IBM WebSphere Information Integrator
OmniFind Edition

テキスト分析機能ガイド

バージョン 8.3

SD88-6728-00
(英文原典：SC18-9674-00)

IBM WebSphere Information Integrator
OmniFind Edition

テキスト分析機能ガイド

バージョン 8.3

SD88-6728-00
(英文原典：SC18-9674-00)

本書および本書で紹介する製品をご使用になる前に、『特記事項』に記載されている情報をお読みください。

本書には、IBM の専有情報が含まれています。その情報は、使用許諾条件に基づき提供され、著作権により保護されています。本書に記載される情報には、いかなる製品の保証も含まれていません。また、本書で提供されるいかなる記述も、製品保証として解釈すべきではありません。

IBM の資料は、オンラインまたは最寄りの事業所の IBM 担当員を通じて注文できます。

- オンラインで資料を注文する場合は、IBM Publications Center (www.ibm.com/shop/publications/order) にアクセスしてください。
- 最寄りの IBM 担当員をお探しになる場合は、IBM Directory of Worldwide Contacts (www.ibm.com/planetwide) にアクセスしてください。

IBM は、お客様が提供するいかなる情報も、お客様に対してなんら義務も負うことのない、自ら適切と信ずる方法で、使用もしくは配布することができるものとします。

本マニュアルに関するご意見やご感想は、次の URL からお送りください。今後の参考にさせていただきます。

http://www.ibm.com/jp/manuals/main/mail.html

なお、日本 IBM 発行のマニュアルはインターネット経由でもご購入いただけます。詳しくは

http://www.ibm.com/jp/manuals/ の「ご注文について」をご覧ください。

(URL は、変更になる場合があります)

お客様の環境によっては、資料中の円記号がバックスラッシュと表示されたり、バックスラッシュが円記号と表示されたりする場合があります。

原　典： SC18–9674–00
IBM WebSphere Information Integrator
OmniFind Edition
Text Analysis Integration
Version 8.3

発　行： 日本アイ・ビー・エム株式会社

担　当： ナショナル・ランゲージ・サポート

第1刷 2005.11

この文書では、平成明朝体™W3、平成明朝体™W7、平成明朝体™W9、平成角ゴシック体™W3、平成角ゴシック体™W5、および平成角ゴシック体™W7を使用しています。この(書体*)は、(財) 日本規格協会と使用契約を締結し使用しているものです。フォントとして無断複製することは禁止されています。

注* 平成明朝体™W3、平成明朝体™W7、平成明朝体™W9、平成角ゴシック体™W3、
平成角ゴシック体™W5、平成角ゴシック体™W7

目次

本書のトピックについて

この情報を使用して、IBM® WebSphere® Information Integrator OmniFind™ Edition バージョン 8.3 システムで、セマンティック検索ソリューションの作成およびデプロイを行います。セマンティック検索では、高水準の概念の検索を行い、テキスト分析で検出された検索照会内の関係を表現することができます。

WebSphere Information Integrator OmniFind Edition (WebSphere II OmniFind Edition) には、エンタープライズ・サーチ という機能が搭載されています。エンタープライズ・サーチのコンポーネントは、WebSphere II OmniFind Edition 製品のインストール時に、インストールされます。エンタープライズ・サーチ という用語は、インストール・パスおよび製品パッケージ化ラベルを参照する場合をのぞき、WebSphere II OmniFind Edition の資料全体で使用されています。

エンタープライズ・サーチのテキスト分析資料では、以下のトピックについて説明します。

- エンタープライズでの言語サポートの紹介
- エンタープライズ・サーチでカスタム・テキスト分析を統合する方法の説明
- XML 文書構造のマップ方法の説明
- 選択した分析結果を JDBC 表に追加する方法の説明
- セマンティック検索を使用可能にするために、分析結果をエンタープライズ・サーチ索引に追加する方法の説明
- 検索に同義語、ストップワードおよびランキング調整ワード辞書を組み込む方法の説明
- 文書処理中に、どのテキスト分析機能が自動的に実行されるかの概要

本書の対象読者

本書は、エンタープライズ・サーチのセマンティック検索ソリューションの作成およびデプロイを担当するシステム管理者およびサーチ・アプリケーション開発者を対象としています。

エンタープライズ・サーチは、テキスト文書のセマンティック検索サポートを提供します。文書は分析され、結果が保管されて、サーチ中にアクセス可能です。テキスト分析を語とテキストのスパンの両方の索引付け機能を持つエンタープライズ・サーチと結合することにより、セマンティック検索が可能になります。検索時のセマンティック・サポートの目的は、文書検索結果の精度を上げることです。つまり、照会に一致する文書の中で最も有効なコレクションに到達することです。

この情報を使用して、エンタープライズ・サーチでカスタム・テキスト分析を統合する方法および、照会結果を向上させるために、同義語、ストップワード、およびランキング調整ワード辞書の使用方法を学習します。またこの情報を使用して、どの基本言語サポートが文書処理の間中にエンタープライズ・サーチに含まれるかも理解します。

本書を活用するためには、Web アプリケーションを十分理解していること、検索したいデータ・ソースについての経験があることが必要です。

セマンティック検索に関する言語サポート

エンタープライズ・サーチは、大部分のインド・ヨーロッパ語族の言語、および日本語などのアジア言語によるテキスト文書に対する言語検索サポートを提供します。

検索時の言語サポートの目的は、文書検索結果の精度を上げることです。そして、照会に一致する文書の中で最も有効なコレクションに到達することです。

言語処理は、2 つのステージで行われます。文書が処理されて索引に追加される際と、検索時にユーザーが照会を入力する際です。

エンタープライズ・サーチには、入力文書の言語の判別および文書入力ストリームの語またはトークンへのセグメント化に必要なおおまかな言語機能しか組み込まれていません。

必要な検索が、主に、基本的なキーワード検索または文書構造を使用するネイティブ XML 検索に限られる場合、検索要件はエンタープライズ・サーチに組み込まれている言語処理で十分カバーされます。

ただし、以下の場合のように、単に文書内の語の検索でなく、より特定化した検索が必要な場合には、言語処理だけでは必ずしも十分ではありません。

- コラボレーション・ケースの情報。情報はいつも明示的にマークされているわけではありません。たとえば、E メール内の住所や電話番号などです。用語 *電話番号* は使用されません。その代わりに、E メールでは、「555-641-1805 で連絡してください」などのフレーズが含まれます。
- 競合に関する情報。競合相手およびその競合相手が提供する製品が記載されている文書、あるいは競合相手の Web サイトが過去 3 カ月にある製品セットから別の製品セットにシフトしたことなど。
- カスタマー・リレーションシップ・マネージメント。サンフランシスコ地区の修理店における車のブレーキ故障について記載されている文書など。修理店のレポートは、「油圧の漏出によるシューの調整」などの状況を記述します。さらにこれらのレポートは、修理店の番地のみを記述し、完全な住所は記述しません。
- 研究分野。特定のたんぱく質とそれに関係する少なくとも 1 つの病気が、同じパラグラフにある文書など。文献には 20 以上の異なる名前のたんぱく質があり、文書はしばしば病気 というワードには触れずに、病気の名前自身を記述します。

これらの例において、今日存在する膨大な情報ソースの集合体の中から必要なものを検索するには、エンタープライズ・サーチが提供するセグメンテーション・レベルおよび辞書ベースの分析よりさらに複雑な分析が新たに必要であることを示しています。興味のある情報のほとんどは、オリジナル文書で何らかの方法で明確にタグ付けされたり、マークされたりしていません。代わりに、興味のある概念 (例えば、個人、組織、場所、機能、製品などの特定のエンティティーやこれらのエンティティー相互の考えられる関係など) を認識し検索するために、情報を分析する必要があります。

IBM Unstructured Information Management Architecture (UIMA) は、文書コレクションの中から興味のある情報を検出および検索するために必要な拡張分析機能をエンタープライズ・サーチにおいて作成する際に役立つ、アーキテクチャーおよびソフトウェア・フレームワークです。

関連概念

3 ページの『カスタム・テキスト分析の組み込み』
Unstructured Information Management Architecture (UIMA) は、テキスト分析機能の作成、発見、構成、および配置を サポートするソフトウェア・アーキテクチャーです。UIMA を使用して、カスタム・テキスト分析を作成することができます。

4 ページの『Unstructured Information Management Architecture (UIMA) の概要』
Unstructured Information Management Architecture (UIMA) は、文書コレクション内の特定の 情報の検索に役立つ拡張分析機能の構築を支援するアーキテクチャーおよび ソフトウェア・フレームワークです。

カスタム・テキスト分析の組み込み

Unstructured Information Management Architecture (UIMA) は、テキスト分析機能の作成、発見、構成、および配置をサポートするソフトウェア・アーキテクチャーです。UIMA を使用して、カスタム・テキスト分析を作成することができます。

UIMA は、概念的に異なる分析機能ごとにコンポーネントを識別し、これらのコンポーネントの再使用や他のコンポーネントとの結合を容易にするオープン・プラットフォームです。

UIMA の中心となる概念は、*分析エンジン* です。これは、テキスト文書における分析内容を明確に表します。分析ロジック・コンポーネントは *アノテーター* と呼ばれます。アノテーターは、分析タスクのみに焦点をあて、他の処理には関与しません。分析エンジンには、1 つのアノテーターが含まれているか、あるいは、それぞれにアノテーターが含まれている多数のエンジンから構成されます。

分析エンジンにより導き出された情報は、*分析結果* と呼ばれます。理想的には、分析結果が、ユーザーが検索したい情報に一致します。

高機能言語分析には、多数の異なる分析タスクが含まれています。分析では、例えば、最初に言語の検出およびセグメンテーションを行い、続いて個々の品詞の認識を行い、その後より詳しい文法的な構文解析を行います。最終ステップには、例えば、科学的な本質と特定の徴候の出現との間の関係などの識別が含まれます。分析プロセスの各ステップは、後続のステップで必要になります。

UIMA は、ユーザー固有の分析エンジンを作成、テスト、および展開するための基本的な構築ブロックを提供します。 UIMA 環境に配置できる事前構成済みの分析エンジンとして、言語分析機能を提供するものではありません。

UIMA Software Development Kit (SDK) には、UIMA コンポーネントのインプリメンテーション、記述、構成およびデプロイメントのための UIMA フレームワークの Java™ インプリメンテーションが含まれます。 これは、 UIMA を使用するための、一連のツールおよびユーティリティーを含む、Eclipse ベースの 開発環境が含まれます。Eclipse について詳しくは、www.eclipse.org を参照してください。

UIMA を使用するには、UIMA Software Development Kit をインストールする必要があります。この Development Kit は、IBM developerWorks® で入手可能です。詳しくは、WebSphere Information Integrator ゾーン (http://www.ibm.com/developerworks/db2/zones/db2ii/) にアクセスしてください。Eclipse 対話式開発環境における UIMA Software Development Kit のインストール方法については、UIMA 文書を参照してください。

関連概念

1 ページの『セマンティック検索に関する言語サポート』
エンタープライズ・サーチは、大部分のインド・ヨーロッパ語族の言語、および日本語などの アジア言語によるテキスト文書に対する言語検索サポートを提供します。

『Unstructured Information Management Architecture (UIMA) の概要』
Unstructured Information Management Architecture (UIMA) は、文書コレクション内の特定の 情報の検索に役立つ拡張分析機能の構築を支援するアーキテクチャーおよび ソフトウェア・フレームワークです。

Unstructured Information Management Architecture (UIMA) の概要

Unstructured Information Management Architecture (UIMA) は、文書コレクション内の特定の情報の検索に役立つ拡張分析機能の構築を支援するアーキテクチャーおよびソフトウェア・フレームワークです。

フィーチャー構造 とは、分析の結果を表す基礎となるデータ構造です。フィーチャー構造は、属性-値の構造をしています。各フィーチャー構造が 1 つのタイプになっており、Java クラスのように、すべてのタイプに、指定された一連の有効なフィーチャーまたは属性 (プロパティー) があります。フィーチャーには、フィーチャーが使用する値のタイプ (例えば、String など) を示す範囲タイプがあります。

ほとんどの分析アルゴリズム (アノテーターとも呼ばれています) は、分析結果を注釈形式で生成します。注釈は、言語分析処理用に指定される特殊な種類のフィーチャー構造です。フィーチャー構造は、1 つの入力テキストをカバーし、入力テキストの先頭位置と終了位置によって定義されます。

例えば、通貨表示を認識するアノテーターは、「100.55 US Dollars」というテキストに対して、 `currencySymbol` フィーチャーを「$」に設定して、テキストをカバーする `monetaryExpression` タイプの注釈を作成します。

UIMA のすべてのアノテーターは、フィーチャー構造を使用して情報の保管および読み取りを行います。つまり、すべてのデータがフィーチャー構造としてモデル化されます。

タイプ・システムは、Java のクラス階層のように、タイプとフィーチャーから考えられるすべてのフィーチャー構造を定義します。

すべてのフィーチャー構造は、*共通分析構造* と呼ばれる中央データ構造で表されます。すべてのデータ交換は、共通分析構造を使用して扱われます。

共通分析構造には、以下のオブジェクトが含まれます。

- テキスト文書
- タイプ、サブタイプ、フィーチャーを示すタイプ・システム記述
- 文書または文書の領域を示す分析結果
- 分析結果へのアクセスおよび反復をサポートする索引リポジトリー

関連概念

1 ページの『セマンティック検索に関する言語サポート』
エンタープライズ・サーチは、大部分のインド・ヨーロッパ語族の言語、および日本語などの アジア言語によるテキスト文書に対する言語検索サポートを提供します。

3 ページの『カスタム・テキスト分析の組み込み』
Unstructured Information Management Architecture (UIMA) は、テキスト分析機能

の作成、発見、構成、および配置を サポートするソフトウェア・アーキテクチャーです。UIMA を使用して、カスタム・テキスト分析を作成することができます。

カスタム分析の組み込みのワークフロー

カスタム・テキスト分析アルゴリズムは、UIMA Software Development Kit を使用して、作成およびテストした後、配置し、エンタープライズ・サーチの文書コレクションにおいて実行します。

分析アルゴリズムを作成し、それらをエンタープライズ・サーチにインプリメントするには、次のようにします。

1. プランおよび設計:
   a. 検索したい情報を決定する。検索したい文書は何ですか ? 特定の検索タスクで必要な概念と関連は何ですか? たとえば、製品および従業員名は、製薬会社の内部 Web サイトでの汎用検索を拡張するのに必要である可能性があります。そのとき、リサーチおよび開発エリアの人々は、薬品名の変形を使用し、薬品-原因-治癒の関連を表示する必要があります。
   b. 検索したい文書で情報を取得するために必要なテキスト分析の種類を指定する。
   c. コレクションに XML 文書がある場合は、ソリューションで XML マークアップを活用するかどうかを決める。エンタープライズ・サーチでは、以下の 2 つのうちいずれかの方法で XML マークアップを使用することができます。
      - カスタム分析で XML マークアップを使用できる場合 (例えば、文書に、要約またはカテゴリー化アノテーターにおいて便利な `<summary>` または `<topic>` エレメントが含まれている場合) は、XML と共通分析構造とのマッピングを定義します。
      - 照会で、XML マークアップを、文書にある通りに使用したい場合は、ネイティブ XML マッピングを使用可能にします。
   d. 共通分析構造に保管するテキスト分析結果情報を決定する。この情報に、セマンティック検索を使用してアクセスできるようにします。共通分析構造と索引とのマッピングを定義します。
   e. 分析結果をリレーショナル・データベースに保管したいかどうかを判別します。たとえば、レポートまたはデータ・マイニング・アプリケーションを使用して、傾向および関連を発見したい場合などです。共通分析構造と JDBC 表とのマッピングを定義します。
   f. セマンティック検索アプリケーションを設計します。検索ユーザーの、追加のセマンティック検索機能の使用を判別します。ユーザー・インターフェースを設計します。
2. 作成: UIMA Software Development Kit アクティビティー
   a. 個々の分析ステップを定義する。
   b. マッピングおよび分析アルゴリズムのタイプ・システムを記述する。
   c. 分析ステップごとに分析アルゴリズム (アノテーター) を作成し、UIMA Software Development Kit を使用して、分析エンジンにアノテーター

を組み込みます。エンタープライズ・サーチの基本アノテーター・パッケージの基本機能 (言語識別およびトークン化) を使用して、カスタム分析を作成します。

d. UIMA で分析アルゴリズムをテストしたら、分析エンジンを PEAR (Processing Engine Archive) ファイルとしてパッケージします。アーカイブには、ユーザーの分析アルゴリズムのみが含まれるようにし、エンタープライズ・サーチの基本言語機能は含まれないようにしてください。

3. 配置: エンタープライズ・サーチを使用したアクティビティー

a. 分析エンジンのアーカイブ・ファイル (.pear) をエンタープライズ・サーチにアップロードします。エンタープライズ・サーチするで参照できるように、分析コンポーネントの名前を提供します。

b. 1 つ以上の文書コレクションを分析エンジンに関連付ける。

c. 適用できる場合は、コレクションごとに、カスタム分析用に定義した、XML エレメントから UIMA へのタイプ・マッピングをアップロードおよび選択します。

d. 適用できる場合は、コレクションごとに、カスタム分析用に定義したデータベース・マッピングをアップロードおよび選択します。

e. コレクションごとに、セマンティック検索用に定義した、索引マッピング構成をアップロードおよび選択します。

f. 必要に応じて、カスタム・セマンティック検索アプリケーションをセットアップします。たとえば、ブラウザー・ベースの検索ユーザー・インターフェースをアプリケーション・サーバーにデプロイします。

g. セマンティック検索コレクション内の文書を、キーワード・ベースのコレクションで使用するときに、クロール、構文解析および索引付けします。

関連タスク

『エンタープライズ・サーチ・ベース・アノテーターのインストールおよび実行』
エンタープライズ・サーチ・ベース・アノテーター・パッケージを使用して、エンタープライズ・サーチ・アノテーターの出力を基にした新規のアノテーターを作成し、 UIMA Software Development Kit (SDK) 内でカスタム・アノテーターをテストできます。

エンタープライズ・サーチ・ベース・アノテーターのインストールおよび実行

エンタープライズ・サーチ・ベース・アノテーター・パッケージを使用して、エンタープライズ・サーチ・アノテーターの出力を基にした新規のアノテーターを作成し、 UIMA Software Development Kit (SDK) 内でカスタム・アノテーターをテストできます。

一連のベース・アノテーターには、次が含まれます。

- **言語 ID アノテーター**

  文書の言語を検出します。機能および構成パラメーターについては、ディスクリプター・ファイル jlangid.xml を参照してください。

- **FROST 辞書検索アノテーター**

IBM LanguageWare 辞書に基づいた、トークン化およびセンテンス検出を提供します。トークン、追加の言語情報 (基本型または見出し語など) が生成されます。機能および構成パラメーターについては、ディスクリプター・ファイル jfrost.xml を参照してください。

- **空白トークナイザー**

  すべてのヨーロッパ言語の文書、またはその他の空白で分離されたスクリプトで、空白に基づいたトークン化を実行します。さらに、アノテーターはアラビア語、Han 語、ヘブライ語、ひらがな、カタカナ、ラオ語、モンゴル語、タイ語、YI 語およびハングルのテキスト・スクリプトで、 n-gram トークン化を実行できるようにします。機能および構成パラメーターについては、ディスクリプター・ファイル jtok.xml を参照してください。

UIMA 内のこれらのアノテーターを実行するには、UIMA Software Development Kit (SDK) がインストールされている必要があります。これは、IBM developerWorks Web サイト (http://www-128.ibm.com/developerworks/db2/zones/db2ii/) にあります。

エンタープライズ・サーチ・ベース・アノテーター・パッケージは、エンタープライズ・サーチで使用されるテキスト分析アノテーターを含む圧縮ファイルです。これらのアノテーターは、エンタープライズ・サーチで文書の構文解析を行う場合に、カスタム分析の前に実行します。

アノテーター・パッケージをインストールするには、次を行います。

1. *ES_INSTALL_ROOT*/packages/uima ディレクトリーにある、エンタープライズ・サーチ (WebSphere Information Integrator OmniFind Edition) インストール内でアノテーター・パッケージ OF_base_annotators.zip を検索します。
2. ZIP されたファイルを UIMA SDK インストールのルート・ディレクトリーにコピーします。
3. ZIP されたファイルを解凍して、エンタープライズ・サーチ・ベース・アノテーター・ファイルを UIMA SDK インストールの、指定したディレクトリー構造に追加します。

ベース・アノテーター・パッケージのインストール後、アノテーター・ファイルは *UIMA_SDK_INSTALL*/docs/examples/descriptors/analysis_engine フォルダーにあります。of_tokenization.xml ファイルは、エンタープライズ・サーチ内で使用されるシーケンス内のベース・アノテーターをリストします。

ディスクリプター・ファイルは、エンタープライズ・サーチで使用される、同じ構成値を含みます。デバッグ目的用に、 UIMA SDK で値を変更できます。しかし、エンタープライズ・サーチ・システムのディスクリプター・ファイルは変更しないでください。これらのファイルへの変更は、システムの安定性の問題や、パフォーマンス上の問題を引き起こす可能性があります。

エンタープライズ・サーチ・ベース・アノテーター・パッケージは、英語の文書を処理するのに必要な辞書のみを含みます。開発環境で他の言語を処理したい場合は、次のステップにしたがってください。

1. *ES_INSTALL_ROOT*/configurations/parserservice/jediidata/frost/resources のエンタープライズ・サーチ・インストールで、エンタープライズ・サーチ辞書を検索します。

2. ディレクトリーの内容をローカルの UIMA SDK インストール (*UIMA_SDK_INSTALL*/data/frost/resources) にコピーします。

アノテーター・パッケージが正常にインストールされたかどうかを検査するには、次のようにします。

1. *UIMA_SDK_INSTALL*/bin/cvd[.bat/.sh] ディレクトリーにある Open the Common Analysis Structure (CAS) Visual Debugger (CVD) を開きます。
2. 「**実行**」 → 「**TAE のロード**」をクリックします。
3. *UIMA_SDK_INSTALL*/docs/examples/descriptors/analysis_engine ディレクトリーにある of_tokenization.xml と呼ばれるテキスト分析エンジン指定子を選択します。
4. サンプル文書をロードし、テキスト分析エンジンを実行します。タイプ `uima.tt.TokenAnnotation` の注釈は CVD に表示されます。

エンタープライズ・サーチ・アノテーターを処理に使用するには、次のようにします。

1. カスタム・アノテーターが、エンタープライズ・サーチ・アノテーターによって定義されたタイプを使用する場合、カスタム・アノテーター指定子の typeSystem セクションの of_typesystem.xml ファイルへの参照を含めます。of_typesystem.xml ファイルは、 *UIMA_SDK_INSTALL*/docs/examples/descriptors/analysis_engine ディレクトリーにある of_typesystem.xml ファイルにあります。ディスクリプター・ファイルへの参照を含む方法のサンプルについては、 analysis_engine ディレクトリーにある jtok.xml を参照してください。

関連資料

46 ページの『エンタープライズ・サーチに定義されているタイプおよびフィーチャー』
エンタープライズ・サーチに定義されているタイプ・システムは、文書メタデータ処理および基本的な言語分析をカバーします。

テキスト分析アルゴリズム

UIMA Software Development Kit には、アノテーター (タイプ・システム記述が含まれている分析アルゴリズム) を作成して、分析エンジンに組み込むための API およびツールが組み込まれています。

UIMA 文書には、これらのコンポーネントの作成に役立つチュートリアル・スタイルのガイドが組み込まれています。 Software Development Kit には、テストや結果の表示を行うユーティリティーや、分析結果の索引を作成する小規模なセマンティック検索エンジンが組み込まれています。索引に保管されている情報について、より高度な検索を行うこともできます。

UIMA Software Development Kit には、事前に構成されている分析エンジンはありません。ただし、 UIMA 環境においてエンタープライズ・サーチで提供される基本的なアノテーターを使用することはできます。ご使用の UIMA 環境でテキスト分析アルゴリズムを作成する前に、言語検出機能およびトークン化機能を組み込む方法を UIMA 文書で確認してください。

UIMA Software Development Kit を使用して分析エンジンの開発およびテストを完了し、エンタープライズ・サーチでドキュメント・コレクションにおいてそのアル

ゴリズムを実行するには、PEAR (Processing Engine ARchive) ファイルを作成する必要があります。このアーカイブ・ファイルには、エンタープライズ・サーチにおける分析エンジンとしてユーザーが作成したカスタム分析機能を配置する際に必要なリソースがすべて含まれています。アーカイブ作成に必要なすべての処理ステップは、Software Development Kit に含まれている UIMA 文書に記載されています。

アーカイブには、必ずカスタム分析が含まれている必要があります。そのカスタム分析が、エンタープライズ・サーチで提供されている基本的な言語機能に基づいている場合でも、アーカイブに含まれていなければなりません。カスタム分析を実行する前に、常に、基本的なエンタープライズ・サーチの分析ステップが実行されます。

エンタープライズ・サーチでセマンティック検索ソリューションを構成およびデプロイする方法を学習するには、http://www.ibm.com/developerworks/db2/zones/db2ii/ に記載されているチュートリアルを実行してください。このチュートリアルは、エンタープライズ・サーチのカスタム・テキスト分析アルゴリズムのデプロイに含まれるステップをガイドし、検索結果を改善するために、照会で分析結果を使用する方法を示します。

