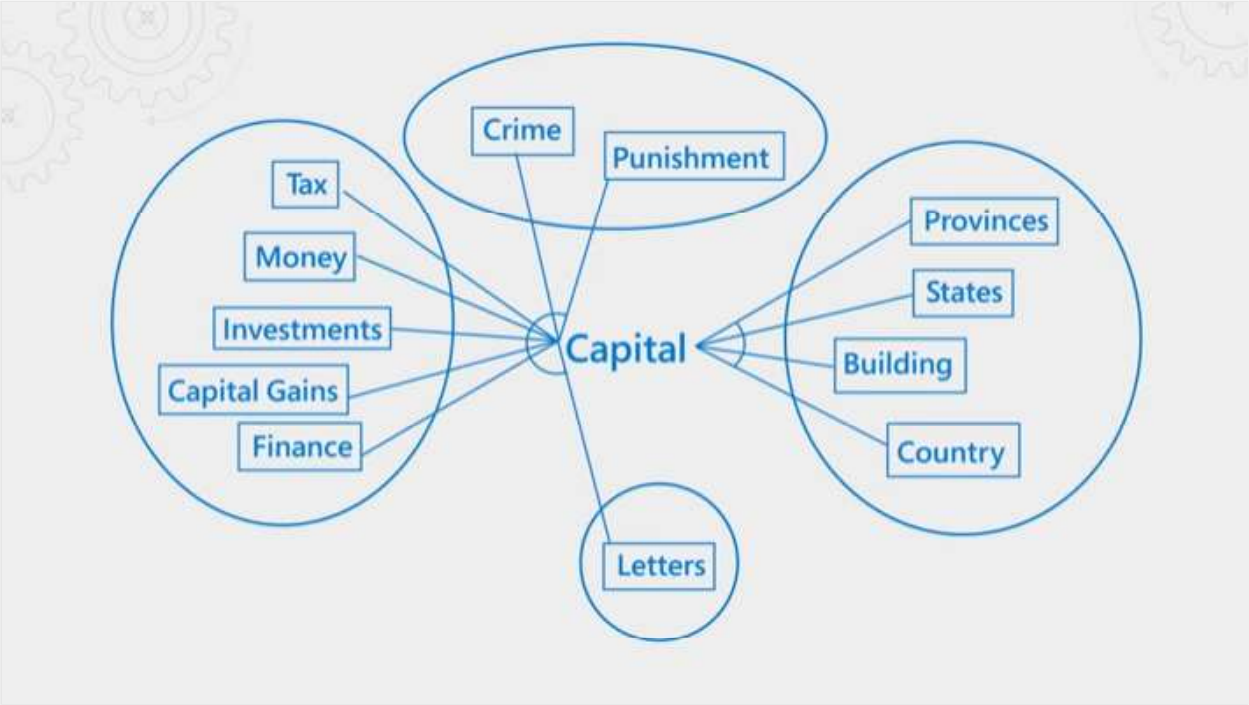
テーブルを展開する

特徴量	説明
セマンティック ランク付け	クエリのコンテキストまたはセマンティックの意味を利用して、事前にランク付けされた結果に対して新しい関連スコアを計算します。
セマンティック キャプションとハイライト	コンテンツを最もよく要約している逐語的な文やフレーズをドキュメントから抽出し、スキャンを簡単にするために重要な部分を強調表示します。 結果を要約するキャプションは、個々のコンテンツ フィールドが検索結果ページに対して高密度である場合に便利です。 強調表示されたテキストにより、最も関連性の高い用語とフレーズが目立つため、ユーザーはその一致が関連していると見なされた理由を迅速に判断できます。
セマンティック回答	セマンティック クエリから返される省略可能な追加のサブ構造体。 これにより、質問のようなクエリに直接回答することができます。 ドキュメントには、回答の特性を持つテキストが含まれている必要があります。

セマンティック ランカーのしくみ

セマンティック ランク付けは、クエリと結果を Microsoft がホストする言語理解モデルにフィードし、より適切な一致をスキャンします。

コンセプトを次の図で説明します。 "capital" という用語を考えてみましょう。 これは、コンテキストが財務、法律、地理、文法なのかによって意味が異なります。 言語理解では、セマンティックのランカーによってコンテキストが検出され、クエリの意図に沿う結果が昇格されます。



セマンティックの順位付けは、リソースと時間の両方を消費します。 クエリ操作の予想待機時間内に処理を完了するために、セマンティック ランク付けへの入力が統合され、削減されるため、再ランク付けの手順をできるだけ早く完了できます。

セマンティック ランク付けには、サマライゼーションとスコアリングという 2 つの手順があります。 出力が再スコアリングされた結果、キャプション、および回答で構成されます。

入力が収集されて要約されるしくみ

セマンティック ランク付けでは、クエリのサブシステムからサマライゼーション モデルとランク付けモデルに検索結果が入力として渡されます。 ランク付けモデルには入力サイズに制約があり、集中的に処理されるため、検索結果は効率的に処理できるようなサイズで構造化 (要約) されている必要があります。

1. セマンティック ランク付けは、テキスト クエリの BM25 でランク付けされた検索結果から、またはハイブリッド クエリの RRF でランク付けされた結果から開始されます。 テキスト フィールドのみが再ランク付け実行で使用され、結果の数が 50 個を超える場合でも、セマンティック ランク付けが行われるのは上位 50 個の結果のみです。 通常、セマンティック ランク付けで使用されるフィールドは、情報を提供する説明的なものです。

2. 検索結果の中の各ドキュメントで、サマライゼーション モデルが受け入れるのは 2,000 トークンまでで、この場合、1 トークンはおよそ 10 文字です。 入力がセマンティック構成の一覧にある "title"、"keyword"、および "content" フィールドからアセンブルされます。

