



テキスト分析機能ガイド

お願い

本書および本書で紹介する製品をご使用になる前に、117ページの『特記事項および商標』に記載されている情報をお読みください。

本書は、IBM OmniFind Enterprise Edition (製品番号 5724-C74) バージョン 8、リリース 5、モディフィケーション 0、および新しい版で明記されていない限り、以降のすべてのリリースおよびモディフィケーションに適用されます。

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。

IBM 発行のマニュアルに関する情報のページ

<http://www.ibm.com/jp/manuals/>

こちらから、日本語版および英語版のオンライン・ライブラリーをご利用いただけます。また、マニュアルに関するご意見やご感想を、上記ページよりお送りください。今後の参考にさせていただきます。

(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： SC18-9674-02
IBM OmniFind Enterprise Edition
Version 8.5
Text Analysis Integration

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

目次

ibm.com および関連リソース	v	カスタム・ストップワード辞書	69
セマンティック検索に関する言語サポート	1	ストップワード用 XML ファイルの作成	70
カスタム・テキスト分析の組み込み	3	ストップワード辞書の作成	71
テキスト分析処理で使用される基本概念	4	カスタム・ランキング調整ワード辞書	73
テキスト分析アルゴリズム	5	ランキング調整ワードに使用できる XML ファイル の作成	74
カスタム分析の組み込みのワークフロー	6	ランキング調整ワード辞書の作成	75
UIMA 内でのエンタープライズ・サーチ・ベース・ア ノテーターの使用	7	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	77
UIMA 内でのデータベース・コンシューマーへの 共通分析構造の使用	10	言語の識別	77
UIMA 内での正規表現アノテーターの使用	12	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	78
ベース・アノテーターとカスタム・テキスト分析結 果の表示	13	数字の N-gram トークンとしてのトークン化	79
タイプ・システム記述	15	辞書ベース・セグメンテーションに関する言語サポ ート	80
基本分析モードから拡張分析モードへの変更	16	日本語における語のセグメンテーション	82
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	17	日本語における変種文字	82
エンタープライズ・サーチに固有なタイプとフィー チャー	22	ストップワードの除去	82
タイプ・システム記述のサンプル	25	文字の正規化	83
分析および検索における XML マークアップ	28	正規表現アノテーター	85
XML エレメントから共通分析構造へのマッピン グ・ファイルの作成	30	正規表現アノテーターを使用した簡単なセマンティ ック検索	86
テキスト分析結果	35	正規表現アノテーターを使用した簡単なセマンティ ック検索の使用可能化	86
フィーチャー・パス	36	規則セット・ファイル	88
組み込みフィーチャー	37	正規表現規則の定義	89
フィルター	40	正規表現アノテーターのカスタマイズ	92
カスタム分析結果の索引マッピング	41	アノテーター・ディスクリプター	93
共通分析構造から索引へのマッピング・ファイル の作成	42	ロギング	96
選択した分析結果のデータベース・マッピング	48	エンタープライズ・サーチの資料	99
分析結果のデータベースへの保管	49	アクセシビリティ機能	101
ロード・ファイル・セットの使用	49	エンタープライズ・サーチの用語集	103
共通分析構造からデータベースへのマッピング・ ファイルの作成	50	特記事項および商標	117
「コンテナ・タイプ」のマッピング	55	特記事項	117
セマンティック検索照会に一致する文書の部分の取 得	59	商標	119
セマンティック検索アプリケーション	62	索引	121
セマンティック検索照会条件	62		
検索アプリケーションの同義語サポート	65		
同義語用 XML ファイルの作成	65		
同義語辞書の作成	67		

ibm.com および関連リソース

製品サポートおよび文書は、ibm.com® より入手できます。

サポートおよび支援

製品サポートを Web 上で受けられます。

IBM® OmniFind™ Enterprise Edition

<http://www.ibm.com/software/data/enterprise-search/omnifind-enterprise/support.html>

IBM OmniFind Discovery Edition

<http://www.ibm.com/software/data/enterprise-search/omnifind-discovery/support.html>

IBM OmniFind Yahoo! Edition

<http://www.ibm.com/software/data/enterprise-search/omnifind-yahoo/support.html>

インフォメーション・センター

Web ブラウザーによる Eclipse ベースのインフォメーション・センターで製品資料を表示できます。インフォメーション・センターは、<http://publib.boulder.ibm.com/infocenter/discover/v8r5m0/> にアクセスしてご覧ください。

PDF 資料

ご使用の OS で Adobe® Acrobat Reader を使用して、オンラインで PDF 資料を閲覧できます。ご使用の OS に Acrobat Reader がインストールされていない場合は、Adobe の Web サイト <http://www.adobe.com> からダウンロードできます。

PDF 資料の Web サイトを以下に示します。

製品	Web サイト・アドレス
OmniFind Enterprise Edition バージョン 8.5	http://www.ibm.com/support/docview.wss?rs=63&uid=swg27010938
OmniFind Discovery Edition バージョン 8.4	http://www.ibm.com/support/docview.wss?rs=3035&uid=swg27008552
OmniFind Yahoo! Edition バージョン 8.4	http://www.ibm.com/support/docview.wss?rs=3193&uid=swg27008932

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、および日本語をはじめとするアジア言語で書かれたテキスト文書の言語検索サポートを提供します。

言語サポートを使用して、検索結果の質を上げることができます。

言語処理は、索引に追加されるために文書が処理されるときと、ユーザーが検索照会を入力するときの 2 つのステージで行われます。

エンタープライズ・サーチには、入力文書の言語の判別と、文書入力ストリームを語またはトークンにセグメント化するために必要なおおまかな基本的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。

テキスト文書の中のほとんどの情報は構造化されておらず、情報の意味に簡単にアクセスできないため、有効に使用することができなくなっています。

キーワードの検索は簡単ですが、次の例に示すように、文書内の単なる語の検索を超えた検索が必要な場合には、それ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・ケースの場合、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。例えば、E メール内の住所や電話番号などです。実際に、電話番号 という用語がまったく使用されないこともあります。その代わりに、E メールに、「555-641-1805 に連絡してください。」などのフレーズが含まれることがあります。多くの場合、ユーザーは検索する情報が文書の中にどのような形で入っているかを知りません。そして、例えばバーバラという人の電話番号を検索するときに、理想的には「バーバラの電話番号」というような照会を入力したいと思います。しかし、この照会は、文書の中に電話番号 という語は入っていないため、失敗に終わります。
- 競合に関する情報において、文書に競合相手およびその競合相手が提供する商品、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が記述されることがあります。この場合、ユーザーは「スミス社 商品」または「スミス社 商品 2004 年 11 月から 2005 年 1 月まで」というような照会を入力するかもしれません。最初の照会では、用語商品は製品または製品群を表しますが、この照会が用語商品 を検索するため、スミス社が提供する製品を戻しません。特定の期間を含んだ照会にも同じことが言えます。キーワード検索を使用して期間を照会することは、ほとんど不可能です。
- カスタマー・リレーションシップ・マネージメントにおいて、文書に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ことがあります。修理工場のレポートには「油圧漏れのため、シュー調整」といった状態の記述があります。ユーザーはさらに詳細な「北部サンフランシスコのブレーキ故障の修理工場」のような照会を入力するかもしれません。しかし、この照会では、用語ブレーキ故障 または修理工場 といった用語がそのレポートに現れない

ため、「油圧漏れのため、シュー調整」を記したどのレポートも戻さないかもしれません。その上、これらのレポートには、都市名のサンフランシスコを含めた完全な住所ではなく、修理工場の地区と通りの名前しか含まれていないかもしれません。

- 研究分野。様々な登録商標の下に広く市場に出された特定の薬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。ユーザーは、症状を含めた様々な病気の詳細な説明が返ってくることを期待して、思い付きで、薬の、一般に使用されている名前の 1 つを使って照会を入力するかもしれません。しかし、薬の一般に使用されている名前が文書で使用されているとは限らず、また、これらの文書は多くの場合、語病気をまったく含まず、病気そのものの名前のみを含むため、照会によって期待した文書が戻されることはないでしょう。

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、文書コンテンツは分析されて、インタレストの概念を認識して検出する必要があります。インタレストの概念には、例えば、人、組織、場所、設備、および製品などの名前の付けられたエンティティと、それらのエンティティ間にありうる関係などがあります。

テキスト文書の中で発見し抽出する情報は、ユーザーおよびドメインに固有です。ユーザー独自の分析アルゴリズムの設計と開発を支援するために、IBM は、IBM Unstructured Information Management Architecture (UIMA) を提供します。これは、エンタープライズ・サーチで、文書コレクションの中のインタレスト情報を検索する、拡張分析機能の作成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。

関連概念

3 ページの『カスタム・テキスト分析の組み込み』

4 ページの『テキスト分析処理で使用される基本概念』

カスタム・テキスト分析の組み込み

Unstructured Information Management Architecture (UIMA) を使用してエンタープライズ・サーチの外でカスタム分析を作成したあと、エンタープライズ・サーチ管理コンソールを使用して分析ロジックをエンタープライズ・サーチに組み込むことができます。

UIMA は、それぞれの、概念的に別個の分析機能の構成要素を特定するオープン・プラットフォームで、これらの構成要素の再使用と結合を簡単に行うことができるようにするものです。

高機能言語分析に、多数のいろいろな分析タスクを含めることができます。分析では、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。最後のタスクでは、例えば、ある科学物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、その前のステップの結果によって異なります。

各ステップの分析ロジックは、アノテーター の中に含まれます。アノテーターは結合して、新規情報を発見しその情報を次の処理のために保管するために、コレクション内の各文書に渡って繰り返して行う処理チェーンを形成します。

テキスト文書内の分析コンテンツの検出と対応付けを担当するアノテーターは、UIMA の中心的な概念である分析エンジン に含まれています。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。

UIMA は、ユーザー固有の分析エンジンを作成、テスト、およびデプロイするための基本的な構築ブロックのみを提供します。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。しかし、エンタープライズ・サーチで適用される言語処理は、UIMA の中で一緒に使用できる一連のアノテーターとして使用可能です。

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。Development Kit は、IBM developerWorks® から入手できます。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。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。

UIMA SDK はまた、Eclipse ベースの開発環境 (Eclipse プラグイン) の中で UIMA と共に機能する一連のツールとユーティリティーも提供します。Eclipse の詳細は、www.eclipse.org を参照してください。また、UIMA Software Development Kit を Eclipse Interactive Development Environment にインストールする方法については、UIMA の資料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』

テキスト分析処理で使用される基本概念

テキスト分析処理で使用される基本概念には、アノテーター、分析結果、フィーチャー構造、タイプ、タイプ・システム、注釈、および共通分析構造が含まれます。

アノテーター は、文書を分析し、文書全体 (文書メタデータと呼ばれる) または文書の部分についての記述データを検出して記録するロジックを含みます。この記述データは分析結果 と呼ばれます。分析結果は、テキスト文書に近いサブスリング (スパンとも呼ばれる) を注釈として付けるものです。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。

フィーチャー構造 は、分析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) 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。

例えば、テキスト・スパン「James Matthew Bloggs」は、フィーチャー `personName`、`age`、`nationality`、および `profession` を持つタイプ `Person` の注釈で補われるでしょう。

タイプ・システム は、文書の中で発見される可能性のあるオブジェクトのタイプ (フィーチャー構造) を定義します。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 (属性) 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。タイプ・システムの中に、異なるタイプをいくつでも定義できます。タイプ・システムはドメインおよびアプリケーションに固有です。

ほとんどのテキスト分析アノテーターは、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。注釈は、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、フィーチャー構造の中のデータをモデルにして保管します。

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

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』

3 ページの『カスタム・テキスト分析の組み込み』

テキスト分析アルゴリズム

UIMA Software Development Kit には、API およびツールが組み込まれています。これらを使用して、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、これらのアノテーターを分析エンジンに組み込むことができます。

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含まれています。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対して、より高度なセマンティック検索を行うこともできます。

UIMA Software Development Kit は事前に構成されたアノテーターを提供しないため、UIMA を使用して作成しエンタープライズ・サーチに組み込んだカスタム・アノテーターはエンタープライズ・サーチ・ベース・アノテーターの結果の上に作成されるために、ベース・アノテーター・パッケージを UIMA 環境に対して使用することができます。使用している UIMA 環境でカスタム・テキスト分析アルゴリズムを実行する前に、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を開発しテストをしたあと、エンタープライズ・サーチのドキュメント・コレクションでそのアルゴリズムを実行するために、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。アーカイブの作成方法は、Software Development Kit 中の UIMA 文書に説明されています。

エンタープライズ・サーチにアップロードするために作成されるアーカイブには、カスタム分析ロジックのみを含めてください。ベース・アノテーターは常にエンタープライズ・サーチの中のカスタム分析よりも先に実行されるため、たとえカスタム分析ロジックがベース・アノテーター結果の上に作成されたとしても、エンタープライズ・サーチ・ベース・アノテーターは 1 つも含めないでください。

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

7 ページの『UIMA 内でのエンタープライズ・サーチ・ベース・アノテーターの使用』

カスタム分析の組み込みのワークフロー

UIMA Software Development Kit を使用してカスタム・テキスト分析アルゴリズムを作成しテストしたあと、それをデプロイし、エンタープライズ・サーチの文書コレクションに対して実行します。

分析アルゴリズムを作成し、エンタープライズ・サーチに組み込むには、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。リトリートしたい文書は何ですか？それぞれの特定の検索タスクにどの概念と関係が必要ですか？例えば、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-治癒の関連を知る必要があります。
 - b. リトリート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で役に立つ <summary> または <topic> エlementが含まれている場合)、XML Elementと共通分析構造とのマッピング・ファイルを作成します。
 - 照会で、XML マークアップを、文書にあるとお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する必要があります。
 - d. セマンティック検索を使用してアクセスできるようにする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。共通分析構造と索引とのマッピング・ファイルを作成します。
 - 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断します。例えば、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。共通分析構造とデータベースとのマッピング・ファイルを作成します。
 - 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断します。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. UIMA Software Development Kit を使用して、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、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あと、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。

テキスト分析ソリューションを設計するときに、複数の PEAR ファイルから提供された分析モジュールをいくつか組み込む場合があります。UIMA は、1 つのファイルをアップロードしてエンタープライズ・サーチで実行できるように、2 つ以上の PEAR ファイルを 1 つの PEAR ファイルにマージする方法を提供します。PEAR ファイルのマージ機能によって、名前の衝突がないこと、入出力機能が正しくマージされること、そして、アノテーター・ディスクリプターの中でマージされたパラメーターが同じ名前を持っていてもパラメーターがオーバーライドされることはないことが保障されます。PEAR ファイルのマージ方法については、UIMA 文書を参照してください。

3. デプロイ: 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. 処理エンジン・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。テキスト分析コンポーネントの名前を指定します。その名前を使用して、そのコンポーネントをエンタープライズ・サーチの中で参照します。
 - b. 1 つ以上の文書コレクションをテキスト分析コンポーネントに関連付けます。
 - c. 該当する場合は、コレクションごとに、カスタム分析用に定義した、XML エlement から共通分析構造へのマッピングをアップロードし選択します。
 - d. 該当する場合は、コレクションごとに、カスタム分析用に定義した共通分析構造からデータベースへのマッピングをアップロードし選択します。
 - 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、共通分析構造から索引へのマッピングをアップロードし選択します。
 - 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。例えば、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、クロール、構文解析および索引付けします。

関連タスク

『UIMA 内でのエンタープライズ・サーチ・ベース・アノテーターの使用』

UIMA 内でのエンタープライズ・サーチ・ベース・アノテーターの使用

エンタープライズ・サーチ・ベース・アノテーター・パッケージ内のアノテーターを使用して、UIMA Software Development Kit (SDK) 内で新規アノテーターを作成し、分析結果を JDBC 表にマップできます。

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できます。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 語およびハンガルのテキスト・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。このリストには、すべての主要なアジアのテキスト・スクリプトが含まれ、アノテーターが日本語、中国語、および韓国語をサポートすることを意味します。

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。

- **正規表現アノテーター**

テキスト文書の中のエンティティまたは情報のスパンを正規表現に基づいて検出します。必要なテキスト・エンティティを検出するために、独自の規則を定義して、正規表現アノテーターをカスタマイズすることができます。テキスト文書の中の電話番号、URL、および E メール・アドレスを検出するサンプルの正規表現アノテーターがベース・アノテーター・パッケージに含まれています。

- **データベース・コンシューマーへの共通分析構造**

データベース・コンシューマーへの共通分析構造は、リレーショナル・データベースに特定のテキスト分析結果を追加します。

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、ベース・テキスト分析アノテーター、正規表現アノテーター、およびデータベース・コンシューマーへの共通分析構造を含んだ圧縮ファイルです。言語 ID アノテーター、FROST 辞書検索アノテーター、および空白トークナイザーはベース・テキスト分析アノテーターで、エンタープライズ・サーチの中で文書が解析されるときに、常にどのカスタム・テキスト分析よりも先に実行されます。

エンタープライズ・サーチの中でベース・テキスト分析アノテーターがどのカスタム・テキスト分析よりも常に先に実行されるため、また、すべてのカスタム・テキスト分析はベース・アノテーターの出力に基づいているため、カスタム・アノテーターを作成しテストするときに、これらのアノテーターを UIMA 環境で使用できます。

データベース・コンシューマーの正規表現アノテーターと共通分析構造は、テキスト処理オプションを構成するときに、エンタープライズ・サーチ管理コンソールで選択できる追加のオプションです。また、これらは UIMA の中でも使用できます。正規表現アノテーターの高度なカスタマイズについては、提供された UIMA SDK ツールを使用してアノテーターをカスタマイズすることをお勧めします。

UIMA の中で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.ibm.com/developerworks/db2/zones/db2ii/>) から入手できます。

アノテーター・パッケージを UIMA SDK インストールにインストールするには、次のようにします。

1. アノテーター・パッケージ `OF_base_annotators.zip` をエンタープライズ・サーチ (OmniFind Enterprise Edition) インストールの `ES_INSTALL_ROOT/packages/uima` ディレクトリーの中で見つけます。
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。ファイル `tt_core_typesystem.xml` は上書きされます。このファイルの古いバージョンを取っておきたい場合は、圧縮ファイルを解凍する前に保存してください。
4. クラス・パスを設定するには、`bin` ディレクトリーの中の `setUIMAClasspath` スクリプトを開き、スクリプトの最後に、`OFAnotEnv` スクリプトを開始する行を追加します。
5. UIMA の中で、カスタムまたはエンタープライズ・サーチに特定のタイプを使用する場合は、その定義方法について UIMA SDK 文書を参照してください。