関連タスク

6 ページの『エンタープライズ・サーチ・ベース・アノテーターのインストールおよび実行』
エンタープライズ・サーチ・ベース・アノテーター・パッケージを使用して、エンタープライズ・サーチ・アノテーターの出力を基にした新規のアノテーターを作成し、 UIMA Software Development Kit (SDK) 内でカスタム・アノテーターをテストできます。

タイプ・システム記述

タイプ・システム記述は、カスタム分析で使用されるフィーチャー構造 (分析結果を示す基礎となるデータ構造) を記述します。

タイプ・システム記述に定義されている同じタイプは、アノテーター (分析アルゴリズム) を持つ分析エンジンによって使用されることが必要です。また、その同じタイプは、カスタム分析に関連付けられた、すべてのマッピング・ファイルでも使用されることが必要です。これらには、XML マッピング構成ファイル、索引作成構成ファイル、および JDBC 対応データベース構成マッピング・ファイルを含みます。

アノテーターのタイプ・システム記述は、アノテーターの記述の一部にするか、別のタイプ・システム記述ファイルに含めることができます。これは、同じ分析エンジンに含まれる他のアノテーターのディスクリプターの一部の場合もあります。

タイプ・システム記述は、 UIMA 環境からエンタープライズ・サーチにインポートされる分析エンジン・アーカイブ (.pear ファイル) の一部でなければなりません。

次の同じサンプル・タイプ・システム記述は、カスタム分析で選択できるマッピングの異なるタイプについての、すべてのトピックで使用されます。

次のタイプ・システム記述のサンプルは、容疑者、犯罪発生場所、犯罪時刻、犯罪日付の情報を含む、警察レポートを記述します。

```
<?xml version="1.0" encoding="UTF-8"?>
<typeSystemDescription>
   <name>Police Reports Type System</name>
   <description>Type system description for
        police reports</description>
   <version>1.0</version>
   <types>
     <typeDescription>
       <name>com.ibm.omnifind.types.PoliceReport</name>
       <description>Annotates a police report</description>
       <superTypeName>uima.tcas.Annotation</superTypeName>
       <features>
         <featureDescription>
           <name>time</name>
           <description>Time the crime was reported to have happened
              </description>
           <rangeTypeName>com.ibm.omnifind.types.Time</rangeTypeName>
         </featureDescription>
         <featureDescription>
           <name>date</name>
           <description>When the crime happened</description>
           <rangeTypeName>com.ibm.omnifind.types.Date</rangeTypeName>
         </featureDescription>
         <featureDescription>
           <name>location</name>
           <description>Where the crime took place</description>
           <rangeTypeName>com.ibm.omnifind.types.City</rangeTypeName>
         </featureDescription>
         <featureDescription>
           <name>knownSuspects</name>
           <description>Contains annotations of type Suspect</description>
           <rangeTypeName>uima.cas.FSArray</rangeTypeName>
         </featureDescription>
         <featureDescription>
           <name>crimeDescription</name>
           <description>Short description of the crime</description>
           <rangeTypeName>uima.cas.String</rangeTypeName>
         </featureDescription>
       </features>
     </typeDescription>
     <typeDescription>
       <name>com.ibm.omnifind.types.City</name>
       <description>The name of a city</description>
       <superTypeName>uima.tcas.Annotation</superTypeName>
       <features>
         <featureDescription>
           <name>cityName</name>
           <description>The name of the city</description>
           <rangeTypeName>uima.cas.String</rangeTypeName>
         </featureDescription>
         <featureDescription>
           <name>cityDistrict</name>
           <description>The name of the district</description>
           <rangeTypeName>uima.cas.String</rangeTypeName>
         </featureDescription>
       </features>
     </typeDescription>
     <typeDescription>
       <name>com.ibm.omnifind.types.Person</name>
       <description>A person annotation</description>
       <superTypeName>uima.tcas.Annotation</superTypeName>
       <features>
         <featureDescription>
           <name>role</name>
           <description>For example, suspect or witness</description>
           <rangeTypeName>uima.cas.String</rangeTypeName>
         </featureDescription>
```

```
    <featureDescription>
      <name>firstName</name>
      <description>The first name of the person</description>
      <rangeTypeName>uima.cas.String</rangeTypeName>
    </featureDescription>
    <featureDescription>
      <name>surName</name>
      <description>The surname of the person</description>
      <rangeTypeName>uima.cas.String</rangeTypeName>
    </featureDescription>
    <featureDescription>
      <name>title</name>
      <description>For example, Mr. or Ms.</description>
      <rangeTypeName>uima.cas.String</rangeTypeName>
    </featureDescription>
    <featureDescription>
      <name>gender</name>
      <description>Male or female</description>
      <rangeTypeName>uima.cas.String</rangeTypeName>
    </featureDescription>
  </features>
</typeDescription>
<typeDescription>
  <name>com.ibm.omnifind.types.Suspect</name>
  <description>A found suspect</description>
  <superTypeName>com.ibm.omnifind.types.Person</superTypeName>
  <features>
    <featureDescription>
      <name>description</name>
      <description>Suspect description,
    for example, bearded with dark glasses</description>
      <rangeTypeName>uima.cas.String</rangeTypeName>
    </featureDescription>
  </features>
</typeDescription>
<typeDescription>
  <name>com.ibm.omnifind.types.Date</name>
  <description>A date</description>
  <superTypeName>uima.tcas.Annotation</superTypeName>
  <features>
    <featureDescription>
      <name>year</name>
      <description>The year, for example, 2005</description>
      <rangeTypeName>uima.cas.Integer</rangeTypeName>
    </featureDescription>
    <featureDescription>
      <name>month</name>
      <description>The month in digits, for example, 7</description>
      <rangeTypeName>uima.cas.Integer</rangeTypeName>
    </featureDescription>
    <featureDescription>
      <name>day</name>
      <description>The day in digits</description>
      <rangeTypeName>uima.cas.Integer</rangeTypeName>
    </featureDescription>
    <featureDescription>
      <name>dayOfWeek</name>
      <description>The day of the week, for example, Monday</description>
      <rangeTypeName>uima.cas.String</rangeTypeName>
    </featureDescription>
    <featureDescription>
      <name>quarter</name>
      <description>The quarter, for example, Q1-2005</description>
      <rangeTypeName>uima.cas.String</rangeTypeName>
    </featureDescription>
    <featureDescription>
      <name>englDate</name>
```

```
            <description>Date as mm/dd/yyyy</description>
            <rangeTypeName>uima.cas.String</rangeTypeName>
          </featureDescription>
        </features>
      </typeDescription>
      <typeDescription>
        <name>com.ibm.omnifind.types.Time</name>
        <description>A time</description>
        <superTypeName>uima.tcas.Annotation</superTypeName>
        <features>
          <featureDescription>
            <name>hours</name>
            <description>Hours from 00-23</description>
            <rangeTypeName>uima.cas.Integer</rangeTypeName>
          </featureDescription>
          <featureDescription>
            <name>minutes</name>
            <description>Minutes in the hour</description>
            <rangeTypeName>uima.cas.Integer</rangeTypeName>
          </featureDescription>
          <featureDescription>
            <name>timeOfDay</name>
            <description>Time periods, such as morning, noon</description>
            <rangeTypeName>uima.cas.String</rangeTypeName>
          </featureDescription>
        </features>
      </typeDescription>
    </types>
</typeSystemDescription>
```

関連資料

46 ページの『エンタープライズ・サーチに定義されているタイプおよびフィーチャー』

エンタープライズ・サーチに定義されているタイプ・システムは、文書メタデータ処理および基本的な言語分析をカバーします。

49 ページの『UIMA に定義されているタイプおよびフィーチャー』

UIMA Software Development Kit は、テキスト分析時に文書内で検出される可能性がある基本的な言語タイプおよびフィーチャーを定義します。

分析および検索における XML マークアップ

文書内の XML 構造の情報を、UIMA アノテーターを作成せずに、共通分析構造に直接マップすることができます。

コレクション内の文書が XML で、テキスト分析またはセマンティック検索中に XML マークアップを活用したい場合、次のオプションがあります。

ネイティブ XML 検索

セマンティック検索中に文書が表示されるときに、すべての XML タグおよび属性を使用したい場合に、このオプションを使用します。たとえば、`<addressee>` エレメントを含む請求書がある場合、ネイティブ XML 検索を使用可能にすると、このエレメント内の特定のカスタマー名の検索に、セマンティック検索照会でこのタグを使用できるようになります。

このオプションで、文書の XML 構造は `com.ibm.es.tt.MarkupTag` タイプを使用する共通分析構造に表示されます。各 XML タグに、このタイプの

注釈が作成されます。この注釈には、タグの名前、その属性および属性内容が含まれます。この情報は常に索引付けされ、セマンティック検索にアクセス可能です。

ネイティブ XML 検索は、マッピング構成ファイルを必要としません。エンタープライズ・サーチの管理コンソールから、ネイティブ XML 検索を使用可能にできます。

XML エレメントから UIMA へのタイプ・マッピング

次のケースでこのオプションを使用します。

- 特定の XML エレメントのセマンティクスが正確で、今後のテキスト分析ステップで使用できる場合です。これらの分析ステップは、注釈および XML 構造から作成されたフィーチャーで直接操作でき、元の文書の潜在的に異なるフォーマットからシールドされます。例えば、請求に関する文書のエレメント `<addressee>` に、常に顧客名が含まれています。XML エレメントからタイプへのマッピングを使用して、このエレメントの内容はタイプ `Customer` の注釈に直接マップできます。アノテーターは、その後 `Customer` 注釈周辺の情報を使用して、顧客-場所-関係を推測します。
- XML 入力の特定の指定された領域への、カスタム・アノテーターの処理の有効範囲を制限したい場合があります。たとえば、車の問題を検出するアノテーターの、 `<technicianComment>` タグの内容を制限できます。
- テキスト分析処理と以降の検索を、XML 文書の特定の部品に制限し、不適切またはテキストでないものをフィルターに掛けたい場合です。
- 異なる名前 (たとえば、`<mainHeading>` または `<doc>`) を持つ XML タグをセマンティック検索で使用される共通スパン (たとえば、タイトル) にマップしたいとします。

これらのケースでは、フィーチャー構造を定義する XML から UIMA へのタイプ・マッピング構成ファイルを作成しなくてはなりません。構成ファイルで定義するフィーチャー構造は、文書が構文解析され、カスタム・アノテーターによってアクセスされた時に作成されます。

文書のコレクションに対して複数の構成ファイルを使用できます。どの構成をどの XML 文書に使用するかは、`<identifier>` エレメントで判別します。構成ファイル内の `<identifier>` エレメントは、XML 文書のルート・エレメントに一致している必要があります。例えば、ユーザー文書のルート・エレメントが `doc` である場合、構成ファイル内の `<identifier>` エレメントの値も「doc」でなければなりません。

一致するものが見つからない場合、プログラムは、構成ファイル内で Default に設定されている `<identifier>` エレメントを検索します。デフォルト構成が見つからない場合は、文書の本文 (タグ情報が無い部分) が共通分析構造の文書注釈にマップされます。

文書の関係のある部分のみに含まれている情報を抽出し、関係の無い部分は無視したい場合は、文書内のどの XML エレメントに関連情報が含まれているかを指定するだけでかまいません。これは、コンテンツ抽出と呼ばれます。例えば、title および body エレメントに指定されている入力情報を抽出して、author、date、ID、publisher は無視することができます。

コンテンツ抽出は、以下のタイプの XML 文書の分析処理を向上させることができます。

- 分析に適合しないコンテンツが大量にある文書 (例えば、バイナリー添付ファイルなど)。コンテンツ抽出を使用することにより、文書サイズがかなり削減され、処理が速くなり、不適合データによる分析エラーが回避されます。
- 文書テキストと関係の無いテキストが混在している文書 (例えば、<note> タグ内に編集情報が含まれている文書など)。この情報を無視することにより、文書コンテンツの分析時により良い結果が得られます。

ネイティブ XML 検索および XML エレメント UIMA からへのタイプ・マッピングのコンテンツ抽出オプションは、すべてのコンテンツを対象とするか、指定されたコンテンツのみを対象とするかという点で互いに矛盾しています。コンテンツ抽出を指定すると、ネイティブ XML マッピングは無視されます。内容抽出しない場合は、XML エレメントから UIMA へのタイプ・マッピングおよびネイティブの XML 検索の両方を持つことができます。

構成ファイルで使用するすべてのタイプおよびフィーチャーは、カスタム分析ステップのシステム記述に示されていなければなりません。ご使用の UIMA 環境で、Component Descriptor Editor Eclipse プラグインを使用して、タイプ・システム記述子を作成することができます。このプラグインによって、必要な XML 構文を知る必要なく、記述子ファイルを作成することができます。

カスタム分析の作成およびテストが完了したら、UIMA PEAR (Processing Engine ARchive) 生成ウィザードを使用して、タイプ・システム記述が組み込まれたカスタム分析ファイルが含まれるアーカイブを作成します。

その後、エンタープライズ・サーチの管理コンソールを使用して、カスタム分析アーカイブと UIMA タイプ・マッピング構成ファイルへの XML エレメントを、エンタープライズ・サーチにアップロードします。

関連タスク

『XML から UIMA タイプのマッピング構成ファイルの作成』
XML から UIMA へのマッピング構成ファイルでは、XML を UIMA データ・タイプに マッピングするための全範囲の構成オプションを使用できます。

XML から UIMA タイプのマッピング構成ファイルの作成

XML から UIMA へのマッピング構成ファイルでは、XML を UIMA データ・タイプにマッピングするための全範囲の構成オプションを使用できます。

このタスクについて

XML から UIMA へのタイプ・マッピング構成ファイルは、以下の例に示すスキーマに適合する必要があります。

サンプルの XML 警察レポートは、犯罪のタイプ、犯罪の日付、犯罪の発生場所、報告した警察官、警察官の所属地区、容疑者の説明、および要約の XML タグがあります。この後に本体セクションが続きます。例えば、以下のようになります。

```
<report>
 <doc>
  <crimeType>Car theft</crimeType>
  <crimeDate>04/23/05 09:23 pm</crimeDate>
```

```
  <crimeLocation>27 Main Street, Brynston, Springfield, New Jersey</crimeLocation>
  <reportingOfficer rank="Lt">Jakob
   <lastname>Collins</lastName>
  </reportingOfficer>
  <policePrecinct>14th Precinct</policePrecinct>
  <suspectDescription>Male, dark haired, dark glasses,
    blue jeans with dark, probably black,
    jacket</suspectDescription>
  <abstract>A Mercedes CLK was stolen on 04/23/2005 from a parking
   lot in front of the Blue Lagoon restaurant on
   27 Main Street, Brynston.(serial number: 32 2761 50871)</abstract>
  <body>A Mercedes CLK was stolen on 04/23/2004 from a parking
   lot in front of the Blue Lagoon restaurant on 27 Main Street,
   Brynston.(serial number: 32 2761 50871)

It has a black color and wide Michelin tires.

Eyewitnesses in front of the restaurant saw two darkly dressed
males drive away in the car at high speed. The car was
found abandoned on Aliway Ave in Brooklyn. The fuel tank was empty.
The seats were badly stained and the back seat was vandalized.
Nothing was stolen out of the car....</body>
 </doc>
 <image>
  </-- image of the crime scene as a base64-encoded string -->
 </image>
</report>
```

サンプルのレポートに基づいて、構成ファイルには次の構造がある可能性があります。サンプルは、警察レポート・シナリオに定義されたタイプ・システムを使用します。

```
<?xml version="1.0"?>
<xmlCasInitializerConfiguration
 xmlns="http://www.ibm.com/2005/uima/jedii_ci_xml">

 <identifier>Default</identifier>
 <description>Sample configuration</description>

 <contentElements>
   <element>/report/doc</element>
 </contentElements>

 <elementToTypeMappings>
   <elementToTypeMapping>
     <element>//doc//reportingOfficer</element>
     <type>com.ibm.omnifind.types.Person</type>
     <featureValueAssignment>
       <feature>role</feature>
       <basicValue default="Reporting officer">
           </basicValue>
     </featureValueAssignment>
     <featureValueAssignment>
       <feature>gender</feature>
       <basicValue default="male"
          useAttributeValue="sex"/>
     </featureValueAssignment>
     <featureValueAssignment>
       <feature>surName</feature>
       <values concatenate="true" delimiter=""/>
         <basicValue useAttributeValue="rank"
                 default="Lt"/>
         <basicValue useElementContent="lastName"/>
       </values>
     </featureValueAssignment>
   </elementToTypeMapping>
```

```
   <elementToTypeMapping>
     <element>//doc</element>
     <type>com.ibm.omnifind.types.PoliceReport</type>
     <featureValueAssignment>
       <feature>crimeDescription</feature>
       <basicValue useElementContent="abstract"
                trim="true">
           </basicValue>
     </featureValueAssignment>
   </elementToTypeMapping>
 </elementToTypeMappings>

</xmlCasInitializerConfiguration>
```

制約事項

XML マッピング構成ファイルは、次の 2 つのセクションに分割されています。

<contentElements> element

特定内容の抽出が必要な場合に、このエレメントを使用します。サンプル構成ファイルは、文書の <doc> セクションの内容を抽出し、文書の他のセクションは無視します。XML 警察レポートでは、イメージは大きく、テキスト処理に役立つわけではないかもしれません。 <image> ではなく、<doc> を内容エレメントとして指定することで、テキスト処理の開始前にイメージはフィルタリングされます。

<elementToTypeMappings>

文書内の個々のどの XML エレメント (<elementToTypeMapping> エレメントで指定されています) を共通分析構造のどのフィーチャー構造にマップするかを指定するには、このエレメントを使用します。

コンテンツ抽出オプションを使用する場合には、<elementToTypeMappings> セクションに指定されている XML エレメントが、<contentElements> セクションに指定されている XML エレメント内に含まれていなければなりません。

手順

XML から UIMA へのタイプ・マッピング構成ファイルを作成するには、次のようにします。

1. XML ファイルを作成します。 XML 構文エラーを避けるために、 XML を妥当性検査する XML エディターまたは XML オーサリング・ツールを使用します。構成ファイルの XSD スキーマは、configuration.xsd と呼ばれ、*ES_INSTALL_ROOT*/packages/uima/ にあるエンタープライズ・サーチのインストールに含まれています。
2. マッピングは必ず、<xmlCasInitializerConfiguration xmlns="http://www.ibm.com/2005/uima/jedii_ci_xml"> エレメントに入れてください。名前空間 (xmlns 属性に指定) は、表示どおりでなければなりません。
3. 特定の内容を、文書のセクションおよび、どの各 XML エレメントを、共通分析領域のどのフィーチャーにマップするを指定する <elementToTypeMappings> エレメントから抽出したい場合に、 <contentElements> エレメントを追加します。
4. <identifier> エレメントおよび <description> エレメントを追加します。 ID は、どの構成をどの XML 文書に使用するかを判別します。 ID には、doc など

の文書のルート・エレメントが含まれていなければなりません。 ID が Default に設定されている場合、文書のルート・エレメントは不適切であり、構成マッピングは XML 文書に適用されます。

5. 文書の関係のある部分のみに含まれている情報を抽出したい場合は、<contentElements> エレメントを追加します。これには、以下のコンポーネント・エレメントがあります。
   - 文書および続く XPath 構文内の XML 文書のパスを含む、1 つ以上の <element> エレメント。たとえば、<element>/doc/crimeType</element> です。
6. 文書内のどの XML エレメントを、共通分析構造内のどのフィーチャー構造にマップするかを指定する <elementToTypeMappings> エレメントを追加します。これには、以下のコンポーネント・エレメントがあります。
   - 1 つ以上の <elementToTypeMapping> エレメント。このエレメントには、以下のネストされたエレメントが含まれている必要があります。
     - <element> エレメント。これは、XML エレメントのパスを指定するために使用され、XPath 構文に従います。先頭のスラッシュ (/) は、絶対パスが指定されていることを意味します。たとえば、ルート・エレメント doc の下にある abstract です。 2 つのスラッシュ (//) は、パスのサブセットを意味します。たとえば、birthDate は reportingOfficer 内になければなりませんが、他のエレメントはこれら 2 つの間にあればかまいません。
     - <type> エレメント。これは、タイプ・システム記述に定義されているタイプを指定します。これは Annotation タイプでなければなりません。
     - ゼロ以上の <featureValueAssignment> エレメント。
7. <featureValueAssignment> エレメントでは、 <feature> エレメントの String タイプのフィーチャーに名前を付け、 <basicValue> エレメントに値を割り当てます。複数の <basicValue> エレメントを、<values> エレメントの間に追加できます。

   <basicValue> エレメントは、属性を持つことができます。それには、useAttributeValue、useElementContent、 default、および trim が含まれます。

フィーチャーの値として属性の値を使用したい場合は、useAttributeValue を使用します。以下は例です。

```
<elementToTypeMapping>
  <element>/doc//reportingOfficer</element>
  <type>com.ibm.omnifind.types.Person</type>
  <featureValueAssignment>
    <feature>role</feature>
    <basicValue default="Reporting officer"/>
  </featureValueAssignment>
  <featureValueAssignment>
    <feature>gender</feature>
    <basicValue default="male" useAttributeValue="sex"/>
  </featureValueAssignment>
</elementToTypeMapping>
```

結果出力は、次のようになります。

- 文書内の <doc> XML タグ内のどこかにある各 <reportingOfficer> XML タグごとに、フィーチャー構造タイプ com.ibm.omnifind.types.Person が作成されます。
- <reportingOfficer> タグに sex 属性が含まれている場合、新規に作成されたフィーチャー構造のフィーチャー gender は、属性の値に設定されます。

フィーチャーの値としてコンテンツを追加するには、useElementContent 属性を使用します。例えば、以下の構成断片の場合、

```
<elementToTypeMapping>
  <element>//doc</element>
  <type>com.ibm.omnifind.types.PoliceReport</type>
  <featureValueAssignment>
    <feature>crimeDescription</feature>
    <basicValue useElementContent="abstract" trim="true"/>
  </featureValueAssignment>
</elementToTypeMapping>
```

<doc> 内の <abstract> エレメントでカバーされたテキストは、フィーチャー構造 crimeDescription の値になります。すべての前後のブランクは除去されます。

以下のケースでは、<values> エレメントの間に複数の値が指定できます。

- 設定されるフィーチャーが StringArray タイプの場合。
- 多数のストリングが区切り文字属性を使用して 1 つのストリングに連結され、String タイプのフィーチャーにマップする場合。以下の例では、Mr. というタイトルが定数で、ファーストネームが属性の値で、ラストネームが XML エレメントによってカバーされます。

```
<elementToTypeMapping>
  <element>//doc//reportingOfficer</element>
  <type>com.ibm.omnifind.types.Person</type>
  <featureValueAssignment>
    <feature>surName</feature>
    <values concatenate="true" delimiter=" ">
      <basicValue default="Mr."/>
      <basicValue useAttributeValue="rank"
         default="Lt."/>
      <basicValue useElementContent="lastName"/>
    </values>
  </featureValueAssignment>
</elementToTypeMapping>
```

ストリング・フィーチャー値は、構成ファイルからそのまま抽出されます。値は、先頭および末尾のブランクが入ったままになります。ただし、タイプおよびフィーチャーの名前には、ブランクが挿入されます。たとえば、
<type> **com.ibm.omnifind.types.Person** </type> は
<type>**com.ibm.omnifind.types.Person**</type> になります。

<condition> エレメントを使用して、属性に条件を設定することができます。例えば、com.ibm.omnifind.types.Person タイプのフィーチャー構造は、 armed 属性が yes に設定されている <suspectDescription> が文書内にある場合にのみ作成されます。

```
<elementToTypeMapping>
  <element>//suspectDescription</element>
  <type>com.ibm.omnifind.types.Person</type>
  <condition attribute="armed"  value="yes"/>
</elementToTypeMapping>
```

サンプルの警察レポートおよび定義されたマッピング構成ファイルに基づいて、以下のフィーチャー構造が作成されます。

com.ibm.omnifind.types.PoliceReport

- covered text: ″Car theft 04/23/05 09:23 pm 27 Main Street, Brynston, Springfield, New Jersey Jakob Collins 14th Precinct Male, dark haired, dark glasses, blue jeans with dark, probably black, jacket A Mercedes CLK was ... Nothing was stolen out of the car.
- begin = 2
- end = 904
- knownSuspects = null
- crimeDescription = ″A Mercedes CLK was stolen on 04/23/2005 from a parking lot in front of the Blue Lagoon restaurant on 27 Main Street, Brynston.(serial number: 32 2761 50871)″

com.ibm.omnifind.types.Person

- covered text = ″Jakob Collins″
- begin = 112
- end = 127
- role = ″Reporting officer″
- firstName = null
- surName = ″Lt Collins″
- gender = ″male″

XML ファイルの作成後に、これをエンタープライズ・サーチにアップロードし、エンタープライズ・XML 管理コンソールを使用して、他のカスタム分析選択のある XML 文書マッピング構成ファイルを選択する必要があります。