3. 長さが非常に長い文字列は、全体の長さが概要作成手順の入力要件を満たすようにトリミングされます。 優先順位の高い順序でセマンティック構成にフィールドを追加することが重要なのは、このトリミングの実行があるためです。 テキストを多用するフィールドを持つ非常に大きいドキュメントがある場合は、最大値制限を超えたテキストは無視されます。

テーブルを展開する

セマンティック フィールド	トークンの限度
「タイトル」	128 トークン
"keywords	128 トークン
"content"	残りのトークン

4. サマライゼーション出力は各ドキュメントの要約文字列であり、各フィールドの中の最も関連した情報で構成されます。 要約文字列がスコアリングのためランカーへ送られて、キャプションと回答を求めて機械読み取り理解モデルへ送られます。

   セマンティック ランカーへ渡されるそれぞれの生成後の要約文字列の最大長さは、256 トークンです。

セマンティック ランカーの出力

各要約文字列から、機械読み取り理解モデルは最も代表的な一節を見つけ出します。

出力は次のようになります:

- ドキュメントのセマンティック キャプション。 各キャプションは、プレーン テキスト バージョンと強調表示バージョンで使用できます。また、多くの場合、ドキュメントあたり 200 語未満です。

- `answers` パラメーターを指定した場合、クエリが質問として提示された場合、その質問に対して適していそうな回答を提供する長い文字列で文節が見つかる場合を想定し、オプションのセマンティック回答が返されます。

キャプションと回答は、常にインデックスからの逐語的なテキストです。 このワークフローには、新しいコンテンツを作成または構成する生成 AI モデルはありません。

要約がスコアリングされるしくみ

スコアリングは、キャプションと、256 のトークン長を埋める要約文字列の中の他のあらゆるコンテンツに対して行われます。

1. キャプションは、指定されたクエリに対して相対的な概念とセマンティックの関連性に対して評価されます。

2. **@search.rerankerScore** が各ドキュメントに、指定のクエリのドキュメントのセマンティック関連性に基づいて割り当てられます。 スコアの範囲は 4 から 0 (高から低) です。スコアが高いほど関連性が高いことを示します。

3. 一致は、スコア順に一覧表示され、クエリ応答ペイロードに含められます。 ペイロードには、回答、プレーンテキスト、強調表示されたキャプション、select 句で取得または指定されたフィールドが含まれます。

> ⓘ **注意**
>
> 2023 年 7 月 14 日より、**@search.rerankerScore** の分布が変更されます。 スコアの結果を、テスト以外で判断することはできません。 この応答プロパティにハードしきい値の依存関係がある場合、テストを返して、しきい値に適切な新しい値を確認してください。

セマンティック機能と制限

セマンティック ランカーは新しいテクノロジなので、実行できることとできないことについての期待値を設定することが重要です。 "できること" は、次のようなことです。

- セマンティック的に元のクエリの意図に近い一致を昇格させます。

- キャプションおよび回答として使用できる文字列を見つけ出します。 キャプションと回答は応答で返され、検索結果ページに表示できる文字列が検出されます。

セマンティック ランク付けで "できない" ことは、コーパス全体に対してクエリを再実行して、セマンティックな関連がある結果を検出することです。 セマンティック ランク付けでは、既定のランク付けアルゴリズムによってスコアリングされた上位 50 個の結果で構成される既存の結果セットが再ランク付けされます。 さらに、セマンティック ランク付けで新しい情報や文字列を作成することはできません。 キャプションと回答は、コンテンツから逐語的に抽出されるので、結果に回答のようなテキストが含まれていない場合、その言語モデルではキャプションや回答は生成されません。

セマンティック ランク付けはすべてのシナリオで有益なわけではありませんが、特定のコンテンツではその機能から多くのメリットが得られます。 セマンティック ランク

付けの言語モデルは、情報が豊富で、散文として構造化された検索可能なコンテンツに最適です。 ナレッジ ベース、オンライン ドキュメント、説明的なコンテンツを含むドキュメントでは、セマンティック ランク付け機能から最も多くのメリットが得られます。

テクノロジは Bing と Microsoft Research が基になっており、Azure AI Search インフラストラクチャにアドオン機能として統合されます。 研究と AI 投資でバックアップされるセマンティック ランク付けに関する詳細については、「Bing の AI が Azure AI Search にパワーを与えるしくみ (Microsoft Research ブログ)」を参照してください。

次の動画では、機能の概要について説明しています。
https://www.youtube-nocookie.com/embed/yOf0WfVd_V0

可用性と料金

セマンティック ランカーは、利用可能なリージョンにおいて、Basic レベル以上の検索サービスで使用できます。

セマンティック ランカーを有効にする場合は、機能に対応する価格プランを選択します。

- クエリ ボリュームが低い場合 (月間 1000 件未満)、セマンティック ランク付けは無料です。
- クエリのボリュームが大きい場合、標準価格プランを選択します。

Azure AI Search の価格ページでは、さまざまな通貨とサイクル間隔での課金レートが表示されます。

セマンティック ランク付けの料金は、`queryType=semantic` を含む、検索文字列が空でないクエリ要求 (たとえば `search=pet friendly hotels in New York`) を使用した場合に発生します。 検索文字が空の場合 (`search=*`)、queryType が semantic に設定されていても課金されません。

関連項目

- セマンティック ランク付けを有効にする
- セマンティック ランク付けを構成する
- ブログ: ハイブリッド検索とランク付け機能でベクトル検索の性能を上回る

キーワード検索での関連性 (BM25 スコアリング)

[アーティクル] • 2024/05/08

この記事では、フルテキスト検索で検索スコアを計算するために使用される BM25 関連性スコアリングのアルゴリズムについて説明します。 BM25 関連性は、フルテキスト検索に限定されます。 フィルター クエリ、オートコンプリート、提案されたクエリ、ワイルドカード検索、あいまい検索の各クエリは、関連性についてスコアリングもランク付けもされません。