ベース・アノテーター・パッケージをインストールすると、`UIMA_SDK_INSTALL/docs/examples/descriptors/analysis_engine` ディレクトリーの中にアノテーター・ディスクリプター・ファイルが含まれていることが分かります。ファイル `of_tokenization.xml` は、ベース・テキスト分析アノテーター (言語 ID アノテーター、FROST 辞書検索アノテーター、および空白トークナイザー) をそれがエンタープライズ・サーチの中で使用された順番でリストします。

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・システムの中のこれらのディスクリプター・ファイルは変更しないでください。これらのファイルを変更すると、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・パッケージには、英語の文書を処理するのに必要な辞書のみが含まれます。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. エンタープライズ・サーチ・インストールに含まれるエンタープライズ・サーチ辞書を見つけてみます。 `ES_INSTALL_ROOT/configurations/parserservice/jediidata/frost/resources` にあります。
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 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テキスト分析エンジン指定子ファイル `of_tokenization.xml` 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。

使用している開発環境の中で、カスタム・アノテーターよりも先にベース・テキスト分析アノテーターを実行し、カスタム・アノテーターがベース・テキスト分析で定義されたタイプを使用する場合は、カスタム・アノテーター指定子のタイプ・システム・セクションに、ファイル `tt_core_typesystem` への参照を含めてください。`tt_core_typesystem` ファイルは `UIMA_SDK_INSTALL/docs/examples/descriptors/analysis_engine` ディレクトリーにあります。ディスクリプター・ファイルへの参照を組み込む方法の例は、`analysis_engine` ディレクトリーにあるファイル `jtok.xml` を参照してください。

関連タスク

13 ページの『ベース・アノテーターとカスタム・テキスト分析結果の表示』

86 ページの『正規表現アノテーターを使用した簡単なセマンティック検索の使用可能化』

『UIMA 内でのデータベース・コンシューマーへの共通分析構造の使用』

12 ページの『UIMA 内での正規表現アノテーターの使用』

UIMA 内でのデータベース・コンシューマーへの共通分析構造の使用

UIMA の中でデータベース・コンシューマーへの共通分析構造を使用する前に、コンシューマー・ディスクリプター・ファイルを変更し、共通分析構造からデータベースへのマッピング・ファイルを作成する必要があります。

UIMA環境の中でデータベース・コンシューマーへの共通分析構造を実行する前に、次のことを行う必要があります。

1. `UIMA_SDK_INSTALL/docs/examples/descriptors/cas_consumer` 中の XML ディスクリプター・ファイル `cas2jdbc.xml` を開きます。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. パラメーター `mappingFile` を、共通分析構造からデータベースへのマッピング・ファイルがある場所への絶対パスが入るように変更します。例えば、`D:¥temp¥MyMapping.xml` です。
3. パラメーター `docMetadata_Type` を、そこから組み込みフィーチャーのすべてのメタデータをリトリブする UIMA タイプを指定するように変更します。例えば、`uima.tcas.DocumentAnnotation` です。
4. パラメーター `docId_Feature` を、そこから文書の数値 ID (整数型) をリトリブするメタデータ・タイプへのフィーチャーまたはフィーチャー・パスを含むよ

うに変更します。docId()、uniqueId()、objectId()、および fsId() など、ID が必要なすべての組み込みフィーチャーで、これが必要になります。

5. パラメーター **encryptionClass** は設定しません。これは、エンタープライズ・サーチの中でのみ使用され、データベース・コンシューマーへの共通分析構造が暗号化マッピング・ファイルを使用できるようにします。
6. ファイルを保存します。
7. エンタープライズ・サーチ・インストールの lib ディレクトリーから EMF ライブラリー・ファイル (common.jar、ecore.jar、および ecore.xmi.jar) を UIMA インストールの lib ディレクトリーにコピーします。cc_cas2jdbc.jar は、既に、UIMA インストールの lib ディレクトリーにあります。
8. データベースに保管するテキスト分析結果を定義する、共通分析構造からデータベースへのマッピング・ファイルを作成します。UIMA_SDK_INSTALL/docs/examples/descriptors/cas_consumer にあるマッピング・ファイル sampleMapping.xml を、独自のマッピング・ファイルを作成するときのサンプルとして使用できます。

UIMA_SDK_INSTALL/docs/examples/descriptors/cas_consumer にある、CasToJDBCMapping.xsd という XML スキーマ・ファイルを使用して、共通分析構造からデータベースへのマッピング・ファイルの妥当性検査を行います。パフォーマンス上の理由から、データベース・コンシューマーへの共通分析構造は、マッピング・ファイルの妥当性検査はしません。自分で検査する必要があります。

UIMA の中でコンシューマーを実行する方法については、UIMA 文書の中に説明があります。

次のサンプルは、ディスクリプターの中の必須パラメーターをどのように定義しなければならないかを示しています。

```
...
<nameValuePair>
  <name>mappingFile</name>
  <value>
    <string>D:/temp/MyMapping.xml</string>
  </value>
</nameValuePair>
<nameValuePair>
  <name>docMetadata_Type</name>
  <value>
    <string>uima.tcas.DocumentAnnotation</string>
  </value>
</nameValuePair>
<nameValuePair>
  <name>docId_Feature</name>
  <value>
    <string>end</string>
  </value>
</nameValuePair>
...
```

次の表は、ディスクリプター・ファイルの中で現れる順に構成パラメーターをリストしたもので、どれが必須であるかを示しています。

表 1. 共通分析構造からデータベース・コンシューマーへのディスクリプター・ファイル中の構成パラメーター

パラメーター	説明	必須
mappingFile	共通分析構造からデータベースへのマッピング・ファイルの絶対パス。例えば、 D:%temp%sample.xml です。 Windows® システムでは、パスの区切り記号として「¥」を使用します。	はい
encryptionClass	このパラメーターは設定しません。これはエンタープライズ・サーチの中でのみ使用され、データベース・コンシューマーへの共通分析構造が暗号化マッピング・ファイルを使用できるようにします。	いいえ
docMetadata_Type	組み込みフィーチャーのすべてのメタデータがリトリートされる UIMA タイプ。	はい
docId_Feature	文書の数値 ID がリトリートされるメタデータ・タイプのフィーチャーまたはフィーチャー・パス。これは、整数型でなければならず、uniqueId()、objectId()、および fsId() といった、ID を必要とするすべての組み込みフィーチャーに必要です。	はい
docUri_Feature	文書の URI が取り出されるメタデータ・タイプのフィーチャーまたはフィーチャー・パス。それはストリング型でなければなりません。	いいえ
IsCompleted_Feature	現在の文書が複数の共通分析構造に渡って取り出されたかどうかを示すメタデータ・タイプのフィーチャーまたはフィーチャー・パス。	いいえ
chunkNumber_Feature	現在のチャンクの後続の番号を示すメタデータ・タイプのフィーチャーまたはフィーチャー・パス。	いいえ

UIMA 内での正規表現アノテーターの使用

正規表現アノテーターを使用して、テキスト文書内のエンティティまたは情報の単位を検出します。検索の必要に合わせて、対象ドメインのアノテーターをカスタマイズできます。

電話番号、URL、および E メール・アドレスを検出するサンプルの正規表現アノテーターを実行するか、または、サンプルのアノテーターを UIMA 環境で正規表現アノテーターの独自のカスタマイズされたバージョンを作成するときのベースとして使用するには、次のことが必要です。

1. 正規表現アノテーター・ディスクリプターが `UIMA_SDK_INSTALL/docs/examples/descriptors/analysis_engine` ディレクトリーにあること。
2. サンプルの規則セットとタイプ・システム記述が `UIMA_SDK_INSTALL/docs/examples/regex` ディレクトリーにあること。
3. サンプルの規則セットが適用できるサンプル・テキスト・ファイルが `UIMA_SDK_INSTALL/docs/data` ディレクトリーの `of_sample_regex.txt` という名前のファイルにあること。

UIMA の中でアノテーターを実行する方法については、UIMA 文書の中に説明があります。

ベース・アノテーターとカスタム・テキスト分析結果の表示

解析によって生成された分析結果、およびエンタープライズ・サーチ内のアノテーターによって生成された分析結果を表示するには、共通分析構造に保管された分析結果の表示可能な XML バージョンを生成するように、文書コレクションのプロパティを更新する必要があります。

このタスクについて

共通分析構造に保管されたアノテーター分析結果の XML 直列化を使用して、次のことを行います。

- 解析のあとの結果 (ベース・アノテーターが処理される前) を表示します。
- 解析とトークン化 (エンタープライズ・サーチ・ベース・アノテーターの実行による) のあとの結果を表示します。これを基に、ベース・アノテーターのあとに常に実行することになる、これから作成するカスタム分析への入力データ構造を決めることができます。
- カスタム分析をコレクション全体に実行するかどうかを決定する前のテストとして、エンタープライズ・サーチ内の比較的小さな文書コレクションにカスタム分析を実行した結果を表示し検証します。

XML 直列化は、2 セットの結果を生成します。

- 解析のあとの結果。これにはフィールド・マッピングおよび文書メタデータが含まれます。
- 解析とトークン化のあとの結果。また、選択された場合は、カスタム・テキスト分析。これにはすべての生成されたトークンと注釈が含まれます。

手順

分析結果の表示可能な XML バージョンを生成するには、次のようにします。

1. コレクションの中の文書の解析を始める前に、`ES_NODE_ROOT/master_config/<CollectionID>.parserdriver` 中のファイル `collection.properties` を開きます。
2. 解析のあとの結果を表示するには、次の行を `collection.properties` ファイルに追加します。`trevi.parser.dumpXCas=<your_dump_directory>`

ダンプ・ディレクトリー (<your_dump_directory>) は既に存在していなければなりません。

- a. 希望する出力のタイプを選択します。出力には常に、解析結果のために使用される `OmniFindParserTypeSystem.xml` と呼ばれるタイプ・システム記述が含まれます。次の行の 1 つを追加します。

- 最後に処理された 25 ファイルの出力を表示するには、`trevi.parser.maxXCasFileCount=25` を追加します。

ファイルの数は自分で決めることができますが、この値をあまり高く設定しないようにしてください。

ファイル出力バッファは最大バッファ・サイズに到達したあと、絶えず上書きされることに注意してください。これはまた、最も大きい番号の文書が最後に処理された文書とは限らないことも意味します。

出力には、ファイル `OmniFindParserXCasDump1.xml` の後に `OmniFindParserXCasDump2.xml` が続くというように、25 個までのファイルがリストされて含まれます。

- 特定の文書の出力を表示するには、文書 URI `trevi.parser.xCasURI.1=file://home/test/file1.txt` を追加します。

追加する文書の数に制限はありませんが、文書には 1 から始まる昇順の連続した番号 (間に抜けない番号) を付ける必要があります。例えば、2 番目の文書は `trevi.parser.xCasURI.2=file://home/test/file2.txt` で、3 番目の文書は `trevi.parser.xCasURI.3=file://home/test/file3.txt` のようになります。

出力には、ファイル

`OmniFindParserXCasDumpURI_1.xml`、`OmniFindParserXCasDumpURI_2.xml` というように、リストされたファイル名までのファイルが含まれます。

3. トークン化のあとの結果を表示するには、次の行を追加します。

```
trevi.tokenizer.dumpXCas=<your_dump_directory>
```

また、ダンプ・ディレクトリーは既に存在していなければなりません。

- a. 希望する出力のタイプを選択します。また、作成された出力には、常に、トークン化とテキスト分析結果に使用された、`OmniFindTypeSystem.xml` というタイプ・システム記述が含まれます。次の行の 1 つを追加します。

- 最後に処理された 25 ファイルの出力を表示するには、`trevi.tokenizer.maxXCasFileCount=25` を追加します。

ファイルの数は自分で決めることができますが、この値をあまり高く設定しないようにしてください。

ファイル出力バッファは最大バッファ・サイズに到達したあと、絶えず上書きされることに注意してください。これはまた、最も大きい番号の文書が最後に処理された文書とは限らないことも意味します。

出力には、ファイル `OmniFindXCasDump1.xml` の後に `OmniFindXCasDump2.xml` が続くというように、25 個までのファイルがリストされて含まれます。

- 特定の文書の出力を表示するには、文書 URI `trevi.tokenizer.xCasURI.1=file://home/test/file1.txt` を追加します。

追加する文書の数に制限はありませんが、文書には 1 から始まる昇順の連続した番号 (間に抜けない番号) を付ける必要があります。例えば、2 番目の文書は `trevi.tokenizer.xCasURI.2=file://home/test/file2.txt` で、3 番目の文書は `trevi.tokenizer.xCasURI.3=file://home/test/file3.txt` のようになります。

出力には、ファイル `OmniFindXCasDumpURI_1.xml`、`OmniFindXCasDumpURI_2.xml` というように、リストされたファイル名までのファイルが含まれます。

エンタープライズ・サーチでは、XCAS Annotation Viewer を使用して、XML ファイルの内容を表示できます。ES_INSTALL_ROOT/bin ディレクトリーにある `xcasAnnotationViewer` スクリプト・ファイルを実行して XCAS Annotation Viewer を開始します。次の入力を求めるプロンプトが出されます。

- 解析とトークン化のあとの結果を入れるダンプ・ディレクトリー
- ディスクリプター・ファイル。同様にダンプ・ディレクトリーに入る `OmniFindParserTypeSystem.xml` (パーサーの結果) または `OmniFindTypeSystem.xml` (トークン化と分析の結果) のどちらか。

リストから文書を選択すると、文書の分析結果が表示されます。文書内の強調表示された注釈をクリックすると、注釈の詳細が表示されます。

タイプ・システム記述

タイプ・システムは、共通分析構造の中でインスタンス化される可能性のあるオブジェクトのタイプと、タイプのプロパティー (またはフィーチャー) を定義します。

各分析エンジンには、分析エンジンの中のアノテーターの入力要件と出力タイプを示す、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、アプリケーション・ドメインに固有です。

タイプ・システムには、タイプの定義、タイプのプロパティー、およびタイプの単一継承階層が含まれます。共通分析構造は特定のタイプ・システムに適合する必要があります。

タイプ・システム記述で定義されるタイプとフィーチャーは、XML エレメントから共通分析構造へのマッピング・ファイル、共通分析構造から索引へのマッピング・ファイル、および共通分析構造からデータベースへのマッピング・ファイルを含む、文書分析に関連付けられたすべてのマッピング・ファイルの中で使用される必要があります。

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。

UIMA 環境で、分析エンジンの開発とテストを完了すると、作成してエンタープライズ・サーチにアップロードするアーカイブ・ファイル (.pear ファイル) には、分析ロジック・ファイルとタイプ・システム記述が入っています。

エンタープライズ・サーチ・ベース・アナテーターは、3 個のタイプ・システム記述を使用します。1 つはコア・タイプ・システム記述で、これは常に含まれます。他の 2 つは文書コレクションの基本分析処理を拡張分析モードに変更するために、オプションで活動化できます。拡張タイプ・システム記述の一方または両方を含める必要があるかどうかは、基本分析処理の間に含める追加のテキスト分析処理結果によって異なります。

拡張タイプ・システムの一方または両方を含めることによって、拡張分析モードを使用可能にすることができます。拡張分析モードでは、基本分析処理の間に追加の分析フィーチャーが使用可能にされ、その分析フィーチャーは共通分析構造に保存されます。例えば、トークンのすべての可能な見出し語といったトークンについてさらに情報 (詳細なフィーチャー情報) が必要である場合、または、見出し語がストップワードであるか、見出し語の品詞であるか、形態処理の特別なフィーチャーである場合、そして日本語の場合は、拡張分析モードを活動化する必要があります。

関連タスク

『基本分析モードから拡張分析モードへの変更』

関連資料

17 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』

基本分析モードから拡張分析モードへの変更

エンタープライズ・サーチ・ベース・アナテーターによって実行される文書コレクション処理を基本分析モードから拡張分析モードに変更するには、拡張分析モードのタイプ・システム記述を組み込む必要があります。

制約事項

拡張分析モードを活動化するために選択できるタイプ・システム記述が 2 つあります。

- `tt_extension_typesystem` 記述。これは見出し語について詳細な字句タイプのフィーチャー情報を含みます。
- `dlt_extension_typesystem` 記述。これは、追加の形態フィーチャーと特別な字句タイプを含みます。

手順

基本コレクション処理を拡張分析モードに変更するには、次のようにします。

1. `ES_NODE_ROOT/master_config/CollectionID.parserdriver/specifiers` ディレクトリーにあるファイル `tt_core_typesystem.xml` を開きます。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. `<imports>` セクションの中の `<import>` エレメントを囲むコメント・タグを除去して、一方または両方の拡張タイプ・システム記述ファイルを含めます。


```

<imports>
<!-- imports the tt_extension_ttypesystem for advanced analysis -->
<!-- <import location="tt_extension_typesystem.xml"/>-->
<!-- imports the dlt extension typesystem -->
<!-- <import location="dlt_extension_typesystem.xml"/> -->
</imports>

```

- 2 つのディスクリプター・ファイル `jfrost.xml` と `jfrost_ngram.xml` を開き、`<capabilities>` セクションの `<description>` エlement にリストされた中で、分析の間に含めたいタイプ (`<type>` Element の中に) とフィーチャー (`<feature>` Element の中に) を含めるように `<outputs>` Element の内容を変更します。変更を保存します。
- ディスクリプター・ファイル `jtok.xml` を開き、`<capabilities>` セクションの `<description>` Element にリストされた中で、分析の間に含めたいフィーチャーを (`<feature>` Element の中に) 含めるように `<outputs>` Element の内容を変更します。変更を保存します。
- ディスクリプター・ファイル `es_tok_no_stw.xml` を開き、ここでもまた、`<capabilities>` セクションの `<description>` Element にリストされた中で、分析の間に含めたいフィーチャーを (`<feature>` Element の中に) 含めるように `<outputs>` Element の内容を変更します。変更を保存します。
- 拡張分析モードに変更したときに、文書コレクションを再度解析する必要があります。