関連概念

12 ページの『分析および検索における XML マークアップ』
文書内の XML 構造の情報を、UIMA アノテーターを作成せずに、 共通分析構造に直接マップすることができます。

関連資料

9 ページの『タイプ・システム記述』
タイプ・システム記述は、カスタム分析で使用されるフィーチャー構造 (分析結果 を示す基礎となるデータ構造) を記述します。

テキスト分析結果

すべてのテキスト分析結果は、共通分析構造に保管されます。

アノテーターは、通常共通分析構造の読み取りおよび書き込みを行います。共通分析構造コンシューマー (*CAS コンシューマー*) は、共通分析構造に格納された分析結果の、最終処理を行います。エンタープライズ・サーチには、以下の 2 つの CAS コンシューマーが含まれています。

- 検索エンジン内の共通分析構造の内容に索引付けするコンシューマー。このコンシューマーは、エンタープライズ・サーチ管理コンソールのカスタム・テキスト分析で選択した、索引作成構成ファイルを必要とします。
- リレーショナル・データベースに特定の分析結果を追加するコンシューマー。このコンシューマーは、エンタープライズ・サーチ管理コンソールのカスタム・テキスト分析オプションで選択する構成ファイルも必要とします。

CAS コンシューマーは、共通分析構造からのみ読み取ります。

必要な場合、カスタム CAS コンシューマーをエンタープライズ・サーチにデプロイできます。コンシューマーの書き込み方法については、UIMA 文書を参照してください。コンシューマーのアップロードおよびエンタープライズ・サーチでの使用方法を学習するには、 IBM UIMA developerWorks Web サイト (http://www.ibm.com/developerworks/db2/zones/db2ii/) を参照してください。

関連概念

25 ページの『カスタム分析結果の索引マッピング』
文書のコレクションにカスタム分析を実行した後、 エンタープライズ・サーチの検索エンジンを使用して、カスタム分析アルゴリズムによって生成される 共通分析構造に保管された情報から索引を作成することができます。

33 ページの『選択した分析結果のデータベース・マッピング』
エンタープライズ・サーチで文書のコレクションのカスタム分析を実行した後で、 選択したテキスト分析結果を JDBC 対応データベースに保管できます。

フィーチャー・パス

フィーチャー・パスは、共通分析構造内のフィーチャー値にアクセスする方法を提供します。これは、XML 文書で XML エレメントにアクセスする際に使用する XPath ステートメントに似ています。

フィーチャー・パスは、複雑なフィーチャー (たとえば、配列値または他のフィーチャー構造に対するフィーチャーなど) を結合するフィーチャー構造にアクセスしたい場合に便利です。フィーチャー・パスを使用して、フィーチャーの値を直接フィーチャー構造に関連付け、この値をセマンティック検索索引またはデータベースに保管できます。

例えば、車と車の製造会社を示すアノテーターの場合を考えてみます。アノテーターは、`make` という属性を持つ `car` タイプの注釈を作成します。ただし、`make` には、実際の会社名 (例えば、 Chevrolet など) は含まれていませんが、それ自体が `companyname` というストリング値属性を持つ `Company` というタイプのフィーチャー構造が含まれています。車の名前と会社名を結合するセマンティック照会を可能にするには、フィーチャー・パス `make/companyname` を使用して、値 `companyname` を車の注釈用に生成される車のスパンに結び付けます。これにより、'/car[@make="Chevrolet"]' という構文を使用して「Chevrolet 社製造の車が含まれている文書を探す」という照会を行うことができます。

フィーチャー・パスは、以下のプロパティーを持つ一連のフィーチャー名 (f1/.../fn) です。

- フィーチャー・パスの値は、String、Integer、Float、またはこれらのタイプのいずれかの配列にできます。

- f1 から fn-1 までのパス内のすべてのフィーチャーは、複合タイプを持つ必要があります。これは、uima.cas.TOP, uima.cas.FSArray タイプ、uima.cas.FSList タイプ、またはこれらのサブタイプの 1 つです。
- パスの最後のフィーチャー fn は、複合タイプを含むことができます。さらに、uima.cas.Float、uima.cas.Integer、uima.cas.String、uima.cas.FloatArray、uima.cas.IntegerArray、uima.cas.StringArray、uima.cas.FloatList、uima.cas.IntegerList、または uima.cas.StringList の (サブ)タイプを含むことができます。
- オプションで、フィーチャー名にタイプを指定できます。フィーチャー名の前に、完全修飾タイプ名を指定し、コロンで区切る必要があります。たとえば、f1/com.ibm.es.SomeType:f2/.../fn のように指定します。

特定のフィーチャーのタイプの有効範囲を絞り込むことができます。例えば、uima.cas.TOP タイプの additionalInfo というフィーチャーの場合を考えてみます。 additionalInfo フィーチャーの値が、実際にフィーチャー salary を持つ EmployeeInfo タイプであることがわかっている場合には、additionalInfo/EmployeeInfo:salary を使用してこのフィーチャーにアクセスできます。この例では、フィーチャー・パス additionalInfo/salary はエラーになります。これは、salary が uima.cas.TOP タイプに対して定義されていないからです。

配列値またはリスト値を持つフィーチャーは、以下の追加プロパティーを持ちます。

- ブラケット ([<番号>]) を使用して、配列またはリストの特定のエレメントを選択します。配列はゼロ (0) から開始します。たとえば、companies 配列内の最初のエレメントを選択する場合には、companies[0] を使用します。配列のサイズに関係なく、配列内の最後のエントリーを選択する場合は、特殊マーカー [last] を使用できます。例えば、 companies[last] のように指定します。
- すべてのエレメントを表示するには、空のブラケット ([]) を使用します。フィーチャー・パスには、空のブラケット ([]) は 1 つだけ使用できます。たとえば、容疑者の配列がある場合、フィーチャー・パス knownSuspects[]/com.ibm.omnifind.types.Suspect:surName は容疑者のすべてのラストネームを String 配列に収集します。
- 配列を戻すフィーチャー・パスは、索引付け中に使用され、配列エレメントは連結 (空白で区切られる) され、索引に単一、複数用語属性、またはフィールドとして書き込まれます。
- フィーチャー・パスの次のエレメントには、タイプを指定しなければなりません。タイプ名は、配列内のエレメントのタイプです。例えば、Info タイプのフィーチャー構造の場合を考えてみます。このタイプには、範囲が FSArray である companies というフィーチャーがあります。配列のエレメントは Company タイプです。Company には、 profit というフィーチャーがあります。3 番目の会社の収益 (profit) 情報を獲得するには、 companies[3]/Company:profit (通常、完全修飾タイプ名を使用します) と指定します。

組み込みフィーチャー

組み込みフィーチャーは、特殊なセマンティクスを持つ、事前定義されたフィーチャー名です。これは、フィーチャー構造自体、たとえばフィーチャー構造のタイプ

または注釈のカバー・テキストなどに含まれない情報へのアクセスに使用できます。これらは、最後または単一エレメントとして、フィーチャー・パスで使用できます。

次の組み込みフィーチャーが、両方のマッピング構成ファイルで使用可能です。

- `fsId()` は、フィーチャー構造の ID を戻します。戻される ID は整数 (32 ビット) です。この組み込みフィーチャーを使用して、照会に完全に一致する文書の部分にアクセスします。
- `typeName()` は、共通分析構造オブジェクト・タイプをストリングとして戻します。タイプは、名前空間接頭部を含む完全修飾タイプ名 (たとえば、`uima.tcas.Annotation`) です。データベース・コンテキストでは、 `typeName()` は、同じ列にタイプおよびサブタイプを保管し、注釈またはフィーチャー構造の実際のタイプを知りたい場合に、特に便利です。次の例は、*suspect* または *witness* などの人物のタイプを、役割列に保管します。

```
<explicitMappingRule  applyToSubTypes="false">
   <type>com.ibm.omnifind.types.Person</type>
   <table>sample.person</table>
   <featureMappings>
     <featureMapping>
       <feature>typeName()</feature>
       <column>role</column>
     </featureMapping>
   </featureMappings>
 </explicitMappingRule>
```

- `coveredText()` は共通分析オブジェクトによってスパンされるテキストを戻します。`coveredText()` は、注釈およびそのサブタイプにのみ使用可能です。注釈タイプによって組み込まれていないフィーチャー構造では、この組み込みフィーチャーは使用しないでください。次の例は、容疑者の名前を suspectName 列に保管します。

```
<implicitMappingRule applyToSubTypes="false">
   <type>com.ibm.omnifind.types.Suspect</type>
   <relation>sample.person</relation>
   <featureMappings>
     <featureMapping>
       <feature>coveredText()</feature>
       <column>suspectName</column>
       <length>128</length>
     </featureMapping>
   </featureMappings>
 </implicitMappingRule>
```

- `[]` は、現行のコンテナー・エントリー (配列またはリスト) へのハンドルを戻します。このフィーチャーは、反復を意味します。これは、配列またはリスト内の各エレメントのデータベース表または索引に作成されたエントリーを意味します。次の例は、組み込みフィーチャー [:index] も許可されている JDBC 構成ファイルからのものです。

```
<implicitMappingRule applyToSubTypes="false">
   <type>uima.cas.FSArray</type>
   <table>sample.knownSuspects</table>
   <featureMappings>
     <featureMapping>
       <feature>uniqueId()</feature>
       <column>arrayId</column>
     </featureMapping>
     <featureMapping>
       <feature>[:index]</feature>
```

```
      <column>arrayIndex</column>
    </featureMapping>
    <featureMapping>
      <feature>[]/com.ibm.omnifind.types.Suspect:uniqueId()</feature>
      <column>suspectId</column>
    </featureMapping>
  </featureMappings>
</implicitMappingRule>
```

次の組み込みフィーチャーが、JDBC マッピング構成ファイルでのみ使用可能です。

- uniqueId() は、フィーチャー構造のグローバル固有 ID を戻します。戻される固有 ID は、固定長 (27 文字) のストリングで、fsId()、 docId()、docTimestamp()、および現行チャンクの数の結果の連結です。これは、文書はエンタープライズ・サーチの複数の共通分析構造でチャンクにできるからです。

  戻されるストリングには、「a から z」および「A から Z」、「0 から 9」の数字、セミコロン (";")、およびコロン (":") を含むことができます。

  uniqueId() の結果は、表の主キーとして使用できます。

- objectId() は、注釈またはフィーチャー構造の ID を戻します。 objectId() は uniqueId() に似ています。 docTimestamp() の結果を含んでいません。戻される ID は、文書が一度構文解析されるコレクションでのみ固有です。すべての文書および文書のバージョン全体で固有性が必要な場合は、uniqueId() を使用する必要があります。

  組み込みフィーチャー objectId() の戻されるストリングは、16 文字の固定長で、「a から z」および「A から Z」、「0 から 9」の数字、セミコロン (";")、およびコロン (":") を含むことができます。

  uniqueId() または objectId() が空のフィーチャー構造を参照する場合、データベース表定義に定義されているデフォルト値が採用され、参照されたタイプの空でないオブジェクトが保管されます。

- docId() は文書 ID を戻します。戻り値は整数タイプ (32 ビット) です。

  以下の例は組み込みフィーチャーを示しています。

```
<explicitMappingRule applyToSubTypes="true">
   <type>com.ibm.omnifind.types.PoliceReport</type>
   <table>sample.PoliceReport</table>
   <featureMappings>
     <featureMapping>
       <feature>uniqueId()</feature>
       <column>policeReportId</column>
     </featureMapping>
     <featureMapping>
       <feature>docId()</feature>
       <column>policeReportDocId</column>
     </featureMapping>
   </featureMappings>
 </explicitMappingRule>
```

- docUri() は、文書 URI を戻します。

- docTimestamp() は、文書が処理された時刻 (ミリ秒) を戻します。組み込みフィーチャーは、文書のバージョンのトラッキングで便利です。たとえば、使用している文書のバージョンが、クローラーの渡した最新のものかどうかを知りたい場合などに使用できます。

```
<explicitMappingRule applyToSubTypes="false">
   <type>com.ibm.omnifind.types.PoliceReport</type>
   <relation>sample.PoliceReport</relationcolumn>
   </StoreFeature>
   <featureMappings>
     <featureMapping>
       <feature>uniqueId()</feature>
       <column>policeReportId</column>
     </featureMapping>
     <featureMapping>
       <feature>docTimestamp()</feature>
       <column>reportVersion</column>
     </featureMapping>
   </featureMappings>
 </explicitMappingRule>
```

- parentId() は、コンテナー・マッピングを囲むフィーチャー構造の fsId() を戻します。parentId() は、コンテナー・マッピングのコンテキスト内でのみ有効です。
- uniqueParentId() は、コンテナー・マッピングに囲まれた注釈またはフィーチャー構造の uniqueId() を戻します。この有効なフィーチャーも、コンテナー・マッピングのコンテキスト内でのみ有効です。
- [:index] は、現行のコンテナー・エントリー (配列またはリスト) の索引を戻します。

関連タスク

43 ページの『セマンティック検索照会に一致する文書の部分の取得』
関連するフィーチャー構造を索引またはデータベースの両方にマッピングし、セマンティック検索照会でスパンを指定することで、完全に照会に一致する文書の 部分だけを検索できます。

フィルター

フィルターは、索引および JDBC 構成ファイルで、マッピング・ルールを制約するのに使用されます。フィルターが true の場合のみ、分析結果は索引または JDBC 表に追加されます。

<filter> エレメントはオプションで、マッピングを特定の属性値を持つフィーチャーのみに制限する際に使用します。これは、何に対して索引付けを行うか、または何をデータベースに追加するかについてのスイッチとして属性を使用する場合に便利です。例えば、個人 (person) と組織 (organization) が EntityAnnotation タイプの注釈に記載されているとします。その type というフィーチャーは、person または organization のいずれかに設定されます。個人 (person) のみを抽出し、組織 (organization) は抽出しないようにするには、以下のフィルターをマッピング・ルールに追加します。

```
<filter syntax="FeatureValue">type = "person"</filter>
```

いずれのフィルター式も以下の形式になります。

```
<FeaturePath> <Operator> <Literal>
```

ここで、

- FeaturePath は、共通分析構造内のフィーチャー・パスです。
- Operator は =、!=、<、<=、> または >= です。< (< のみ) は、`<` と表記することに注意してください。
- Literal は、整数、浮動小数点数 (指数構文はサポートされていません) または二重引用符で囲まれたストリング・リテラル (ストリング内の引用符および円記号は円記号を付けて拡張します) です。

`<FeaturePath>`、`<Operator>` および `<Literal>` は、ブランク・スペースで区切ってください。

以下は有効なフィルターの例です。

- `<filter syntax="FeatureValue"> foo = "hello world" </filter>`

  フィーチャー foo に、hello world というストリングが含まれています。
- `<filter syntax="FeatureValue"> foo &lt; 42 </filter>`

  フィーチャー foo に、整数値 42 が含まれています。
- `<filter syntax="FeatureValue"> make/company = "Chevrolet" </filter>`

  フィーチャー make に値が Chevrolet のフィーチャー company があるフィーチャー構造が含まれているフィーチャー・パス make/company。
- `<filter syntax="FeatureValue"> bar7 >= 0.5 </filter>`

  フィーチャー bar7 に、浮動小数点値 0.5 が含まれています。

カスタム分析結果の索引マッピング

文書のコレクションにカスタム分析を実行した後、エンタープライズ・サーチの検索エンジンを使用して、カスタム分析アルゴリズムによって生成される共通分析構造に保管された情報から索引を作成することができます。

エンタープライズ・サーチ索引で、分析結果をフィールド、テキストのスパン、および属性にマッピングすることにより、照会でその情報を使用できるようになります。カスタム分析を語とテキストのスパンの両方の索引付け機能を持つエンタープライズ・サーチと結合することにより、セマンティック検索が可能になります。

索引作成構成ファイルを使用して、共通分析構造内のどの分析結果の索引付けを行うかを判別することができます。

さまざまなスタイルを使用して、共通分析構造内のフィーチャー構造をエンタープライズ・サーチ索引にマップできます。

注釈 注釈スタイルを使用して、共通分析構造内のフィーチャー構造の索引付けを行うと、指定したタイプのすべての注釈が、検索可能なスパンとして索引に保管されます。

例えば、テキストの一定範囲に渡るフィーチャー構造が `person` タイプで、注釈スタイルを使用して索引付けを行う場合には、以下の照会が可能です。

表 1. 照会例

必要な情報	可能な照会
少なくとも 1 人の個人 (person) 名が含まれている文書を検索する	<person/>
個人 (person) 注釈に上司 (boss) が含まれている文書を検索する	<person>boss</person>
言葉 (Lang) が競合相手 (competitors) のいずれかと同じセンテンス (sentence) に記載されている文書をすべて検索する	<sentence><person>Lang</person> <competitor/></sentence>

フィーチャー構造の属性も、スパンの一部として索引付けることができます。例えば、車を検出し、`car` 注釈の `make` フィーチャーとして車の製造会社を保管するアノテーターの場合を考えてみます。この場合、「Chevrolet が製造した車が記載されている文書を探す」という照会が可能になります。

フィールド

エンタープライズ・サーチのフィールド検索機能を使用して、検索時にフィーチャー構造のコンテンツをアクセス可能にする場合は、このスタイルを使用します。この方法では、フィーチャー構造のコンテンツを検索結果に表示したり、パラメトリック検索で使用したりすることができます。

例えば、薬の服用量をパラメトリック・フィールドにマップすると、「服用時に 100 ミリグラムを超える薬について記載されている文書をすべて探す」という照会を行うことができます。

ブレーク

特定のフィーチャー構造を明確な区切りとして解釈する (例えば、セクションやパラグラフなど) 場合に、このスタイルを使用します。エンタープライズ・サーチは、デフォルトで、センテンス (文) およびパラグラフ (段落) を検出します。このスタイルは、カスタム分析が、文書内で、個別に解釈したい追加の構造化エレメントを検出する場合にのみ使用します。

分析結果は、単純なキーワード照会に対しても、エンタープライズ・サーチの文書のランキングに影響するように使用できます。これは、以下の 2 つのステップで行います。

1. 「注釈」または「フィールド」マッピング・スタイルを使用して、フィーチャー構造を検索可能なスパンまたはフィールドにマップします。
2. エンタープライズ・サーチ管理コンソールを使用して、ランキング調整クラスを定義し、このランキング調整クラスにスパンまたはフィールド名をマップします。

フィーチャー構造に含まれる検索語を入力した場合、文書のランクは高くなります。例えば、人物と会社名を示すアノテーターの場合を考えてみます。これらのフィーチャー構造をスパン (″person″ および ″company″ など) にマッピングし、これらのスパンをランキング調整クラスにマッピングすることで、″gap″ の検索結果は、単に ″gap″ という語を含む文書よりも、″Gap″ という会社についての文書で高くランクされます。

索引作成構成ファイルの書き込み後に、管理コンソールを使用してこのファイルをエンタープライズ・サーチにアップロードできます。

関連タスク

『索引作成構成ファイルの作成』

索引作成構成ファイルを使用して、検索を使用可能にするために、 共通分析構造内のどの分析結果の索引付けを行うかを判別することができます。

索引作成構成ファイルの作成

索引作成構成ファイルを使用して、検索を使用可能にするために、共通分析構造内のどの分析結果の索引付けを行うかを判別することができます。

このタスクについて

索引作成構成ファイルは、以下の例に示すスキーマに適合する必要があります。サンプル構成ファイルは、警察レポート・シナリオに定義されたタイプ・システムに基づいています。

```
<?xml version="1.0" encoding="UTF-8"?>
<indexBuildSpecification
xmlns="http://www.ibm.com/of/822/consumer/index/xml">
 <skipCondition>
   <type>com.ibm.uima.tt.DocumentAnnotation</type>
   <filter syntax="FeatureValue">toBeprocessed = 0</filter>
 </skipCondition>

 <indexBuildItem>
   <name>com.ibm.omnifind.types.Person</name>
   <indexRule>
     <style name="Annotation">
       <attributeMappings>
         <mapping>
           <feature>role</feature>
           <indexName>role</indexName>
         </mapping>
         <mapping>
           <feature>title</feature>
           <indexName>title</indexName>
         </mapping>
         <mapping>
           <feature>gender</feature>
           <indexName>gender</indexName>
         </mapping>
       </attributeMappings>
     </style>
   </indexRule>
 </indexBuildItem>
 <indexBuildItem>
   <name>com.ibm.omnifind.types.Suspect</name>
   <indexRule>
     <style name="Annotation"/>
     <style name="Field">
       <attribute name="parametric" value="false"/>
       <attribute name="fieldSearchable" value="true"/>
       <attribute name="returnable" value="true"/>
     </style>
   </indexRule>
 </indexBuildItem>
  <indexBuildItem>
   <name>com.ibm.omnifind.types.City</name>
   <indexRule>
     <style name="Annotation">
```

```
      <attributeMappings>
        <mapping>
          <feature>cityDistrict</feature>
          <indexName>district</indexName>
        </mapping>
      </attributeMappings>
    </style>
  </indexRule>
 </indexBuildItem>
 <indexBuildItem>
  <name>com.ibm.omnifind.types.Date</name>
  <indexRule>
    <style name="Field">
      <attribute name="fixedName" value="Date"/>
      <attribute name="fieldSearchable" value="true"/>
      <attribute name="returnable" value="true"/>
    </style>
    <style name="Field">
      <attribute name="fixedName" value="hour"/>
      <attribute name="valueFeature"  value="hour"/>
      <attribute name="parametric" value="true" />
    </style>
  </indexRule>
  <filter syntax="FeatureValue">year="2005"</filter>
 </indexBuildItem>
 <indexBuildItem>
  <name>com.ibm.omnifind.types.PoliceReport</name>
  <indexRule>
    <style name="Annotation">
      <attribute name="fixedName"
         value="PoliceReport"/>
      <attributeMappings>
        <mapping>
          <feature>crimeDescription</feature>
          <indexName>crimeDescription</indexName>
        </mapping>
        <mapping>
          <feature>time/coveredText()</feature>
          <indexName>time</indexName>
        </mapping>
        <mapping>
          <feature>date/englDate</feature>
          <indexName>date</indexName>
        </mapping>
        <mapping>
          <feature>location/coveredText()</feature>
          <indexName>location</indexName>
        </mapping>
        <mapping>
          <feature>knownSuspects[]/com.ibm.omnifind.types.Suspect:surName</feature>
          <indexName>suspectsLastNames</indexName>
        </mapping>
      </attributeMappings>
    </style>
  </indexRule>
 </indexBuildItem>
</indexBuildSpecification>
```

制約事項

索引マッピング構成ファイルには、照会で検索できるようにしたい、すべての分析結果が含まれていなければなりません。

手順

索引マッピング構成ファイルを作成するには、次のようにします。

1. XML ファイルを作成します。 XML 構文エラーを避けるために、選択した XML エディターまたは XML オーサリング・ツールを使用します。構成ファイルの XSD スキーマは、CasToIndexMapping.xsd と呼ばれ、*ES_INSTALL_ROOT*/packages/uima/ にあるエンタープライズ・サーチのインストールに含まれています。
2. マッピングは必ず、<indexBuildSpecification xmlns="http://www.ibm.com/of/822/consumer/index/xml"> エレメントに入れてください。名前空間 (xmlns 属性に指定) は、表示どおりでなければなりません。
3. 一定のフィーチャー値に基づいて、特定の文書については索引付けを行わないようにするために、<skipCondition> エレメントを追加します。このエレメントはオプションです。上記の例では、toBeProcessed というフィーチャーがゼロに設定されている com.ibm.uima.tt.DocumentAnnotation タイプのデータ構造が含まれている文書は、索引付けが行われません。
4. 共通分析構造内のある特定のフィーチャーから、索引内の構造へのマッピングを含む、1 つ以上 <indexBuildItem> エレメントを追加します。
5. XML ファイルを保管して妥当性検査を行います。

<indexBuildItem> エレメント

索引作成仕様構成ファイルには、1 つ以上の <indexBuildItem> エレメントが含まれています。各エレメントは、共通分析構造内のある特定のフィーチャーを索引内の構造 (スパンまたはフィールド) にマッピングするための情報を示します。

<name> エレメントには、フィーチャー構造タイプが含まれています。タイプの指定には 2 つの方法があります。

- 完全なタイプ名。たとえば、com.ibm.omnifind.types.Suspect です。
- ワイルドカード。たとえば、com.ibm.omnifind.types.* です。ワイルドカード文字は、タイプ指定の最後にのみ追加できます。

索引作成項目として、uima.tcas.Annotation のサブタイプのみを使用してください。フィーチャー構造がサブタイプ uima.cas.TOP (かつ、 uima.tcas.Annotation のフィーチャー構造でない場合) は、注釈から開始するフィーチャー・パスを使用してこのフィーチャー構造にアクセスできます。

タイプ A がタイプ B のサブタイプで (サンプルでは、com.ibm.omnifind.types.Suspect は、 com.ibm.omnifind.types.Person のサブタイプ)、両方のタイプに <indexBuildItem> エレメント Ia と Ib が定義されている場合、処理は以下のようになります。

- Ib に定義されているそれぞれの索引規則は、タイプ B のフィーチャー構造およびタイプ A のフィーチャー構造に適用される。
- Ia に定義されているそれぞれの索引規則は、タイプ A のフィーチャー構造にのみ適用される。