フルテキスト検索で使用されるスコアリング アルゴリズム

Azure AI Search には、フルテキスト検索用の次のスコアリング アルゴリズムが用意されています：

テーブルを展開する

アルゴリズム	使用法	Range
`BM25Similarity`	2020 年 7 月以降に作成されたすべての検索サービスでアルゴリズムを修正しました。 このアルゴリズムは構成可能ですが、古いアルゴリズム (クラシック) に切り替えることはできません。	無制限。
`ClassicSimilarity`	以前の検索サービスに存在します。 BM25 をオプトインし、インデックスごとにアルゴリズムを選択できます。	0 < 1.00

BM25 とクラシックはいずれも TF-IDF タイプの取得関数です。この関数では、単語の出現頻度 (TF) と逆文書頻度 (IDF) が変数として使用され、ドキュメントとクエリの組みごとに関連スコアが計算されます。ドキュメントとクエリの組みはその後、ランク付けの結果に使用されます。 概念的にはクラシックと似ていますが、BM25 は確率論的情報取得に根ざしており、ユーザーの調査で測定した場合のように、より直感的な一致が生成されます。

BM25 には高度なカスタマイズ オプションがあります。たとえば、ユーザーは、一致した単語の出現頻度で関連性スコアが変動するしくみを決定できます。 詳細については、スコアリング アルゴリズムの構成に関するページを参照してください。

> **注意**

2020 年 7 月より前に作成された検索サービスを使用している場合、スコアリング アルゴリズムは当時の既定のアルゴリズム (`ClassicSimilarity`) である可能性が高く、その場合は、インデックスごとにアップグレードを行うことができます。 詳細については、「**以前のサービスで BM25 スコアリングを有効にする**」を参照してください。

次のビデオ セグメントでは、Azure AI Search で使用される一般提供の優先度付けアルゴリズムの説明に早送りしています。 詳しい背景情報については、ビデオ全編をご覧ください。
https://www.youtube-nocookie.com/embed/Y_X6USgvB1g?version=3&start=322&end=643

BM25 ランク付けのしくみ

関連性のスコアリングとは、検索スコア (**@search.score**) を計算することです。これは、現在のクエリのコンテキストにおける項目の関連性のインジケーターとして機能します。 範囲は無制限です。 ただし、スコアが高いほど、項目の関連性が高くなります。

検索スコアは、文字列入力とクエリ自体の統計プロパティに基づいて計算されます。 Azure AI Search では、検索語句に一致するドキュメント (searchModeに応じて一部または全部) が検索され、検索語句の多くのインスタンスを含むドキュメントが優先されます。 データ インデックス全体での語句の出現頻度は低いがドキュメント内ではよく使用されている場合、検索スコアはより高くなります。 関連性を計算するこのアプローチの基礎となる手法は、*TF-IDF* (単語の出現頻度 - 逆文書頻度) と呼ばれています。

検索スコアは、結果セット全体で繰り返すことができます。 同じ検索スコアを持つ項目が複数ヒットした場合、同じスコアを持つ項目の順序付けは定義されていないので安定しません。 クエリを再度実行すると、特に、複数のレプリカで無料のサービスまたは課金対象サービスを使用している場合は、項目の位置が変わる場合があります。 同一スコアの項目が 2 つ存在する場合、どちらが最初に表示されるかは特定できません。

繰り返しスコアの間の関係を解除するには、**$orderby** 句を追加することで、まずスコアで並べ替えを行い、次に別の並べ替え可能なフィールド (`$orderby=search.score() desc,Rating desc` など) で並べ替えを行うことができます。 詳細については、$orderby に関するページを参照してください。

スコア付けには、インデックスで `searchable` としてマークされているフィールド、またはクエリで `searchFields` としてマークされているフィールドが使用されます。

`retrievable` としてマークされたフィールド、またはクエリの `select` で指定されたフィールドのみが、検索スコアと共に検索結果に返されます。

> **注意**
>
> `@search.score = 1` は、スコア付けまたは順位付けが行われていない結果セットを示します。 スコアは、すべての結果にわたって均一です。 スコア付けされていない結果が発生するのは、クエリ フォームがファジー検索、ワイルドカード、または正規表現のクエリである場合、または空の検索である場合 (`search=*` がフィルターとペアで指定され、一致を返す主な手段がフィルターである場合があります) です。

テキスト結果のスコア

結果がランク付けされるたびに、`@search.score` プロパティには結果の順序付けに使用される値が含まれます。

次の表に、各一致、アルゴリズム、および範囲で返されるスコアリング プロパティを示します。

テーブルを展開する

検索メソッド	パラメーター	スコアリング アルゴリズム	Range
フルテキスト検索	`@search.score`	BM25 アルゴリズム。インデックスで指定されたパラメーターを使用します。	無制限。

スコア バリエーション

検索スコアは、一般的な意味での関連性を示すものであり、同じ結果セット内の他のドキュメントと比べた場合の一致の強さが反映されています。 ただし、スコアは、あるクエリと次のものとの間で必ずしも一貫しているとは限らないため、クエリを操作していると、検索ドキュメントの順序付け方法における小さな不一致に気付くことがあります。 これが発生する理由については、次のいくつかの説明があります。