関連概念

15 ページの『タイプ・システム記述』

関連資料

『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。

エンタープライズ・サーチで使用されるタイプは、3 個の別々のタイプ・システム記述ファイルの中に定義されます。その内の 1 つは、すべての基本言語分析に常に必要とされるコア・タイプを含むタイプ・システム記述ファイルで、他に、通常は拡張分析モードでのみ必要とされる拡張言語フィーチャーを定義するタイプ・システム記述があります。

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。文書の基本的な分析のときに `tt_core_typesystem` 記述が使用され、そのあとのカスタム分析の中で使用できる次の情報が、共通分析構造に追加されます。

- タイプ `com.ibm.es.tt.DocumentMetaData` の文書メタデータ。
- タイプ `uima.tt.SentenceAnnotation`、および `uima.tt.ParagraphAnnotation` の、センテンスおよびパラグラフ注釈といった文書構造情報。
- タイプ `uima.tt.TokenAnnotation` の、トークンおよび複合といった字句注釈。

`tt_core_typesystem` 記述は、ほとんどのテキスト分析処理に適しています。

コレクション処理を拡張分析モードに変更したければ、次の 2 つのタイプ・システムを含めることができます。タイプ・システムは基本的な言語処理の間に作成されない追加のフィーチャーを主として含みます。

- `tt_extension_typesystem`。これは、トークン、見出し語、パラグラフおよびセンテンスのさらに詳細なフィーチャー情報を含みます。
- `dlt_core_typesystem`。これは、いくつかの IBM LanguageWare 拡張注釈タイプ、例えば URL とアドレスを含みます。これには、頻繁には使用されない形態フィーチャーも含まれます。

tt_core_typesystem

次のタイプとフィーチャーが `tt_core_typesystem` 記述の中で定義されています。

uima.tcas.DocumentAnnotation

文書メタデータが含まれる文書注釈で、次のフィーチャーを持ちます。

- `categories` にはテキスト・カテゴライザーによって追加される文書カテゴリーが入ります。追加された各カテゴリーは、タイプ `com.tt.CategoryConfidencePair` です。
- `languageCandidates` には解析時に自動的に検出された文書言語が入ります。言語はタイプ `com.tt.LanguageConfidencePair` のリストに追加され、最も可能性の高い言語が先頭にリストされます。
- `id` には、例えば URL などの文書 ID が入ります。

uima.tt.TTAnnotation

これは、`tt_core_typesystem` に定義された注釈のルート・タイプです。このスーパータイプは `uima.tcase.Annotation` です。以下のタイプがあります。

uima.tt.DocStructureAnnotation

文書構造に関する注釈です。次のサブタイプがあります。

uima.tt.SentenceAnnotation

センテンス

uima.tt.ParagraphAnnotation

文書パラグラフ

uima.tt.LexicalAnnotation

トークンまたは複数語表現といった字句注釈です。次のサブタイプがあります。

uima.tt.TokenLikeAnnotation

単一トークン注釈で、次のフィーチャーを持つことができます。

- `tokenProperties` にはトークン・プロパティが入ります。
- `lemma` には見出し語または用語の語幹が入ります。
- `normalizedCoveredText` にはカバーされたテキストの正規化表現が入ります。

この注釈タイプは次のサブタイプを持ちます。

uima.tt.TokenAnnotation

複合部分と区別される実際のトークン。

uima.tt.CompPartAnnotation

用語の複合部分。

uima.tt.CompoundAnnotation

複合トークンの注釈。複合トークンは、通常、複数のトークン注釈をスパンします。

uima.tt.MultiTokenAnnotation

複数のトークンから成る字句注釈。この注釈タイプは次のサブタイプを持ちます。

uima.tt.StopwordAnnotation

ストップワードの注釈。ストップワードは複数語から成るワードであることもできます。

uima.tt.SynonymAnnotation

同義語が存在する用語の注釈。これは、用語の検出された同義語をリストするフィーチャー `synonyms` を持ちます。

uima.tt.SpellCorrectionAnnotation

スペル訂正が存在する用語の注釈。これは、最も可能性の高い順にソートされた修正をリストする、フィーチャー `correctionTerms` を持ちます。

uima.tt.MultiWordAnnotation

複合語の注釈。

uima.CAS.TOP

タイプ・システムのルート。次のサブタイプがあります。

uima.tt.KeyStringEntry

String データ構造の抽象タイプ。ストリング・キーと次のサブタイプを持つ、フィーチャー `key` を含みます。

uima.tt.Lemma

ディクショナリー見出し語エントリー。

uima.tt.CategoryConfidencePair

検出されたカテゴリーの信頼値。次のフィーチャーを持ちます。

- `categoryString` にはカテゴリーの名前が入ります。
- `categoryConfidence` にはカテゴリーの信頼値が入ります。
- `mostSpecific` には、このカテゴリーがその文書を最も特定するかどうかを示すフラグが入ります。
- `taxonomy` には、そこからカテゴリーが引き出された分類の名前が入ります。

uima.tt.LanguageConfidencePair

検出されたカテゴリーの信頼値。このタイプは、フィーチャー `languageConfidence`、`language`、および `languageID` を含みます。

tt_extension_typesystem

tt_extension_typesystem は、さらに拡張された処理のための追加のテキスト分析フィーチャーを含みます。

uima.tt.TokenLikeAnnotation

tt_extension_typesystem の中のこの注釈タイプは、次のフィーチャーを持ちます。

- lemmaEntries は、トークンのすべての可能な見出し語をリストします。リスト項目はタイプ uima.tt.Lemma です。
- tokenNumber
- stopwordToken

uima.tt.Lemma

タイプ uima.tt.KeyStringEntry のこの注釈は、次のフィーチャーを持ちます。

- isStopword は、見出し語がストップワードであれば true です。
- isDeterminer は、見出し語が限定詞であれば true です。
- partOfSpeech。次の品詞番号記述コードがあります。
 - 0: 不明
 - 1: 代名詞
 - 2: 動詞
 - 3: 名詞
 - 4: 形容詞
 - 5: 副詞
 - 6: 接置詞
 - 7: 間投詞
 - 8: 接続詞

uima.tt.DocStructureAnnotation

文書構造に関する注釈です。これは次のサブタイプを持ちます。

uima.tt.SentenceAnnotation

文書センテンス。フィーチャー sentenceNumber を持ちます。

uima.tt.ParagraphAnnotation

文書パラグラフ。フィーチャー paragraphNumber を持ちます。

dlt_extension_typesystem

dlt_extension_typesystem は、IBM LanguageWare が使用する追加のフィーチャーを含みます。

uima.tt.LexicalAnnotation

この注釈は次のサブタイプを持ちます。

uima.tt.TokenLikeAnnotation

dlt_extension_typesystem の中で、この注釈は次のフィーチャーを持ちます。

- synonymEntries

- frost_TokenType
- inflectedForms
- spellAid
- decomposition

com.ibm.dlt.uimatypes.FilePath

com.ibm.dlt.uimatypes.Email

com.ibm.dlt.uimatypes.Number

com.ibm.dlt.uimatypes.URL

com.ibm.dlt.uimatypes.Date

com.ibm.dlt.uimatypes.Time

com.ibm.dlt.uimatypes.Tel

com.ibm.dlt.uimatypes.Currency

com.ibm.dlt.uimatypes.Acronym

uima.tt.TokenLikeAnnotation

dlt_extension_typesystem の中のこの注釈タイプは、次のタイプを持ちます。

com.ibm.dlt.uimatypes.MWU

このタイプは IBM LanguageWare が使用して複数語表現の注釈を付けます。

uima.tt.KeyStringEntry

String 注釈。これは次のサブタイプを持ちます。

uima.tt.Lemma

次のフィーチャーを持ちます。

- frost_Constraints には制約フラグが入ります。
- frost_MorphBitMasks には形態ビット・マスク配列が含まれません。
- frost_ExtendedPOS には、例えば日本語の JPOS、中国語の CPOS などの、拡張品詞情報が入ります。
- frost_JKom には日本語の形態データが含まれます。
- frost_JPStart には日本語の開始分析データが含まれます。
- morphID には見出し語プロパティが含まれます。

uima.tcas.Annotation

これは次のサブタイプを持ちます。

com.ibm.dlt.uimatypes.Decomp_Analysis

複合の完全構造分析。これは次のフィーチャーを持ちます。

- headComponentIndex には複合の先頭コンポーネントが入ります。
- route には、単一分解ルートを構成するトークンのリストが含まれます。

関連資料

25 ページの『タイプ・システム記述のサンプル』

エンタープライズ・サーチに固有なタイプとフィーチャー

of_typesystem 記述に定義されているタイプとフィーチャーは、OmniFind Enterprise Edition に固有なタイプをカバーします。これらのタイプは、文書固有のメタデータに使用されます。また、これらのタイプは、フィールドおよび XML マークアップ情報または HTML アンカーの表記を説明します。

of_typesystem 記述は、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、これらのタイプのいずれかを使用する場合、分析エンジンのタイプ・システム記述に、これらのタイプを再定義する必要があります。例えば、文書セキュリティ情報にアクセスしたり、クローラー・タイプまたは文書タイプにアクセスする場合などです。

以下のタイプとフィーチャーは、of_typesystem 記述の中で定義されています。

uima.tcas.DocumentAnnotation

標準 UIMA 文書アノテーションは、以下のフィーチャーによって拡張されます。

esDocumentMetaData

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

com.ibm.es.tt.DocumentMetaData

文書メタデータ・タイプには、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。フィーチャー値は、`uima.cas.String` タイプです。

- **CM**。DB2 Content Manager クローラーによってクロールされる文書に使用します。
- **Database**。JDBC データベース クローラーによってクロールされる文書に使用します。
- **DB2**。DB2 クローラーによってクロールされる文書に使用します。
- **DominoDoc**。Domino Document Manager クローラーによってクロールされる文書に使用します。
- **Exchange**。Exchange Server クローラーによってクロールされる文書に使用します。
- **NNTP**。NNTP クローラーによってクロールされる文書に使用します。
- **Notes**。Notes クローラーによってクロールされる文書に使用します。
- **QuickPlace**。QuickPlace クローラーによってクロールされる文書に使用します。

- **Seedlist**。シード・リスト クローラーによってクロールされる文書に使用します。
- **UnixFS**。UNIX ファイル・システム クローラーによってクロールされる文書に使用します。
- **VBR**。Content Edition クローラーによってクロールされる文書に使用します。
- **WCM**。Web Content Management クローラーによってクロールされる文書に使用します。
- **Web**。Web クローラーによってクロールされる文書に使用します。
- **WinFS**。Windows ファイル・システム クローラーによってクロールされる文書に使用します。
- **WP**。WebSphere Portal クローラーによってクロールされる文書に使用します。

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。フィーチャー値は、`uima.cas.String` タイプです。

- `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)

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

フィールドがフリー・テキストでの検索が可能であることを示すフラグ。

fieldSearchable

フィールドがフィールドとして検索可能であることを示すフラグ。

parametric

フィールドがパラメトリック照会で検索可能であることを示すフラグ。

showInSearchResult

注釈付きのデータが検索結果詳細に含まれていることを示すフラグ。

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できます。フィーチャー値は、`uima.cas.String` タイプです。

sortable

フィールドがストリングでソート可能であることを示すフラグ。

exactMatch

検索が照会条件に完全一致している必要があることを示すフラグ。

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.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。

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。

タイプ・システム記述のサンプル

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）に入っていないとなりません。

次のタイプ・システム記述のサンプルは、被疑者、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。

このサンプルのタイプ・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さまざまなタイプについて説明するときのすべてのテキスト分析トピックで使用されます。

```
<?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>eng1Date</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>

```

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で、文書に XML タグおよび属性が現れる場合、すべての XML タグおよび属性を使用する場合はこのオプションを使用します。例えば、<addressee> エレメントを含む請求書の文書に対して、ネイティブ XML 検索を使用可能にすると、このエレメント内で特定のカスタマー名を検索するとき、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 エレメントから共通分析構造へのマッピング

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明確で、そのあとのテキスト分析ステップで使用できる場合。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> には、たいてい顧客名が含まれています。XML エレメントから共通分析構造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。

- カスタム・アノテーターの処理の有効範囲を、XML 入力の指定した領域に制限したい場合があります。例えば、<technicianComment> タグの内容の分析を、車の故障を検出するアノテーターの中のみタグに制限するなどです。
- 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。
- 名前の異なる XML タグをセマンティック検索で使用される共通スパンにマップしたいとします。例えば、<mainHeading> または <doc> をタイトルにマップするなどです。

これらのケースでは、XML エlementをどのフィーチャー構造にマップするかを定義する、XML エlementから共通分析構造へのマッピング・ファイルを作成する必要があります。マッピング・ファイルの中で定義するフィーチャー構造は、文書が解析され、カスタム・アノテーターによってアクセスされたときに作成されます。

文書のコレクションに対して、XML エlementから共通分析構造へのマッピング・ファイルを複数使用できます。どのマッピング・ファイルをどの XML 文書に使用するかは、<identifier> エlementによって決定されます。マッピング・ファイル内の <identifier> エlementは、XML 文書内のルート・Element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・Elementが doc である場合、マッピング・ファイル内の <identifier> エlement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、<identifier> エlementが Default に設定されているマッピング・ファイルを検索します。デフォルトのマッピングが見つからない場合、文書の本文セクション (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lement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、コンテンツの抽出と呼ばれます。例えば、title および body Element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。

コンテンツの抽出は、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)。コンテンツの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 検索と、XML エlementから共通分析構造へマッピングでのコンテンツの抽出オプションは、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の違いがあるために、同時に使用することはできません。コンテンツの抽出を指定すると、ネイティブ XML マッピングは無視されま

す。コンテンツの抽出をしない場合は、XML エlementから共通分析構造へのマッピングとネイティブ XML 検索の両方を使用できます。

構成ファイルで使用するすべてのタイプとフィーチャーは、カスタム分析ステップのタイプ・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 構文を知らなくても、ディスクリプター・ファイルを作成できます。

カスタム分析の作成とテストを完了したあと、UIMA PEAR (Processing Engine ARchive、処理エンジン・アーカイブ) 生成ウィザードを使用して、タイプ・システム記述が含まれるカスタム分析ファイルが入ったアーカイブを作成します。そのあと、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、XML エlementから共通分析構造へのマッピング・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

『XML エlementから共通分析構造へのマッピング・ファイルの作成』

XML エlementから共通分析構造へのマッピング・ファイルの作成

XML から共通分析構造へのマッピング・ファイルで、XML を UIMA データ・タイプにマッピングするための広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 から共通分析構造へのマッピング・ファイルを次の例に示します。

サンプルの警察レポートには、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、警察官の所属警察管区、被疑者の説明、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。次に例を示します。

```
<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、XML から共通分析構造へのマッピング・ファイルは次の構造を持ちます。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

マッピング・ファイルは次の 2 つのセクションに分かれています。

<contentElements> エレメント

特定コンテンツの抽出が必要な場合に、このエレメントを使用します。サンプルのマッピング・ファイルは、文書の <doc> セクションのコンテンツを抽出し、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> をコンテンツ・エレメントとして指定することで、テキスト処理が始まる前にイメージはフィルタリングされ除外されます。

<elementToTypeMappings>

このエレメントを使用して、文書内の個々の XML エレメント (<elementToTypeMapping> エレメントで指定) 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します。

コンテンツの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 から共通分析構造へのマッピング・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。マッピング・ファイルの XSD スキーマは XMLCasInitSchema.xsd と呼ばれ、エンタープライズ・サーチのインストールの `ES_INSTALL_ROOT/packages/uima/configuration_xsd/` 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. 文書内のセクションから特定のコンテンツを抽出する場合は <contentElements> エレメントを追加し、文書内の個々の XML エレメントを共通分析領域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。
4. <identifier> エレメントと <description> エレメントを追加します。ID によって、どの XML 文書にどのマッピングを使用するかが決まります。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・エレメントは無関係で、マッピングはどの XML 文書にも適用されます。
5. 文書の該当する部分にのみ含まれている情報を抽出する場合、<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。
 - 1 つ以上の <element> エレメント。これには、文書内の XML エレメントのパスが含まれ、XPath 構文に従います。例えば、<element>/doc/crimeType</element> です。

6. 文書内の XML エlement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、<elementToTypeMappings> Elementを追加します。これには、以下のコンポーネント・Elementがあります。
 - 1 つ以上の <elementToTypeMapping> Element。このElementには、以下のネストされたElementが含まれている必要があります。
 - <element> Element。これは、XML Element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。例えば、ルート・Element 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。例えば、birthDate は reportingOfficer 内になければなりません。他のElementはこれら 2 つの間にあればかまいません。
 - <type> Element。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。
 - ゼロ個以上の <featureValueAssignment> Element。
7. <featureValueAssignment> Elementの中で、<feature> Elementに String タイプのフィーチャーの名前を指定し、<basicValue> Elementに値を割り当てます。複数の <basicValue> Elementを <values> Elementの間に追加できます。

<basicValue> Elementは属性を持つことができます。それには、useAttributeValue、useElementContent、default、および trim が含まれません。

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 <reportingOfficer> XML タグ内のどこかに現れるそれぞれの <doc> XML タグに、タイプ com.ibm.omnifind.types.Person のフィーチャー構造が作成されます。
- <reportingOfficer> タグに sex 属性が含まれていれば、新規に作成されたフィーチャー構造のフィーチャー gender には、その属性の値が設定されます。

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。次のマッピング・スニペットを例にして説明します。