例では、com.ibm.omnifind.types.Person 注釈に定義された <indexBuildItem> エレメントは、 com.ibm.omnifind.types.Suspect 注釈にも適用されます。 suspect 注釈には、Person および Suspect の 2 つのスパンが作成されます。

<filter> エレメントはオプションで、 <indexBuildItem> によるマッピングを特定の属性値を持つフィーチャー構造のみに制限する際に使用します。これは、何に対して索引付けを行うかについてのスイッチとして属性を使用する場合に便利です。例えば、個人 (person) と組織 (organization) が EntityAnnotation タイプの注釈に記載されているとします。その type というフィーチャーは、person または organization のいずれかに設定されます。個人 (person) のみを抽出し、組織 (organization) は抽出しないようにするには、以下のフィルターを追加します。

```
<filter syntax="FeatureValue">type = "person"</filter>
```

さらに、個人 (person) と組織 (organization) を異なるスパン名、例えば person と organization で索引付けを行うことができます。このようにするには、EntityAnnotation タイプの 2 つの<indexBuildItem> エレメントを定義し、 type フィーチャーで 2 つのフィルターを使用して、個人 (person) または組織 (organization) のいずれかをトリガーするようにします。

<indexRule> エレメント

各 <indexBuildItem> エレメントには、 1 つの <indexRule> エレメントが含まれています。各 <indexRule> エレメントには、共通分析構造内のフィーチャー構造をフィールド、注釈、ブレーク・スタイルとして索引にマップするために必要な情報がすべて含まれています。注釈スタイルおよびフィールド・スタイルは、多くの属性をサポートします。エンタープライズ・サーチの、 UIMA Software Development Kit でサポートされている条件スタイルは使用できません (条件スタイルはスキップされます)。

注釈スタイルおよびフィールド・スタイルの場合、索引に注釈名またはフィールド名を指定する際に、次のような代替手段があります。

- 各フィーチャー構造が、索引内で同じ名前でアクセスできるようにしたい場合は、fixedName を使用します。以下の例で、Person タイプの各フィーチャー構造は、索引内で「Person」というスパンにマップされます。

  ```
  <indexBuildItem>
     <name>com.ibm.omnifind.types.Person</name>
     <indexRule>
       <style name="Annotation">
         <attribute name="fixedName" value="Person" />
       </style>
     </indexRule>
  <indexBuildItem>
  ```

  これにより、「個人名として Boss が含まれている文書を探す」というような照会が可能になります。照会は、次のように XML フラグメントを使用して表されます。 @xmlf2::'<Person>Boss</Person>'

- 注釈の特定のフィーチャーの値に基づいて異なるスパンを使用してアクセスできるさまざまなエンティティーが注釈にある場合は、nameFeature を使用します。以下の例で、EntityAnnotation は、type というフィーチャーの値に基づいて、person スパンまたは organization スパンとして索引付けが行われます。フィーチャーはフィーチャー・パスであっても構いません。

  ```
  <indexBuildItem>
     <name>com.ibm.tt.EntityAnotation</name>
     <indexRule>
       <style name="Annotation">
  ```

```
        <attribute name="nameFeature" value="type" />
      </style>
    </indexRule>
  <indexBuildItem>
```

これにより、「組織 WHO に関する文書を探す」(英単語 who ではなく) というような照会が可能になります。照会は、限定 XPath 構文では次のように表されます。 `@xmlp::'/organization[ftcontains="WHO"]'`

- 上記の属性がなにも使用されない場合は、<indexBuildItem> エレメント内の注釈タイプのショート・ネームが使用されます。これはデフォルトです。例えば、以下のようになります。

```
<indexBuildItem>
   <name>com.ibm.uima.tutorial.RoomNumber</name>
   <indexRule>
     <style name="Annotation" />
     <style name="Field" />
   </indexRule>
 </indexBuildItem>
```

この <indexBuildItem> エレメントは、 com.ibm.uima.tutorial.RoomNumber によってカバーされるテキストがある RoomNumber という注釈およびフィールドになります。

<style name="Annotation" /> エレメント

<style> エレメントの Annotation は、エンタープライズ・サーチにおけるスパン情報へのアクセス方法を指定します。 fixedName および nameFeature 属性が使用できる以外に、このスタイルは、<attributemappings> エレメントもサポートします。このエレメント内で、フィーチャーの値を索引内の結果スパンの属性にマップすることができます。以降、検索式でそのエレメントを使用することができます。

各マッピングは、それぞれ別個の <mapping> エレメント内で行われます。<feature> エレメントにはフィーチャー・パスが含まれ、<indexName> エレメントには <feature> の値を保管するために索引で使用される属性の名前が含まれます。例えば、以下のようになります。

```
<mapping>
  <feature>make/companyname</feature>
  <indexName>company</indexName>
</mapping>
```

この <mapping> エレメントは、パス make/companyname にあるフィーチャーの値を索引属性 company に直接保管します。

フィーチャー値の索引属性へのマッピングは、ネストされたフィーチャー構造が多く含まれているなど、テキスト分析時に使用されるタイプ・システムが複雑な場合に特に便利です。 <mapping> エレメントを使用して、関係のある属性を明らかにすることにより、オリジナルのタイプ・システム構造の詳細を知らなくても、照会でこれらの属性を使用することができます。

<style name="Field" /> エレメント

<style> エレメントの Field は、エンタープライズ・サーチにおけるフィールド情報へのアクセス方法を指定します。 fixedName および nameFeature 属性以外にも、以下の属性を設定することができます。

parametric
true に設定すると、パラメトリック検索を使用して、フィールド値を検索できます (例えば、#dosage:>100)。

fieldSearchable
true に設定すると、検索でフィールド値を使用できます (例えば、make:Bayer)。

returnable
true に設定すると、フィールドとその値が検索結果に戻されます。

フィールド情報は、常に検索可能なコンテンツです。つまり、フィールド情報は、通常のキーワード検索でアクセス可能です。

オプション属性 valueFeature は、フィールド値として、どのフィーチャー値をとるかを定義します。フィーチャー構造が注釈で、属性が設定されていない場合、注釈のカバー・テキストがフィールド値として使用されます。以下の例では、

```
<indexBuildItem>
  <name>com.ibm.omnifind.types.Date</name>
  <indexRule>
    <style name="Field">
      <attribute name="fixedName" value="date"/>
      <attribute name="fieldSearchable" value="true"/>
      <attribute name="returnable" value="true"/>
    </style>
    <style name="Field">
      <attribute name="fixedName" value="hour"/>
      <attribute name="valueFeature"  value="hour"/>
      <attribute name="parametric" value="true" />
    </style>
  </indexRule>
  <filter syntax="FeatureValue">year="2005"</filter>
 </indexBuildItem>
```

com.ibm.omnifind.types.Date に対して 2 つのフィールドが生成されます。 date というフィールドには、カバー・テキスト (例えば 5:15pm) が含まれます。もう 1 つのフィールドには、属性 hour の値が含まれます。この場合、「hour::<17」を使用して照会することができます。

<style name="Breaking" /> エレメント

<style> エレメントの値 Breaking には、これ以外のエレメントは含まれません。

XML ファイルの作成後に、これをエンタープライズ・サーチにアップロードし、エンタープライズ・XML 管理コンソールを使用して、他のカスタム分析選択のある索引マッピング構成ファイルを選択する必要があります。

関連概念

25 ページの『カスタム分析結果の索引マッピング』
文書のコレクションにカスタム分析を実行した後、 エンタープライズ・サーチの検索エンジンを使用して、カスタム分析アルゴリズムによって生成される 共通分析構造に保管された情報から索引を作成することができます。

20 ページの『フィーチャー・パス』
フィーチャー・パスは、共通分析構造内のフィーチャー値にアクセスする 方法を提供します。これは、XML 文書で XML エレメントにアクセスする際に使用する XPath ステートメントに似ています。

関連資料

24 ページの『フィルター』
フィルターは、索引および JDBC 構成ファイルで、マッピング・ルール を制約するのに使用されます。フィルターが true の場合のみ、分析結果は 索引または JDBC 表に追加されます。

9 ページの『タイプ・システム記述』
タイプ・システム記述は、カスタム分析で使用されるフィーチャー構造 (分析結果 を示す基礎となるデータ構造) を記述します。

選択した分析結果のデータベース・マッピング

エンタープライズ・サーチで文書のコレクションのカスタム分析を実行した後で、選択したテキスト分析結果を JDBC 対応データベースに保管できます。

このバージョンは、DB2® Universal Database、バージョン 8.2.2 (com.ibm.db2.jcc.DB2Driver Version 2.3) および Oracle 10g (oracle.jdbc.driver.OracleDriver Version 1.0) のみをサポートします。

DB2 Universal Database および Oracle では分析結果を直接データベースに挿入するか、または同等のデータベース特定ロード・ファイルおよび、ロード・コマンドを実行する、対応するスクリプトの生成を選択できます。

データベースの表を分析結果をマッピングすると、この情報を以降のビジネス・インテリジェンス処理ステップに使用するか、セマンティック検索照会に一致する文書の関係のある部品に直接アクセスできるようになります。

XML マッピング構成ファイルには、データベース接続構成情報が含まれ、どのカスタム分析結果が、どの表および列に保管されるかを示します。構成ファイル内の表および列名は、データベースに作成された表および列に対応していなければなりません。

構成ファイルの書き込み後に、管理コンソールを使用してこのファイルをエンタープライズ・サーチにアップロードできます。

関連タスク

34 ページの『XML マッピング構成ファイルの作成』
分析結果をデータベースに追加するために、データベース接続構成情報および、どのカスタム・テキスト分析結果が、どの表および列に保管されるかの記述を含む 構成ファイルを作成しなければなりません。

分析結果のデータベースへの保管

選択した分析結果を JDBC 対応データベースに保管するには、エンタープライズ・サーチ用の構成ファイルを書き込まなくてはなりません。また、必要な JDBC ライブラリーが、構成ファイルに定義したパスになければなりません。

分析結果を JDBC 対応データベースに保管するには、以下のようにします。

1. データベースに保管したい分析結果を決定します。対応するデータ・タイプに必要なすべての列をもつ表を含む、データベースを作成します。

重要: 選択した分析結果を保管するための、独自の DB2 データベースを作成します。エンタープライズ・サーチのインストールに含まれている DB2 データベースは使用しないでください。

2. XML エディターで、データベース構成データおよび保管したい分析結果を構成ファイルに書き込みます。構成ファイルに含む分析結果を判別するには、カスタム分析に使用された、基礎となるタイプ・システムを知っていなければなりません。
3. JDBC ドライバー・ライブラリーを、エンタープライズ・サーチ・システムの索引ノードからアクセス可能な JDBC ドライバー・ライブラリーに置きます。
4. エンタープライズ・サーチ管理コンソールを使用して、選択したカスタム・テキスト分析付きの構成ファイルをアップロードします。

XML マッピング構成ファイルの作成

分析結果をデータベースに追加するために、データベース接続構成情報および、どのカスタム・テキスト分析結果が、どの表および列に保管されるかの記述を含む構成ファイルを作成しなければなりません。

このタスクについて

XML マッピング構成ファイルは、以下の例に示すスキーマに適合する必要があります。サンプルは、警察レポート・シナリオに定義されたタイプ・システムに基づいています。

この例では、これらの警察犯罪レポートに表示されている警察レポートおよび都市のみがデータベースに追加されます。例では、組み込みフィーチャーおよび `<constant>` エレメント・マッピングの使用を示しています。

```
<?xml version="1.0" encoding="UTF-8"?>
<cas2JdbcConfiguration xmlns="http://www.ibm.com/uima/consumer/jdbc/100/xml">
   <databaseConnection>
     <connectionUrl>db2://myMachine:myPort/myDatabase</connectionUrl>
     <driver type="jdbc">com.ibm.db2.jcc.DB2Driver</driver>

     <driverLibraries>
       <driverLibrary>C:\db2\db2jcc.jar</driverLibrary>
       <driverLibrary>C:\db2\db2jcc_license_cu.jar</driverLibrary>
       <driverLibrary>C:\db2\db2jcc_license_cisuz.jar</driverLibrary>
     </driverLibraries>

     <authentication>
       <username>myUser</username>
       <password>myPassword</password>
     </authentication>

     <loadFile>
       <loadFileDirectory>/home/cas2jdbc/load/</loadFileDirectory>
       <loadScript>/home/cas2jdbc/load/load.sh</loadScript>
     </loadFile>

   </databaseConnection>

   <jdbcMappingSpec>
     <skipCondition>
       <name>com.ibm.uima.tt.DocumentAnnotation</name>
       <filter syntax="FeatureValue">toBeProcessed=0</filter>
     </skipCondition>
```

```
    <cas2Jdbcmappings>
     <explicitMappings>
      <explicitMappingRule applyToSubtypes="false">
       <type>com.ibm.omnifind.types.PoliceReport</type>
       <table>sample.policeReport</table>
       <featureMappings>
        <featureMapping>
         <feature>uniqueId()</feature>
         <column>policeReportId</column>
        </featureMapping>
        <featureMapping>
         <feature>location/uniqueId()</feature>
         <column>crimeLocationId</column>
        </featureMapping>
       </featureMappings>
       <filter syntax="FeatureValue">location/coveredText()="Los Angeles"</filter>
      </explicitMappingRule>
     </explicitMappings>

     <implicitMappings>
      <implicitMappingRule applyToSubtypes="false">
       <type>com.ibm.omnifind.types.City</type>
        <table>sample.City</table>
       <featureMappings>
        <featureMapping>
         <feature>uniqueId()</feature>
         <column>crimeLocationId</column>
        </featureMapping>
        <featureMapping>
         <feature>coveredText()</feature>
         <column>cityName</column>
         <length>150</length>
        </featureMapping>
        <featureMapping>
         <constant>USA</constant>
         <column>country</column>
        </featureMapping>
       </featureMappings>
      </implicitMappingRule>
     </implicitMappings>
    </cas2JdbcMappings>
   </jdbcMappingSpec>
</cas2JdbcConfiguration>
```

制約事項

選択した分析結果を保管するための、独自の DB2 データベースを作成します。エンタープライズ・サーチのインストールに含まれている DB2 データベースは使用しないでください。

手順

XML データベース構成ファイルを作成するには、次のようにします。

1. XML ファイルを作成します。 XML 構文エラーを避けるために、選択した XML エディターまたは XML オーサリング・ツールを使用します。構成ファイルの XSD スキーマは、CasToJDBCMapping.xsd と呼ばれ、*ES_INSTALL_ROOT*/packages/uima/ にあるエンタープライズ・サーチのインストールに含まれています。

2. マッピングは必ず、<cas2JdbcConfiguration xmlns="http://www.ibm.com/uima/consumer/jdbc/100/xml"> エレメントに入れてください。名前空間 (xmlns 属性に指定) は、表示どおりでなければなりません。
3. すべてのデータベース接続構成情報および、データベースまたはロード・ファイルに保管されている、分析結果のマッピング規則を記述する <jdbcMappingSpec> エレメントを含む <databaseConnection> エレメントを追加します。
4. 次のコンポーネント・エレメントを、<databaseConnection> エレメントに追加します。
   - 必須: <connectionUrl> エレメント。このエレメントには、データベース接続 URL が含まれます。JDBC ドライバーのインプリメンテーションによって、データベースのローカルまたはリモート・アクセスを使用できます。
   - 必須: <driver> エレメント。このエレメントには、 JDBC ドライバー・クラスの名前 (たとえば、DB2 では com.ibm.db2.jcc.DB2Driver、または Oracle では oracle.jdbc.driver.OracleDriver)。
   - 必須: <driverLibraries> エレメント。このエレメントは、ドライバー・ライブラリーをリストします。各ライブラリーは、<driverLibrary> エレメントにリストされています。これらのライブラリーは、DB2 または Oracle のインストール・ディレクトリーにあります。DB2 の場合、ライブラリーは c:\your_db2_dir\db2jcc.jar、c:\your_db2_dir\db2jcc_license_cu.jar および c:\your_db2_dir\db2jcc_license_cisuz.jar にあります。Oracle の場合、組み込むライブラリーは c:\your_oracle_dir\classes12.zip です。
   - 必須: <authentication> エレメント。このエレメントには、データベースのユーザー名およびパスワードが含まれます。
   - オプション: <loadFile> エレメント。このエレメントには、<loadFileDirectory> エレメントのロード・ファイル・ディレクトリーおよび、 <loadScript> エレメントのロード・スクリプト名が含まれます。<loadFile> エレメントを指定しない場合、すべてのデータは、 JDBC を使用してデータベースに直接保管されます。

     データベース特定のロード・ファイルおよびスクリプトを使用する場合は、すべてのデータベース構成パラメーターも追加する必要があります。
5. 次のコンポーネント・エレメントを、<jdbcMappingSpec> エレメントに追加します。
   - オプション: <skipCondition> エレメント。スキップ条件が定義されていない場合は、すべての文書が処理されます。

     ```
     <skipCondition>
       <name>com.ibm.uima.tt.DocumentAnnotation</name>
       <filter syntax="FeatureValue">toBeProcessed=0</filter>
     </skipCondition>
     ```

     上記の例では、toBeProcessed というフィーチャーがゼロに設定されている com.ibm.uima.tt.DocumentAnnotation タイプの注釈を含む文書は、考慮されません。

- どのタイプおよびフィーチャーが、どのデータベース表および列にマップされるかを示す <cas2JdbcMappings> エレメント。このエレメントには、明示的および暗黙的なマッピング・セクションが含まれています。

6. <explicitMappings> エレメントを追加します。このエレメントは必須です。これには、明示的なマッピングを定義する、1 つ以上の <explicitMappingRule> エレメントが必要であり、注釈タイプおよびそのサブタイプに対してのみ定義可能です。 マッピングが明示的なマッピング・セクションに定義されている場合、マッピング定義に一致するすべての注釈が、データベースに保管されます。
7. オプション: <implicitMappings> エレメントを追加します。このエレメントは、すべてのフィーチャー構造タイプをサポートします。このエレメントがある場合、最低 1 つの <implicitMappingRule> エレメントを含む必要があります。 暗黙的なマッピング・セクションに定義されているマッピングは、一致する注釈タイプが明示的または暗黙的なマッピング規則のいずれかに一致する、他の注釈によって参照されている場合にのみ、データベースに追加されます。

   暗黙的なマッピングの目的は、 特定のコンテキストで表示される分析結果のみを保管できるようにすることです。たとえば、com.ibm.omnifind.types.City タイプの注釈が暗黙的な場合、明示的なマッピング・セクションで com.ibm.omnifind.types.PoliceReport マッピング定義によって参照されている都市のみ、データベースに保管されます。これは、警察レポートで記述されている都市のみが、データベースに追加されることを意味します。

   City 注釈に、明示的なマッピング規則がある場合は、すべての都市がデータベースに追加されます。いずれの場合でも、複数の警察レポートで参照された都市は、データベースに 1 度だけ追加されます。
8. <explicitMappingRule> および <implicitMappingRule> エレメントは、applyToSubtypes 属性を含む必要があります。これが true に設定されている場合、<type> エレメントにリストされているフィーチャー構造だけでなく、そこから派生したすべてのフィーチャー構造も保管します。次のコンポーネント・エレメントを、 <explicitMappingRule> and <implicitMappingRule> に追加します。
   - フィーチャー構造タイプを含む <type> エレメント。
   - データベース・スキーマおよび表名を含む <table> エレメント。スキーマが定義されていない場合、構文は schema.table_name 規則に、または table_name の規則にのみ従います。
   - 1 つ以上の <featureMapping> エレメントまたは、1 つ以上の <contanerMapping> エレメントのある <featureMappings> エレメント。
   - オプション: マッピング規則が一致するたびに評価される条件を含む <filter> エレメント。条件が true を評価する場合、注釈またはフィーチャー構造はデータベースに保管されます。この例では、ロサンゼルスで発生した犯罪を扱う警察レポートのみ、データベースに保管されます。
9. <featureMapping> エレメント・コンポーネント構造は、フィーチャーまたは定数にマッピングしているかにしたがって、変化します。

   フィーチャーまたはフィーチャー・パスをマッピングする場合、以下がコンポーネント・エレメントに含まれます。

- フィーチャーの名前のある <feature> エレメント。フィーチャーは、タイプ・エレメントのフィーチャー構造に対して定義される必要があります。フィーチャー・パス構成またはシステム定義の組み込みフィーチャーのいずれかも、使用できます。
- オプション: 指定したデータベース列に持てるストリングの長さのある <length> エレメント。長いストリングは切り捨てられます。
- フィーチャー値が保管される列の名前のある <column> エレメント。どのフィーチャー・マッピングでも使用されないデータベース列は、データベースに構成されているデフォルト値 (通常はヌル) を使用します。

  フィーチャー・エレメントの値が、適切なタイプの列に保管されることを確認してください。以下の表は、どの UIMA タイプが、どのデータベース・タイプとマッチングするするかを示しています。

表 2. UIMA タイプおよび対応するデータベース・タイプ間のマッピング

UIMA タイプまたは組み込みフィーチャー	推奨される DB2 データ・タイプ	推奨される Oracle データ・タイプ
Float	REAL	FLOAT
String	VARCHAR	VARCHAR2
Integer	INTEGER	INTEGER
uniqueId()、uniqueParentId()	CHAR(27)	CHAR(27)
objectId()、parentId()	CHAR(16)	CHAR(16)
docTimestamp()	BIGINT	LONG

定数の場合、コンポーネント・フィーチャー・マッピング・エレメントは以下のようになります。

- 定数の値を含む <constant> エレメント。
- 定数値が追加される列の名前のある <column> エレメント。

10. コンテナー・タイプ・フィーチャー (配列またはリスト) のマッピングを含む <containerMapping> エレメント。このエレメントは、コンテナー・タイプにのみ使用されなければなりません。これには、以下のコンポーネント・エレメントがあります。
    - フィーチャーの名前のある <feature> エレメント。フィーチャー・パス構成またはシステム定義の組み込みフィーチャーのいずれかも、使用できます。
    - データベース・スキーマおよび表名を含む <table> エレメント。スキーマが定義されていない場合、構文は schema.table_name 規則に、または table_name の規則にのみ従います。
    - フィーチャーが追加されるフィーチャー構造または列名の名前を含む、 1 つ以上の <featureMapping> エレメント。
11. 提供されたスキーマを使用して、XML ファイルを保管し、妥当性検査します。

XML ファイルの作成後に、これをエンタープライズ・サーチにアップロードし、エンタープライズ・XML 管理コンソールを使用して、他のカスタム分析選択のあるデータベース・マッピング構成ファイルを選択する必要があります。

関連概念

33 ページの『選択した分析結果のデータベース・マッピング』
エンタープライズ・サーチで文書のコレクションのカスタム分析を実行した後で、 選択したテキスト分析結果を JDBC 対応データベースに保管できます。

20 ページの『フィーチャー・パス』
フィーチャー・パスは、共通分析構造内のフィーチャー値にアクセスする 方法を提供します。これは、XML 文書で XML エレメントにアクセスする際に使用する XPath ステートメントに似ています。

関連資料

24 ページの『フィルター』
フィルターは、索引および JDBC 構成ファイルで、マッピング・ルール を制約するのに使用されます。フィルターが true の場合のみ、分析結果は 索引または JDBC 表に追加されます。

21 ページの『組み込みフィーチャー』
組み込みフィーチャーは、特殊なセマンティクスを持つ、事前定義されたフィーチャー名です。 これは、フィーチャー構造自体、たとえばフィーチャー構造のタイプまたは 注釈のカバー・テキストなどに含まれない情報へのアクセスに使用できます。 これらは、最後または単一エレメントとして、フィーチャー・パスで 使用できます。

9 ページの『タイプ・システム記述』
タイプ・システム記述は、カスタム分析で使用されるフィーチャー構造 (分析結果 を示す基礎となるデータ構造) を記述します。

「コンテナー・タイプ」のマッピング

コンテナー・タイプとは、共通分析構造の組み込み配列または、リスト・タイプの 1 つです。コンテナー・タイプ・マッピングとは、リレーショナル・データベースに対する配列のマッピングまたは値のリストの方法です。

構成ファイルでコンテナー・タイプを処理する方法は、2 つあります。 1 つ目の方法は、定義済みの組み込みフィーチャー構造および、配列またはフィーチャー・マッピングの値のリストを含む汎用リンク・テーブルを使用するものです。異なる配列またはリストが同じリンク・テーブルに保管されている場合、表は保管されている情報の関係については、何も示しません。

2 つ目の方法は、`<containerMapping>` エレメントを使用して定義されたリンク・テーブル定義が、使用したい指定した情報間の関係を、明示的に表示することです。

汎用リンク・テーブル・マッピングがどのように見えるかの例は、次のとおりです。警察レポートと容疑者の間には、n:m の関係があります。これは、一人の容疑者が複数の警察レポートで記述される場合があり、1 つの警察レポートが、複数の容疑者について記述している場合があるということを意味します。

例の中の汎用の `sample.fsarray` 表は、警察レポートと容疑者の間の、リンク・テーブルです。タイプ `com.ibm.omnifind.types.FSArray` のフィーチャーを持つ `com.ibm.omnifind.types.PoliceReport` 以外の、別のマッピング・タイプがある場合は、それもまたこの表にマップされます。警察レポートと容疑者の間の関係の表は、これで正しく照会することができます。しかしながら、警察レポートと考えられる容疑者の間の関係またはリンクを含む表を単純に見ただけで、断定することはできません。

```
<cas2Jdbcmappings>
  <explicitMappings>
    <explicitMappingRule applyToSubtypes="false">
      <type>com.ibm.omnifind.types.PoliceReport</type>
      <table>sample.policeReport</table>
      <featureMappings>
        <featureMapping>
          <feature>uniqueId()</feature>
          <column>policeReportId</column>
        </featureMapping>
        <featureMapping>
          <feature>knownSuspects/uniqueId()</feature>
          <column>suspectArrayId</column>
        </featureMapping>
        <featureMapping>
          <feature>location/cityName</feature>
          <column>city</column>
        </featureMapping>
      </featureMappings>
    </explicitMappingRule>
  </explicitMappings>

  <implicitMappings>
    <implicitMappingRule applyToSubtypes="false">
      <type>com.ibm.omnifind.types.Suspect</type>
      <table>sample.suspect</table>
      <featureMappings>
        <featureMapping>
          <feature>uniqueId()</feature>
          <column>suspectID</column>
        </featureMapping>
        <featureMapping>
          <feature>surName</feature>
          <column>lastName</column>
        </featureMapping>
        <featureMapping>
          <feature>description</feature>
          <column>description</column>
        </featureMapping>
    </implicitMappingRule>
    <implicitMappingRule applyToSubtypes="false">
      <type>uima.cas.FSArray</type>
      <table>sample.fsarray</table>
      <featureMappings>
        <featureMapping>
          <feature>uniqueId()</feature>
          <column>arrayId</column>
        </featureMapping>
        <featureMapping>
          <feature>[:index]</feature>
          <column>arrayIndex</column>
        </featureMapping>
        <featureMapping>
          <feature>[]/uniqueId()</feature>
          <column>suspectId</column>
        </featureMapping>
      </featureMappings>
    </implicitMappingRule>
  </implicitMappings>

</cas2JdbcMappings>
```

以下は、上の一般マッピング規則に基づくデータベース表を示します。

表 3. sample.policeReport 表

policeReportId	**suspectArrayId**	**city**
aaa...1	**bbb...1**	Springfield
aaa...2	**bbb...2**	Ladysmith

表 4. sample.fsarray 表

arrayId	**arrayIndex**	**suspectId**
bbb...1	1	*ccc...1*
bbb...1	2	*ccc...2*
bbb...2	1	*ccc...3*

表 5. sample.suspect 表

suspectID	**lastname**	**説明**
ccc...1	Brown	Dark complexion
ccc...2	Smith	Wears glasses
...	...	...