テーブルを展開する

原因	説明
同一のスコア	複数のドキュメントのスコアが同じである場合、それらはいずれも最初に表示される可能性があります。
データの不安定性	ドキュメントを追加、変更、または削除すると、インデックス コンテンツは変動します。 インデックスの更新が徐々に処理されるに従って用語頻度は変化し、一致するドキュメントの検索スコアに影響を与えます。
複数のレプリカ	複数のレプリカを使用するサービスの場合、クエリは、各レプリカに対して並列に発行されます。 検索スコアを計算するために使用されるインデックス統計はレプリカごとに計算され、クエリ応答の中で結果がマージされ、順序付けられます。 レプリカのほとんどは互いのミラーですが、状態の小さな違いのために統計は異なる場合があります。 たとえば、あるレプリカで、他のレプリカからマージされた、その統計に寄与しているドキュメントが削除されることがあります。 通常、レプリカごとの統計の違いは、小さなインデックスの方がより顕著です。 この条件について詳しくは、以下のセクションで説明します。

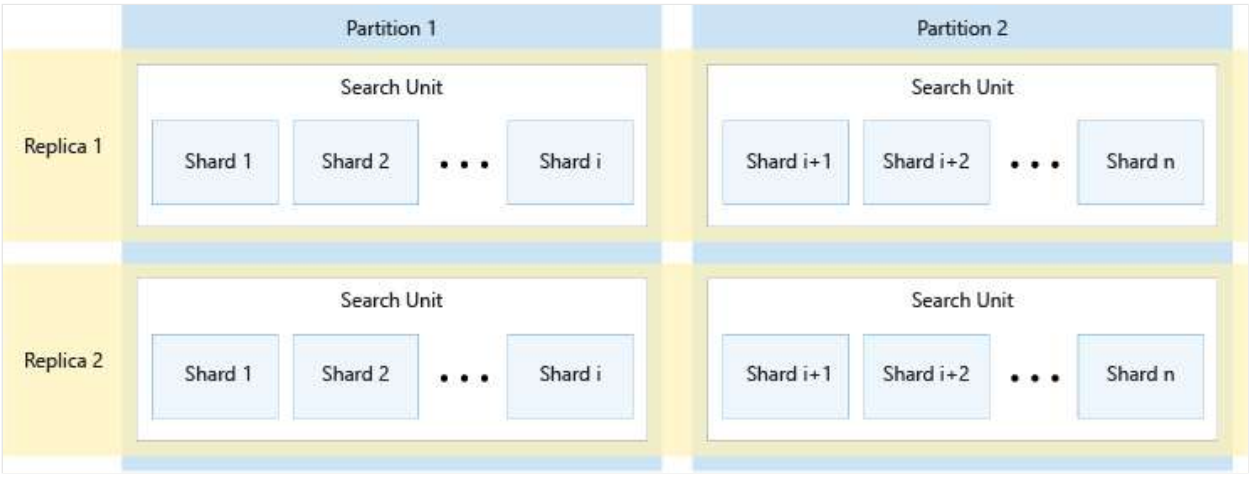
クエリ結果に対するシャーディング効果

"シャード" とは、インデックスのチャンクです。 Azure AI Search では、インデックスをさらに "シャード" に分割し、(シャードを新しい検索単位に移動することによって) パーティションを追加するプロセスを高速化しています。 検索サービスでは、シャード管理は実装の詳細であり、構成できませんが、インデックスがシャード化されているとわかっていれば、順位付けやオートコンプリートの動作で不定期に発生する異常を容易に把握できます。

- 異常のランク付け: 検索スコアは最初にシャード レベルで計算され、続いて単位の結果セットに集計されます。 シャード コンテンツの特性に応じて、あるシャードからの一致が別のシャードの一致よりも高い順位になる場合があります。 検索結果の順位付けが直観に反しているように感じられる場合は、シャーディングの影響が原因である可能性が最も高いです (特にインデックスが小さい場合)。 インデックス全体でグローバルにスコアを計算するように選択すれば、これらの順位付けの異常を回避できますが、その場合、パフォーマンスが低下します。

- オートコンプリートの異常: オートコンプリート クエリでは、部分的に入力された語句の最初のいくつかの文字で照合が行われますが、スペルの少しの間違いを許容するあいまいパラメーターが使用されます。 オートコンプリートの場合、あいまい一致は現在のシャード内の用語に限定されます。 たとえば、シャードに "Microsoft" が含まれており、"micro" という部分的な語句が入力された場合、検索エンジンはそのシャード内の "Microsoft" と一致しますが、インデックスの残りの部分を保持した他のシャードでは一致しません。

次の図は、レプリカ、パーティション、シャード、および検索単位間の関係を示しています。 ここでは、2 つのレプリカと 2 つのパーティションを持つサービスで、どのように 1 つのインデックスが 4 つの検索単位にわたっているかを例で示しています。 4 つの検索単位それぞれには、インデックスのシャードの半分だけが格納されます。 左側の列の検索単位は、シャードの前半を格納して最初のパーティションを構成し、右側の列の検索単位は、シャードの後半を格納して 2 番目のパーティションを構成しています。 レプリカが 2 つあるため、各インデックス シャードのコピーは 2 つあります。 上部の行の検索単位は、1 つのコピーを格納して最初のレプリカを構成し、下部の行の検索単位は、別のコピーを格納して 2 番目のレプリカを構成しています。



上の図は 1 つの例にすぎません。 パーティションとレプリカはさまざまに組み合わせることができ、最大で合計 36 の検索単位が可能です。

> **① 注意**
>
> レプリカとパーティションの数は、均等に 12 分割されます (具体的には、1、2、3、4、6、12)。 Azure AI Search では各インデックスを 12 のシャードに事前に分割して、それぞれがすべてのパーティションに均等に分散されるようにします。 たとえば、サービスに 3 つのパーティションがあり、インデックスを作成する場合、各パーティションにはインデックスの 4 つのシャードを含めます。 Azure AI Search でインデックスがどのようにシャードされるかは実装の詳細であり、今後のリリースで変更される場合があります。 今日 12 個であっても、今後も必ず 12 個になるとは限りません。