```
<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。

次の場合には、<values> エレメントの間に複数の値を指定できます。

- 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>

```

String フィーチャー値は、マッピング・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。例えば、
 <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、属性に条件を設定します。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・ファイルに基づいて、次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"

マッピング・ファイルを作成したあと、エンタープライズ・サーチ管理コンソールを使用して、これをエンタープライズ・サーチにアップロードし、他のカスタム分析の選択に加えて、XML から共通分析構造へのマッピング・ファイルを選択する必要があります。

関連概念

28 ページの『分析および検索における XML マークアップ』

関連資料

25 ページの『タイプ・システム記述のサンプル』

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、共通分析構造からのみ読み取ります。CAS コンシューマーが、共通分析構造に保管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、共通分析構造から索引へのマッピング・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また、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した、共通分析構造からデータベースへのマッピング・ファイルも必要とします。

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。コンシューマーをエンタープライズ・サーチにアップロードする方法と、エンタープライズ・サーチの中での使用方法については、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

41 ページの『カスタム分析結果の索引マッピング』

48 ページの『選択した分析結果のデータベース・マッピング』

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lement 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 (例えば、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。例えば、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を持ちます。

- 大括弧 ([<番号>]) を使用して、配列またはリストの特定の要素を選択します。配列はゼロ (0) から開始します。例えば、`companies` 配列内の最初の要素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。
- すべての要素を表示するには、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。例えば、被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` は被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、配列要素は連結 (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・パスの次の要素には、タイプを指定しなければなりません。タイプ名は、配列内の要素のタイプです。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。配列の要素は `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[2]/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、例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、最後または単一要素として、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 (例えば、`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>
    </featureMapping>
  </featureMappings>
</explicitMappingRule>
```



```

        <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、被疑者の名前を 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・エンタリー (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タリーを意味します。次の例は、共通分析構造からデータベースへのマッピング・ファイルからのもので、組み込み関数 [:index] も許可されています。

```

<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は、共通分析構造からデータベースへのマッピング・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 (";"), およびコロンの (":") 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。例えば、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</relation>
  <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・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

59 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』

フィルター

フィルターは、共通分析構造から索引へのマッピング・ファイルと共通分析構造からデータベースへのマッピング・ファイルの中のマッピング規則を制限するために使用されます。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ために使用します。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、空白スペースで区切る必要があります。

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 < 42 </filter>`

フィーチャー `foo` は 42 より小さい整数値を持ちます。

- `<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 は、0.5 以上の浮動小数点値を持ちます。

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。

共通分析構造から索引へのマッピング・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決定できます。

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 2. 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	<code><person/></code>
個人 (person) 注釈に上司 (boss) が含まれている文書を検索する	<code><person>boss</person></code>
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	<code><sentence><person>Lang</person> <competitor/></sentence></code>

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。

共通分析構造から索引へのマッピング・ファイルを作成したあと、それを、管理コンソールを使用してエンタープライズ・サーチにアップロードできます。

関連タスク

『共通分析構造から索引へのマッピング・ファイルの作成』

共通分析構造から索引へのマッピング・ファイルの作成

共通分析構造から索引へのマッピング・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決定できます。

このタスクについて

共通分析構造から索引へのマッピング・ファイルは、XML で記述されます。サンプルの共通分析構造から索引へのマッピング・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>
        </attributemappings>
      </style>
    </indexRule>
  </indexBuildItem>
</indexBuildSpecification>
```

```

        </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>
    </indexRule>

```

```

        <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

共通分析構造から索引へのマッピング・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

共通分析構造から索引へのマッピング・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。マッピング・ファイルの XSD スキーマは CasToIndexMapping.xsd と呼ばれ、エンタープライズ・サーチのインストールの `ES_INSTALL_ROOT/packages/uima/configuration_xsd/` 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には、`<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構造から、索引内の構造へのマッピングを含んだ `<indexBuildItem>` エレメントを 1 つ以上追加します。
5. XML ファイルを保存して妥当性検査を行います。

<indexBuildItem> エレメント

共通分析構造から索引へのマッピング・ファイルは、1 つ以上の `<indexBuildItem>` エレメントを含みます。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せん。

<name> エレメントは、フィーチャー構造タイプを含みます。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。例えば、`com.ibm.omnifind.types.Suspect` です。
- ワイルドカード。例えば、`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、処理は次のようになります。

- 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。被疑者 (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ために使用します。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、ブレイク・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、索引内で同じ名前アクセスできるようにしたい場合は、`fixedName` を使用します。次の例で、`com.ibm.omnifind.types.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。次の例で、`com.ibm.tt.EntityAnnotation` は、`type` というフィーチャーの値に応じて、`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lement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。次に例を示します。

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

この `<indexBuildItem>` エlement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lement

`<style>` エlement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実験します。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 ファイルを作成したあと、エンタープライズ・サーチ管理コンソールを使用して、これをエンタープライズ・サーチにアップロードし、他のカスタム分析の選択に加えて、共通分析構造から索引へのマッピング・ファイルを選択する必要があります。

関連概念

41 ページの『カスタム分析結果の索引マッピング』

36 ページの『フィーチャー・パス』

関連資料

40 ページの『フィルター』

25 ページの『タイプ・システム記述のサンプル』

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・サーチで文書を分析したあと、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) 以降、および Oracle 10g (oracle.jdbc.driver.OracleDriver Version 1.0) 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分に直接アクセスできるようになります。

共通分析構造からデータベースへのマッピング・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。マッピング・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。

共通分析構造からデータベースへのマッピング・ファイルを作成したあと、そのファイルを、管理コンソールを使用してエンタープライズ・サーチにアップロードできます。

関連タスク

50 ページの『共通分析構造からデータベースへのマッピング・ファイルの作成』

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、どの分析結果をデータベースに保管するかを定義した、共通分析構造からデータベースへのマッピング・ファイルを作成しなければなりません。また、必要な JDBC ドライバー・ライブラリーが、マッピング・ファイルの中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。
2. XML エディターで、データベース構成データと保管したい分析結果を記述した、共通分析構造からデータベースへのマッピング・ファイルを作成します。マッピング・ファイルにどの分析結果を含めるかを決めるには、文書が処理されるときに使用された、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・システムからアクセス可能なインデクサー・ノード上のディレクトリーに置きます。
4. エンタープライズ・サーチ管理コンソールを使用して、マッピング・ファイルをアップロードし、選択します。

ロード・ファイル・セットの使用

分析結果は、JDBC 対応データベースに直接保管するか、あるいは、ロード・ファイル・セットを使用し、あとの段階でデータをデータベースにロードするように処理を構成することもできます。

ロード・ファイル・セットを使用すると次の利点があります。

- 全体で、ロード・ファイルのセットは、オペレーティング・システムでサポートされる最大ファイル・サイズよりも大きくなることはありません。
- ロード・ファイル・セットがいっぱいになると同時にデータをデータベースにロードし始めることができ、ファイル・アクセスの衝突を避けるために文書パーサーを停止して再始動する必要はありません。

たとえば文書が複数の共通分析構造に分けられたとしても、1 つのロード・ファイル・セットから次のセットへの切り替えは文書レベルで行われます。文書が処理されたあと、現在のロード・ファイル・セット内の 1 つのロード・ファイルが定義された限界を超えると、新しいロード・ファイル・セットが使用されます。これにより、ロード・ファイル・セットの整合性が保証されます。1 つのロード・ファイル・セットのコンテンツがデータベースにロードされたあと、マスター表内のすべての項目は一致する項目がデータベース表内に含まれるため、データ・モデルの整合性は保たれます。

ロード・ファイルとスクリプト・ファイルは、ファイル拡張子 .cur によって特定されます。ロード・ファイル・セットが閉じられると、ファイルは、拡張子 .dat に名前変更されます。これは、文書パーサーがまだ実行中であっても、ファイルをデータベース・サーバーにコピーまたは移動できることを示しています。

ロード・ファイルのサイズを指定できます。ロード・ファイル・サイズの限界に達すると、新規ロード・ファイル・セットが開始します。ロード・ファイル・サイズは、共通分析構造からデータベースへのマッピング・ファイルの <loadFile> XML エlement・セクションに指定できます。パラメーター loadFileSize は、<loadFileSize> エlementを使用して定義でき、 $10 \leq \text{loadFileSize} \leq 10240$ (10MB $\leq \text{loadFileSize} \leq 10\text{GB}$) の M バイトで指定します。<loadFileSize> エlementはオプションです。値が設定されなければ、デフォルト値は 1024 MB (1 GB) です。

セット内の各ロード・ファイルは 10 桁の数字で番号付けされ、これにより、ファイルがどのロード・ファイル・セットに属するかが特定されます。ロード・ファイル・セットは次の場合に閉じます。

- セット内の 1 つのロード・ファイルが定義されたサイズの限界を超えたとき
- パーサーが停止したかエラーが発生したために処理が停止したとき

パーサーが再始動すると、処理は停止した地点から新規のロード・ファイル・セットを使用して継続されます。

重要: Cas2Jdbc を使用してロード・ファイルを生成する場合、構成されているのが 1 つのパーサー・スレッドのみであることを確認してください。Cas2Jdbc ロード・ファイルを生成するのに構成されているコレクションに対し、複数のパーサー・スレッドを使用すると、無効なロード・ファイルができる可能性があります。使用するパーサー・スレッドの数を指定するには、エンタープライズ・サーチの管理コンソールを使用して、コレクションを編集します。「解析」ページを選択して、解析オプションを構成するオプションを選択してから、パーサー・スレッドの数を 1 に設定します。

共通分析構造からデータベースへのマッピング・ファイルの作成

分析結果をデータベースに追加するには、データベース接続構成情報と、どのカスタム・テキスト分析結果がどのデータベース表と列に保管されるかの記述を含んだ、共通分析構造からデータベースへのマッピング・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

共通分析構造からデータベースへのマッピング・ファイルは、XML で記述されます。次のサンプルは、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。例には、組み込みフィーチャーと <constant> エlement・マッピングの使用が示されています。

```
<?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>
  <loadFileSize>1048</loadFileSize>
  <loadScript>/home/cas2jdbc/load/load.sh</loadScript>
</loadFile>

</databaseConnection>

<cas2JdbcMappingSpec>
  <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>
</cas2JdbcMappingSpec>

```



```
</implicitMappings>
</cas2JdbcMappings>
</cas2JdbcMappingSpec>
</cas2JdbcConfiguration>
```

手順

共通分析構造からデータベースへのマッピング・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。マッピング・ファイルの XSD スキーマは CasToJDBCMapping.xsd と呼ばれ、エンタープライズ・サーチのインストールの `ES_INSTALL_ROOT/packages/uima/configuration_xsd/` 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を含む `<databaseConnection>` エレメントと、データベースまたはロード・ファイルに保管される、分析結果のマッピング規則を記述した `<cas2JdbcMappingSpec>` 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 (例えば、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します。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。

ドライバー・ライブラリーは、必ず DB2 アプレット・サーバーと常に同じ保守レベルにしてください。

- 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、次のコンポーネント・エレメントが含まれます。
 - `<loadFileDirectory>` エレメント。ロード・ファイル・ディレクトリーが入ります。

- オプション: <loadFileSize> エlement。ロード・ファイル・サイズが入ります。ロード・ファイル・サイズの限度は $10 \leq \text{loadFileSize} \leq 10240$ (10MB $\leq \text{loadFileSize} \leq 10\text{GB}$) です。値が定義されなければ、デフォルトは 1024 MB (1GB) です。

- <loadScript> Element。ロード・スクリプト名が入ります。

<loadFile> Elementを指定しなければ、すべてのデータは、JDBC を使用して、直接、データベースに保管されます。

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・Elementを <jdbcMappingSpec> Elementに追加します。

- オプション: <skipCondition> Element。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。

- <cas2JdbcMappings> Element。どのタイプとフィーチャーが、どのデータベース表と列にマップされるかを示します。このElement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> Elementを追加します。このElement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> Elementが必要で、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> Elementを追加します。このElement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Elementがある場合、少なくとも 1 つの <implicitMappingRule> Element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。例えば、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。<explicitMappingRule> および <implicitMappingRule> エレメントに、次のコンポーネント・エレメントを追加します。
- <type> エレメント。フィーチャー構造タイプを含みます。
 - <table> エレメント。データベース・スキーマおよび表名を含みます。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。
 - <featureMappings> エレメント。1 つ以上の <featureMapping> エレメント、または 1 つの <containerMapping> エレメントを含みます。
 - オプション: <filter> エレメント。マッピング規則が一致するたびに評価される条件を含みます。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のどちらかをマッピングしているかによって異なります。

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。

- <feature> エレメント。フィーチャーの名前を含みます。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、使用できます。
- オプション: <length> エレメント。指定したデータベース列に入れることができるストリングの長さを含みます。長いストリングは切り捨てられます。
- <column> エレメント。フィーチャー値が保管される列の名前を含みます。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かを示しています。

表3. 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
fsId()	INTEGER	INTEGER

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。

- <constant> エレメント。定数の値を含みます。
 - <column> エレメント。定数値が追加される列の名前を含みます。
10. <containerMapping> エレメントは、コンテナ・タイプ・フィーチャー (配列またはリスト) のマッピングを含みます。このエレメントは、コンテナ・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。
- <feature> エレメント。フィーチャーの名前を含みます。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。
 - <table> エレメント。データベース・スキーマおよび表名を含みます。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。フィーチャー構造の名前とフィーチャーが追加される列名を含みます。
11. 提供されたスキーマを使用して、XML ファイルを保存し、妥当性検査します。

XML ファイルを作成したあと、エンタープライズ・サーチ管理コンソールを使用して、それをエンタープライズ・サーチにアップロードし、他のカスタム分析の選択に加えて、共通分析構造からデータベースへのマッピング・ファイルを選択する必要があります。

関連概念

48 ページの『選択した分析結果のデータベース・マッピング』

36 ページの『フィーチャー・パス』

関連資料

40 ページの『フィルター』

37 ページの『組み込みフィーチャー』

25 ページの『タイプ・システム記述のサンプル』

「コンテナ・タイプ」のマッピング

コンテナ・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・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。

共通分析構造からデータベースへのマッピング・ファイルでコンテナ・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 つ目の方法はリンク・テーブル定義を使用します。リンク・テーブル定義は <containerMapping> エレメントによって定義し、使用したい指定した情報間の関係を明示的に表します。

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。警察レポートと被疑者の間には、n:m の関係があります。これは、一人の被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、複数の被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、警察レポートと被疑者の間の、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。警察レポートと被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被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>
      </featureMappings>
    </implicitMappingRule>

    <implicitMappingRule applyToSubtypes="false">
      <type>uima.cas.FSArray</type>
      <table>sample.fsarray</table>
      <featureMappings>
        <featureMapping>
          <feature>uniqueId()</feature>
          <column>arrayId</column>
        </featureMapping>
      </featureMappings>
    </implicitMappingRule>
  </implicitMappings>
</cas2JdbcMappings>
```

```

    <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 4. *sample.policeReport* 表

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

表 5. *sample.fsarray* 表

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

表 6. *sample.suspect* 表

suspectID	lastname	description
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・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、警察レポートと被疑者の間には、`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被疑者の間のリンク・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の反復を定義するた
めに使用されます。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> エレメントは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 7. *sample.policeReport* 表

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

表 8. *sample.reports_suspect* 表

policeReportId	suspectId
bbb...1	<i>ccc...1</i>
bbb...2	<i>ccc...2</i>
...	...

表 9. *sample.suspect* 表

suspectID	lastname	description
<i>ccc...1</i>	Brown	Dark complexion
<i>ccc...2</i>	Smith	Wears glasses
...

関連資料

37 ページの『組み込みフィーチャー』

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および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リトリブできます。

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には (例えば、すべての人物を取得するなど)、共通分析構造から索引へのマッピング・ファイルの中に、注釈タイプのフィールド・スタイル・マッピングを含め、それに、戻すことができるというマークを付けます。次に例を示します。

```
<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。これらの注釈値をリトリブするには、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結果処理に制限したい場合には、不適切です。例えば、文書に 5 人の人物が記載されているとします。ただし、セマンティック検索照会 '`<sentence><person/>IBM</sentence>`' で、「IBM」という用語が現れたのと同じ文にある人物にのみ興味があります。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。次に例を示します。

```
<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。次に例を示します。

```
<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 をリトリブします。検索および索引 API (SI-API) 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。次に例を示します。

```
$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。プロパティ内の各オカレンスは、整数値に変換できます。
5. SI-API 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リトリブするには、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、スパン名が出てくる共通分析構造から索引へのマッピング・ファイルと、表名が出てくる共通分析構造からデータベースへのマッピング・ファイルから導き出されます。
 - 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。例えば、"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 (例えば、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、結果にフィルターを掛けることができます。

関連概念

62 ページの『セマンティック検索照会条件』

関連タスク

42 ページの『共通分析構造から索引へのマッピング・ファイルの作成』

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 (SI-API) 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。

SI-API 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となります。

セマンティック検索照会条件は、よりの確な文書のリトリートに役立ちます。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リトリートしたり、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条件

セマンティック検索照会条件は、あいまい条件と呼ばれます。

検索および索引 API (SI-API) であいまい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

XML フラグメント照会条件は、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、あいまい条件記号 @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 照会条件は、@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 (SI-API) インターフェースの一般的な照会条件と同じように、
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)

照会条件の同義語も考慮します。

波形記号 (~) (接尾部)

照会条件と同じ見出し語を持つ語も考慮します。

番号記号 (#)

用語は強調表示されます。

以下に XML フラグメント照会の例を示します。

@xmlf2::'<City>Springfield</City>'

スパン (注釈) City を含み、それにストリング Springfield が含まれる文書
を検出します。

@xmlf2::'<Person gender="female"/>'

女性という注釈が付けられ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>'**

PoliceReport 注釈の crimeDescription 属性がストリング robbery または theft
で、ストリング accident ではない、被疑者 (suspect) とみなされる男性
(male) を指定する文書を検出します。文書には、テキスト・ワード
Springfield をカバーする City 注釈が含まれ、district 属性が Brynston また
は Brooklyn である注釈も含まれている必要があります。

対応する XPath 照会には、以下の構造があります。

@xmlxp::'//City ftcontains ("Springfield")'

スパン (注釈) City を含み、それにストリング Springfield が含まれる文書
を検出します。

@xmlp::'//PoliceReport[City ftcontains("Springfield")]'

スパン `PoliceReport` に、スパン (注釈) `City` を含み、それにストリング `Springfield` が含まれる文書を検出します。

@xmlp::'//Person[@gender="female" or @title ftcontains("Ms") or @title ftcontains("Mrs")]'

女性という注釈が付けられた文書を検索します。`gender` 属性では値は厳密に一致する必要がありますが、`title` 属性では、`Ms` および `Mrs` が属性値と完全に一致する必要はありません。