この例は、フィーチャー構造配列のマッピングを示しています。このタイプのマッピングを StringArray、IntegerArray、および FloatArray にも適用できます。これらの単純な値配列に対するマッピング規則を含む場合は、`[]/uniqueId()` を `[]` で置き換えます。

同じ汎用表の方法が、フィーチャー構造リストおよび単純タイプのリスト (StringList、IntegerList および FloatList) でも使用できます。

関係を処理するより簡単な方法は、配列またはリストに含まれるエレメント上の反復を定義する、明示的なコンテナー・マッピング・エレメントを使用することです。

以下は、明示的なリンク・テーブルを表示するマッピングの例です。再度、警察レポートと容疑者の間には、 n:m 関係があります。しかし今回は、`sample.reports_suspects` 表は警察レポートと容疑者の間のリンク・テーブルです。

この方法では、配列 ID、またはリスト・タイプのヘッドおよびテール・エントリー・マッピングの処理を考慮する必要はありません。リンク・テーブルには、明示的な関係が含まれています。

```
<cas2Jdbcmappings>
  <explicitMappings>
    <explicitMappingRule applyToSubtypes="false">
      <type>com.ibm.omnifind.types.PoliceReport</type>
      <table>sample.policeReport</table>
      <featureMappings>
        <featureMapping>
          <feature>uniqueId()</feature>
          <column>policeReportID</column>
        </featureMapping>
        <featureMapping>
```

```
        <feature>location/cityName</feature>
        <column>city</column>
      </featureMapping>
      <featureMapping>
        <feature>knownSuspects</feature>
        <containerMapping>
          <table>sample.reports_suspects</table>
          <featureMapping>
            <feature>com.ibm.omnifind.types.PoliceReport
              /objectId()</feature>
            <column>policeReportId</column>
          </featureMapping>
          <featuremapping>
            <feature>knownSuspects/[]/objectId()</feature>
            <column>suspectId</column>
          </featureMapping>
        </containerMapping>
      </featureMapping>
    </featureMappings>
  </explicitMappingRule>
</explicitMappings>

<implicitMappings>
  <implicitMappingRule applyToSubtypes="false">
    <type>com.ibm.omnifind.types.Suspect</type>
    <table>sample.suspect</table>
    <featureMappings>
      <featureMapping>
        <feature>objectId()</feature>
        <column>suspectID</column>
      </featureMapping>
      <featureMapping>
        <feature>surName</feature>
        <column>lastName</column>
      </featureMapping>
      <featureMapping>
        <feature>description</feature>
        <column>description</column>
      </featureMapping>
    </featureMappings>
  </implicitMappingRule>
</implicitMappings>

</cas2JdbcMappings>
```

<containerMapping> エレメントは、配列に含まれるエレメント上の反復を定義します。この例では、sample.reports_suspects リンク・テーブルには、policeReportId および suspectId 列へのリンクが含まれています。<containerMapping> エレメントをネストしないでください。

以下は、明示的なリンク・テーブル・マッピング規則に基づくデータベース表を示します。

表 6. sample.policeReport 表

policeReportId	city
aaa...1	Springfield
aaa...2	Ladysmith

表 7. sample.reports_suspect 表

policeReportId	suspectId
bbb...1	*ccc...1*
bbb...2	*ccc...2*
...	...

表 8. sample.suspect 表

suspectID	lastname	description
ccc...1	Brown	Dark complexion
ccc...2	Smith	Wears glasses
...	...	...

関連資料

21 ページの『組み込みフィーチャー』
組み込みフィーチャーは、特殊なセマンティクスを持つ、事前定義されたフィーチャー名です。 これは、フィーチャー構造自体、たとえばフィーチャー構造のタイプまたは 注釈のカバー・テキストなどに含まれない情報へのアクセスに使用できます。 これらは、最後または単一エレメントとして、フィーチャー・パスで 使用できます。

セマンティック検索照会に一致する文書の部分の取得

関連するフィーチャー構造を索引またはデータベースの両方にマッピングし、セマンティック検索照会でスパンを指定することで、完全に照会に一致する文書の部分だけを検索できます。

検索結果の特定の注釈タイプの、すべてのインスタンスにアクセスする、たとえばすべての人物を取得するには、注釈タイプのフィールド・スタイル・マッピングを含み、それを索引構成ファイルで戻せるようにマークします。例えば、以下のようになります。

```
<indexBuildItem>
 <name>com.ibm.omnifind.types.Person</name>
 <indexRule>
  <style name="Annotation"/>
  <style name="Field">
   <attribute name="returnable" value="true"/>
  </style>
 </indexRule>
</indexBuildItem>
```

この例では、com.ibm.omnifind.types.Person タイプの注釈は、エンタープライズ・サーチ索引内の Person という名前のスパンにマップされ、セマンティック検索でアクセス可能です。さらに、完全な人名などの、注釈のカバー・テキストは、戻すことが可能なフィールドとして保管されます。これらの注釈値を検索するには、検索照会 (キーワードまたはセマンティック) から戻される、それぞれの結果オブジェクトで getFields(″Person″) を呼び出します。このメソッドは、String 配列および注釈値 (この場合は人物名) を戻します。

しかしこの方法では、指定された注釈タイプのすべてのインスタンスを戻し、照会に完全に一致する文書のみの結果処理に制限したい場合には、不適切です。たとえば、文書に 5 人の人物が記載されているとします。しかし、セマンティック検索照会 '<sentence><person/>IBM</sentence> で、「IBM」という用語と同じ文にある人物にのみ興味があるものとします。他の人物には興味はありません。

照会に完全に一致するフィーチャー構造にアクセスし、処理するには、次のようにします。

1. 注釈マッピング・スタイルを使用して、関連するフィーチャー構造タイプをエンタープライズ・サーチ索引にマップします。 例えば、以下のようになります。

```
<indexBuildItem>
 <name>com.ibm.omnifind.types.Person</name>
 <indexRule>
  <style name="Annotation"/>
 </indexRule>
</indexBuildItem>
```

2. 関連するフィーチャー構造タイプを JDBC 表にマップします。マッピングの一部として、文書 URI およびフィーチャー構造 ID に対して 2 つの列を組み込む必要があります。すべてのフィーチャー構造タイプを同じデータベース表にマップできますが、各タイプを異なる表にマップする必要があります。 例えば、以下のようになります。

```
<explicitMappingRule applyToSubtypes="false">
 <type>com.ibm.omnifind.types.Person</type>
 <table>sample.person</table>
 <featureMappings>
  <featureMapping>
   <feature>objectId()</feature>
   <column>primaryId</column>
  </featureMapping>
  <!-- Contains the covered text of the annotation-->
  <featureMapping>
   <feature>coveredText()</feature>
   <column>personName</column>
  </featureMapping>
  <!-- Other mapping go in here-->
  <!-- To access the relevant person annotations in the query result-->
  <featureMapping>
   <feature>docUri()</feature>
   <column>docUri</column>
  </featureMapping>
  <featureMapping>
   <feature>fsId()</feature>
   <column>annotationId</column>
  </featureMapping>
 </featureMappings>
</explicitMappingRule>
```

3. 文書をクロール、構文解析および索引付けします。
4. 照会に一致するインスタンスの ID を検索します。検索および索引 API (SIAPI) では、これらのインスタンスはターゲット・エレメントとして参照されます。ターゲット・エレメントは、戻される入力スパンを指定します。これは、次のように定義されます。
   - XML フラグメントでは、ターゲット・エレメントは前に付けられた番号記号 (#) によって識別されます。番号記号は 1 度だけ許可され、XML フラグメント照会のどこにでも入れることができます。例:
     `$xmlf2::'<sentence><#person/>IBM</sentence>'`

- デフォルトの XPath では、ターゲット・エレメントは XPath 式の最後のフィールドです。
- Result.getProperty(″TargetElement″) メソッドを使用して、これらのインスタンスにアクセスします。戻されたプロパティーは、スペースで分離されたすべてのオカレンス ID の連結ストリングです。プロパティー内の各オカレンスは、整数値に変換できます。

5. SIAPI は、フィーチャー構造自体は戻さず、オカレンス ID のみを戻します。これらの ID は、データベース表に保管されている fsId() 値に対応します。これらのインスタンスおよび関連する情報を検索するには、アプリケーションが次を行う必要があります。
   a. ターゲット・エレメントのスパン名に基づいて、右のデータベース表を選択します。例では、アプリケーションに person から sample.Person 表へのマッピングが含まれています。この情報は、スパン名を生み出す索引マッピングと、表名を生み出す JDBC マッピングの両方から、構成ファイルを推測します。
   b. 検索結果の各結果オブジェクトごとに、以下のようになります。
      1) オカレンス ID で検索するために、Result.getProperty (″TargetElement″) で戻されるストリングを構文解析します。
      2) 結果 URI (Result.getDocumentId() を使用してアクセス可能) を docUri 列の値として、オカレンス ID を annotationId 列の値として使用し、表に対して SELECT ステートメントを実行します。列名は、マッピング・ファイルによって異なります。列名は、前の例から取られます。

      戻される行には、フィーチャー構造のために保管された情報が含まれます。たとえば、″last name″ または ″city of birth″ などの、カバー・テキストまたはフィーチャー構造の特定の属性などです。

データベースへの更新がエンタープライズ・サーチの索引更新と同期していることを確認してください。データベースに古い情報が含まれている場合 (たとえば、データベース・ロード・ファイルを使用し、データベースを更新せず、しかし索引を更新または再編成した場合)、データベース内で検出されないオカレンス ID がある場合があります。エンタープライズ・サーチは、索引内に最新の文書のバージョンのレコードのみを保持します。そのため、オカレンス ID は、最新の文書でのみ有効です。

同じ文書の複数のバージョンを、同じデータベース表に保管する場合、文書の異なるそれぞれのバージョンに対応する、同じオカレンス ID に一致する複数の行が存在する場合があります。このケースでは、文書バージョンの列を定義し、アプリケーション・ロジックまたは docTimestamp() などの組み込みフィーチャーを使用してデータを追加する必要があります。この方法で、最新の文書のバージョンのみを取得するために、結果にフィルターに掛けることができます。

関連概念

53 ページの『セマンティック検索照会用語』

セマンティック検索照会条件は、不透明条件と呼ばれます。

関連タスク

27 ページの『索引作成構成ファイルの作成』
索引作成構成ファイルを使用して、検索を使用可能にするために、 共通分析構造内のどの分析結果の索引付けを行うかを判別することができます。

34 ページの『XML マッピング構成ファイルの作成』
分析結果をデータベースに追加するために、データベース接続構成情報および、どのカスタム・テキスト分析結果が、どの表および列に保管されるかの記述を含む 構成ファイルを作成しなければなりません。

エンタープライズ・サーチに定義されているタイプおよびフィーチャー

エンタープライズ・サーチに定義されているタイプ・システムは、文書メタデータ処理および基本的な言語分析をカバーします。

文書の言語認識およびセグメンテーションを行う基本的な言語分析は、カスタム分析が選択されているかどうかにかかわらず、文書の索引付けを行う際に常に実行されます。基本的な文書の分析時に、カスタム分析で使用できる共通分析構造に以下の情報が追加されます。

- `com.ibm.es.tt.DocumentMetaData` タイプの文書メタデータ
- タイプ `uima.tt.TokenAnnotation`、`uima.tt.SentenceAnnotation`、および `uima.tt.ParagraphAnnotation` のトークン、センテンス、およびパラグラフ注釈。トークン注釈には、フィーチャー `lemma` が含まれています。

エンタープライズ・サーチに定義されているタイプ・システムには、テキスト分析に固有の複雑なタイプやフィーチャーは含まれていません。これらは、UIMA 環境でカスタム分析タイプおよびフィーチャーを定義する際に使用および拡張できる UIMA タイプ・システムに含まれます。おそらく、エンタープライズ・サーチ・システムを拡張する必要はありません。

エンタープライズ・サーチ・タイプ・システムは、 UIMA Software Development Kit (SDK) には定義されていません。UIMA でアノテーターを作成する際にエンタープライズ・サーチ・タイプ・システムのいずれかのタイプを使用する場合 (文書のセキュリティー情報にアクセスする場合や、クローラー・タイプや文書タイプにアクセスする場合など) には、分析エンジンのシステム記述にこれらのタイプを再度定義する必要があります。

エンタープライズ・サーチには、以下のタイプおよびフィーチャーが定義されています。

uima.tcas.Annotation
注釈は以下のタイプからなります。

uima.tcas.DocumentAnnotation
文書注釈には、以下のフィーチャーがあります。

esDocumentMetaData
`com.ibm.es.tt.DocumentMetaData` タイプの文書メタデータが含まれています。

com.ibm.es.tt.ContentField
コンテンツ・フィールド注釈には、以下のフィーチャーがあります。

parameters
com.ibm.es.tt.CommonFieldParameters タイプのコンテンツ・フィールド・パラメーター。

com.ibm.es.tt.Anchor
HTML 文書のアンカー・テキスト用のアンカー注釈。以下のフィーチャーがあります。

uri アンカー・テキストのターゲット URI。フィーチャー値は、uima.cas.String タイプです。

com.ibm.es.tt.MarkupTag
マークアップ情報注釈。例えば、XML タグの注釈など。マークアップ情報は、以下のフィーチャーに保管されています。

name マークアップ・タグの名前。フィーチャー値は、uima.cas.String タイプです。

depth ネストの深さ。フィーチャー値は、uima.cas.Integer タイプです。

attributeName
フィーチャー属性の名前。フィーチャー値は、uima.cas.StringArray タイプです。

attributeValues
属性の値のストリング。フィーチャー値は、uima.cas.StringArray タイプです。

uima.CAS.TOP
タイプ・システムのルート。以下のタイプがあります。

com.ibm.es.tt.DocumentMetaData
文書メタデータには、以下のフィーチャーがあります。フィーチャーは、文書注釈フィーチャー esDocumentMetaData に結び付けられます。

crawlerId
クローラー名。フィーチャー値は、uima.cas.String タイプです。

dataSource
以下のいずれかのデータ・ソース・タイプ。

- Web (Web クローラーによる文書)
- NNTP (ニュース・グループ・クローラーによる文書)
- DB2 (DB2 クローラーによる文書)
- Notes® (Notes クローラーによる文書)
- CM (コンテンツ・マネージメント・クローラーによる文書)
- FS (UNIX® ファイル・システム・クローラーによる文書)
- WinFS (Windows® ファイル・システム・クローラーによる文書)
- Exchange (Exchange クローラーによる文書)

- VBR (VeniceBridge クローラーによる文書)

フィーチャー値は、uima.cas.String タイプです。

dataSourceName
クローラー (データ・ソース) の名前。フィーチャー値は、uima.cas.String タイプです。

docType
以下のいずれかの文書タイプ。

- text/html
- application/postscript
- application/pdf
- application/x-mspowerpoint
- application/msword
- application/x-msexcel
- application/rtf
- application/vnd.lotus-wordpro
- application/x-lotus-123
- application/vnd.lotus-freelance
- text/xml
- text/plain
- application/x-js-taro (一太郎)

フィーチャー値は、uima.cas.String タイプです。

securityTokens
文書のセキュリティー・トークン。フィーチャー値は、uima.cas.StringArray タイプです。

date 文書の日付。フィーチャー値は、uima.cas.String タイプです。

baseUri
ページの基本 URI。フィーチャー値は、uima.cas.String タイプです。

metaDataFields
フィーチャー値は、uima.cas.FSArray タイプです。この配列の各エレメントは、com.ibm.es.tt.MetaDataField タイプです。

redirectUrl
リダイレクトされた URL。フィーチャー値は、uima.cas.String タイプです。

mimeType
MIME タイプ、または文書タイプ。例えば、XML など。フィーチャー値は、uima.cas.String タイプです。

url 文書の URL。フィーチャー値は、uima.cas.String タイプです。

com.ibm.es.tt.CommonFieldParameters
共通フィールド・パラメーターには、以下が含まれています。

searchable
フィールドが検索可能かどうかを示すフラグ。

fieldSearchable
フィールドがフィールドとして検索可能かどうかを示すフラグ。

parametric
パラメトリック検索を示すフラグ。

showInSearchResult
検索結果詳細に注釈付きのデータが含まれているかどうかを示すフラグ。

resolveConflict
`MetadataPreferred`、`ContentPreferred`、および `Coexist` 間のメタデータの競合を解決するフラグ。フィーチャー値は、`uima.cas.String` タイプです。

name フィールドの名前。フィールド名を使用して、このフィールドを検索することができます。フィーチャー値は、`uima.cas.String` タイプです。

com.ibm.es.tt.MetaDataField
メタデータ・フィールド・データは、文書コンテンツの一部ではありませんが、「text」フィーチャーに保管されます。

parameters
タイプ `com.ibm.es.tt.CommonFieldParameters` のメタデータ・フィールド・パラメーター。

text メタデータ・テキストは、タイプ `uima.cas.String` のこのフィーチャーに保管されます。

関連資料

『UIMA に定義されているタイプおよびフィーチャー』
UIMA Software Development Kit は、テキスト分析時に文書内で検出される可能性がある基本的な言語タイプおよびフィーチャーを定義します。

UIMA に定義されているタイプおよびフィーチャー

UIMA Software Development Kit は、テキスト分析時に文書内で検出される可能性がある基本的な言語タイプおよびフィーチャーを定義します。

各分析エンジンには、分析エンジン内のアノテーターの入力要件および出力タイプを示す、固有のタイプ・システム記述があります。タイプ・システム記述は、ドメインおよびアプリケーションに固有のものです。

UIMA タイプ・システムにユーザー独自のタイプおよびフィーチャーが含まれるように拡張することができます。UIMA 環境には、アノテーターのタイプ・システム

記述子の編集を支援する Eclipse プラグインがあります。Component Descriptor Editor プラグインのインストールおよび使用についての詳細は、 UIMA の資料を参照してください。

UIMA 環境で、分析エンジンの開発およびテストを完了したら、分析エンジン・ファイルが含まれるユーザー作成のアーカイブ・ファイル (.pear ファイル) にもタイプ・システム記述を組み込みます。

UIMA には、以下のタイプおよびフィーチャーが定義されています。

uima.tcas.Annotation
注釈は以下のタイプからなります。

uima.tcas.DocumentAnnotation

uima.tt.TTAnnotation

uima.tcas.DocumentAnnotation
文書注釈には、以下のフィーチャーがあります。

categories
文書のカテゴリー名またはラベルのリスト。フィーチャー値は、`uima.cas.FSList` タイプです。

languageCandidates
文書の言語参照のリスト。フィーチャー値は、 `uima.cas.FSList` タイプです。

id 文書識別形式。例えば URL など。フィーチャー値は、`uima.cas.String` タイプです。

uima.tt.TTAnnotation
TT 注釈には、以下のタイプがあります。

uima.tt.DocStructureAnnotation
文書に関する構造的な情報。文書構造注釈には、以下のタイプがあります。

uima.tt.SentenceAnnotation
開始句読点および終了句読点を含むセンテンス。以下のフィーチャーが含まれます。

sentenceNumber
パラグラフ内のセンテンスのシーケンス番号。各パラグラフの先頭で 1 にリセットします。フィーチャー値は、`uima.cas.Integer` タイプです。

uima.tt.ParagraphAnnotation
パラグラフ。これには、以下のフィーチャーがあります。

paragraphNumber
パラグラフのシーケンス番号。フィーチャー値は、`uima.cas.Integer` タイプです。

uima.tt.LexicalAnnotation
文書に関するコンテンツ情報。字句の注釈は、以下のタイプからなります。

uima.tt.CompPartAnnotation
複合語の一部。多くのゲルマン言語の複合語は、ブランクで区切らずに一緒に書かれています。例えば、ドイツ語の ″Abteilungsleiter″ (部門管理者) という語は、 ″Abteilung″ (部門) と ″Leiter″ (管理者) から構成されています。

uima.tt.TokenAnnotation
空白で囲まれていないトークン。これには、以下のフィーチャーがあります。

lemmaEntries
与えられたトークンに対して考えられるすべての見出し語のリスト。各項目は、トークンに有効な辞書項目です。

lemma `lemmaEntries` 内のトークンに対して考えられるすべての見出し語のリストからの 1 つの見出し語。この見出し語は検索で使用されます。

tokenNumber
センテンス内のトークンのシーケンス番号。各センテンスの先頭で 1 にリセットします。フィーチャー値は、`uima.cas.Integer` タイプです。

tokenProperties
トークンのプロパティー。例えばトークンが uppercase または numeric であるかなど。フィーチャー値は、`uima.cas.Integer` タイプです。

stopwordToken
ストップワードとしてマークされているトークン。フィーチャー値は、`uima.cas.Integer` タイプです。

synonymEntries
`uima.tt.Synonym` タイプの項目に対する参照リスト。各項目は、トークンに有効な同義語項目です。

normalizedCoveredText
注釈によってカバーされるテキストの正規化表現。フィーチャー値は、 `uima.cas.String` タイプです。

uima.CAS.TOP
タイプ・システムのルート。以下のタイプがあります。

uima.tt.KeyStringEntry
以下のフィーチャーがあるストリング。

key ストリング。

uima.tt.Lemma
以下の形態素情報を持つ辞書項目。

partOfSpeech
見出し語の品詞の整数エンコード。

morphID
形態素情報の整数エンコード。

uima.tt.Synonym
指定された uima.tt.keyStringEntry タイプの語に対する同義語項目。

uima.tt.LanguageConfidencePair
文書の言語選択を示す以下のフィーチャーがあるタイプ。

uima.tt.LanguageConfidencePair

languageConfidence
選択された言語が文書の言語に実際にどの程度適合するかを示す標識 (0 から 1 の間の浮動小数点)。

language
文書の言語 (ISO 値)。値は、uima.cas.String タイプです。

languageID
言語 ID。値は、uima.cas.Integer タイプです。

uima.tt.CategoryConfidencePair
文書のカテゴリー選択を示す以下のフィーチャーがあるタイプ。

uima.tt.CategoryConfidencePair
カテゴリーには、以下のフィーチャーがあります。

categoryString
カテゴリーの名前。値は、uima.cas.String タイプです。

categoryConfidence
カテゴリーが文書にどの程度適合するかを示す標識。値は浮動小数点タイプです。

mostSpecific
カテゴリーが文書に最も適しているかどうかを示すフラグ (uima.cas.Integer タイプ)。

taxonomy
カテゴリーが属する分類法の名前。文書は、異なる分類法のカテゴリーを持つことができます。値は、uima.cas.String タイプです。

関連資料

46 ページの『エンタープライズ・サーチに定義されているタイプおよびフィーチャー』
エンタープライズ・サーチに定義されているタイプ・システムは、文書メタデータ処理および基本的な言語分析をカバーします。