スコアリング統計とスティッキー セッション

スケーラビリティのために、Azure AI Search はシャーディング プロセスを通じて各インデックスを水平方向に配布します。つまり、インデックスの 部分は物理的に個別です。

既定では、ドキュメントのスコアは、"シャード内" のデータの統計プロパティに基づいて計算されます。 このアプローチは、一般に、データの大規模なコーパスでは問題にならず、すべてのシャードの情報に基づいてスコアを計算する必要がある場合よりもパフォーマンスが向上します。 ただし、このパフォーマンスの最適化を使用すると、2 つの非常に類似したドキュメント (またはまったく同一のドキュメント) は、それぞれが異なるシャードになる場合、関連性スコアが異なる可能性があります。

すべてのシャードの統計プロパティに基づいてスコアを計算する場合、これを行うには、`scoringStatistics=global` をクエリ パラメーターとして追加します (またはクエリ要求の本文パラメーターとして `"scoringStatistics": "global"` を追加します)。

HTTP

```
POST https://[service name].search.windows.net/indexes/hotels/docs/search?
api-version=2020-06-30
{
    "search": "<query string>",
    "scoringStatistics": "global"
}
```

`scoringStatistics` を使用すると、同じレプリカのすべてのシャードで同じ結果が得られるようになります。 ただし、レプリカはインデックスの最新の変更で常に更新されるため、それぞれ若干異なる場合があります。 一部のシナリオでは、ユーザーが "クエリ セッション" 中により一貫した結果を得られるようにすることが必要な場合があります。 このようなシナリオでは、クエリの一部として `sessionId` を指定できます。 `sessionId` は、一意のユーザー セッションを参照するために作成する一意の文字列です。

HTTP

```
POST https://[service name].search.windows.net/indexes/hotels/docs/search?
api-version=2020-06-30
{
    "search": "<query string>",
    "sessionId": "<string>"
}
```

同じ `sessionId` が使用されていれば、同じレプリカをターゲットにするためにベストエフォートの試行が行われるので、ユーザーに表示される結果の一貫性が向上します。

> **注意**
>
> 同じ `sessionId` 値を繰り返し再利用すると、レプリカ間での要求の負荷分散が妨げられ、検索サービスのパフォーマンスに悪影響を与える可能性があります。 sessionId として使用される値は、'_' 文字で始めることはできません。

関連性のチューニング

Azure AI Search では、BM25 アルゴリズム パラメーターを構成し、次のメカニズムを使用して検索の関連性を調整し、検索スコアを向上させることができます：

テーブルを展開する

アプローチ	実装	説明
スコアリング アルゴリズムの構成	Search index	
スコアリング プロファイル	Search index	コンテンツ特性に基づいて一致の検索スコアを向上させる基準が指定されます。 たとえば、収益の見込みに基づいて一致をブーストしたり、より新しい項目のレベルを上げたり、場合によっては在庫期間が長すぎる項目をブーストしたりできます。 スコアリング プロファイルは、インデックス定義の一部であり、重み付けされたフィールド、関数、およびパラメーターで構成されます。 スコアリング プロファイルの変更によって既存のインデックスを更新できます。 インデックスの再構築は必要ありません。
セマンティック ランク付け	クエリ要求	検索結果に機械読解を適用し、意味的に関連性の高い結果を上位に昇格させます。
featuresMode パラメーター	クエリ要求	このパラメーターは、多くの場合、スコアのアンパックに使用されますが、カスタム スコアリング ソリューション を提供するコードで使用できます。

featuresMode パラメーター (プレビュー)

ドキュメントの検索の要求には、フィールド レベルでの関連性に関するさらに詳細な情報を提供できる新しい featuresMode パラメーターがあります。 `@searchScore` はドキュメント全体に対して計算されますが (このクエリのコンテキストにおけるこのドキ

ュメントの関連度)、featuresMode を使用すると、`@search.features` 構造体で表現された、個々のフィールドに関する情報を取得できます。 この構造体には、クエリで使用されるすべてのフィールド (クエリ内の **searchFields** を介した特定のフィールド、またはインデックス内で**検索可能**として属性が付けられているすべてのフィールド) が含まれます。 フィールドごとに、次の値が取得されます。

- フィールド内で見つかった一意のトークン数
- 類似性スコア。つまり、クエリ用語に対するフィールド内容の類似度のメジャー
- 用語の頻度。つまり、フィールド内でクエリ用語が見つかった回数

"Description" および "title" フィールドを対象とするクエリの場合、`@search.features` を含む応答は次のようになります。

JSON

```
"value": [
 {
    "@search.score": 5.1958685,
    "@search.features": {
        "description": {
            "uniqueTokenMatches": 1.0,
            "similarityScore": 0.29541412,
            "termFrequency" : 2
        },
        "title": {
            "uniqueTokenMatches": 3.0,
            "similarityScore": 1.75451557,
            "termFrequency" : 6
        }
    }
 }
]
```

カスタムのスコアリング ソリューション でこれらのデータ ポイントを使用したり、この情報を使用して検索の関連性の問題をデバッグしたりできます。

フルテキスト クエリ応答のランク付けされた結果の数

既定では、改ページを使用していない場合、検索エンジンはフルテキスト検索では上位 50 位のランクの一致を返します。 `top` パラメーターを使用して、返される項目数を減らしたり増やしたりできます (1 回の応答で 1000 個まで)。 フルテキスト検索には、最大 1,000 件の一致という制限が適用されます (API 応答の制限を参照)。 1,000 件の一致が見つかると、検索エンジンはそれ以上の検索を行いません。

返す結果をさらに増やす、または減らすには、ページング パラメーター `top`、`skip`、`next` を使用します。 ページングは、各論理ページ上の結果の数を決定し、完全なペイロードを導く方法です。 詳細については、「検索結果の操作方法」を参照してください。