検索アプリケーションの同義語サポート

照会条件の同義語を含む文書を検索することで拡張された検索結果を得ることができます。

同義語は、通常 *OmniFind Enterprise Edition* 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。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、ユーザーが照会条件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条件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。照会条件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、照会条件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、すべての照会ワードについて同義語の検索が行われます。検索結果には、照会条件か照会条件の同義語のいずれかを含む文書が入ります。SIAPI は、実行依頼された照会の同義語拡張のリスト生成もサポートします。

- N-gram コレクションの同義語拡張によって、照会テキストの句の分割が可能になります。同義語辞書に句全体が表示される場合、検索は正常に行われます。以下の区切り文字に従って、句が抽出されます。

句読点 以下の文字は区切り文字です。 - () + . ,

引用符は無視されるので句を区切りません。

英字での変更

例えば、N-gram コレクションの場合、同義語辞書に ABC があると、照会は、以下のサンプル照会に ABC の同義語を含むように拡張されます。

ABC run DEF stand (ABC と DEF は日本語テキストです)

ABC+DCF+GHI

同義語用 XML ファイルの作成

照会条件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 XML ファイルからバイナリー辞書ファイルを作成し、それをエンタープライズ・サーチにアップロードし、該当するコレクションに割り当てます。

このタスクについて

同義語をリストする XML ファイルは、特定のスキーマに従っている必要があります。次は同義語の XML ファイルの例です。

```
<?xml version="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

互いに同義語であるワード (<synonym> エlement) を <synonymgroup> Element 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 (ワードの単数形と複数形など) 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も (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)、大文字小文字が異なる複数の用語を含む必要もありません。例えば、用語 météo を同義語として含む場合に、用語 METEO も含む必要はありません。

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。XML ファイルの XSD スキーマは、synonyms.xsd と呼ばれ、エンタープライズ・サーチのインストールの `ES_INSTALL_ROOT/packages/uima/` に含まれています。
2. <synonymgroup>Element 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> Element を挿入します。

マッピングは、必ず、<synonymgroups xmlns="http://www.ibm.com/of/822/synonym/xml">Element 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 ファイルを保存して、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

XML ファイルに同義語のリストを作成するか、更新したあと、その XML ファイルをバイナリー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、`essyndictbuilder` というコマンド行ツールを使用します。これは OmniFind Enterprise Edition に付属しています。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、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。

エンタープライズ・サーチ内の `.dic` の最大サイズは、8 MB です。

手順

エンタープライズ・サーチ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、OmniFind Enterprise 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 つ以上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

検索の適合度を高めるために、照会から外す、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辞書は言語と無関係です。

ユーザー定義のストップワードは、通常 *OmniFind Enterprise Edition* 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語は、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。

例えば、照会条件 *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。ワード *Police* がストップワード辞書にリストされていて、複合照会条件 *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。

ストップワード用 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、XML 文書内に指定された特定のスキーマに従っている必要があります。次はストップワードの XML ファイルの例です。

```
<?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、*météo* をストップワードとして含む場合に、用語 *METEO* も含む必要はありません。

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。XML ファイルの XSD スキーマは、`stopWords.xsd` と呼ばれ、エンタープライズ・サーチのインストールの `ES_INSTALL_ROOT/packages/uima/` に含まれています。
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 ファイルを保存して、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、OmniFind Enterprise 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、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。例えば、`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。

エンタープライズ・サーチ内の `.dic` の最大サイズは、8 MB です。

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、OmniFind Enterprise 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 つ以上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

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

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、まった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条件が、リトリート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、 $B1 > B2$ であれば、 $\text{ランキング調整}(X) \geq \text{ランキング調整}(Y)$ になります。

ランキング調整ワードには、通常 *OmniFind Enterprise Edition* という製品名などの、複数のワードから成る用語が含まれます。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。

ランキング調整ワード辞書は言語と無関係です。

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語は、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。複合語の各用語にランキング調整値が存在する場合、リトリートされた文書はランク付けされますが、割り当てられた値は、その用語自体が (複合語の一部としてではなく) 文書に現れる場合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。

例えば、照会条件 *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。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、複合照会条件 *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、XML ファイル内に指定された特定のスキーマに従っている必要があります。次はランキング調整ワードの XML ファイルの例です。

```
<?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 separately-->
    <term useVariants="true">OmniFind Edition</term>
    <term useVariants="false">Edition</term>
    <term>OmniFind</term>
  </boostTermList>
  <boostTermList boost="8">
    <term useVariants="true">WAS</term>
    <term>term9</term>
  </boostTermList>
</boostTerms>
```

制約事項

`<boostTermList>` エレメントの中でランキング調整値が同じ用語をグループにできますが、ランキング調整値が複数回発生する場合があります。例えば、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、頭字語または省略語といった変形を持つことができます。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、同義語辞書に既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。リトリート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む場合に、用語 `METEO` も含む必要はありません。

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。XML ファイルの XSD スキーマは、boostTerms.xsd と呼ばれ、エンタープライズ・サーチのインストールの `ES_INSTALL_ROOT/packages/uima/` に含まれています。
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. `<boostTermList>` エレメントを追加して、指定したランキング調整値と同じ値を持つすべての用語をグループ化します。

ランキング調整値は -10 から 10 の範囲です。例えば、`<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>` エレメントを追加し、そこに、指定したランキング調整値を使用する各用語を含めます。

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 ファイルを保存して、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、`esboostworddictbuilder` というコマンド行ツールを使用します。これは `OmniFind Enterprise Edition` に付属しています。このツールは、`ES_INSTALL_ROOT/bin` ディレクトリにあります。

ツールへの入力、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。例えば、`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。

エンタープライズ・サーチ内の `.dic` の最大サイズは、8 MB です。

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、OmniFind Enterprise 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 つ以上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

67 ページの『同義語辞書の作成』

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。基本的な非辞書ベースのサポートにはセンテンス・セグメンテーションも含まれます。
- 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レンマタイゼーションがあります。

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』

78 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、レンマタイゼーションを行う前に、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。

表 10. 自動言語識別でサポートされる言語

アフリカーンス語	アラビア語	バリ語
バスク語	カタロニア語	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
アイスランド語	アイルランド語 (ゲール語)	イタリア語
日本語	韓国語	マレー語
ノルウェー語 (ブークモール)	ポーランド語	ポルトガル語

表 10. 自動言語識別でサポートされる言語 (続き)

ルーマニア語	ロシア語	スペイン語
スウェーデン語	タガログ語	タイ語
トルコ語	ベトナム語	

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。エンタープライズ・サーチ管理コンソールでクローラーのプロパティを編集して、文書の言語を指定します。デフォルトの言語は英語です。

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、言語特定の辞書が無い文書が処理されます。

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で制限することができます。例えば、複数言語の文書コレクションの中で Jacques Chirac に関する文書を検索する場合、検索結果を、フランス語で書かれている文書のみが含まれるように制限することができます。結果文書の言語の設定は、エンタープライズ・サーチ管理コンソールで選択できる拡張検索オプションです。

関連概念

77 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』

『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』

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

字句解析テクノロジーによってサポートされない言語の文書の場合、エンタープライズ・サーチは、unicode ベースの空白文字の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の基本サポートを提供します。

unicode ベースの空白文字の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。この単純なセグメンテーション方式は、多くの検索タスクで有効です。

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 ベースの空白文字の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。

コレクションを作成するときに、オプションで、中国語および日本語の文書の N-gram セグメンテーションを使用したトークン化を選択することもできます。

N-gram セグメンテーションの間に、例えば改行文字やタブ文字などのすべての空白文字を除去するには、文書の解析を始める前に、`collection.properties` ファイルの中の `ES_NODE_ROOT/master_config/<CollectionID>.parserdriver` のパラメーター設定をオンにする必要があります。空白文字を除去するために必要なパラメーターは次のとおりです。

- **removeCjNewLineChars:** このパラメーターを `true` に設定すると、中国語または日本語の文字の間の改行文字とタブ文字は、すべて除去されます。デフォルトは `removeCjNewlineChars=false` です。
- **removeCjNewLineCharsMode:** このパラメーターを `all` に設定すると、文字コンテキストとは関係なく空白文字が除去されます。例えば、英語のテキストの空白文字も除去されます。このオプションを指定する場合は、プロパティ・ファイルにパラメーターを追加する必要があります。 `removeCjNewlineCharsMode=all` のみ有効で、これ以外の値はすべて無視されます。

関連概念

77 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』

77 ページの『言語の識別』

数字の N-gram トークンとしてのトークン化

2 バイト文字に加えて数字を N-gram トークンとしてトークン化するには、アナテーター・ディスクリプター・ファイルのパラメーター設定を変更する必要があります。

このタスクについて

空白および N-gram トークナイザーでのデフォルトの数字の処理は、すべての数字を空白でセグメント化されたトークンとして扱うことです。数字を N-gram トークンとしてトークン化するには、アナテーター・ディスクリプター・ファイルの N-gram モード設定を変更しなければなりません。この設定は、エンタープライズ・サーチ管理コンソールを使用して変更することはできません。

ヒント: N-gram トークン化には、3 つのモード (標準、数字、完全) を使用できます。この手順では、数字 N-gram トークン化を使用可能にする方法を説明します。

エンタープライズ・サーチ・コレクションで完全 N-gram トークン化のサポートを構成する方法について、および、完全 N-gram サポートに構成されたコレクションでの文字の処理方法について詳しくは、<http://www.ibm.com/support/docview.wss?rs=63&uid=swg27011088> を参照してください。

手順

デフォルトの N-gram モード設定は `normal` (標準) と呼ばれ、数字と SBCS 文字を、空白でセグメント化された文字として扱います。数字 N-gram モードを使用可能にするには、次のようにします。

1. コレクションのパarserを停止します。
2. コレクションのランタイムを停止します。
3. `ES_NODE_ROOT/master_config/collection_ID.parserdriver/specifiers` ディレクトリーにある `jtok.xml` という名前のアノテーター・ディスクリプター・ファイルを開きます。`collection_ID` は、コレクションが作成されたときに、コレクションに指定された ID (またはシステムによって割り当てられた ID) です。
4. **NgramMode** パラメーター設定を `normal` (標準) から `numeric` (数字) に変更します。
5. コレクションのパarserを再始動します。
6. ランタイムを再始動します。

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イ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。

レンマタイゼーション

レンマタイゼーションは、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)。

レンマタイゼーションでは、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。レンマタイゼーションは、照会において、さまざまな語形変化した語を含む文書を検

出すことにより、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* という語を含む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。例えば、次のとおりです。

wouldn't は *would* と *not* に分割されます。

Horse's は *Horse* と *'s* に分割されます。

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。例えば、次のとおりです。

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、アポストロフィは、接語の場合は語の一部とみなされ、不明な省略形の場合は終止符 (またはピリオド) とみなされます。URL、E メール・アドレス、および日付は、いくつかのトークンに分割されます。

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。

表 11. サポートされる言語

アラビア語	イタリア語
中国語 (繁体字および簡体字)	日本語
チェコ語	韓国語
デンマーク語	ノルウェー語 (ブークモール)
オランダ語	ポーランド語
英語	ポルトガル語 (本国およびブラジル)

表 11. サポートされる言語 (続き)

フィンランド語	ロシア語
フランス語 (本国およびカナダ)	スペイン語
ドイツ語 (本国およびスイス)	スウェーデン語
ギリシャ語	

関連概念

『日本語における語のセグメンテーション』

『日本語における変種文字』

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テクノロジーを使用して、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

80 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』

『日本語における変種文字』

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、標準表記でないカタカナをその基本型 (見出し語のようなもの) にマップし、標準的な表記法から外れたカタカナが照会ストリングに含まれていても、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

80 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』

『日本語における語のセグメンテーション』

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、語が名詞であるか動詞であるかを認識します。他の言語の場合、エンタープライズ・サーチは特別ナリストを使用します。

次の場合は、照会処理でストップワードは除去されません。

- 照会の中のすべてのワードがストップワードである。ストップワード処理ですべての照会条件が除去されると、結果セットは空です。検索結果が戻されるようにするため、すべての照会条件がストップワードであれば、ストップワードの除去は使用不可にされます。例えば、ワード *car* がストップワードで、検索対象が *car* の場合、検索結果にはワード *car* に一致する文書が含まれます。検索対象が *car buick* の場合、検索結果にはワード *buick* に一致する文書のみが含まれます。
- 照会の中のワードの前に正符号 (+) が置かれている。
- ワードが完全一致の一部である。
- ワードが句の中に入っている。例えば、"I love my car"。

関連概念

『文字の正規化』

文字の正規化

文字の正規化は、文書のヒット率を上げるプロセスです。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、より多くの文書がリトリートされることになります。

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

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* を検索する場合、*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* を検索する場合、*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* を検索する場合、*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* を検索する場合、*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* を検索する場合、*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* を検索する場合、*usa* が含まれている文書を検索することも、*é* を検索する場合、*e* が入っているワード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で使用することもできます。例えば、*METEO* を検索する場合、*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

82 ページの『ストップワードの除去』

正規表現アノテーター

正規表現アノテーターによって、ユーザー独自のテキスト分析エンジンのインプリメントを必要とせずに、カスタム・テキスト分析を実行できます。ユーザーが自分で定義できる規則のセット (正規表現) に基づいて、正規表現アノテーターはテキスト文書内の情報構造を検出し、検出した情報の注釈を共通分析構造の中に作成します。

正規表現アノテーターは、テキスト文書内のエンティティまたは情報単位 (例えば、電話番号、製品コード、建物と部屋番号、住所など) を正規表現に基づいて検出します。正規表現の 1 つが文書テキストの一部と一致すると、正規表現アノテーターは、一致した情報の部分をカバーする、対応する注釈を作成します。これらの注釈は共通分析構造に保管され、あとで、共通分析構造から索引へのマッピング・ファイルを使用して、これらの分析結果をエンタープライズ・サーチ索引にマッピングすることによって検索できます。一方、注釈を JDBC 対応データベースに保管するために、共通分析構造からデータベースへのマッピング・ファイルが作成されます。

ユーザーが定義した規則のセット (正規表現) は XML 構成ファイル (規則セット・ファイルとも呼ばれる) に保管されます。正規表現アノテーターは、これらの正規表現を処理する分析ロジックを含みます。これは Java 1.4 の正規表現構文をサポートします。

正規表現アノテーターのタイプ・システム記述は、正規表現アノテーターによって使用され作成される注釈タイプとフィーチャーを定義しなければなりません。正規表現アノテーターのアプリケーション領域の複雑さに応じて (例えば、さらにタイプが必要であれば、提供された正規表現アノテーターの中でタイプが定義されます)、正規表現アノテーター・ディスクリプターの中に追加の入出力機能を定義しなければなりません。ディスクリプターの中で使用されるタイプは、アノテーターのタイプ・システム記述の中のタイプと一致しなければなりません。

正規表現アノテーターは、電話番号、URL、および E メール・アドレスを検出するサンプルの規則を含むよう構成された、デプロイ可能な PEAR (Processing Engine ARchive) ファイルとしてエンタープライズ・サーチの中に組み込まれています。

関連概念

88 ページの『規則セット・ファイル』

関連タスク

89 ページの『正規表現規則の定義』

関連資料

93 ページの『アノテーター・ディスクリプター』

96 ページの『ロギング』

正規表現アノテーターを使用した簡単なセマンティック検索

エンタープライズ・サーチには、テキスト文書内の電話番号、URL、および E メール・アドレスの検出を可能にする規則のセットが事前に構成された正規表現分析エンジンが組み込まれています。

正規表現分析エンジンのこのサンプル構成を使用すると、エンタープライズ・サーチが文書内でキーワード *phone number* を検索せずに、文書内にある実際の電話番号を検索できるようにすることができます。正規表現アノテーターによって検出される構成を照会するために、サンプルの共通分析構造から索引へのマッピング・ファイルも提供されます。さらに、単純なキーワードを使って強力なセマンティック照会を発行できる、簡単な方式もあります。この方式はエンタープライズ・サーチ同義語サポートを使用して、単純なキーワード照会をセマンティック照会に自動的に展開します。このメカニズムを説明するサンプルの同義語辞書が提供されています。サンプルの構成で正規表現アノテーターを使用するために必要なすべてのファイルは、`ES_INSTALL_ROOT/packages/uima/regex` にあります。

多くのアプリケーション・シナリオでは、ユーザーの必要に合わせて正規表現アノテーターを調整しようとするときに、サンプル構成と一緒に提供された正規表現規則を少し変更するだけで十分かもしれません。

しかし、アノテーターを全面的にカスタマイズする場合は、UIMA SDK を使用することをお勧めします。このために、正規表現アノテーターも `ES_INSTALL_ROOT/packages/uima/` にあるエンタープライズ・サーチ・ベース・アノテーター・パッケージの中に含まれています。

関連タスク

- 『正規表現アノテーターを使用した簡単なセマンティック検索の使用可能化』
92 ページの『正規表現アノテーターのカスタマイズ』
- 13 ページの『ベース・アノテーターとカスタム・テキスト分析結果の表示』

正規表現アノテーターを使用した簡単なセマンティック検索の使用可能化

同義語を使用した簡単なセマンティック検索を使用可能にするには、正規表現アノテーター、共通分析構造から索引へのマッピング・ファイル、およびサンプル同義語辞書をエンタープライズ・サーチ・システムに追加し、これらのリソースを対象のコレクションに関連付ける必要があります。

そのあと、正規表現アノテーターは解析フェーズで文書を処理し、インデクサーはカスタム分析結果を索引に追加します。サーチ・サービスは提供された同義語辞書を使用し、自動的にセマンティック照会に展開される単純なキーワードを通してカスタム分析結果を検索できます。

手順

簡単なセマンティック検索を使用可能にするには、次のようにします。

- エンタープライズ・サーチ管理コンソールを使用して、`ES_INSTALL_ROOT/packages/uima/regex` にある `of_regex.pear` と呼ばれる正規表現カスタム・テキスト分析エンジンをエンタープライズ・サーチ・システムに追加します。
- 正規表現テキスト分析エンジンをコレクションに関連付けます。