セマンティック検索アプリケーション

4 つのタイプの文書情報が、検索および索引 API (SIAPI) インターフェースを使用して、検索アプリケーションで照会できるエンタープライズ・サーチ索引に保管されます。

以下の 4 つの異なるタイプの情報があります。

- 文書で検出されるテキストの語。例えば、*computer software* などの句。
- スパン名。例えば、<author>James</author> が含まれている XML 文書では、スパン <author> が生じます。
- 属性名。例えば、<author countryOfBirth=USA>James</author> が含まれている XML 文書では、属性「countryOfBirth」が生じます。
- 属性値。例えば、USA は属性「countryOfBirth」の値です。

SIAPI 照会言語は、セマンティック検索照会条件を組み込みます。条件は、 twig (小枝) のパターンを指定します。twig (小枝) は、葉の付いた小さな木です。それぞれの葉は、4 つのタイプの情報 (テキスト・ワード、スパン名など) を示します。ツリーの内部ノードは、文書内のオカレンスの相互関係を指定します。関連を指定する内部ノードには、以下の 5 つのタイプがあります。

- and
- or
- not
- in_the_span_of
- attribute_in_the_span_of

文書に葉のオカレンスが含まれており、内部ノードによって指定された制約 (定義されている関係) が成り立つ場合に、その文書は指定されたセマンティック検索条件を満たしていることになります。

セマンティック検索照会条件は、より的確な文書の検索に役立ちます。語および注釈のブール組み合わせを使用した検索だけでなく、例えば、author というスパンに *James* が含まれている文書を検索したり、同じセンテンス内に *ibm* と *search* という用語がある文書を検索することができるようになります。

セマンティック検索照会用語

セマンティック検索照会条件は、不透明条件と呼ばれます。

検索および索引 API (SIAPI) で不透明条件を表す構文には、以下の 2 つの形式があります。

- XML フラグメント
- 限定 XPath

正しく定義された XML 文書フラグメントのように見えます。 XML フラグメント照会条件は、不透明条件記号 `@xmlf2::` の後に単一引用符 ('...') で囲まれた XML フラグメント式が続きます。

これに対して、限定 XPath 照会用語は、`@xmlxp::` の後に単一引用符 ('...') で囲まれた XPath 照会が続きます。

検索および索引 API (SIAPI) インターフェースの一般的な照会用語と同じように、各用語に出現修飾子を付けることができます。

正符号 (+)
必ずその用語がなければなりません。

= (接頭部)
用語が完全一致していなければなりません。

波形記号 (~) (接頭部)
照会用語の同義語も考慮します。

波形記号 (~) (接尾部)
照会用語と同じ見出し語を持つ語も考慮します。

以下に XML フラグメント照会の例を示します。

@xmlf2::'<City>Springfield</City>'
ストリング Springfield が含まれているスパン (注釈) city が含まれている文書を検索します。

@xmlf2::'<Person gender="female">'
女性が注釈を付けた文書を検索します。

@xmlf2::'<Person><.or><@gender>female</@gender>
<@title>Mrs</@title><@title>Ms</@title><.or></Person>'
性別またはタイトルのいずれかで女性と指定される文書を検索します。

@xmlf2::'<Person gender="male" role="suspect"/>
<PoliceReport><@crimeDescription><.or>robbery theft</.or>-accident
</@crimeDescription></PoliceReport> <City>Springfield<.or>
<@district>Brynston</@district><@district>Brooklyn</@district><.or></City>'
容疑者としてみなされている男性を指定する文書および、*robbery* および *theft* を含む policeReport 注釈を crimeDescription から検索しますが、*accident* は検索しません。この文書には、*Brynston* および *Brooklyn* 地区の city 注釈も含まれていなければなりません。

対応する XPath 照会には、以下の構造があります。

@xmlxp::'//City[ftcontains(Springfield)]'
ストリング *Springfield* が含まれているスパン (注釈) city が含まれている文書を検索します。

@xmlxp::'//Person[@gender="female"]'
女性が注釈を付けた文書を検索します。

@xmlxp::'//Person[@gender="female" or @title ftcontains(Ms) or @title
ftcontains(Mrs)]'
性別またはタイトルのいずれかで女性と指定される文書を検索します。

@xmlxp::'//Person[@gender="male" and @role="suspect"] //PoliceReport
[@crimeDescription ftcontains(robbery) or @crimeDescription ftcontains(theft)]
//City [(@district="Brynston" or @district="Brooklyn") and
ftcontains(Springfield)]'
容疑者としてみなされている男性を指定する文書および、robbery および

theft を含む policeReport 注釈を crimeDescription から検索します。この文書には、*Brynston* および *Brooklyn* 地区の city 注釈も含まれていなければなりません。

検索アプリケーションの同義語サポート

照会用語の同義語を含む文書を検索することで検索結果を拡張することができます。

同義語は、通常 *WebSphere Information Integrator OmniFind* というプロダクト名のような、複数のワードから成る用語を含みます。同義語辞書に含まれる複数のワードから成る用語は、ユーザー照会で正しく識別され、引用符で囲んで表示する必要はありません。

エンタープライズ・サーチの Search and Index API (SIAPI) は、ユーザーが照会用語の同義語を検索する方法を複数サポートしています。

- SIAPI 照会構文は、同義語の拡張のチルド (~) 演算子をサポートしています。ユーザーがこの演算子を照会用語の先頭に付加すると、そのワードに対する同義語の拡張が実施されます。例えば、`~WAS` という照会は、WebSphere Application Server を扱う文書を返し、この省略語について存在するその他の同義語を返します。
- 同義語の拡張は、検索アプリケーション内から SIAPI 同義語の拡張インターフェースを使用して使用可能にすることができます。 照会用語が自動的に拡張されて同義語を含むようになるか、あるいは、検索アプリケーションに、照会用語の同義語を検索結果に戻すかどうかをユーザーが指定できるオプションが含まれている場合があります。

  同義語の自動拡張の間に、照会のすべてのワードとコンテンツ・フィールドに対して同義語の検索が行われます。検索結果には、照会用語か照会用語の同義語のいずれかを含む文書が入ります。また、検索結果には、どの用語がどの同義語に拡張されたかが表示されます。

  ユーザー主導型のシナリオでは、照会が実際に行われる前に、それぞれの照会ワードに対してどの同義語が検出されたかを検索アプリケーションがユーザーに示します。その後、ユーザーはどの用語を検索に含めるかを選択するか、あるいは、再度検索の式を立て直して、元の照会用語を削除します。このシナリオで、ユーザーは、照会にどの用語を組み込むか、すなわち、厳密に同じ値か、あるいは、ワードの多様な意味および用法を含むか、いずれかを制御します。

同義語に使用できる XML ファイルの作成

照会用語の同義語を含めるためにエンタープライズ・サーチで照会を展開するには、XML ファイルで、互いに同義語とみなされるワードを指定する必要があります。

このタスクについて

同義語をリストする XML ファイルは、以下の例に示すスキーマに適合する必要があります。

```
<?xmlversion="1.0" encoding="UTF-8"?>
<synonymgroups xmlns="http://www.ibm.com/of/822/synonym/xml">
  <synonymgroup>
    <synonym>Think Pad</synonym>
    <synonym>Notebook</synonym>
    <synonym>Notebooks</synonym>
  </synonymgroup>
  <synonymgroup>
    <synonym>WebSphere Application Server</synonym>
    <synonym>WAS</synonym>
  </synonymgroup>
</synonymgroups>
```

制約事項

互いに同義語 (<synonym> エレメント) であるワードを <synonymgroup> エレメントでグループ化する必要があります。同義語には、空白文字を含めることができますが、コンマ (,) または垂直バー (|) などの句読文字は含めることができません。これらの文字は、エンタープライズ・サーチ照会構文を妨げる可能性があるためです。

同義語として追加する用語の、考えられるすべての語尾変化を列挙する必要があります。アクセントまたはウムラウトの除去といった、用語の正規化を列挙する必要はありません (エンタープライズ・サーチは正規化を自動的に処理します)。例えば、用語 météo を同義語として含めたい場合、用語 METEO も含める必要はありません。

手順

エンタープライズ・サーチの同義語のリストを作成するには、次のようにします。

1. XML ファイルを作成します。 XML 構文エラーを避けるために、選択した XML エディターまたは XML オーサリング・ツールを使用します。
2. <synonymgroup> エレメントを追加してから、同義語グループの他のワードの同義語として扱われる各ワードの <synonym> エレメントを挿入します。

   マッピングは、必ず、<synonymgroups xmlns="http://www.ibm.com/of/822/synonym/xml">エレメントに入れてください。名前空間 (xmlns 属性に指定) は、表示どおりでなければなりません。
3. エンタープライズ・サーチ・コレクションで文書を検索するのに使用したい同義語をすべて指定するまで、上のステップを繰り返します。
4. XML ファイルを保管して、終了します。

XML ファイルを作成したら、エンタープライズ・サーチ・システムに追加できるように、それを同義語辞書に変換する必要があります。

同義語辞書の作成

同義語のリストを XML ファイルで作成または更新した後で、その XML ファイルを同義語辞書に変換する必要があります。

このタスクについて

同義語辞書を作成するには、WebSphere II OmniFind Edition と一緒に提供される、essyndictbuilder というコマンド行ツールを使用します。このツールは、*ES_INSTALL_ROOT*/bin ディレクトリーにあります。

ツールへの入力は、同義語がリストされる XML ファイルで、ツールからの出力は同義語辞書です。辞書は、接尾部 `.dic` をもっている必要があります。例えば、`c:\mydictionaries\products.dic` です。

どちらのファイルも、デフォルトの場所は、スクリプトが呼び出されるディレクトリーです。同じ名前をもつ辞書が存在する場合、スクリプトがエラーを出します。

手順

エンタープライズ・サーチの同義語辞書を作成するには、次のようにします。

1. 索引サーバーで、エンタープライズ・サーチ管理者としてログインします。 このユーザー ID は、WebSphere II OmniFind Edition のインストール時に指定されたものです。
2. 以下のコマンドを入力します。ここで、*XML_file* は、同義語のリストが含まれている XML ファイルまでの完全修飾パスであり、*DIC_file* は、同義語辞書までの完全修飾パスです。

   AIX、Linux または Solaris: `essyndictbuilder.sh` *XML_file DIC_file*
   Windows: `essyndictbuilder.bat` *XML_file DIC_file*

同義語辞書を作成したら、エンタープライズ・サーチ管理コンソールを使用して、辞書をエンタープライズ・サーチ・システムに追加し、それを 1 つまたは複数のコレクションと関連付けます。

エンタープライズ・サーチ・システムには、生成された .dic ファイルだけがアップロードされます。ソース XML ファイルは、アクセス制御された環境に保持し、ファイルを定期的にバックアップしてください。同義語辞書の更新には、この XML ファイルが必要です。

カスタム・ストップワード辞書

検索の妥当性を高めるために、照会から除去されるエンタープライズ特定の語彙を定義できます。

エンタープライズ・サーチには、次の 2 つの種類のストップワード・サポートがあります。

- マルチ・ワード照会から、頻繁に使用される共通ワード (*a* および *the* など) をすべて除去する、言語特定のストップワード認識。各言語ごとに存在するストップワード辞書は、ユーザーは変更できません。このストップワード認識は、検索の妥当性を高めるために、すべての照会で自動的に実行されます。
- 照会からエンタープライズ特定の語彙を除去する、ユーザー定義またはカスタム・ストップワード認識。このストップワード辞書は、管理者が定義し、特別な語彙のみを含むことができます。ユーザー定義のストップワード辞書は、共通のワードを含むエンタープライズ・サーチの言語特定のストップワード辞書を置き換えません。

ユーザー定義のストップワードは、通常 *WebSphere Information Integrator OmniFind* という製品名などの、複数のワードから成る用語を含みます。ストップワード辞書に含まれる複数のワードから成る用語は、ユーザー照会で正しく識別され、引用符で囲んで表示する必要はありません。

ゲルマン系言語の複合語も、照会で正しく識別されます。複合語とは、単一のワードとして使用される、複数ワードの組み合わせです。 *Reisebüro* (旅行代理店) などの語彙化された複合は、複合とはみなされません。

照会での複合語は、複合を形成するそれぞれの用語に分割されます。複合を形成するそれぞれの用語のいずれかが、ストップワード辞書にある場合、その複合語は照会から除去されません。

たとえば、照会用語 `Versicherungspolice` (保険証書) は、複合語 *Lebensversicherungspolice* (生命保険証書) および *Haftpflichtversicherungspolice* (第三者保険証書) を含む文書を戻します。`Haftpflicht` (第三者保険) などの照会では、2 番目の用語も戻されます。ワード *Police* がストップワード辞書にリストされていても、複合照会用語 `Versicherungspolice` は照会から除去からは除去されません。

エンタープライズ・サーチ・システムに追加できるように、XML ファイルにエンタープライズ特定語彙をリストしてから、ストップワード辞書に変換する必要があります。

どのストップワード辞書を使用するか、エンタープライズ・サーチ管理コンソールで選択できます。それぞれのコレクションに対して、1 つのストップワード辞書を選択できます。ストップワード辞書は、複数のコレクションで共有できます。

ストップワードに使用できる XML ファイルの作成

照会からエンタープライズ特定の語彙を除去するには、どのワードがストップワードとして使用できるか、XML ファイルに指定する必要があります。

このタスクについて

ストップワードをリストする XML ファイルは、以下の例に示すスキーマに適合する必要があります。

```
<?xml version="1.0" encoding="UTF-8"?>
<stopWords xmlns="http://www.ibm.com/of/83/stopwordbuilder/xml">
   <stopWord>OmniFind Edition</stopWord>
   <stopWord>WAS<stopWord>
   <stopWord>...</stopWord>
</stopWords>
```

制約事項

ストップワードには、空白文字を含めることができますが、コンマ (,) または垂直バー (|) などの句読文字は含めることができません。これらの文字は、エンタープライズ・サーチ照会構文を妨げる可能性があるためです。

アクセントまたはウムラウトの除去といった、用語の正規化を列挙する必要はありません (エンタープライズ・サーチは正規化を自動的に処理します)。例えば、用語 météo をストップワードとして含めたい場合、用語 METEO も含める必要はありません。

手順

エンタープライズ・サーチのストップワードのリストを作成するには、次のようにします。

1. XML ファイルを作成します。 XML 構文エラーを避けるために、 XML を妥当性検査できる XML エディターまたは XML オーサリング・ツールを使用します。
2. ストップワードとして扱われる各ワードの `<stopWord>` エレメントを追加します。

   マッピングは、必ず `<stopWords xmlns="http://www.ibm.com/of/83/stopwordbuilder/xml">` エレメントに入れてください。名前空間 (xmlns 属性に指定) は、表示どおりでなければなりません。
3. エンタープライズ・サーチ・コレクションの検索時に、照会から除去したいストップワードをすべて指定するまで、上のステップを繰り返します。
4. XML ファイルを保管して、終了します。

XML ファイルを作成したら、エンタープライズ・サーチ・システムに追加できるように、それをストップワード辞書に変換する必要があります。

ストップワード辞書の作成

ユーザー定義のストップワードのリストを XML ファイルで作成または更新した後で、その XML ファイルをストップワード辞書に変換する必要があります。

このタスクについて

ストップワード辞書を作成するには、WebSphere II OmniFind Edition と一緒に提供される、esstopworddictbuilder というコマンド行ツールを使用します。このツールは、*ES_INSTALL_ROOT*/bin ディレクトリーにあります。

ツールへの入力は、ストップワードがリストされる XML ファイルで、ツールからの出力はストップワード辞書です。辞書は、接尾部 `.dic` をもっている必要があります。たとえば、`c:\mydictionaries\productstopwords.dic` です。

どちらのファイルも、デフォルトの場所は、スクリプトが呼び出されるディレクトリーです。同じ名前をもつ辞書が存在する場合、スクリプトがエラーを出します。

手順

エンタープライズ・サーチのストップワード辞書を作成するには、次のようにします。

1. 索引サーバーで、エンタープライズ・サーチ管理者としてログインします。 このユーザー ID は、WebSphere II OmniFind Edition のインストール時に指定されたものです。
2. 以下のコマンドを入力します。ここで、*XML_file* は、ストップワードのリストが含まれている XML ファイルまでの完全修飾パスであり、 *DIC_file* は、ストップワード辞書までの完全修飾パスです。

   AIX、Linux または Solaris: `esstopworddictbuilder.sh` *XML_file DIC_file*
   Windows: `esstopworddictbuilder.bat` *XML_file DIC_file*

ストップワード辞書を作成したら、エンタープライズ・サーチ管理コンソールを使用して、辞書をエンタープライズ・サーチ・システムに追加し、それを 1 つまたは複数のコレクションと関連付けます。

エンタープライズ・サーチ・システムには、生成された .dic ファイルだけがアップロードされます。ソース XML ファイルは、アクセス制御された環境に保持し、ファイルを定期的にバックアップしてください。ストップワード辞書の更新には、この XML ファイルが必要です。

カスタム・ランキング調整ワード辞書

その用語が出現する文書のランク値を上下する、特定の用語または複数のワードから成る用語を定義できます。

ランキング調整辞書の各用語は、-10 から +10 の範囲のランキング調整因子に関連付けられています。結果文書に特に表示したい用語は、高いランキング調整因子を割り振られ、全く表示したくない用語、または高いランキング調整の用語との組み合わせは、低い値を与えられます。値 -1、0 および 1 には、ランキング調整効果はありません。

特定のランキング調整因子のあるランキング調整辞書にリストされている照会用語が、検索された文書に表示された場合、文書ランク値はランキング調整値にしたがって上下します。用語に割り当てられたランキング調整値は相対で、他の因子によっても影響されます。したがって、用語 X が B1 で、用語 Y が B2 でランキング調整され、B1 > B2 で boost(X) >= boost(Y) になります。

ランキング調整ワードは、通常 *WebSphere Information Integrator OmniFind* という製品名などの、複数のワードから成る用語を含みます。ランキング調整ワード辞書に含まれる複数のワードから成る用語は、ユーザー照会で正しく識別され、引用符で囲んで表示する必要はありません。

ゲルマン系言語の複合語も、照会で正しく識別されます。複合語とは、単一のワードとして使用される、複数ワードの組み合わせです。 *Reisebüro* (旅行代理店) などの語彙化された複合は、複合とはみなされません。

照会での複合語は、複合を形成するそれぞれの用語に分割されます。ランキング調整値がそれぞれの複合の用語に存在する場合、検索された文書はランクされますが、割り当てられた値は、その用語が複合用語の一部でない用語よりも低くなります。これは、検索の有効範囲を広げ、完全な複合を含む文書が、あまり見つからなかった場合にのみ便利です。

たとえば、照会用語 `Versicherungspolice` (保険証書) は、複合語 *Lebensversicherungspolice* (生命保険証書) および *Haftpflichtversicherungspolice* (第三者保険証書) を含む文書を戻します。`Haftpflicht` (第三者保険) などの照会では、後の用語も戻されます。ワード *Police* (ポリシー) がランキング調整辞書に存在する場合は、複合照会用語 `Versicherungspolice` を含む文書は、ランキング調整値を割り当てます。

エンタープライズ・サーチ・システムに追加できるように、XML ファイルにランキング調整値付きの用語をリストしてから、ランキング調整ワード辞書に変換する必要があります。

どのランキング調整ワード辞書を使用するか、エンタープライズ・サーチ管理コンソールで選択できます。各コレクションに対して、1 つのランキング調整辞書を選択できます。ランキング調整ワード辞書は、複数のコレクションで共有できます。

ランキング調整ワードに使用できる XML ファイルの作成

特定の結果文書の重要度を上下するには、XML ファイルにどのワードが文書のランキングに影響するかを指定する必要があります。

このタスクについて

ランキング調整ワードをリストする XML ファイルは、以下の例に示すスキーマに適合する必要があります。

```
<?xml version="1.0" encoding="UTF-8"?>
<boostTerms xmlns="http://www.ibm.com/of/83/boostbuilder/xml">
    <!-- group boost terms by boost value-->
   <boostTermList boost="5">
    <!-- each term can specify the synonym expansion separatly-->
     <term useVariants="true">OmniFind Edition</term>
     <term useVariants="false">Edition</term>
         <term useVariants>OmniFind</term>
   </boostTermList>
   <boostTermList boost="8">
        <term useVariants="true">WAS</term>
        <term>term9</term>
    </boostTermList>
</boostTerms>
```

制約事項

<boostTermList> エレメントで同じランキング調整値を共用する用語をグループ化する必要があります。ランキング調整値は複数回発生する場合があります。たとえば、XML ファイルでランキング調整ワードをアルファベット順にソートしたい場合などです。

ランキング調整ワードには、空白文字を含めることができますが、コンマ (,) または垂直バー (|) などの句読文字は含めることができません。これらの文字は、エンタープライズ・サーチ照会構文を妨げる可能性があるためです。

ランキング調整用語は、通常頭字語または省略語などの変形を持ちます。ランキング調整辞書内のすべての変形を列挙できますが、ランキング調整辞書と共に同義語辞書も使用する計画で、同義語辞書にすでに用語および変形を追加している場合は、ランキング調整辞書にこれらの変形を追加する必要はありません。その代わりに、ランキング調整辞書に追加する変形のために、単に useVariants 属性を true に設定できます。検索された文書に表示される、同義語辞書にリストされたこの用語のすべての変形は、これらの文書に割り当てられたランク値に影響します。

アクセントまたはウムラウトの除去といった、用語の正規化を列挙する必要はありません (エンタープライズ・サーチは正規化を自動的に処理します)。例えば、用語 météo をランキング調整ワードとして含めたい場合、用語 METEO も含める必要はありません。

手順

エンタープライズ・サーチのランキング調整ワードのリストを作成するには、次のようにします。

1. XML ファイルを作成します。 XML 構文エラーを避けるために、選択した XML エディターまたは XML オーサリング・ツールを使用します。

2. マッピングは必ず、<boostTerms xmlns="http://www.ibm.com/of/83/boostbuilder/xml"> エレメントに入れてください。名前空間 (xmlns 属性に指定) は、表示どおりでなければなりません。
3. 指定したランキング調整値を共用する、すべての用語をグループ化するために、<boostTermList> エレメントを追加します。

   ランキング調整値は -10 から 10 の範囲です。たとえば、<boostTermList boost="-5"> または <boostTermList boost="5"> のようになります。

   指定した用語を含む文書の重要度は、指定されたランキング調整値にしたがって上下します。
4. <term> エレメントを、指定したランキング調整値を使用する各用語のために追加します。

   同義語辞書にリストされたランキング調整ワードの変形を組み込みたい場合は、<term> エレメントの useVariants 属性を true に設定します。デフォルトは false です。変形が同義語辞書に見つからない場合、エラー・メッセージは生成されません。
5. エンタープライズ・サーチ・コレクションの検索時に、ランキング調整ワードとして使用される用語をすべて指定するまで、上のステップを繰り返します。
6. XML ファイルを保管して、終了します。

XML ファイルを作成したら、エンタープライズ・サーチ・システムに追加できるように、それをランキング調整ワード辞書に変換する必要があります。

ランキング調整ワード辞書の作成

ランキング調整ワードのリストを XML ファイルで作成または更新した後で、その XML ファイルをランキング調整ワード辞書に変換する必要があります。

このタスクについて

ランキング調整ワード辞書を作成するには、WebSphere II OmniFind Edition と一緒に提供される、esboostworddictbuilder というコマンド行ツールを使用します。このツールは、ES_INSTALL_ROOT/bin ディレクトリーにあります。

ツールへの入力は、ランキング調整ワードがリストされる XML ファイルで、ツールからの出力はランキング調整ワード辞書です。辞書は、接尾部 .dic をもっている必要があります。たとえば、c:\mydictionaries\productboostwords.dic です。

どちらのファイルも、デフォルトの場所は、スクリプトが呼び出されるディレクトリーです。同じ名前をもつ辞書が存在する場合、スクリプトがエラーを出します。

手順

エンタープライズ・サーチのランキング調整ワード辞書を作成するには、次のようにします。

1. 索引サーバーで、エンタープライズ・サーチ管理者としてログインします。 このユーザー ID は、WebSphere II OmniFind Edition のインストール時に指定されたものです。

2. 以下のコマンドを入力します。ここで、*XML_file* は、ランキング調整ワードのリストが含まれている XML ファイルまでの完全修飾パスであり、 *DIC_file* は、ランキング調整ワード辞書までの完全修飾パスです。同義語辞書も使用したい場合は、ランキング調整辞書名の後に、同義語辞書の完全修飾パスを追加します。同義語辞書への命名はオプションです。

   UNIX: `esboostworddictbuilder.sh` *XML_file DIC_file SYNDIC_file*
   Windows: `esboostworddictbuilder.bat` *XML_file DIC_file SYNDIC_file*

ランキング調整ワード辞書を作成したら、エンタープライズ・サーチ管理コンソールを使用して、辞書をエンタープライズ・サーチ・システムに追加し、それを 1 つまたは複数のコレクションと関連付けます。

エンタープライズ・サーチ・システムには、生成された .dic ファイルだけがアップロードされます。ソース XML ファイルは、必ず、適切なバックアップ方針が整った状態で、アクセス制御された環境に保持してください。ランキング調整ワード辞書の更新には、この XML ファイルが必要です。