関連項目

- スコアリング プロファイル
- REST API リファレンス
- ドキュメント API の検索
- Azure AI Search .NET SDK

ベクトル検索での関連性

[アーティクル] • 2024/04/18

ベクトル クエリの実行時、検索エンジンは類似のベクトルを検索して、検索結果に返される最適な候補を見つけます。 ベクトル コンテンツのインデックス付け方法に応じて、関連する一致の検索は網羅的であるか、ニアネイバーに制限されて処理が高速化されます。 候補が見つかると、類似性メトリックを使用して、一致の強度に基づいて各結果のスコアが付けられます。

この記事では、関連性のある一致を見つけるために使用されるアルゴリズムと、スコアリングに使用される類似性メトリックについて説明します。 また、検索結果が期待に沿っていない場合に関連性を向上させるヒントも提供します。

ベクトル検索で使用されるアルゴリズム

ベクトル検索アルゴリズムには、完全な k 二アレスト ネイバー (KNN) や Hierarchical Navigable Small World (HNSW) があります。

- 完全な KNN は、ベクトル空間全体をスキャンするブルート フォース検索を実行します。
- HNSW は、近似二アレスト ネイバー (ANN) 検索を実行します。

検索とスコアリングに使用されるのは、インデックスに `searchable` としてマークされているフィールド、またはクエリの `searchFields` としてマークされたベクトル フィールドだけです。

完全な KNN を使用する場合

完全な KNN は、データ ポイントのすべてのペア間の距離を計算し、クエリ ポイントの正確な `k` 二アレスト ネイバーを見つけます。 これは、高い再現率が最も重要であり、ユーザーにクエリ待ち時間のトレードオフを受け入れる意思があるシナリオを対象としています。 計算負荷が高いため、小規模から中規模のデータセット、または精度要件がクエリ パフォーマンスの考慮事項を上回る場合は、完全な KNN を使用します。

2 番目の用途は、近似二アレスト ネイバー アルゴリズムの呼び戻しを評価するデータセットの構築です。 完全な KNN を使用して、二アレストネイバーのグラウンド トゥルースのセットを構築できます。

完全な KNN サポートは、2023-11-01 REST API および 2023-10-01-Preview REST API と、そのいずれかの REST API バージョンを対象とする Azure SDK クライアント ライブ

ラリで利用できます。

HNSW を使用する場合

インデックス作成中に、HNSW は、データ ポイントを階層グラフ構造に編成して、より高速な検索のために追加のデータ構造を作成します。 HNSW には、検索アプリケーションのスループット、待機時間、および再現目標を達成するために調整できるいくつかの構成パラメーターがあります。 たとえば、クエリ時に、ベクトル フィールドに HNSW のインデックスが付いている場合でも、包括的な検索のオプションを指定できます。

クエリの実行中、HNSW はグラフ内を移動して高速な近隣クエリを有効にします。 このアプローチにより、検索の正確さと計算効率のバランスが取れます。 HNSW は、大規模なデータ セットを検索するときの効率が高いため、ほとんどのシナリオに推奨されます。

ニアレストネイバー検索のしくみ

ベクトル クエリは、同じ埋め込みモデルから生成されたベクトルで構成される埋め込み空間に対して実行されます。 一般に、クエリ要求内の入力値は、ベクトル インデックスに埋め込みを生成したのと同じ機械学習モデルにフィードされます。 出力は、同じ埋め込み空間内のベクトルです。 同様のベクトルが近接してクラスター化されるため、一致を見つけることは、クエリ ベクトルに最も近いベクトルを見つけ、関連するドキュメントを検索結果として返すのと同じです。

たとえば、クエリ要求がホテルに関する場合、モデルは、ホテルに関するドキュメントを表すベクトルのクラスター内のどこかに存在するベクトルにクエリをマップします。 類似度メトリックに基づいて、クエリに最も似ているベクトルを特定すると、最も関連性の高いドキュメントが決まります。

完全な KNN に対してベクトル フィールドのインデックスが作成されると、クエリは "all neighbors" に対して実行されます。 HNSW 用にインデックスが作成されたフィールドの場合、検索エンジンは HNSW グラフを使用して、ベクトル インデックス内のノードのサブセットを検索します。

HNSW グラフの作成

インデックス作成中、検索サービスは HNSW グラフを構築します。 HNSW グラフに新しいベクトルのインデックスを作成する目的は、効率的なニアレストネイバー検索できるような方法でグラフ構造に追加することです。 次の手順は、このプロセスをまとめたものです。

1. 初期化: 空の HNSW グラフ、または新しいインデックスでない場合は既存の HNSW グラフから開始します。

2. エントリ ポイント: これは階層グラフの最上位レベルであり、インデックス作成の開始点として機能します。

3. グラフへの追加: 階層レベルが異なると、グラフの細分性が異なり、レベルが高いほどグローバルになり、レベルが低いほど細かく表示されます。 グラフ内の各ノードは、ベクトル ポイントを表します。

    - 各ノードは、近隣にある最大 `m` 個のネイバーに接続されます。 これが `m` パラメーターです。

    - 候補接続と見なされるデータ ポイントの数は、`efConstruction` パラメーターによって管理されます。 この動的リストは、アルゴリズムが考慮する既存のグラフ内の最も近いポイントのセットを形成します。 `efConstruction` 値が大きいほど、多くのノード数が考慮され、多くの場合、各ベクトルのローカル近傍が高密度になります。

    - これらの接続では、構成された類似性 `metric` を使用して距離を決定します。 一部の接続は、さまざまな階層レベルで接続する "長距離" 接続であり、検索効率を向上させるショートカットをグラフに作成します。