3. ディレクトリー `ES_INSTALL_ROOT/packages/uima/regex` にある `of_sample_regex_cas2index.xml` という共通分析構造から索引へのマッピング・ファイルを追加します。これは、正規表現アノテーターが生成したカスタム分析結果 (注釈) を、エンタープライズ・サーチ索引内の検索可能なスパンにマップします。その結果、XML フラグメントまたは XPath 照会を使用して、これらのスパンを検索できます。
4. コレクションのクロール、解析、および索引作成を行います。索引作成が終了した時点で、XML フラグメント式、例えば `@xmlf2::'<#phonenumber>'` を使用し、検索アプリケーションを使用して XML 検索照会に入ることでもあります。しかし、同義語によるセマンティック検索を使用可能にする目的は、`Barbara phone number` のような照会を使用できるようにし、システムに、その照会を `Barbara @xmlf2::'<#phonenumber>'` に変換させることです。
5. 管理コンソールを使用して、ディレクトリー `ES_INSTALL_ROOT/packages/uima/regex` にある `of_sample_synonym_dic.dic` という、提供されたサンプルのバイナリー同義語辞書をエンタープライズ・サーチ・システムに追加します。サンプル辞書のソース XML を変更するか、それを基に独自の辞書を作成することもできます。そして、それを `essyndictbuilder` ツールを使用して新しい辞書ファイルに変換します。XML サンプル同義語辞書は、`of_sample_synonym_dic.xml` と呼ばれ、これもまた `ES_INSTALL_ROOT/packages/uima/regex` にあります。
6. 同義語辞書をコレクションに関連付け、コレクションのサーチ・サービスを開始 (または再開) します。
7. 検索アプリケーションで、意味展開を使用して自動的に同義語を検索するオプションを選択します。このオプションを使用可能にしたあと、検索アプリケーションは基本的なキーワード照会を XML フラグメント照会に書き直し、電話番号、E メール・アドレス、および URL を識別する検索可能スパンを検出する式を組み込みます。
8. 検索アプリケーションで、電話番号を要求する照会、例えば、`barbara telephone number` を入力します。この照会は、3 つのキーワード `barbara`、`telephone`、および `number` を含んだ文書を検索するとともに、キーワード `barbara` を含み、文書の中に電話番号に定義された正規表現に一致する数字と文字のスパンを含む文書を検索します。検出されたキーワードと電話番号は、検索結果の中で強調表示されます。

提供されたサンプル同義語辞書で、どのキーワードがセマンティック照会に変換されるかが分かります。

```
<?xml version="1.0" encoding="UTF-8"?>
<synonymgroups xmlns="http://www.ibm.com/of/822/synonym/xml">
  <synonymgroup>
    <synonym>telephone number</synonym>
    <synonym>phone number</synonym>
    <synonym>telephone nbr</synonym>
    <synonym>phone nbr</synonym>
    <synonym>@xmlf2::'&lt;#phonenumber/&gt; '</synonym>
  </synonymgroup>
  <synonymgroup>
    <synonym>facsimile number</synonym>
    <synonym>fax number</synonym>
    <synonym>facsimile nbr</synonym>
    <synonym>fax nbr</synonym>
    <synonym>@xmlf2::'&lt;#phonenumber/&gt; '</synonym>
  </synonymgroup>
</synonymgroups>
```



```
<synonym>e-mail address</synonym>
<synonym>email address</synonym>
  <synonym>@xmlf2:.'&lt;#email/&gt;.'</synonym>
</synonymgroup>
<synonymgroup>
  <synonym>URL</synonym>
  <synonym>unified resource locator</synonym>
  <synonym>Web address</synonym>
  <synonym>@xmlf2:.'&lt;#url/&gt;.'</synonym>
</synonymgroup>
</synonymgroups>
```

関連概念

86 ページの『正規表現アノテーターを使用した簡単なセマンティック検索』

規則セット・ファイル

正規表現アノテーターの中で、XML 規則セット・ファイルは正規表現の形で規則を定義し、これらはテキスト文書の解析に使用されます。

規則は、文書テキスト内の場所、アノテーターが検出しなければならないもの、一致が見つかったときに行うアクションを順番に指定します。

正規表現アノテーターが呼び出されると、正規表現パターンが含まれた XML 規則セット・ファイルはコンパイルされ、文書テキストの部分に対して突き合わせが行われます。一致または部分一致が見つかったら、特定の規則に関連付けられた注釈が作成され共通分析構造に保管されます。

規則の中で使用されるタイプは、正規表現アノテーターのタイプ・システム記述の中で定義されなければなりません。

正規表現アノテーターは、XML 規則セット・ファイルの最初の規則から始めて、一度に 1 つの規則を処理します。各規則について、コンパイルされた対応する正規表現が、前のステップで作成された注釈 (例えば、正規表現アノテーターの前に文書を処理したアノテーターによって作成された注釈) に突き合わされます。規則に一致する注釈は、正規表現アノテーター・ディスクリプターの中で指定された入力機能タイプと同じタイプでなければなりません。

一致が見つかった場合、呼び出される規則の中で作成される注釈タイプもまた、正規表現アノテーター・ディスクリプターの中で有効な出力機能タイプとして指定されなければなりません。前の規則で作成された新規の注釈は、XML 規則セットの中であとで呼び出される規則の入力注釈として使用されることが可能です。

関連概念

85 ページの『正規表現アノテーター』

関連タスク

89 ページの『正規表現規則の定義』

関連資料

93 ページの『アノテーター・ディスクリプター』

96 ページの『ロギング』

正規表現規則の定義

規則セットは、文書内のテキストに対して突き合わせる正規表現と、パターンが一致したときに正規表現アノテーターが行う必要のあるアクションを定義します。

このタスクについて

XML 規則セット・ファイルは、次の例に概要を示した規則構文に従う必要があります。次は、電話番号、URL、および E メール・アドレスを認識するサンプル正規表現アノテーターの規則セット・ファイルです。

トップレベル・エレメントは <ruleSet> エレメントで、これは 1 つ以上の <rule> エレメントから成ります。各 <rule> エレメントは、属性 regEx、matchStrategy、および matchType から構成される Java 正規表現を同様に定義します。アクションは <createAnnotation> エレメントの中で定義され、注釈 ID と注釈タイプを指定します。

```
<?xml version="1.0" encoding="UTF-8"?>
<ruleSet xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:noNamespaceSchemaLocation="ruleSet.xsd">
  <!-- Phone Number -->
  <!-- This rule matches different ways of writing telephone numbers,
    for example, 01234-12345, 01234 / 122-32, (001234)12345,
    +49 (0) 123412345, (123) 123 1234,
    1-800-IBM-4YOU -->
  <rule regEx="(?(x)(%s|%b)) (
0{1,2}[1-9]{1}[0-9]{1,5}%x20?[-/%%]x20?[1-9]{1}([0-9]{1,8}-?)
  {1,3}[0-9]{1,}
|%(0[1-9]{1}[0-9]{1,3}%x20?[1-9]{1}[0-9]{2,8}
|%(00[1-9]{1}[0-9]{1,8}%x20?[1-9]{1}[0-9]{2,10}
|%(0%x20?[1-9]{1}[0-9]{1,3}|00%x20?[1-9]{1}[0-9]{1,8})%x20?[1-9]
{1}[0-9]{1,3}(%x20[0-9]{2,4}){1,5}
|0%x20?[1-9]{1}[0-9]{1,3}|00%x20?[1-9]{1}[0-9]{1,8})%x20?[/%%]x20?
[1-9]{1}[0-9]{1,3}(%x20[0-9]{2,4}){1,5}
|%(?%+[1-9]{1}[0-9]{0,3})%?([-x20]|x20?(0%))[-x20]?[1-9]
{1}[0-9]{1,10}
|%(?%+[1-9]{1}[0-9]{0,3})%?([-x20]|x20?(0%))[-x20]?[1-9]
{1}[0-9]{1,3}[-x20]([0-9]{2,5}[-x20]?) {1,4}
|1-)?[0-9]{3}-[0-9]{3}-[0-9]{4}
|%(1-9){1}[0-9]{2}%x20[0-9]{3}[-x20][0-9]{4}
|1-(800|888|877|866)-([A-Z0-9]{7}|[A-Z0-9]{3}-[A-Z0-9]
{4}|[A-Z0-9]{4}-[A-Z0-9]{3})
) (?!(%d|x20%d|-%d)) (%s|%b)"
  matchStrategy="matchAll" matchType="uima.tcas.DocumentAnnotation">
  <createAnnotation id="phonenummer" type="com.ibm.es.uima.PhoneNumber">
  <begin group="0"/>
  <end group="0"/>
  </createAnnotation>
</rule>
<!-- potential Phone Number -->
<!-- This rule matches numbers that resemble telephone numbers but could
also be anything else. For example, 0123 1234 123,
+123456789, 123 123 1234 -->
<rule regEx="(?(x)(%s|%b)) (
0[1-9]{1}[0-9]{1,3}%x20[1-9]{1}[0-9]*x20?([0-9]{2,}x20?)+
|00%x20?[1-9]{1}[0-9]{0,3}%x20[1-9]{1}[0-9]{1,3}%x20?[1-9]
{1}([0-9]{2,}x20?)+
|%+[1-9]{1}[0-9]{0,3}[1-9]{1}[0-9]{6,}
|[1-9]{1}[0-9]{2}%x20[0-9]{3}%x20[0-9]{4}
) (?!(%d|x20%d|-%d)) (%s|%b)"
  matchStrategy="matchAll" matchType="uima.tcas.DocumentAnnotation">
<createAnnotation id="potential_phonenummer"
  type="com.ibm.es.uima.PotentialPhoneNumber">
```

```

    <begin group="0"/>
    <end group="0"/>
  </createAnnotation>
</rule>
<!-- URL Annotation -->
<!-- This rule matches URLs, for example, http://www.ibm.com -->
<rule regex="(?(x) (¥s|¥b) (
  http://[¥w¥-]+([¥.][¥w¥-]+)([/] [¥w¥~¥(¥)¥-¥?=%¥u0026¥#]*)*
  |www.[¥w¥-]+([¥.][¥w¥-]+)([/] [¥w¥~¥(¥)¥-¥?=%¥u0026¥#]*)*
  )(¥s|¥b)"
  matchStrategy="matchAll" matchType="uima.tcas.DocumentAnnotation">
  <createAnnotation id="url" type="com.ibm.es.uima.URL">
    <begin group="0"/>
    <end group="0"/>
  </createAnnotation>
</rule>
<!-- Email Annotation -->
<!-- This rule matches e-mail addresses, for example, yourName@domain.com -->
<rule regex="(?(x) (¥s|¥b) (
  [a-zA-Z0-9][¥w¥.-]*[a-zA-Z0-9]@[a-zA-Z0-9]([¥.-]?¥w)*¥.[a-zA-Z]
  {2,3})(¥s|¥b)"
  matchStrategy="matchAll" matchType="uima.tcas.DocumentAnnotation">
  <createAnnotation id="email" type="com.ibm.es.uima.Email">
    <begin group="0"/>
    <end group="0"/>
  </createAnnotation>
</rule>
</ruleSet>

```

手順

カスタム正規表現を定義する正規表現アノテーターの XML 規則セットを作成するには、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。XML 規則セット・ファイルの XSD スキーマは `ruleSet.xsd` と呼ばれ、エンタープライズ・サーチのインストールの `ES_INSTALL_ROOT/packages/uima/regex/` ディレクトリにあります。
2. マッピングを `<ruleSet xmlns="http://www.w3.org/2001/XMLSchema-instance" xsi:noNamespaceSchemaLocation="ruleSet.xsd">` エlement に含めます。名前空間が `xmlns` 属性の中に指定され、示されたものと完全に一致している必要があります。
3. `<rule>` Element を追加し、それに、正規表現パターン、`matchStrategy` 属性、および `matchType` 属性を含んだ `regex` 属性を入れます。

アノテーターは、Java 1.4 正規表現構文を完全にサポートします。正規表現の概要および完全な構文を表示するには、Java の資料 (<http://java.sun.com/j2se/1.4.2/docs/api/java/util/regex/Pattern.html>) を参照してください。

`matchStrategy` は検索方法を指定します。例えば、すべての一致が文書内で見つからなければならないのか、または、テキストの突き合わせは完全一致でなければならないのかなどです。次の 3 つの突き合わせ方法を使用できます。

- `matchFirst` は、正規パターンに一致した最初のテキスト・シーケンスで停止します。
- `matchAll` は、正規パターンに一致した、文書内のすべてのテキスト・シーケンスを検出します。

- `matchComplete` は、完全に一致するテキスト・シーケンスのみを一致とします。例えば、パターンが「foo」であれば、用語「foo」のみが一致で、「foobar」は一致という結果にはなりません。

`matchType` は規則が突き合わせの対象とする注釈タイプを決定します。この方法により、突き合わせる正規表現を、例えば存在するトークン注釈内に制限できます。これにより、規則内の突き合わせ内容が多くなりすぎないようにできます。可能なタイプは、アノテーターに入力できる注釈タイプ (アノテーター・ディスクリプターの中で定義される)、例えば、`uima.tt.DocumentAnnotation`、`uima.tt.ParagraphAnnotation` や、`foo.bar.MyAnnotation` といったユーザー定義タイプです。ある規則の出力タイプが、その後の規則の入力タイプとして使用されることもあります。`matchType` によって、特定の規則の検索有効範囲を制限できます。

4. `<createAnnotation>` エレメントを追加し、その中に、一致が見つかったときに正規表現アノテーターが行うアクションを定義します。

各 `createAnnotation` エレメントは次の 2 つの属性を持ちます。

- `id` は注釈を一意的に識別し、注釈の参照に使用されます。
 - `type` は作成される注釈のタイプを指定します。
5. `<createAnnotation>` エレメントの突き合わせの場所を定義する次のコンポーネント・エレメントを追加します。
 - 必須: `<begin>` は突き合わせの開始位置を指定します。このエレメントは 2 つの属性を持ちます。
 - 必須: `group` は Java キャプチャリング・グループを識別します。これには、0 (完全なテキスト・シーケンス一致) から 9 (複数のキャプチャリング・グループ) の値を指定できます。
 - オプション: `location` は突き合わせグループ内の位置を (小括弧の位置決めに関して)、`start` (左括弧) または `end` (右括弧) のどちらかで指定します。
 - 必須: `<end>` は突き合わせの終了位置を指定します。このエレメントは 2 つの属性を持ちます。
 - 必須: `group` はキャプチャリング・グループを識別します。これには、0 (完全なテキスト・シーケンス一致) から 9 (後続およびより小さな一致グループ) の値を指定できます。
 - オプション: `location` は突き合わせグループ内の位置を (小括弧の位置決めに関して)、`start` (左括弧) または `end` (右括弧) のどちらかで指定します。
 - オプション: `<setFeature>` はフィーチャーを作成し、それを注釈に割り当てます。このエレメントは 2 つの属性を持ちます。
 - `name` はタイプ・システム記述の中で定義したとおりのフィーチャーの名前です。
 - `type` はフィーチャー値のタイプを、`String`、`Integer`、`Float`、および `Reference` のいずれかで指定します。タイプは、アノテーター・タイプ・システム記述の中でフィーチャーに定義した範囲タイプと同じでなければなりません。

タイプ Reference のフィーチャーは、2 つの注釈間のセマンティック関係を表すリンクを作成するために使用されます。<setFeature> エLEMENTの内容には、作成するリンク先の <createAnnotation> ELEMENTの id を設定する必要があります。

関連概念

88 ページの『規則セット・ファイル』

正規表現アノテーターのカスタマイズ

サンプルの規則セットとタイプ・システム・ファイルを少し変更することで、正規表現アノテーターのサンプル構成をカスタマイズして、新規のエンティティを検出したり (例えば製品シリアル番号など)、既存のエンティティに正規表現規則を付け加えたり (例えば、会社を特定した電話番号を検出するなど) できます。

変更された規則セット・ファイルとタイプ・システム記述は、正規表現処理エンジン・アーカイブ・ファイル (PEAR ファイル) に追加する必要があります。PEAR ファイルを更新したあと、カスタマイズされた正規表現テキスト分析エンジンを再度エンタープライズ・サーチ・システムに追加できます。

正規表現アノテーターをさらに複雑にカスタマイズする場合は、UIMA SDK ツールを使用することを強くお勧めします。これらのツールは、タイプ・システム記述とディスクリプター・ファイルの作成または更新、可能であればアノテーターと他のアノテーターとの結合による集合分析エンジンの形成、およびエンタープライズ・サーチの中でアノテーターを使用するために必要なすべてのリソースを含んだ新規処理エンジン・アーカイブ (PEAR ファイル) の作成を支援します。これらの作業をサポートできるツールについての情報は、UIMA SDK の資料を参照してください。

手順

新規規則およびエンティティの追加によって正規表現アノテーターを適合させたり、既存の規則を変更したりする場合、提供されたサンプルの正規表現アノテーター PEAR ファイルを更新できます。次にその手順を示します。

1. システムの中に、xml という新規ディレクトリーを作成します。
2. `ES_INSTALL_ROOT/packages/uima/regex/` ディレクトリーの中のサンプルの規則ファイル `of_sample_regex_rules.xml` を xml ディレクトリーにコピーし、そのファイルを変更して、カスタム・パターン・マッチング規則を含むようにします。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。
3. 対応するタイプ・システム記述ファイル `of_sample_typesystem.xml` を、ディレクトリー `ES_INSTALL_ROOT/packages/uima/regex/` から xml ディレクトリーにコピーし、そのファイルを変更して、新規の規則が必要とするタイプの定義を含むようにします。
4. 新規規則を少しだけ追加したり、既存の規則を変更しただけであれば、アノテーター・ディスクリプターを変更する必要はありません。他にも変更を行うつもりであったり、追加のカスタム分析ステップを使用するのであれば、アノテーター・ディスクリプターを変更する必要があるかどうか確認してください。
5. 好みのアーカイブ・ユーティリティーを使用して、正規表現アノテーター PEAR ファイルのコピーを更新して、更新した 2 つのファイルを含めるように

します。例えば、ES_INSTALL_ROOT/packages/uima/regex/ から of_regex.pear ファイルを、作成した xml ディレクトリーの親ディレクトリーにコピーします。そのあと、Java jar コマンド行ツール (例えば、IBM Java SDK のパーツ) を使用して、次のコマンドをその親ディレクトリーから発行します。

```
"jar -uf of_regex.pear -C xml/ of_sample_regex_rules.xml"  
"jar -uf of_regex.pear -C xml/ of_sample_regex_typesystem.xml"
```