関連タスク

58 ページの『同義語辞書の作成』
同義語のリストを XML ファイルで作成または更新した後で、その XML ファイルを同義語辞書に変換する必要があります。

エンタープライズ・サーチに組み込まれているテキスト分析

エンタープライズ・サーチに組み込まれているテキスト分析には、言語検出とセグメンテーションが含まれています。

文書処理時に、エンタープライズ・サーチはその文書の言語を判別し、入力テキストのストリームを別個の単位またはトークンに分割します。

検索時に、ユーザー、つまりアプリケーションは、手動で照会言語を選択する必要があります。照会ストリングは、セグメント化され、分析され、索引内で検索されます。

文書分析も照会ストリング分析も、以下に分割されます。

- 基本的な非辞書ベースのサポート。これには、空白によるセグメンテーションと N-gram セグメンテーションがあります。
- 辞書ベースの言語サポート。これには、語およびセンテンス・セグメンテーションと見出し語処理があります。

  言語処理では、字句解析が行われます。これは、入力テキストの代替表記を作成する処理で、有効なすべての辞書データを、入力テキストにおいて認識されたトークンに関連付けます。拡張言語処理を使用することにより、検索品質は一段と向上します。

  関連概念

  『言語の識別』
  エンタープライズ・サーチでは、語およびセンテンスのセグメンテーション、文字の正規化、 見出し語処理を行う前に、ソース・ドキュメントの言語を判別 する必要があります。

  70 ページの『非辞書ベース・セグメンテーションに関する言語サポート』
  言語検出および字句解析テクノロジーによってサポートされない言語による文書の場合、 エンタープライズ・サーチは、Unicode ベースの空白によるセグメンテーションおよび N-gram セグメンテーション の形式で基本サポートを提供します。

言語の識別

エンタープライズ・サーチでは、語およびセンテンスのセグメンテーション、文字の正規化、見出し語処理を行う前に、ソース・ドキュメントの言語を判別する必要があります。

エンタープライズ・サーチは、以下の言語を自動的に検出することができます。

アラビア語
中国語 (繁体字および簡体字)
チェコ語
デンマーク語
オランダ語
英語
フィンランド語
フランス語
ドイツ語
ギリシャ語
ヘブライ語
ハンガリー語
イタリア語
日本語
韓国語
ポーランド語
ポルトガル語
ロシア語
スペイン語
スウェーデン語
トルコ語

エンタープライズ・サーチの言語処理では、照会処理時ではなく、索引作成時にソース・ドキュメントの言語を検出します。

エンタープライズ・サーチでは、文書の言語を自動的に検出するか、または使用する言語を選択するかを指定できます。

自動言語検出を選択していて、パーサーが文書の言語を検出できない場合、パーサーは、エンタープライズ・サーチ管理コンソールでクローラーを作成した時に指定した言語を使用します。

自動言語検出を選択しない場合は、指定した言語が常に使用されます。デフォルト値は英語です。

空白セグメンテーションおよび n-gram セグメンテーションなどの、基本的な言語独自のテクノロジーを使用して、言語特定の辞書が処理されていない文書用です。

エンタープライズ・サーチの言語検出テクノロジーは、単一言語文書に最も適しています。文書が複数の言語で書かれている場合は、その文書で最も多く使用されている言語を判別します。ただし、その場合、分析結果が常に満足できるものであるとは限りません。

文書の言語を使用して、検索結果を特定言語で書かれている文書のみに制限することができます。例えば、 Jacques Chirac に関する文書を検索する場合、検索結果にフランス語で書かれている文書のみが含まれるように指定することができます。

関連概念

69 ページの『エンタープライズ・サーチに組み込まれているテキスト分析』
エンタープライズ・サーチに組み込まれているテキスト分析には、言語検出とセグメンテーションが含まれています。

『非辞書ベース・セグメンテーションに関する言語サポート』
言語検出および字句解析テクノロジーによってサポートされない言語による文書の場合、 エンタープライズ・サーチは、Unicode ベースの空白によるセグメンテーションおよび N-gram セグメンテーション の形式で基本サポートを提供します。

非辞書ベース・セグメンテーションに関する言語サポート

言語検出および字句解析テクノロジーによってサポートされない言語による文書の場合、エンタープライズ・サーチは、Unicode ベースの空白によるセグメンテーションおよび N-gram セグメンテーションの形式で基本サポートを提供します。

Unicode ベースの空白によるセグメンテーション
この言語処理方式は、語間の空白 (またはブランク・スペース) を語の区切り文字として使用します。

N-gram セグメンテーション
この言語処理方式は、*n* 文字のオーバーラップするシーケンスを単一の語として扱います。この簡単なセグメンテーション方式は、多くの検索タスクで十分に使用できます。

これらの方式は、言語辞書に依存せず、基本型への変換などの複雑な言語処理テクノロジーも組み込まれていません。

N-gram セグメンテーションは、区切り文字としてブランク・スペースを使用しないタイ語などの言語に使用されます。同じ方式が、ヘブライ語やアラビア語に適用されます。これらの 2 つの言語は、空白区切り文字を使用しますが、N-gram セグメンテーションを使用した方が、Unicode ベースの空白によるセグメンテーションの基本形式を使用するより、良い結果が得られます。

関連概念

69 ページの『エンタープライズ・サーチに組み込まれているテキスト分析』
エンタープライズ・サーチに組み込まれているテキスト分析には、言語検出とセグメンテーションが含まれています。

69 ページの『言語の識別』
エンタープライズ・サーチでは、語およびセンテンスのセグメンテーション、文字の正規化、 見出し語処理を行う前に、ソース・ドキュメントの言語を判別 する必要があります。

辞書ベース・セグメンテーションに関する言語サポート

文書の言語が正しく検出され、言語固有の辞書が使用可能である場合には、該当する言語処理が適用されます。

セグメンテーションとは、入力テキストを個別の字句単位にブレークダウンするプロセスのことです。このプロセスには、以下のいくつかの言語処理アクティビティーが含まれます。

語のセグメンテーション

語のセグメンテーションは、日本語や中国語のように、語の間に空白 (または区切り文字) を使用しない言語に使用されます。

見出し語処理

見出し語処理は、テキスト内のそれぞれの語形の見出し語を判別する言語処理形式です。語の *見出し語* は、語の基本型に加えて、同じ品詞を共用する語形変化型も含めます。例えば、見出し語 *go* には、*go*、*goes*、*went*、*gone* および *going* が含まれます。名詞グループの見出し語には、単数形と複数形が含まれます (*calf* と *calves* など)。形容詞グループの見出し語には、比較級、最上級形が含まれます (*good*、*better*、*best* など)。代名詞グループの見出し語には、同じ代名詞のさまざまな格が含まれます (*I*、*me*、*my*、*mine* など)。

見出し語処理では、索引付けと検索の両方に辞書が必要です。

エンタープライズ・サーチは、見出し語と語形変化した語の索引付けを行い、照会内のすべての語形変化した語に見出し語を対応させます。見出し語処理は、照会において、さまざまな語形変化した語が含まれている文書を検出することにより、検索の品質を向上させます。例えば、照会に *mouse* という語が含まれている場合、 *mice* という語が含まれている文書も検出されます。

短縮形の分割

短縮形を識別して、それをコンポーネント・パーツに分割することによって、検索の品質が向上します。例:

wouldn't は *would* と *not* に分割されます。
Horse's は *Horse* と *is* または *'s* に分割されます (照会のあいまいさを考慮するため)

接語の識別

接語は特殊な形式の短縮形で、接語の構成要素を判別することにより、検索の品質が向上します。*接語* は、接辞および語のような性質を持っています。ただし、接語は、語形成の一部でもあるため、識別が難しくなります。他の形態構造的な (語構造) 事象と異なり、接語は統語的な構造内にあり、語に結び付いている接語は、語形成規則の一部ではありません。例:

reparti-lo-emos には、 *repartir* と *lo* と *emos* の構成要素があります。
l'avenue には、*le* と *avenue* の構成要素があります。
dell'arte には、*dello* と *arte* の構成要素があります。

英字以外の文字認識

言語処理は、英字以外の文字を認識します。英字以外の文字は、内部的な言語依存ロジックに従って、さまざまなタイプの個別の字句単位として戻されたり、グループ化されたりします。

例えば、アポストロフィやハイフンは、接語内にある場合は語の一部とみなされ、不明な省略形内にある場合は終止符 (またはピリオド) とみなされます。また、言語処理は、一部の特殊なシーケンスの文字 (例えば、URL、E メール・アドレス、日付など) をトークンとして認識します。

省略形の認識

言語処理は、辞書にある省略形を 1 つの字句単位として認識します。省略形が辞書に無い場合、その省略形は字句項目として認識されますが、その省略形には関連した辞書情報がありません。

省略形を正しく認識することは、文の認識においてきわめて重要です。例えば、省略形の語尾にあるピリオドは、必ずしもセンテンスの終わりを示すものではありません。

センテンスの終わりを示すマーカーの認識

言語処理は、センテンスのセグメンテーションのためにセンテンスの終わりを示すマーカーを正しく識別します。

辞書ベースの言語サポートは、以下の言語で有効です。

アラビア語
中国語 (繁体字および簡体字)
チェコ語
デンマーク語
オランダ語
英語
フィンランド語
フランス語 (本国およびカナダで使用されているもの)
ドイツ語 (本国およびスイスで使用されているもの)
ギリシャ語
イタリア語
日本語
韓国語
ノルウェー語 (ブークモールおよびニーノシュク)
ポーランド語
ポルトガル語 (本国およびブラジルで使用されているもの)
ロシア語
スペイン語
スウェーデン語

関連概念

73 ページの『日本語における語のセグメンテーション』
テキスト文書または照会ストリングが日本語であると認識されると、 エンタープライズ・サーチは、日本語に最適化された形態学的な分析を使用して、 適切な語のセグメンテーションを行います。

『日本語における変種文字』
日本語は、多数の変種文字を使用します。カタカナは、外来語のスペルや発音によく使用されるため、最も重要です。 日本語では、多数のカタカナがよく使用されます。

日本語における語のセグメンテーション

テキスト文書または照会ストリングが日本語であると認識されると、エンタープライズ・サーチは、日本語に最適化された形態学的な分析を使用して、適切な語のセグメンテーションを行います。

この最適化の例が語分解です。日本語は、多数の複合語を使用します。これらの語は、検索結果の精度を向上させるために、最適なサイズのトークンに分解されます。語形変化した語句や接頭語も、検索効率を上げるために、分解されます。

関連概念

71 ページの『辞書ベース・セグメンテーションに関する言語サポート』
文書の言語が正しく検出され、言語固有の辞書が使用可能である場合には、 該当する言語処理が適用されます。

『日本語における変種文字』
日本語は、多数の変種文字を使用します。カタカナは、外来語のスペルや発音によく使用されるため、最も重要です。 日本語では、多数のカタカナがよく使用されます。

日本語における変種文字

日本語は、多数の変種文字を使用します。カタカナは、外来語のスペルや発音によく使用されるため、最も重要です。日本語では、多数のカタカナがよく使用されます。

エンタープライズ・サーチは、変種文字対応辞書を使用して、標準的なカタカナをその基本型 (見出し語に類似したもの) にマップし、照会ストリングにカタカナが含まれている文書を含む、すべての文書を検索できるようにします。

また、エンタープライズ・サーチは、漢字の語尾にひらがなで書かれている標準的な送り仮名もサポートします。

関連概念

71 ページの『辞書ベース・セグメンテーションに関する言語サポート』
文書の言語が正しく検出され、言語固有の辞書が使用可能である場合には、 該当する言語処理が適用されます。

『日本語における語のセグメンテーション』
テキスト文書または照会ストリングが日本語であると認識されると、 エンタープライズ・サーチは、日本語に最適化された形態学的な分析を使用して、 適切な語のセグメンテーションを行います。

ストップワードの除去

エンタープライズ・サーチでは、検索効率を上げるために、複数語照会からすべてのストップワード (例えば *a* や *the* などの共通の語) が除去されます。

日本語におけるストップワードの認識は、文法的な情報に基づいて行われます。例えば、エンタープライズ・サーチは、他の言語では特殊なリストに基づいてストップワードを認識しますが、日本語については、語が名詞であるか動詞であるかに基づいてストップワードを認識します。

関連概念

『文字の正規化』
文字の正規化は、想起性を改善するプロセスです。文字を正規化して想起性を改善すると、文書が照会に完全に一致していなくても、より多くの文書が検索されることになります。

文字の正規化

文字の正規化は、想起性を改善するプロセスです。文字を正規化して想起性を改善すると、文書が照会に完全に一致していなくても、より多くの文書が検索されることになります。

エンタープライズ・サーチは、アジア言語の全角および半角文字の正規化を含む Unicode 互換の正規化を使用します。

例えば、日本語では、全角の英数字は半角文字に正規化され、半角のカタカナは全角文字に正規化されます。また、エンタープライズ・サーチは、日本語で複合語の区切り文字として使用されるカタカナの中黒を除去します。

文字の正規化のその他の形式には、以下のものがあります。

大/小文字の正規化
例えば、*usa* と指定された検索で *USA* が含まれている文書を検索します。

ウムラウトの拡張
例えば、*schön* と指定された検索で *schoen* が含まれている文書を検索します。

アクセントの除去
例えば、*e* と指定された検索で *é* が含まれている文書を検索します。

その他の発音符の除去
例えば、*c* と指定された検索で *ç* が含まれている文書を検索します。

合字の拡張
例えば、*ae* と指定された検索で *Æ* が含まれている文書を検索します。

すべての正規化は、両方向に作用します。*USA* と指定した検索で *usa* が含まれている文書を検索することも、*é* と指定した検索で *e* の付く語が含まれている文書を検索することもできます。これらの正規化を組み合わせて使用することもできます。例えば、*METEO* と指定した検索で *météo* が含まれている文書を検索することができます。

正規化は、Unicode 文字特性に基づいており、言語に依存しません。例えば、エンタープライズ・サーチは、ヘブライ語における発音符の除去、およびアラビア語における合字の拡張をサポートします。

関連概念

73 ページの『ストップワードの除去』
エンタープライズ・サーチでは、検索効率を上げるために、複数語照会からすべてのストップワード (例えば *a* や *the* などの共通の語) が除去されます。

エンタープライズ・サーチの資料

WebSphere Information Integrator OmniFind Edition の資料は、 PDF または HTML で読むことができます。

WebSphere Information Integrator OmniFind Edition インストール・プログラムは、「インフォメーション・センター」を自動的にインストールします。インストール・プログラムは検索サーバーに「インフォメーション・センター」をインストールします。複数サーバー・インストールの場合、「インフォメーション・センター」は、両方の検索サーバーにインストールされます。「インフォメーション・センター」をインストールしていない場合、「ヘルプ」をクリックすると、 IBM Web サイトの「インフォメーション・センター」が開きます。エンタープライズ・サーチの「HTML トピック」を参照するには、「インフォメーション・センター」を開始します。

PDF 資料を参照するには、docs/*locale*/pdf に移動します。例えば、英語で資料を見つけるには、docs/en_US/pdf に移動します。 PDF 資料、ダウンロード、フィックス、技術情報、およびインフォメーション・センターを、 WebSphere Information Integrator OmniFind Edition サポート・サイトで表示できます。

以下のテーブルは、使用可能な資料、ファイル名、ロケーションを示します。

表 9. エンタープライズ・サーチ用 PDF 資料

ヘッダー	ヘッダー	ヘッダー
エンタープライズ・サーチのインストール・ガイド (この文書のトピックは、「インフォメーション・センター」でも使用可能です。)	iiysi.pdf	docs/*locale*/pdf/
エンタープライズ・サーチの管理 (この文書のトピックは、「インフォメーション・センター」でも使用可能です。)	iiysa.pdf	docs/*locale*/pdf/
エンタープライズ・サーチ プログラミング・ガイドおよび API リファレンス (この文書のトピックは、「インフォメーション・センター」でも使用可能です。)	iiysp.pdf	docs/*locale*/pdf/
メッセージ・リファレンス (この文書のトピックは、「インフォメーション・センター」でも使用可能です。)	iiysm.pdf	docs/*locale*/pdf/

表 9. エンタープライズ・サーチ用 *PDF* 資料 (続き)

ヘッダー	ヘッダー	ヘッダー
エンタープライズ・サーチのインストール要件 (この文書のトピックは、「インフォメーション・センター」でも使用可能です。)	iiysr.txt または iiysr.htm	docs/*locale*/ (このファイルは、「ファースト・ステップ」プログラムから起動することも可能です。)
リリース情報	iiysn.pdf	IBM WebSphere Information Integrator OmniFind Edition 資料 の Web サイトでのみ入手可能です。
テキスト分析機能ガイド	iiyst.pdf	docs/*locale*/pdf/

WebSphere II OmniFind Edition アクセシビリティ

IBM WebSphere Information Integrator OmniFind Edition のユーザー・インターフェースおよび資料はアクセス可能です。

インストール・プログラム

キーボード・ショートカットを使用して、 WebSphere II OmniFind Edition インストール・プログラム全体を移動することができます。以下のテーブルは、キーボード・ショートカットを説明しています。

表 10. インストール・プログラム用キーボード・ショートカット

アクション	ショートカット
ラジオ・ボタンの強調表示	矢印キー
ラジオ・ボタンの選択	タブ・キー
プッシュボタンの強調表示	タブ・キー
プッシュボタンの選択	Enter キー
次のウィンドウまたは前のウィンドウへ移動、またはキャンセル	タブ・キーを押してプッシュボタンを強調表示し、Enter キーを押す
アクティブ・ウィンドウを非アクティブにする	Ctrl + Alt + Esc

エンタープライズ・サーチ管理コンソールおよびインフォメーション・センター

管理コンソールおよびインフォメーション・センターは、 Microsoft® Internet Explorer または Mozilla FireFox で表示することのできるブラウザー・ベースのインターフェースです。ブラウザーのキーボード・ショートカットのリストおよび他のアクセシビリティ機能については、Internet Explorer または FireFox のオンライン・ヘルプを参照してください。

PDF 資料

エンタープライズ・サーチ資料のすべてを PDF で表示できます。PDF 文書は、Adobe Acrobat Version 6.0 によって利用できます。PDF 文書は、ほとんどのスクリーン・リーダー用に構造化され、それによって読むことができるはずです。

エンタープライズ・サーチの用語集

この用語集では、エンタープライズ・サーチのインターフェースおよび資料で使用される用語を定義します。

アクセス制御リスト (access control list)
関連付けられたオブジェクトにアクセスできるユーザーを識別し、そのオブジェクトへのユーザーのアクセス権限を指定するリスト。

管理役割 (administrative role)
エンタープライズ・サーチ管理コンソールで実行できる機能を決めるユーザー種別。この役割により、ユーザーが管理できるコレクションも決まる。

分析エンジン (analysis engine)
『テキスト分析エンジン』を参照。

分析結果 (analysis results)
アノテーターが生成する情報。分析結果は、検索したい情報に相当するもので、共通分析構造と呼ばれるデータ構造に書き込まれます。

注釈 (annotation)
テキストのスパンに関する情報。例えば、注釈は、テキストのスパンが会社名を表すことを指示することもあり得ます。UIMA では、注釈は、特別な種類のフィーチャー構造です。

アノテーター (annotator)
特定の言語分析タスクを実行して、注釈を生成し、記録するソフトウェア・コンポーネント。アノテーターは、分析エンジンにおける分析論理コンポーネントです。

ブール検索 (boolean search)
1 つ以上の検索語が、AND、NOT、OR などの演算子を使って結合された検索。

ランキング調整クラス (boost class)
検索結果内の文書の相対的ランクに影響を与えることのできる仕様。

ランキング調整ワード (boost word)
検索結果内の文書の関連ランクに影響を与えることのできるワード。照会処理中に、ワードに事前定義したスコアに従って、ランキング調整ワードを含む文書の重要度を調整することも可能です。

カテゴリー (category)
類似した特性を持つ文書のグループ。

カテゴリー・ツリー (category tree)
エンタープライズ・サーチ管理コンソールに表示されるカテゴリー階層。

証明書 (certificate)
公開鍵を証明書の所有者の ID に結合するデジタル文書で、それによって証明書の所有者を認証済みにすることができます。証明書は、認証局 (CA) によって発行されます。

認証局 (certificate authority)
証明書を発行する組織。CA は、電子トランザクションに含まれるエンティティー (個人または法人) を認証します。認証局は、情報を交換する二者が、実際に、主張しているとおりの者であることを保証します。

文字の正規化 (character normalization)
大文字化や発音区別符号など、文字の異体形式が共通形式に合わせられる処理。

接語 (clitic)
構文的には分離して機能するが、音声学的には別のワードに接続するワード。接語は、結合されるワードとは、接続して書かれたり、分離して書かれたりします。接語の一般的な例としては、英語における縮小語の終わりの部分が含まれます (*would**n't*** または *you**'re***)。

コレクション (collection)
データ・ソースと、そのクロール、解析、索引作成、検索用のオプションの集合。

共通分析構造 (common analysis structure)
テキスト分析エンジンによって分析される文書を保管する構造。情報は、注釈の形式の共通分析構造および他の機能構造で保管されます。

共通通信層 (CCL) (Common Communication Layer)
WebSphere Information Integrator OmniFind Edition のさまざまなコンポーネント (コントローラー、パーサー、クローラー、インデクサー) を結びつける通信インフラストラクチャー。

概念抽出 (concept extraction)
テキスト文書にある重要な語彙項目 (人、場所、製品など) を識別し、その項目リストを生成する検索機能。『テーマ抽出』も参照。

クロール・スペース (crawl space)
指定パターン (データベース名、ファイル・システム・パス、ドメイン・ネーム、 IP アドレス、URL など) に一致するソースの集合。クローラーはここから読み取って索引用の項目を取り出す。

クローラー (crawler)
データ・ソースから文書を取り出し、検索索引作成用の情報を収集するソフトウェア・プログラム。

信用証明情報 (credential)
認証中に獲得される詳細情報で、ユーザー、グループ・アソシエーション、および他のセキュリティー関連識別属性を記述しています。信用証明情報は、許可、監査、および委任など、多数のサービスを実行するのに使用できます。

データ・ソース (data source)
文書を検索できるデータ・リポジトリー。 Web、リレーショナルおよび非リレーショナル・データベース、およびコンテンツ・マネージメント・システムなど。

データ・ソース・タイプ (data source type)
データ・アクセス用のプロトコルに応じたデータ・ソースのグループ。

デキュー (dequeue)
キューから項目を除去すること。

付加記号 (diacritic)
アクセント記号、またはドイツ語のウムラウトなど、ワードの発音を変更したり、同じワードの間で区別したりする場合に、文字に追加されるマーク。

ディスカバラー (discoverer)
クローラー機能の 1 つで、クローラーが情報検索に使用できるデータソースを判別する機能。

識別名 (distinguished name)
ディレクトリーのエントリーを一意的に識別する名前。識別名は、コンマで分離された「属性:値 (attribute:value)」ペアで構成されます。また、デジタル証明書のエンティティーを一意的に識別する名前/値ペアのセット (例: CN=個人の名前および C=国または地域)。

Document Object Model (DOM)
XML ファイルなど、構造化文書を、プログラマチックにアクセスおよび更新できるオブジェクトのツリーとして表示するシステム。

Domino® Document Manager キャビネット (Domino Document Manager cabinet)
文書を編成するのに使用される Domino Document Manager データベース。キャビネットが Domino データベースを保持します。

Domino Document Manager ライブラリー (Domino Document Manager library)
Domino Document Manager に対するエントリー・ポイントである Domino Document Manager データベース。

Domino Internet Inter-ORB Protocol (DIIOP)
サーバー上で稼働し、Domino Object Request Broker と連動して、 Notes Java クラスを使用して作成される Java アプレットと Domino サーバーとの間の通信を可能にするサーバー・タスク。ブラウザー・ユーザーおよび Domino サーバーは、DIIOP を使用して通信し、オブジェクト・データを交換します。

動的ランキング (dynamic ranking)
照会の条件を検索中の文書に関して分析し、結果のランクを決定するランキングのタイプ。『テキスト・ベースのスコアリング』も参照。『静的ランキング』と対比。

動的要約 (dynamic summarization)
要約タイプの 1 つ。検索語が強調表示され、検索結果には検索している文書の概念を最もよく表す句が含まれる。『静的要約』と対比。

エンキュー (enqueue)
キューに項目を入れること。

エンタープライズ・サーチ管理者 (enterprise search administrator)
エンタープライズ・サーチ・システム全体を管理できる管理役割。

エスケープ文字 (escape character)
後続の 1 つ以上の文字に対して特殊な意味を抑制または設定する文字。

外部データ・ソース (external data source)
WebSphere Information Integrator OmniFind Edition によってクロール、構文

解析、または索引付けされていないフェデレーションに対するデータ・ソース。外部データ・ソースの検索は、それらのデータ・ソースの照会アプリケーション・プログラミング・インターフェースに委任されます。