4. グラフの枝刈りと最適化: これは、すべてのベクトルのインデックス作成後に発生する可能性があり、HNSW グラフのナビゲート性と効率が向上します。

クエリ時の HNSW グラフ内の移動

ベクトル クエリは、一致をスキャンするために階層グラフ構造内を移動します。 プロセスの手順の概要を次に示します:

1. 初期化: アルゴリズムは、階層グラフの最上位レベルで検索を開始します。 このエントリ ポイントには、検索の開始点として機能するベクトルのセットが含まれています。

2. トラバーサル: 次に、グラフをレベルごとに走査し、最上位レベルから下位レベルに移動し、コサイン類似性など、構成された距離メトリックに基づいてクエリ ベクトルに近い候補ノードを選択します。

3. 枝刈り: 効率を向上させるために、アルゴリズムは、ニアレストネイバーを含む可能性が高いノードのみを考慮して、検索領域を刈り込みます。 これは、潜在的な候補の優先順位キューを維持し、検索が進むにつれてそれを更新することによっ

て達成されます。 このキューの長さは、パラメーター `efSearch` によって構成されます。

4. 絞り込み: アルゴリズムがより低く、より細かいレベルに移行すると、HNSW はクエリの近くでより多くのネイバーを考慮し、ベクトルの候補セットを絞り込み、精度を向上させます。

5. 完了: 検索は、二アレストネイバーの必要な数が特定された場合、または他の停止条件が満たされたときに完了します。 この二アレスト米バーのこの必要な数は、クエリ時間パラメーター `k` によって制御されます。

類似性の測定に使用される類似性メトリック

アルゴリズムは、類似性を評価する候補ベクトルを検索します。 このタスクを実行するために、類似性メトリック計算では、候補ベクトルとクエリ ベクトルを比較し、類似性を測定します。 アルゴリズムは、検索された最も類似しているベクトルの順序付けされたセットを追跡し続け、アルゴリズムが完了したときにランク付けされた結果セットを形成します。

テーブルを展開する

メトリック	説明
`cosine`	このメトリックは、2 つのベクトル間の角度を測定し、ベクトルの長さの違いによる影響を受けません。 数学的には、2 つのベクトル間の角度を計算します。 コサインは Azure OpenAI 埋め込みモデルで使用される類似性メトリックであるため、Azure OpenAI を使用している場合は、ベクトル構成で `cosine` を指定します。
`dotProduct`	このメトリックは、2 つのベクトルの各ペアの長さと、それらの間の角度の両方を測定します。 数学的には、ベクトルの大きさとそれらの間の角度の積を計算します。 正規化されたベクトルの場合、これは `cosine` 類似性と同じですが、パフォーマンスは若干高くなります。
`euclidean`	(別名 `l2 norm`) このメトリックは、2 つのベクトル間のベクトル長の差を測定します。 数学的には、2 つのベクトルの差の l2ノルムである 2 つのベクトル間のユークリッド距離を計算します。

ベクトル検索結果のスコア

スコアが計算されて、それぞれの一致に割り当てられ、最も高い一致が `k` の結果として返されます。 `@search.score` プロパティにスコアが含まれています。 次の表は、スコアが該当する範囲を示しています。

テーブルを展開する

検索メソッド	パラメーター	スコアリング メトリック	Range
ベクトル検索	`@search.score`	コサイン	0.333 - 1.00

`cosine` メトリックについては、計算された `@search.score` がクエリ ベクトルとドキュメント ベクトルの間のコサイン値ではないことに注意することが重要です。 代わりに、Azure AI Search では、スコア関数が単調に減少するような変換が適用されます。 つまり、スコア値は、類似性が悪化すると常に値が減少します。 この変換により、検索スコアをランク付け目的で使用できます。

類似性スコアにはいくつかの微妙な違いがあります:

- コサイン類似性は、2 つのベクトル間の角度のコサインとして定義されます。
- コサイン距離は `1 - cosine_similarity` として定義されます。

単調に減少する関数が作成されるように `@search.score` は `1 / (1 + cosine_distance)` として定義されます。

合成値の代わりにコサイン値が必要な開発者は、数式を使用して、検索スコアをコサイン距離に戻すことができます:

C#

```
double ScoreToSimilarity(double score)
{
    double cosineDistance = (1 - score) / score;
    return  -cosineDistance + 1;
}
```

元のコサイン値を持たせておくと、低品質の結果をトリミングするためのしきい値を設定するカスタム ソリューションにおいて有用です。

関連性のチューニングに関するヒント

関連性のある結果が得られない場合は、クエリの構成の変更を試してください。 ベクトル クエリには、スコアリング プロファイルやフィールド、用語ブーストなどの特定のチューニング機能はありません。

- チャンク サイズと重複について実験します。 チャンク サイズを大きくし、チャンク間のコンテキストまたは継続性を維持するのに十分な重複を確保します。

- HNSW の場合は、近接グラフの内部構成を変更するさまざまなレベルの `efConstruction` を試します。 既定値は 400 です。 範囲は 100 ～ 1,000 です。
- チャット モデルを使用している場合は、`k` の結果を増やして、より多くの検索結果をチャット モデルにフィードします。
- セマンティック ランク付けを使用してハイブリッド クエリを試します。 ベンチマーク テストでは、この組み合わせが一貫して最も関連性の高い結果をもたらしました。

次のステップ

- クイックスタートを試す
- 埋め込みの詳細
- データ チャンクの詳細

Reciprocal Rank Fusion (RRF) を使用したハイブリッド検索での関連性スコアリング

[アーティクル] • 2023/10/26

> **ⓘ 重要**
>
> ハイブリッド検索では、現在パブリック プレビュー段階にある**ベクター機能**を**補足の利用規約**のもとに使用します。

Reciprocal Rank Fusion (RRF) は、以前にランク付けされた複数の結果の検索スコアを評価して、統合された結果セットを生成するアルゴリズムです。 Azure Cognitive Search では、並列で実行される 2 つ以上のクエリがある場合は、常に RRF が使用されます。 各クエリがランク付けされた結果セットを生成し、RRF はランク付けをマージして、クエリ応答で返される 1 つの結果セットに均質化するために使用されます。 RRF が必要なシナリオの例としては、ハイブリッド検索、同時に実行される複数のベクター クエリなどがあります。