6. エンタープライズ・サーチ管理コンソールを使用して、正規表現アノテーターをカスタム・テキスト分析エンジンとしてエンタープライズ・サーチ・システムに追加し、それをテスト文書コレクションに関連付けます。
7. 文書コレクションのプロパティを更新し、XCAS ダンプ機能を使用して共通分析構造の中に保管された分析結果の表示可能な XML 出力を生成して、正規表現アノテーターによって生成された分析結果を確認します。
8. テスト文書进行处理し、XCAS Annotation Viewer を使用して、XML ファイルの内容を表示します。
9. カスタム正規表現に基づいてアノテーターにより作成された注釈が満足できるものであれば、再度文書コレクションのプロパティを編集して、パーサーが分析結果の表示可能な XML 出力を生成できないようにします。さらに規則セット・ファイルの変更が必要であれば、PEAR ファイルを更新するステップを繰り返す必要があります。
10. 分析結果の索引作成を行うために共通分析構造から索引へのマッピング・ファイルを作成するか、あるいは、結果をデータベースに追加する場合は共通分析構造からデータベースへのマッピング・ファイルを作成します。提供されたサンプルの共通分析構造から索引へのマッピング・ファイルを開始点として使用して、専用の共通分析構造から索引へのマッピング・ファイルを作成できます。
11. エンタープライズ・サーチ管理コンソールを使用して、マッピング・ファイルを追加し、それをユーザーの文書コレクション全体に関連付けます。
12. XML フラグメントまたは XPath 照会を使用するか、代わりに同義語検索で意味展開を使用して、注釈を検索します。

関連概念

86 ページの『正規表現アノテーターを使用した簡単なセマンティック検索』

関連タスク

13 ページの『ベース・アノテーターとカスタム・テキスト分析結果の表示』

アノテーター・ディスクリプター

正規表現アノテーター XML ディスクリプターは、アノテーターの実行に必要な正規表現アノテーターの記述情報を含みます。

正規表現アノテーターのみを使用し、追加のカスタム分析ステップがない場合で、さらに次に当てはまる場合は、必要なのはディスクリプターの変更のみです。

- 規則セット・ファイルのファイル名 (<externalResourceDependencies> エレメント内) を変更する。
- 複数の規則セット・ファイルを使用する。
- タイプ・システム記述ファイルの名前を変更する。

追加のカスタム分析ステップを使用し、さらに次に当てはまる場合は、ディスクリプターの変更が必要です。

- 正規表現アノテーターによって作成された注釈をカスタム分析が使用するようになる。この場合、アノテーター・ディスクリプターの中の出力機能を更新する必要があります。
- 前のカスタム分析ステップで作成された注釈タイプに一致する必要がある正規表現規則を定義した。この場合、アノテーター・ディスクリプターの中の入力機能を更新する必要があります。

UIMA SDK ツールを使用して、アノテーター・ディスクリプターを作成または更新し、エンタープライズ・サーチの中でアノテーターを使用するために必要なすべてのリソースが含まれた処理エンジン・アーカイブ (.pear ファイル) を再作成します。これらの作業をサポートできるツールについての情報は、<http://www.alphaworks.ibm.com/tech/uima/> にある UIMA SDK の資料を参照してください。

構成パラメーター

正規表現アノテーターは、String2NumberImpl と呼ばれる 1 つの構成パラメーターのみを持ち、これには、com.ibm.uima.an_regex.String2Number インターフェースをインプリメントするクラスの名前を設定する必要があります。正規表現アノテーターは、このクラスのインプリメンテーションと一緒に提供する必要があります。そうでないと、例外が発生します。正規表現アノテーターを必要に合わせてカスタマイズする場合、XML ディスクリプター・ファイルの中にクラス名を渡して、ユーザー独自の String2Number インターフェースのインプリメンテーションを提供できます。

String2Number インターフェースは、toInt(String) と toFloat(String) の 2 つのメソッドを宣言します。これらのメソッドは、整数または浮動小数点値のストリング表現を、対応する整数または浮動小数点値に変換します。これらの 2 つのメソッドを使用して、分離文字を含む数字を、有効な Java Integer または Float 値に変換します。

com.ibm.uima.an_regex.String2Number_impl のデフォルトのインプリメンテーションでは、ピリオド (.) を小数点とみなし、コンマ (,) を 1000 の単位の分離文字とみなします。例えば、テキスト文書に 1,999.00 が見つかり、toInt は、それを 1999 に変換します。toFloat は 1999.00 を戻します。

サンプル

ディスクリプターの構成パラメーター・セクションは次のようになっています。

```
<configurationParameters>
  <configurationParameter>
    <name>String2NumberImpl</name>
    <description>Implementation of the
      com.ibm.uima.an_regex.String2Number interface</description>
    <type>String</type>
    <multiValued>>false</multiValued>
    <mandatory>>true</mandatory>
  </configurationParameter>

  <configurationParameterSettings>
    <nameValuePair>
```



```

    <name>String2NumberImpl</name>
    <value>
      <string>com.ibm.uima.an_regex.impl.String2Number_impl</string>
    </value>
  </nameValuePair>
</configurationParameterSettings>
</configurationParameters>

```

機能

正規表現アノテーターの入出力機能と、それをサポートする言語は、アノテーター・ディスクリプターの機能セクションで定義されます。

ディスクリプター・ファイルの中の入力機能 (入力タイプ) は、規則セット・ファイルで使用される一致タイプに適合する必要があります。規則が `uima.tt.DocumentAnnotation` タイプのみを使用する場合、このタイプは常に定義されているため、入力機能を宣言する必要はまったくありません。他のすべてのタイプは定義する必要があります。

正規表現アノテーターによって作成された注釈タイプは、出力機能セクションに指定されます。これらのタイプは、規則セット・ファイルの中で宣言された出力タイプと一致している必要があります。

正規表現アノテーターは言語から独立しているため、`x-undefined` を指定しません。これは任意の言語を表します。

タイプ・システム記述

正規表現アノテーター XML ディスクリプターの中のタイプ・システム記述セクションで、アノテーターが使用するタイプ・システムを定義します。規則セット XML ファイルの中で使用されるタイプと、アノテーター・ディスクリプターの入出力機能セクションに指定されるタイプは、タイプ・システム記述で定義されるタイプと一致している必要があります。

サンプル

ディスクリプターのタイプ・システム記述セクションは、タイプ・システム・ディスクリプター XML ファイルをインポートします。

```

<typeSystemDescription>
  <imports>
    <import location="./xml/of_sample_regex_typesystem.xml"/>
  </imports>
</typeSystemDescription>

```

外部リソース

ディスクリプターの外部リソース・セクションは、アノテーターが必要とするファイルとクラスを含みます。

正規表現アノテーターは、規則セット・ファイルを必要とします。規則セット・ファイルは、`com.ibm.uima.an_regex.FileResource` インターフェースを通して正規表現アノテーターが使用できるようになり、そのインターフェースは、クラス `com.ibm.uima.an_regex.impl.FileResource_impl` によってインプリメントされます。カスタム規則を正規表現アノテーターに渡すには、規則セット・ファイルの名

前をアノテーター・ディスクリプターの中に指定し、ファイルの場所をクラス・パスに追加する必要があります。正規表現アノテーターが規則セット・ファイルにアクセスするために使用するキーは、RuleSetDefinition という名前です。このキーは変更しないでください。変更すると、正規表現アノテーターは規則セットを見つけることができず、アノテーターは初期化されません。

エンタープライズ・サーチ用にデプロイするカスタム・アノテーターは、UIMA datapath 設定を使用して、外部リソースを検索できません。外部リソースを検索するには、カスタム・アノテーターのクラスパスにあるリソースにパス名を指定します。PEAR 生成ウィザードを使用してカスタム・アノテーターのクラスパス設定を指定するには、<http://www.alphaworks.ibm.com/tech/uima/> の UIMA SDK 文書を参照してください。

サンプル

ディスクリプターの外部リソース・セクションは次のようになっています。

```
<externalResourceDependencies>
  <externalResourceDependency>
    <key>RuleSetDefinition</key>
    <description>Rule set definition</description>
    <interfaceName>com.ibm.uima.an_regex.FileResource</interfaceName>
    <optional>>false</optional>
  </externalResourceDependency>
</externalResourceDependencies>
<resourceManagerConfiguration>
  <externalResources>
    <externalResource>
      <name>of_samples_regex_rules</name>
      <description>Rule set definition file for room numbers</description>
      <fileResourceSpecifier>
        <fileUrl>file:of_samples_regex_rules.xml</fileUrl>
      </fileResourceSpecifier>
      <implementationName>
        com.ibm.uima.an_regex.impl.FileResource_impl</implementationName>
      </externalResource>
    </externalResources>
  <externalResourceBindings>
    <externalResourceBinding>
      <key>RuleSetDefinition</key>
      <resourceName>of_samples_regex_rules</resourceName>
    </externalResourceBinding>
  </externalResourceBindings>
</resourceManagerConfiguration>
```

関連概念

85 ページの『正規表現アノテーター』

88 ページの『規則セット・ファイル』

関連資料

『ロギング』

ロギング

正規表現アノテーターからのすべてのログ・メッセージは、現在のコレクションのログ・ファイルに書き込まれます。

コレクション・ログ・ファイルは `ES_NODE_ROOT/logs/` にあり、`<collection_id>_<current_date>.log` の形の名前です。ログ・ファイルは、`esviewlogs.sh/bat` スクリプトを使用して表示できます。

次の 7 つのロギング・レベルがあります。

- Error (エラー)
- Warning (警告)
- Info (情報)
- Config (構成)
- Fine (詳細-低)
- Finer (詳細-中)
- Finest (詳細-高)

エラーおよび警告メッセージのマッピングは変更できません。デフォルトでは、Info (情報)、Warning (警告)、および Error (エラー) メッセージのみがログ・ファイルに書き込まれます。これらはエンタープライズ・サーチが使用する標準のログ・レベルです。他のログ・レベルは、より詳細な情報にマップできます。

正規表現アノテーターからログ・メッセージを受け取るには、ログ・レベルは少なくとも Config (構成) に設定する必要があります。このレベルでは、アノテーターは、使用された規則セット・ファイル、および `com.ibm.uima.an_regex.String2Number` インターフェースのインプリメンテーション・クラス名などの構成設定をログに記録します。

ログ・レベルを例えば、Finer (詳細-中) に設定すると、アノテーターは、作成できなかった注釈をログに記録します。これによって、期待した注釈の一部が作成されなかった理由を判別できます。例えば、正規表現の 1 つにエラーがあったかもしれませんが。あるいは、オプションのキャプチャリング・グループが文書内のどのテキストとも一致しなかったかもしれません。同様に、フィーチャーがキャプチャリング・グループに一致するテキスト・シーケンスに設定されて、一致するテキスト・シーケンスがない場合、フィーチャーは NULL に設定されます。

最も詳細な情報を得るには、ログ・レベルを Finest (詳細-高) に設定します。このレベルでは、アノテーターは、現在の正規表現パターン、現在分析中の文書テキストの部分、および作成されたすべての注釈とフィーチャーをログに記録します。詳細なロギング、特にログ・レベルを Finer (詳細-中) および Finest (詳細-高) に設定すると、アノテーターの全体のパフォーマンスに悪影響を与えます。

詳細なログ・レベル・マッピングが必要であれば、`ES_NODE_ROOT/master_config/parserservice/` にある `tokenizer.properties` という構成ファイルを変更して、構成設定 `trevi.tokenizer.jedii.InformationalLevelMapping=Info` を `trevi.tokenizer.jedii.InformationalLevelMapping=Finest` に変更します。例えば次のとおりです。

ログ・レベルの変更をアクティブにするには、管理コンソールを使用して、すべてのパーサー・プロセスを停止する必要があります。そのあと、コマンド行から次を呼び出して、パーサー・サービス・セッションを停止して再始動する必要があります。

```
>esadmin session parserservice stop  
>esdamin session parserservice start
```

その後、解析が再開され、新規のログ・レベルが使用されます。ログ・レベルを変更するたびに、これらのステップを繰り返す必要があります。

関連概念

85 ページの『正規表現アノテーター』

88 ページの『規則セット・ファイル』

関連資料

93 ページの『アノテーター・ディスクリプター』

エンタープライズ・サーチの資料

OmniFind Enterprise Edition の資料は、PDF 形式または HTML 形式で読むことができます。

OmniFind Enterprise Edition のインストール・プログラムは、エンタープライズ・サーチ用資料の HTML バージョンを含む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。複数のサーバーをインストールする場合は、インフォメーション・センターが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なかった場合は、「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。

PDF 文書を参照するには、ES_INSTALL_ROOT/docs/locale/pdf に移動します。例えば、英語の資料を見つけるには、ES_INSTALL_ROOT/docs/en_US/pdf に移動します。

使用可能なすべての言語の PDF バージョンの文書にアクセスするには、OmniFind Enterprise Edition Version 8.5 documentationサイトを参照してください。

製品ダウンロード、フィックスパック、技術情報、およびインフォメーション・センターには、OmniFind Enterprise Edition Support サイトからアクセスすることもできます。

以下の表は、使用可能な資料、ファイル名、ロケーションを示します。

表 12. エンタープライズ・サーチの資料

タイトル	ファイル名	場所
インフォメーション・センター		http://publib.boulder.ibm.com/infocenter/discover/v8r5/
エンタープライズ・サーチ インストール・ガイド	iiysi.pdf	ES_INSTALL_ROOT/docs/locale/pdf/
クイック・スタート・ガイド (英語、フランス語、および日 本語ではハードコピー版も用 意されています。)	OmniFindEE850_qsg_ locale を表す 2 文字 .pdf	ES_INSTALL_ROOT/docs/locale/pdf/
エンタープライズ・サーチの 管理	iiysa.pdf	ES_INSTALL_ROOT/docs/locale/pdf/
<i>Programming Guide and API Reference for Enterprise Search</i>	iiysp.pdf	ES_INSTALL_ROOT/docs/en_US/pdf/
トラブルシューティング・ガ イドおよびメッセージ	iiysm.pdf	ES_INSTALL_ROOT/docs/locale/pdf/
テキスト分析機能ガイド	iiyst.pdf	ES_INSTALL_ROOT/docs/locale/pdf/
Google デスクトップ検索用 プラグイン	iiysg.pdf	ES_INSTALL_ROOT/docs/locale/pdf/

アクセシビリティ機能

アクセシビリティ機能は、運動障害や視覚障害といった身体的障害を持つユーザーが IT 製品を快適に使用できるように支援するものです。

IBM は、年齢や能力にかかわらず誰もが使用できる製品を提供するように努めております。

アクセシビリティ機能

OmniFind Enterprise Edition における主要なアクセシビリティ機能は次のとおりです。

- キーボードのみの操作
- スクリーン・リーダー (読み上げソフトウェア) が通常使用するインターフェース

OmniFind Enterprise Edition のインフォメーション・センターと関連資料はアクセシビリティ対応になっています。このインフォメーション・センターのアクセシビリティ機能の説明は http://publib.boulder.ibm.com/infocenter/discover/v8r5m0/topic/com.ibm.classify.nav.doc/dochome/accessibility_info.htm にあります。

キーボード・ナビゲーション

この製品では、標準的な Microsoft® Windows ナビゲーション・キーを使用します。

さらに、以下のキーボード・ショートカットを使用して、OmniFind Enterprise Edition インストール・プログラム内をナビゲートすることができます。

表 13. 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	Tab キー
プッシュボタンの強調表示	Tab 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	Tab 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

インターフェース情報

管理コンソール、サンプル検索アプリケーション、および検索アプリケーション・カスタマイザーのユーザー・インターフェースは、Microsoft Internet Explorer または Mozilla FireFox で表示できる、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。

関連するアクセシビリティ情報

Adobe Acrobat Reader を使用すれば、OmniFind Enterprise Edition の資料を、Adobe PDF で表示できます。これらの PDF は、本製品と同梱の CD に収録されていますが、<http://www.ibm.com/support/docview.wss?rs=63&uid=swg27010938> でアクセスすることもできます。

IBM とアクセシビリティ

IBM のアクセシビリティに対する取り組みの詳細については、IBM Human Ability and Accessibility Center を参照してください。

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)

コンピューター・セキュリティーにおいて、特定のオブジェクトに関連付けられているリスト。そのオブジェクトにアクセスできるすべてのサブジェクトとそれらのアクセス権限を識別する。

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。

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。

エンキュー (enqueue)

メッセージや項目をキューに入れること。

エンタープライズ・サーチ・ベース・アノテーター (enterprise search base annotators)

エンタープライズ・サーチの中でデフォルトの文書分析の処理に使用される標準テキスト分析エンジンのセット。

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。

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。

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。

開始 Uniform Resource Locator (URL) (start Uniform Resource Locator (URL))

クロールの開始点。

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テキスト分析。『テーマ抽出』も参照。

外部データ・ソース (external data source)

OmniFind Enterprise Edition によってクロール、解析、または索引付けされていないフェデレーションのデータ・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。

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と個人証明書に保管される秘密鍵の両方を含む鍵リング。

鍵データベース・ファイル (key database file)

『鍵リング』を参照。

鍵リング (key ring)

コンピューター・セキュリティーにおいて、公開鍵、秘密鍵、トラステッド・ルート、および証明書を含むファイル。『鍵ストア・ファイル』も参照。

カスタム・テキスト分析エンジン (custom text analysis engine)

構成解除情報管理アーキテクチャー (UIMA) Software Development Kit (SDK) の使用によって作成されるテキスト分析エンジンで、標準エンタープライズ・サーチ・テキスト分析エンジン (エンタープライズ・サーチ・ベース・アノテーターとも呼ばれる) のセットに追加できる。『テキスト分析エンジン』も参照。

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。

カテゴリー・ツリー (category tree)

カテゴリーの階層。

管理役割 (administrative role)

ユーザーに対してアクセス権限を規定するユーザーの種別。

共通通信層 (CCL) (Common Communication Layer)