フィーチャー・パス (feature path)
UIMA フィーチャー構造内のフィーチャーの値にアクセスするのに使用されるパス。

フィーチャー構造 (feature structure)
テキスト分析の結果を表す、基礎となるデータ構造。フィーチャー構造は、属性値構造です。各フィーチャー構造は、タイプに属します。すべてのタイプは、Java クラスと非常に類似している、有効なフィーチャーまたは属性の指定されたセットを持ちます。

フェデレーテッド・サーチ (federated search)
複数の検索サービスにわたって検索を可能にし、検索結果の統合化されたリストを戻す検索機能。

フェデレーション (federation)
命名システムを結合する処理。それによって、集合システムは、命名システムをスパンする複合名を処理できます。

フィールド (field)
レコードの最小識別可能パート。

フィールド検索 (fielded search)
特定のフィールドに限定された照会。

フリー・テキスト検索 (free text search)
フリー・フォーム・テキストで検索語を表現した検索。

フルテキスト索引 (full text index)
データ項目を参照し、照会用語を含む文書を検索で迅速に見つけられるようにするデータ構造。

ファジー検索 (fuzzy search)
検索語にスペルが似た語を戻す検索。

ハイブリッド検索 (hybrid search)
ブール検索とフリー・テキスト検索を組み合わせたもの。

ID 管理 (identity management)
セキュア保管でユーザー信用証明情報を暗号化する能力。

索引 (index)
『フルテキスト索引』を参照。

索引キュー (index queue)
索引の再編成要求、または処理される索引のリフレッシュ要求のリスト。

索引のリフレッシュ (index refresh)
エンタープライズ・サーチ・システム内の既存の索引に新しい情報を追加する処理。『索引の再編成』と対比。

索引の再編成 (index reorganization)
エンタープライズ・サーチ・システムで索引を作成する処理。『索引のリフレッシュ』と対比。

情報抽出 (information extraction)
概念抽出のタイプの 1 つで、テキスト文書内の重要な語彙項目 (名前、用語、式など) を自動的に認識するもの。

IP アドレス (IP address)
ネットワーク上のホストを識別する固有 32 ビット・アドレス。

Java Database Connectivity (JDBC)
Java プラットフォームと広範なデータベースとの間のデータベース依存接続の業界標準。 JDBC インターフェースは、SQL ベースのデータベース・アクセスに対する呼び出しレベルの API を提供します。

JavaScript™
ブラウザーおよび Web サーバーで使用される Web スクリプト言語。

JavaServer Pages (JSP)
動的コンテンツをクライアントに戻すために、Java コードを動的に Web ページ (HTML ファイル) に組み込むことができるようにし、そのページのサービスが提供されるとき実行するサーバー・スクリプト・テクノロジー。

Java 仮想マシン (JVM) (Java virtual machine (JVM))
コンパイル済み Java コード (アプレットおよびアプリケーション) を実行するプロセッサーのソフトウェア・インプリメンテーション。

カタカナ (Katakana)
2 つの一般的日本語表音文字の 1 つで使用されるシンボルから構成される文字セット。外国語のワードを表音的に書く場合に主に使用されます。

鍵ストア・ファイル (keystore file)
署名者証明書として保管される公開鍵、および個人証明書に保管される秘密鍵を含むキー・データベース・ファイル。

言語の識別 (language identification)
文書の言語を判別するエンタープライズ・サーチ機能。

見出し語 (lemma)
ワードの正規書式。見出し語は、チェコ語など、大きく屈折する言語では重要です。

見出し語化 (lemmatization)
辞書に提示されているワードに代わって見出し語を検索する処理。見出し語化は、ステミングとは異なります。ステミングは、アルゴリズム的であり、ある言語のワードをリストする辞書を操作しません。

字句類縁性 (lexical affinity)
文書内で相互に密接な関係で出現する検索語間の関係。字句類縁性を使用して、結果の適合度を算出する。

ライブラリー (library)
他のオブジェクトにディレクトリーとしてサービスを提供するシステム・オブジェクト。『Domino Document Manager ライブラリー』を参照。

合字 (ligature)
結びつけることによって 1 つの文字として表示される複数の文字。例: f と i を結びつけて合字 fi を形成。

Lightweight Directory Access Protocol (LDAP)
X.500 モデルをサポートするディレクトリーへのアクセス権を提供する TCP/IP を使用し、より複雑な X.500 ディレクトリー・アクセス・プロトコルのリソース要件の影響を受けないオープン・プロトコル。

言語分析検索 (linguistic search)
基本形に戻したり (例: *mice* は *mouse* として索引付けされる)、または基本形を使用して拡張したり (複合語語のように) した語句を使用して文書を表示、取得、索引付けする検索タイプ。

リンク分析 (link analysis)
文書間のハイパーリンクの分析に基づき、コレクション内のどのページがユーザーにとって重要かを判別するための方法。

ローカル・フェデレーター (Local Federator)
検索可能なオブジェクトのセット全体をフェデレートするクライアント・フェデレーター。

Lotus® QuickPlace® プレース (Lotus QuickPlace place)
地理的に分散した参加者が、構造化されてセキュアなワークスペースにおいて、プロジェクトで共同作業し、オンラインで通信することができるようになる、Lotus QuickPlace によって提供される Web の場。

Lotus QuickPlace ルーム (Lotus QuickPlace room)
共通の興味、および集合的作業の必要を共有する、許可されたメンバーに制限された Lotus QuickPlace プレースのパーティション化領域。

マスク文字 (masking character)
検索語の先頭、中間、および末尾にある任意の文字を表す文字。マスク文字は通常、索引で語の異形を検索するために使用される。『ワイルドカード文字』も参照。

MIME タイプ (MIME type)
インターネット全体にわたって転送されるオブジェクトのタイプを識別するためのインターネット標準。

モデル・ベースのカテゴリー (model-based category)
類似した内容を含む文書で文書を索引付けおよび検索するために、文書の主題の判別に使われる事前定義用語を使用した分類構造。

モニター担当者 (monitor)
コレクション・レベルのプロセスを監視する権限を持つエンタープライズ・サーチ・ユーザー。

自然言語照会 (natural language query)
キーワードを単純に並べるのではなく、文語表現 (「Who runs the finance department?」など) を分析する検索のタイプ。

改行文字 (newline character)
印刷または表示位置を 1 行下へ移動させる制御文字。システムによっては、複数の文字を要求するものもあります。

N-gram セグメンテーション (n-gram segmentation)
Unicode ベースの空白文字セグメンテーションのように、ワードを区切るのにブランク・スペースを使用するよりはむしろ、所与の数の文字の重複シーケンスを単一文字とみなす分析方法。

リンクをたどらないディレクティブ (no-follow directive)
Web ページで検出されるリンクをたどらないように、ロボット (Web クローラーなど) を指示する、それらのページ内のディレクティブ。

索引付けしないディレクティブ (no-index directive)
索引内の Web ページのコンテンツを含まないように、ロボット (Web クローラーなど) を指示する、それらのページ内のディレクティブ。

Notes リモート・プロシージャー・コール (NRPC) (Notes remote procedure call (NRPC))
すべての Notes-to-Notes 通信に使用される Lotus Notes® のアーキテクチャー層。

オペレーター (operator)
コレクション・レベルのプロセスを監視、開始、停止する権限を持つエンタープライズ・サーチ・ユーザー。

パラメトリック検索 (parametric search)
指定された範囲内の数値または属性 (日付、整数、その他のデータ・タイプなど) を含むオブジェクトを探す検索のタイプ。

パーサー (parser)
エンタープライズ・サーチ・データ・ストアに追加された文書を解釈するプログラム。パーサーは、文書から情報を抽出し、索引付け、検索、取得の準備を行う。

プレース (place)
個人やグループが共同作業するために出会うポータルで、可視になる仮想ロケーション。ポータルでは、各ユーザーは、専用作業のための個人用プレースを持ち、個人やグループは、さまざまな共有スペースへのアクセス権を持ちます。そこは、パブリック・プレースにも、制限されたプレースにもなり得ます。『Lotus QuickPlace プレース』をも参照。

高頻度ランキング (popular ranking)
文書の既存のランキングに、その検索頻度を基に加算するランキング・タイプ。

処理エンジン・アーカイブ (processing engine archive)
UIMA 分析エンジン、およびエンタープライズ・サーチのカスタム分析に使用するために要求されるリソースのすべてを含む .pear zip アーカイブ・ファイル。

近接検索 (proximity search)
同一のセンテンス、パラグラフ、または文書にある一定の語を探す検索タイプ。

プロキシー・サーバー (proxy server)
アプリケーションまたは Web サーバーがホストする HTTP Web 要求に対する中継として動作するサーバー。プロキシー・サーバーは、エンタープライズのコンテント・サーバーの代理として動作します。

クイック・リンク (quick link)
URI とキーワードと句の間のアソシエーション。

ランキング (ranking)
照会検索結果の各文書に整数値を割り当てるプロセス。検索結果における文

書の順序は、照会への適合度に基づいて決まる。ランクが高いほど、緊密な一致を表す。『動的ランキング』と『静的ランキング』も参照。

リモート・フェデレーター (Remote Federator)
検索可能なオブジェクトのセットをフェデレートするサーバー・フェデレーター。

ロボット排他プロトコル (Robots Exclusion Protocol)
サイトのある部分をロボットが訪問しないように、Web サイト管理者が、訪問するロボットに指示できるようにするプロトコル。

ルーム (room)
ユーザーが、他の人々が読む文書を作成し、他の人々からのコメントに応答し、プロジェクトの状況と期限を検討することができるようにするプログラム。ユーザーは、同じルームにいる他の人々とチャットすることもできます。『Lotus QuickPlace ルーム』をも参照。

ルール・ベースのカテゴリー (rule-based category)
どの文書が、どのカテゴリーと関連付けられるかを指定する規則によって作成されるカテゴリー。例えば、一定の語を含む、または含まない文書や一定の URI パターンに一致する文書を、特定のカテゴリーと関連付ける規則を定義する。

有効範囲 (scope)
検索要求の範囲を定義するのに使用される関連 URI のグループ。

検索アプリケーション (search application)
エンタープライズ・サーチ・システムで、照会の処理、索引の検索、検索結果の表示、コレクション用のソース文書の取得を行うプログラム。

検索キャッシュ (search cache)
以前の検索要求のデータと結果を保持するバッファー。

検索エンジン (search engine)
検索要求を受け取り、文書リストをユーザーに戻すプログラム。

検索索引ファイル (search index files)
検索エンジンで索引が保管されているファイルのセット。

検索結果 (search results)
検索要求に一致する文書のリスト。

Secure Sockets Layer (SSL)
通信プライバシーを提供するセキュリティー・プロトコル。

セキュリティー・トークン (security token)
コレクションの文書へのアクセス許可に使用される ID とセキュリティーに関する情報。データ・ソース・タイプによって、サポートするセキュリティー・トークンのタイプは異なる。例えば、ユーザー役割、ユーザー ID、グループ ID や、コンテンツへのアクセス制御用のその他の情報などがある。

シード URL (seed URL)
クロールの開始点。

セグメンテーション (segmentation)
パス制御が基本情報単位を、BIU セグメントと呼ばれる、より小さな単位に分割する処理。これによって、隣接サーバーの、より小さなバッファー・サイズに対応します。

サーブレット (servlet)
Web サーバー上で稼働し、Web クライアント要求に対する応答として動的コンテンツを生成することにより、サーバーの機能性を拡張する Java プログラム。サーブレットは、一般的に、データベースを Web に接続するのに使用されます。

ソフト・エラー・ページ (soft error page)
クライアントが要求したページを HTTP サーバーが戻すことができなかった場合に、詳細に問題を説明し、問題の内容を示す戻りコード付きのヘッダーのみで構成される応答の代わりに、これらのページを戻すように HTTP サーバーを構成する特別なページ。

静的ランキング (static ranking)
ランキング・タイプの 1 つ。日付や、その文書を指すリンク数など、ランキングされる文書に関する係数でランキングが上がる。『動的ランキング』と対比。

静的要約 (static summarization)
要約タイプの 1 つ。検索結果には、文書の指定および保管された要約が含まれる。『動的要約』と対比。

ステミング (stemming)
『語のステミング』を参照。

ストップワード (stop word)
共通に使用されるワードで、*the*、*an*、または*and* など、検索アプリケーションが無視するもの。

ストップワードの除去 (stop word removal)
共通ワードを無視して、より関連性のある結果を戻すために、照会から索引からストップワードを共通を除去するプロセス。

要約 (summarization)
文書の内容を簡潔に記述する文を検索結果に組み込むプロセス。『動的要約』と『静的要約』も参照。

同義語辞書 (synonym dictionary)
ユーザーがコレクションを検索するときに、その照会用語の同義語を検索できるようにする辞書。

分類構造 (taxonomy)
類似性に基づいてオブジェクトをグループに分類したもの。エンタープライズ・サーチでは、分類構造によってデータはカテゴリーとサブカテゴリーに編成される。『カテゴリー・ツリー』も参照。

テキスト分析 (text analysis)
コレクションのデータの検索性を高めるために、テキストから意味やその他の情報を抽出するプロセス。

テキスト分析エンジン (text analysis engine)
テキスト内のコンテキストおよびセマンティック・コンテンツを検索および表すことに関与するソフトウェア・コンポーネント。

テキスト・ベースのスコアリング (text-based scoring)
照会内の語に対する文書の適合度を表す整数値を、文書に割り当てるプロセス。整数値が大きいほど、照会への一致が緊密であることを表す。『動的ランキング』も参照。

テーマ抽出 (theme extraction)
概念抽出のタイプの 1 つで、テキスト文書内の重要な語彙項目を自動的に認識して、文書のテーマやトピックを抽出するもの。『概念抽出』も参照。

トークン (token)
エンタープライズ・サーチによって索引付けされる基本テキスト単位。トークンは、言語内のワードにすることもできますし、索引付けに適切な、他のテキスト単位にすることもできます。

トークナイザー (tokenizer)
テキストをスキャンし、一続きの文字をトークンとして認識できる場合に、それを判別するテキスト・セグメンテーション・プログラム。

末尾の文字 (trailing character)
ワードにおける最後の位置を保持する文字。

Unicode ベースの空白スペース・セグメンテーション (Unicode-based white space segmentation)
トークンと区切り文字を区別するために Unicode 文字プロパティーを使用するトークン化の方式。

URI (Uniform Resource Identifier)
抽象的または物理的リソースを識別するコンパクトな文字ストリング。

URL (Uniform Resource Locator)
コンピューター上、またはインターネットなどのネットワーク内の情報リソースを表す一続きの文字。この一続きの文字には、その情報リソースへのアクセスに使用されるプロトコルの省略名、およびそのプロトコルが情報リソースを見つけるために使用する情報が含まれる。

Universal Resource Name (URN)
特定の構文に準拠する文字の短ストリングからなるインターネット・プロトコル・エレメント。そのストリングは、リソースを参照するのに使用できる名前またはアドレスから構成されます。

非構造化情報管理アーキテクチャー (UIMA) (Unstructured Information Management Architecture (UIMA))
非構造化データの分析用システムをインプリメントするフレームワークを定義するn IBM アーキテクチャー。

ユーザー・エージェント (user agent)
Web をブラウズし、アクセスしたサイトに自身の情報を残すアプリケーション。エンタープライズ・サーチで、Web クローラーはユーザー・エージェント。

Web クローラー (Web crawler)
Web 文書を検索したり、文書内のリンクをたどったりすることによって、Web を探索するロボット・ソフトウェアのクラス。

用語加重検索 (weighted term search)
一定の用語が重視される照会。

ワイルドカード文字 (wildcard character)
検索語の先頭、中間、または末尾にある任意の文字を表す文字。

語のステミング (word stemming)
言語学的な正規化のプロセス。1 つのワードの異形を一般形に分解する。例えば、*connections*、*connective*、および *connected* のようなワードは *connect* に戻されます。

XML パス言語 (XPath) (XML Path Language (XPath))
一意的に識別されたり、またはソースの XML 文書の一部にアドレッシングしたりする言語。XPath はまた、ストリング、数値、およびブール演算子の操作のための基本機能を提供します。

WebSphere Information Integration に関する情報へのアクセス

WebSphere Information Integration 製品に関する情報は、Web により入手できます。

WebSphere Information Integration に関する情報は、次の Web で入手できます: www.ibm.com/software/data/integration/db2ii/。このサイトには、次の最新情報が入っています。

- 製品資料
- 製品のダウンロード
- フィックスパック
- リリース情報およびその他のサポート資料
- WebSphere Information Integration に関するニュース
- Web リソースへのリンク (ホワイト・ペーパーおよび IBM Redbooks™ など)
- ニュースグループおよびユーザー・グループへのリンク
- WebSphere Information Integration 製品のオンライン・インフォメーション・センターへのリンク
- 資料の注文方法

製品資料にアクセスするには、次のようにします。

1. Web サイト www.ibm.com/software/data/integration/db2ii/ にアクセスします。
2. ドロップダウン・リストから製品 (例えば、WebSphere Information Integrator OmniFind Edition) を選択します。
3. ページの左側にある「サポート」リンクをクリックします。
4. 「学習」セクションで、必要なリンクを選択します。選択した製品でインフォメーション・センターが使用可能な場合は、インフォメーション・センターのリンクを選択できます。例については、94 ページの図 1 を参照してください。

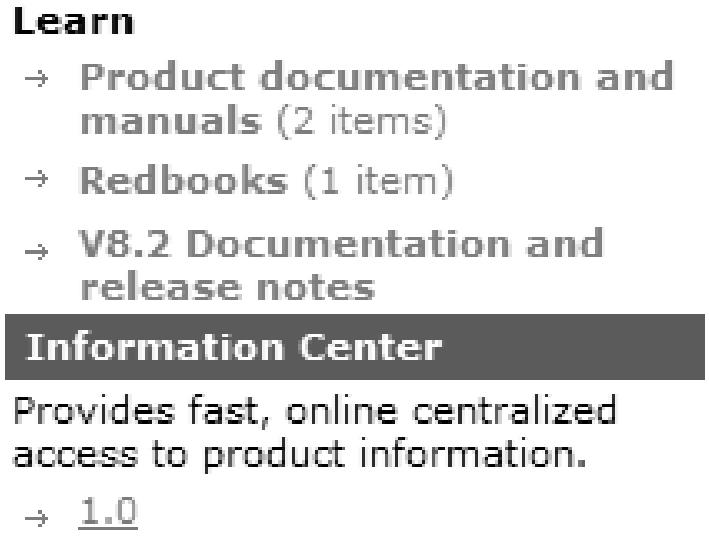

図 1. WebSphere Information Integration サポート Web サイト上の製品資料へのリンクの例

IBM と連絡を取る

お客様の国または地域で IBM に連絡する方法については、Web の www.ibm.com/planetwide にある「IBM Directory of Worldwide Contacts」にアクセスしてください。

商標

ここでは、IBM の商標と、特定の IBM 以外の商標をリストします。

IBM の商標に関する情報については、http://www.ibm.com/legal/copytrade.shtml を参照してください。

以下は、それぞれ各社の商標または登録商標です。

Java およびすべての Java 関連の商標およびロゴは、Sun Microsystems, Inc. の米国およびその他の国における商標または登録商標です。

Microsoft、Windows、Windows NT および Windows ロゴは、Microsoft Corporation の米国およびその他の国における商標です。

Intel、Intel Inside (ロゴ)、および Pentium は、Intel Corporation の米国およびその他の国における商標です。

UNIX は、The Open Group の米国およびその他の国における登録商標です。

Linux は、Linus Torvalds の米国およびその他の国における商標です。

他の会社名、製品名およびサービス名等はそれぞれ各社の商標です。

特記事項

本書は米国 IBM が提供する製品およびサービスについて作成したものであり、米国以外の国においては本書で述べる製品、サービス、またはプログラムを提供しない場合があります。日本で利用可能な製品、サービス、および機能については、日本 IBM の営業担当員にお尋ねください。本書で IBM 製品、プログラム、またはサービスに言及していても、その IBM 製品、プログラム、またはサービスのみが使用可能であることを意味するものではありません。これらに代えて、IBM の知的所有権を侵害することのない、機能的に同等の製品、プログラム、またはサービスを使用することができます。ただし、IBM 以外の製品とプログラムの操作またはサービスの評価および検証は、お客様の責任で行っていただきます。

IBM は、本書に記載されている内容に関して特許権 (特許出願中のものを含む) を保有している場合があります。本書の提供は、お客様にこれらの特許権について実施権を許諾することを意味するものではありません。実施権についてのお問い合わせは、書面にて下記宛先にお送りください。

〒106-0032
東京都港区六本木 3-2-31
IBM World Trade Asia Corporation
Licensing

以下の保証は、国または地域の法律に沿わない場合は、適用されません。 IBM およびその直接または間接の子会社は、本書を特定物として現存するままの状態で提供し、商品性の保証、特定目的適合性の保証および法律上の瑕疵担保責任を含むすべての明示もしくは黙示の保証責任を負わないものとします。国または地域によっては、法律の強行規定により、保証責任の制限が禁じられる場合、強行規定の制限を受けるものとします。

この情報には、技術的に不適切な記述や誤植を含む場合があります。本書は定期的に見直され、必要な変更は本書の次版に組み込まれます。 IBM は予告なしに、随時、この文書に記載されている製品またはプログラムに対して、改良または変更を行うことがあります。

本書において IBM 以外の Web サイトに言及している場合がありますが、便宜のため記載しただけであり、決してそれらの Web サイトを推奨するものではありません。それらの Web サイトにある資料は、この IBM 製品の資料の一部ではありません。それらの Web サイトは、お客様の責任でご使用ください。

IBM は、お客様が提供するいかなる情報も、お客様に対してなんら義務も負うことのない、自ら適切と信ずる方法で、使用もしくは配布することができるものとします。

本プログラムのライセンス保持者で、(i) 独自に作成したプログラムとその他のプログラム（本プログラムを含む）との間での情報交換、および (ii) 交換された情報の相互利用を可能にすることを目的として、本プログラムに関する情報を必要とする方は、下記に連絡してください。

IBM Corporation J46A/G4
555 Bailey Avenue
San Jose, CA 95141-1003 U.S.A.

本プログラムに関する上記の情報は、適切な使用条件の下で使用することができますが、有償の場合もあります。

本書で説明されているライセンス・プログラムまたはその他のライセンス資料は、IBM 所定のプログラム契約の契約条項、IBM プログラムのご使用条件、またはそれと同等の条項に基づいて、IBM より提供されます。

この文書に含まれるいかなるパフォーマンス・データも、管理環境下で決定されたものです。そのため、他の操作環境で得られた結果は、異なる可能性があります。一部の測定が、開発レベルのシステムで行われた可能性がありますが、その測定値が、一般に利用可能なシステムのものと同じである保証はありません。さらに、一部の測定値が、推定値である可能性があります。実際の結果は、異なる可能性があります。お客様は、お客様の特定の環境に適したデータを確かめる必要があります。

IBM 以外の製品に関する情報は、その製品の供給者、出版物、もしくはその他の公に利用可能なソースから入手したものです。 IBM は、それらの製品のテストは行っておりません。したがって、他社製品に関する実行性、互換性、またはその他の要求については確証できません。 IBM 以外の製品の性能に関する質問は、それらの製品の供給者にお願いします。

IBM の将来の方向または意向に関する記述については、予告なしに変更または撤回される場合があり、単に目標を示しているものです。

本書には、日常の業務処理で用いられるデータや報告書の例が含まれています。より具体性を与えるために、それらの例には、個人、企業、ブランド、あるいは製品などの名前が含まれている場合があります。これらの名称はすべて架空のものであり、名称や住所が類似する企業が実在しているとしても、それは偶然にすぎません。

著作権使用許諾:

本書には、様々なオペレーティング・プラットフォームでのプログラミング手法を例示するサンプル・アプリケーション・プログラムがソース言語で掲載されています。お客様は、サンプル・プログラムが書かれているオペレーティング・プラットフォームのアプリケーション・プログラミング・インターフェースに準拠したアプリケーション・プログラムの開発、使用、販売、配布を目的として、いかなる形式においても、IBM に対価を支払うことなくこれを複製し、改変し、配布することができます。このサンプル・プログラムは、あらゆる条件下における完全なテストを経ていません。従って IBM は、これらのサンプル・プログラムについて信頼性、利便性もしくは機能性があることをほのめかしたり、保証することはできません。お客様は、IBM のアプリケーション・プログラミング・インターフェースに準拠したアプリケーション・プログラムの開発、使用、販売、配布を目的として、いかなる形式においても、 IBM に対価を支払うことなくこれを複製し、改変し、配布することができます。

それぞれの複製物、サンプル・プログラムのいかなる部分、またはすべての派生した創作物には、次のように、著作権表示を入れていただく必要があります。

Outside In (®) Viewer Technology, ©1992-2004 Stellent, Chicago, IL., Inc. All Rights Reserved.

IBM XSLT Processor Licensed Materials - Property of IBM ©Copyright IBM Corp., 1999-2004. All Rights Reserved.

索引

日本語, 数字, 英字, 特殊文字の順に配列されています。なお, 濁音と半濁音は清音と同等に扱われています。

[ア行]

[カ行]

[サ行]

[タ行]

[ナ行]

[ハ行]

[マ行]

[ラ行]

D

E

J

N

P

U

W

X

Printed in Japan

SD88-6728-00

日本アイ・ビー・エム株式会社
〒106-8711　東京都港区六本木3-2-12