RRF は、"逆順位" の概念に基づいています。これは、検索結果のリスト内の最初の関連ドキュメントのランクの逆数です。 この手法の目標は、元のランキング内の項目の位置を考慮して、複数のリストで上位にランク付けされた項目により高い重要性を与えることです。 これにより、最終的なランク付けの全体的な品質と信頼性が向上し、複数の順序付けされた検索結果を融合するタスクに役立ちます。

RRF ランク付けのしくみ

RRF は、複数の方法から検索結果を取得し、結果の各ドキュメントに逆順位スコアを割り当て、スコアを組み合わせて新しいランク付けを作成することで機能します。 この概念では、複数の検索方法で上位の位置に表示されるドキュメントはより関連性が高い可能性があるため、結合された結果の上位にランク付けされます。

RRF プロセスの簡単な説明を次に示します。

1. 並列で実行される複数のクエリからランク付けされた検索結果を取得します。

2. ランク付けされた各リストの結果に対して逆順位スコアを割り当てます。 RRF は、各結果セットの一致ごとに新しい `@search.score` を生成します。 検索結果のドキュメントごとに、エンジンはリスト内の位置に基づいて逆順位スコアを割り

当てます。 スコアは `1/(rank + k)` として計算されます。ここで、 `rank` はリスト内のドキュメントの位置であり、 `k` は定数で、60 などの小さな値に設定されている場合に最適に実行されることが実験で確認されています。 **この `k` 値は RRF アルゴリズムの定数であり、最も近い近傍の数を制御する `k` とは完全に別の定数であることに注意してください。**

3. スコアを結合します。 各ドキュメントについて、エンジンは各検索システムから取得した逆順位スコアを合計し、各ドキュメントの結合スコアを生成します。

4. エンジンは、結合されたスコアに基づいてドキュメントをランク付けし、それらを並べ替えます。 結果のリストは融合されたランキングです。

スコアリングには、インデックスで `searchable` としてマークされているフィールド、またはクエリで `searchFields` としてマークされているフィールドのみが使用されます。 `retrievable` としてマークされたフィールド、またはクエリの `select` で指定されたフィールドのみが、検索スコアと共に検索結果に返されます。

並列クエリの実行

RRF は、複数のクエリが実行されるたびに使用されます。 次の例は、並列クエリ実行が行われるクエリ パターンを示しています。

- フルテキスト クエリと 1 つのベクター クエリ (単純なハイブリッド シナリオ) は、2 つのクエリ実行に相当します。
- フルテキスト クエリと、2 つのベクター フィールドを対象とする 1 つのベクター クエリは、3 つのクエリ実行に相当します。
- フルテキスト クエリと、5 つのベクター フィールドを対象とする 2 つのベクター クエリは、11 個のクエリ実行に相当します

ハイブリッド検索結果のスコア

結果がランク付けされるたびに、 `@search.score` プロパティには結果の順序付けに使用される値が含まれます。 スコアは、方法ごとに異なるランク付けアルゴリズムによって生成されます。 各アルゴリズムには、独自の範囲と大きさがあります。

次の表に、各関連性ランク付けアルゴリズムの各一致で返されるスコアリング プロパティ、アルゴリズム、スコアの範囲を示します。

検索メソッド	パラメーター	スコアリング アルゴリズム	Range
フルテキスト検索 (full-text search)	`@search.score`	BM25 アルゴリズム	上限なし。
ベクトル検索	`@search.score`	HNSW アルゴリズム。HNSW 構成で指定された類似性メトリックを使用します。	0.333 - 1.00 (Cosine)、Euclidean と DotProduct の場合は 0 から 1。
ハイブリッド検索	`@search.score`	RRF アルゴリズム	上限は結合されるクエリの数によって制限され、各クエリは RRF スコアに対して最大約 1 を寄与します。 たとえば、3 つのクエリをマージすると、2 つの検索結果のみがマージされる場合よりも RRF スコアが高くなります。
セマンティック ランク付け	`@search.rerankerScore`	セマンティック ランク付け	1.00 - 4.00

セマンティック ランク付けは RRF には関係していません。 そのスコア (`@search.rerankerScore`) は、常にクエリ応答で個別に報告されます。 セマンティック ランク付けでは、フルテキスト検索結果とハイブリッド検索結果を再ランク付けでき、それらの検索結果には意味的に豊富なコンテンツを持つフィールドが含まれていると仮定しています。

ハイブリッド クエリ応答のランク付けされた結果の数

既定では、改ページを使用していない場合、検索エンジンは、フルテキスト検索では上位 50 位のランクの一致、ベクトル検索では最も類似した `k` 個の一致を返します。 ハイブリッド クエリでは、`top` によって応答の結果の数が決まります。 既定に基づいて、統合結果セットの上位 50 にランク付けされた一致が返されます。

多くの場合、検索エンジンは `top` や `k` よりも多くの結果を見つけます。 より多くの結果を返すには、ページング パラメーター `top`、`skip`、`next` を使用します。 ページングは、各論理ページ上の結果の数を決定し、完全なペイロードを導く方法です。

フルテキスト検索には、最大 1,000 件の一致という制限が適用されます (「API 応答の制限」を参照)。 1,000 件の一致が見つかると、検索エンジンはそれ以上の検索を行いません。

詳細については、「検索結果の操作方法」を参照してください。

関連項目

- ハイブリッド検索の詳細を確認する
- ベクトル検索の詳細を確認する