OmniFind Enterprise Edition の各種コンポーネント (コントローラー、パーサー、クローラー、索引サーバー) を結合する通信インフラストラクチャー。

共通分析構造 (common analysis structure)

文書のコンテンツとメタデータ、および、テキスト分析エンジンによって生成されたすべての分析結果を保管する構造。文書分析中のすべてのデータ交換は、共通分析構造を使用して処理されます。

共通分析構造コンシューマー (CAS コンシューマー) (common analysis structure consumer (CAS consumer))

共通分析構造に保管された分析結果の最終処理を行うコンシューマー。例えば、コンシューマーは、検索エンジン内の共通分析構造のコンテンツの索引付けを行うか、あるいは、リレーショナル・データベースに特定の分析結果を取り込みます。

近接検索 (proximity search)

同一文内や同一段落内など、お互いから一定の距離内に 2 つ以上の一致条件があるとき結果を戻すテキスト検索。

クイック・リンク (quick link)

URI とキーワード、または URI と句との間の関連。

クレデンシャル (credential)

認証時に取得される詳細情報で、ユーザー、グループ関連、およびその他のセキュリティー関係識別属性を記述するもの。クレデンシャルは、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。例えば、あるユーザーのサインオン情報 (ユーザー ID とパスワード) は、そのユーザーがアカウントにアクセスすることを許可するクレデンシャルです。

クローラー (crawler)

データ・ソースから文書をリトリートし、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。

クロール・スペース (crawl space)

指定パターン (URL、データベース名、ファイル・システム・パス、ドメイン名、IP アドレスなど) に一致するソースの集合。クローラーはここからの読み取りを行って、索引用の項目をリトリートします。

言語の識別 (language identification)

エンタープライズ・サーチにおいて、文書の言語を決定する検索機能。

言語分析検索 (linguistic search)

基本型に戻したり (例: *mice* は *mouse* として索引付けされる)、または基本型を使用して拡張したり (複合語のように) した語句を使用して文書を表示、リトリート、索引付けする検索タイプ。

検索アプリケーション (search application)

エンタープライズ・サーチにおいて、照会の処理、索引の検索、検索結果の戻し、およびソース文書のリトリートを行うプログラム。

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。

検索結果 (search results)

検索要求に一致する文書のリスト。

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。

検出機能 (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。

合字 (ligature)

2 文字以上を連結することによって 1 文字として表示されるようにしたもの。例えば、ff や ffi は合字として表示できる文字です。

高頻度ランキング (popular ranking)

文書の検索頻度に基づいて文書の既存ランキングを上げるランキング・タイプ。

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。

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。

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。

索引 (index)

『フルテキスト索引』を参照。

索引キュー (index queue)

処理される主索引作成および差分索引作成の要求リスト。

索引付けしないディレクティブ (no-index directive)

Web ページ内のディレクティブで、そのページの内容を索引に含めないようロボット (Web クローラーなど) に指示するもの。

差分索引作成 (delta index build)

エンタープライズ・サーチ・システムにおいて、新しい情報を既存の索引に追加する処理。『主索引作成』と対比。

シード・リスト・ページ (seed list page)

WebSphere Portal の中で、ポータルで使用可能なページへのリンクを含む XML ページ。クローラーは、シード・リストを使用してクロールする文書を識別します。シード・リスト・ページには、クロールされた文書と一緒にエンタープライズ・サーチ索引に保管されるメタデータも含まれます。

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)。

字句類縁性 (lexical affinity)

文書内で互いに近い意味を持つ検索語間の関係。字句類縁性を使用して、結果の適合度を算出します。

主索引作成 (main index build)

エンタープライズ・サーチにおいて、索引全体を作成する処理。『差分索引作成』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。

証明書 (certificate)

コンピューター・セキュリティーにおいて、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るもの。証明書は、認証局によって発行され、その認証局によってデジタル署名されます。

処理エンジン・アーカイブ (processing engine archive)

Unstructured Information Management Architecture (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。

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、照会からストップワードを除去するプロセス。

正規表現アノテーター (regular expression annotator)

文書テキストで検索される厳密パターンを記述する正規表現に基づいて、テキスト文書内の情報エンティティや情報単位 (製品番号など) を検出するソフトウェア・コンポーネント。正規表現の 1 つが文書テキストの一部と一致すると、正規表現アノテーターは、一致の一部または全体を取り入れた、対応する注釈を作成します。これらの注釈の付いたテキストは、索引マッピング・ファイルを使用してエンタープライズ・サーチ索引に保管されるか、データベース・マッピング・ファイルを使用して JDBC 可能データベースに保管されます。

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。

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。

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。

セグメンテーション (segmentation)

テキストを明確な字句単位に分割すること。非辞書ベースの処理には空白文字と N-gram セグメンテーションが含まれ、辞書ベースのサポートには、ワード、文、段落のセグメンテーションとレンマタイゼーションが含まれます。

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 (*wouldn't* または *you're*)。

セマンティック検索 (semantic search)

キーワード検索のタイプで、言語分析と文脈分析を統合したもの。『テキスト分析』も参照。

ソフト・エラー・ページ (soft error page)

Web ページの 1 つのタイプで、要求された Web ページを戻せない理由を説明する情報を提供するもの。例えば、HTTP サーバーは、単純な状況コードを戻す代わりに、状況コードを詳しく説明するページを戻すことができます。

タイプ・システム (type system)

タイプ・システムは、文書の中でテキスト分析エンジンによって発見される可能性のあるオブジェクト (フィーチャー構造) のタイプを定義する。タイプ・システムは、タイプとフィーチャーに関してすべての可能なフィーチャー構造を定義します。タイプ・システムの中に、異なるタイプをいくつでも定義できます。タイプ・システムはドメインおよびアプリケーションに固有です。

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。Unstructured Information Management Architecture (UIMA) では、注釈は、特別な種類のフィーチャー構造です。

データ・ストア (data store)

文書が解析された形式で保持されるデータ構造。

データ・ソース (data source)

文書をリトリートできるデータ・リポジトリ。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。

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。

テキスト・セグメンテーション (text segmentation)

『セグメンテーション』を参照。

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。『セマンティック検索』も参照。

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。

デキュー (dequeue)

キューから項目を除去すること。

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。

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。

トークン化 (tokenization)

入力を解析してトークンを生成する処理。

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、その照会条件の同義語を検索できるようにする辞書。

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。

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。

ドキュメント・ハッシュ (shingle)

文から取り出される連続トークン (ワード) のストリング。例えば、「This is a very short sentence.」からの 3 ワード・ドキュメント・ハッシュ (つまり隣り合う 3 ワード) は次のとおりです。

This is a
is a very
a very short
very short sentence

ドキュメント・ハッシュは、統計言語学で使用できます。例えば、2 つの異なるテキストに多くの共通するドキュメント・ハッシュが含まれている場合、これらのテキストはおそらく何らかの形で関連しています。

認証局 (certificate authority)

信頼できる第三者機関または企業で、デジタル署名および公開鍵と秘密鍵のペアの作成に使用されるデジタル証明書を発行するもの。認証局は、固有の証明書を付与される個人の身元を保証します。

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。

パーサー・サービス (parser service)

文書コレクション全体にわたって、すべての文書の解析とテキスト分析処理を行うエンタープライズ・サーチ・サービス。少なくとも 1 つのパーサー・サービスが常に実行中です。

パーサー・ドライバー (parser driver)

エンタープライズ・サーチにおいて、パーサー・サービスに文書を供給するサービス。それぞれのコレクションに 1 つのパーサー・ドライバーがあります。コレクションのパーサー・ドライバー・サービスは、エンタープライズ・サーチ管理コンソールの中のコレクションのパーサーに対応します。

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。

発音区別符号 (diacritic)

文字または文字の組み合わせの音価の変化を示す符号。

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。

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。

ファジー検索 (fuzzy search)

検索語にスペルが似た語を戻す検索。

フィーチャー・パス (feature path)

Unstructured Information Management Architecture (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 - 値の構造をしています。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。

フィールド (field)

特定のカテゴリのデータや制御情報が入力される領域。

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りリストを戻す検索機能。

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。

フリー・フォーム・テキスト (free-form text)

語や文から成る非構造化テキスト。

フルテキスト索引 (full-text index)

データ項目を参照して、照会条件を含む文書を検索によって見つけれられるようにするデータ構造。

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。

プロキシ・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・サーバーは、エンタープライズのコンテンツ・サーバーの代理として動作します。

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)

アナテーターが生成する情報。分析結果は、共通分析構造と呼ばれるデータ構造に書き込まれます。カスタム・テキスト分析エンジン (アナテーター) によって生成された分析結果は、エンタープライズ・サーチ索引に含めることによって、検索可能にできます。

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とサブカテゴリに編成される。『カテゴリ・ツリー』も参照。

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。

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。

見出し語 (lemma)

ワードの基本型。見出し語は、チェコ語など、大きく語形変化する言語では重要です。

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。

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。

ユーザー・エージェント (user agent)

Web をブラウザし、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。

用語加重検索 (weighted term search)

一定の用語が重視される照会。

要約 (summarization)

文書の内容を簡潔に説明する非冗長文を検索結果に組み込む処理。『動的的要約』と『静的要約』も参照。

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。『Lotus Domino Document Manager ライブラリー』を参照。

ランキング (ranking)

照会によって得られた検索結果の各文書に整数値を割り当てること。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。

ランキング調整クラス (boost class)

検索結果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を含むオブジェクト。

ランキング調整ワード (boost word)

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。

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。

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。

リンクをたどらないディレクティブ (no-follow directive)

Web ページ内のディレクティブで、そのページで検出されたリンクをたどらないようロボット (Web クローラーなど) に指示するもの。

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回答し、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 (rule-based category)

どの文書が、どのカテゴリ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。例えば、一定の語を含む、または含まない文書や一定の Uniform Resource Identifier (URI) パターンに一致する文書を、特定のカテゴリと関連付ける規則を定義する。

レンマタイゼーション (lemmatization)

語の原形とさまざまな文法的形式とを識別する処理。例えば、「mouse」という語を検索すると語「mice」を含む文書も検出され、「go」という語を検索すると「going」、「gone」、または「went」を含む文書も検出されます。

ロー・データ・ストア (raw data store)

クローリング済みの文書がパーサーに送られる前に保管されるデータ構造。クローラーはロー・データ・ストアに書き込み、パーサーはロー・データ・ストアから読み取ります。文書が解析されると、ロー・データ・ストアから除去されます。データ・ストアと混同しないでください。

ローカル・フェデレーター (Local Federator)

エンタープライズ・サーチ・アプリケーションにおいて、検索および索引 API によって作成されるクライアント・オブジェクトで、ユーザーが異種コレクションの集合を検索し、一連の統一された検索結果を取得できるようにするもの。

ロボット排除プロトコル (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。

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。

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。

ID 管理 (identity management)

保護データへのアクセスを制御し、ユーザーがコレクション内の各リポジトリごとにユーザー ID とパスワードを指定しなくてもそのコレクションを検索できるようにするエンタープライズ・サーチ API の集合。

IP アドレス (IP address)

ネットワーク上のデバイスあるいは論理装置の固有のアドレスで、IP 標準に従うもの。

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。

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のソフトウェア・インプリメンテーション。

JavaScript™

ブラウザおよび 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に実行するサーバー・スクリプト・テクノロジー。

Lightweight Directory Access Protocol (LDAP)

オープン・プロトコルの 1 つで、TCP/IP を使用して、X.500 モデルをサポートするディレクトリーにアクセスできるようにするが、より複雑な X.500 ディレクトリー・アクセス・プロトコル (DAP) のリソース要件を負わないもの。例えば、LDAP を使用して、インターネットまたはイントラネット・ディレクトリーで個人、組織その他のリソースを検索できます。

Lotus 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 Lotus Domino Document Manager データベース。キャビネットが Lotus Domino データベースを保持します。

Lotus Domino Document Manager ライブラリー (Domino Document Manager library)

Lotus Domino Document Manager に対するエントリー・ポイントである Lotus Domino Document Manager データベース。

Lotus Domino Internet Inter-ORB Protocol (IIOP)

サーバー上で稼働し、Lotus Domino Object Request Broker と連動して、

Lotus Notes® Java クラスを使用して作成される Java アプレットと Lotus 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・ユーザーおよび Lotus Domino サーバーは、DIOP を使用して通信し、オブジェクト・データを交換します。

Lotus 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® の通信機構。

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、および集会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。

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。

N-gram セグメンテーション (n-gram segmentation)

Unicode ベースの空白文字の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。

Portal Document Manager (PDM)

チーム・コラボレーションのために、ユーザーに文書の中央リポジトリを 1 つ持たせるもの。管理者は文書を効果的に管理する能力を持ち、ユーザーが情報と対話する方法を制御できます。

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。SSL により、クライアント/サーバー・アプリケーションは通信中の盗聴、改ざん、およびメッセージ偽造を回避することができます。

Unicode ベースの空白文字の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 文字プロパティを使用するトークン化の方式。

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 IBM アーキテクチャー。

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)

インターネットなどのネットワークを使用してアクセスできる情報リソースの固有アドレス。URL 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と、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とが含まれます。

Web クローラー (Web crawler)

クローラーの 1 つのタイプで、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 を探索するもの。

XML パス言語 (XPath) (XML Path Language (XPath))

XSLT、XQuery、および XML パーサーなどの XML 関連技術との併用目的で、ソース XML データの各部分を一意的に識別したり処置したりするように設計された言語。XPath は World Wide Web Consortium 標準です。

特記事項および商標

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す。

本書に記載の製品、サービス、または機能が日本においては提供されてい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。

〒106-8711

東京都港区六本木 3-2-12

日本アイ・ビー・エム株式会社

法務・知的財産

知的財産権ライセンス渉外

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を提供し、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せんが、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認できません。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をお願いします。

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ます、単に目標を示しているものです。

表示されている IBM の価格は IBM が小売り価格として提示しているもので、現行価格であり、通知なしに変更されるものです。卸価格は、異なる場合があります。

本書はプランニング目的としてのみ記述されています。記述内容は製品が使用可能になる前に変更になる場合があります。

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。

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。

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的創作物にも、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。

© (お客様の会社名) (西暦年). このコードの一部は、IBM Corp. のサンプル・プログラムから取られています。 © Copyright IBM Corp. _年を入れる_. All rights reserved.

この製品には次のものが含まれています。

- Oracle® Outside In Content Access, Copyright © 1992, 2008, Oracle. All rights reserved.
- IBM XSLT Processor Licensed Materials - Property of IBM © Copyright IBM Corp., 1999-2008. All Rights Reserved.

商標

IBM の商標については、<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。

Adobe、PostScript は、Adobe Systems Incorporated の米国およびその他の国における登録商標または商標です。

Intel、Intel (ロゴ)、Intel Inside、Intel Inside (ロゴ)、Intel Centrino、Intel Centrino (ロゴ)、Celeron、Intel Xeon、Intel SpeedStep、Itanium、Pentium は、Intel Corporation または子会社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。

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です。

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

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。

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。

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。

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]

エンタープライズ・サーチ用 HTML 資料 99
エンタープライズ・サーチ用 PDF 資料 99
送り仮名 82

[カ行]

カスタム分析
 カスタム分析結果の索引付けの方法 41
 基本分析モードから拡張分析モードへの変更 16
 タイプ・システム記述 15
 タイプ・システム記述のサンプル 25
 テキスト分析アルゴリズム 5
 分析および検索における XML マークアップの使用法 28
 ワークフロー 6
 JDBC 対応データベースへの分析結果のマッピング 48, 49, 50, 55
カスタム分析結果の索引付け
 共通分析構造から索引へのマッピング・ファイルの作成 42
 説明 41
カスタム分析結果へのアクセス
 組み込み フィーチャー 37
 フィーチャー・パスの定義 36
 フィルター 40
簡単なセマンティック検索
 正規表現アノテーターの使用 86
言語検出 77
言語サポート
 送り仮名 82
 言語検出 77
 サポートされる言語 80
 辞書ベース・セグメンテーション 80
 システム定義タイプおよびフィーチャー 17
 システムに組み込まれているサポート 77
 数字の N-gram セグメンテーション 79

言語サポート (続き)
 ストップワードの除去 82
 接語 80
 説明 1
 セマンティック検索 62
 日本語における語のセグメンテーション 82
 日本語における変種文字 82
 非辞書ベースのセグメンテーション 78
 見出し語 80
 文字の正規化 83
 レンマタイゼーション 80
 N-gram セグメンテーション 78
 Unicode の正規化 83
 unicode ベースの空白文字のセグメンテーション 78
検索アプリケーション
 ストップワード・サポート 69
 同義語サポート 65
 ランキング調整 ワード・サポート 73
検索サーバー
 ストップワード XML ファイル 70
 ストップワード辞書の作成 71
 同義語 XML ファイル 66
 同義語辞書の作成 67
 ランキング調整ワード XML ファイル 74
 ランキング調整ワード辞書の作成 75
語のセグメンテーション、日本語 82

[サ行]

サポートされる言語
 言語検出 77
 辞書ベースの言語処理 80
辞書ベースの分析 80
辞書ベース・セグメンテーション 80
資料
 検索 99
 HTML 99
 PDF 99
スクリプト
 esboostworddictbuilder 75
 esstopworddictbuilder 71
 essyndictbuilder 67
ストップワード 82
ストップワード辞書
 検索アプリケーション・サポート 69
 DIC ファイルの作成 71
 XML ファイルの作成 70

ストップワードの除去 82
正規表現アノテーター
 アノテーター・ディスクリプター 93
 カスタマイズ 92
 簡単なセマンティック検索 86
 簡単なセマンティック検索の使用可能化 86
 正規表現規則の定義 89
 説明 85
 ログイン 97
 XML 規則セットの説明 88
セグメンテーション
 辞書ベースの 80
 非辞書ベースの 78
 Unicode ベースの空白 78
接語 80
セマンティック検索
 照会に一致する文書の部品の取得 59
 説明 62
 セマンティック検索照会 62

[タ行]

テキスト分析結果へのアクセス
 CAS コンシューマーの定義 35
同義語辞書
 検索アプリケーション・サポート 65
 DIC ファイルの作成 67
 XML ファイルの作成 66

[ナ行]

日本語における変種文字 82

[ハ行]

非辞書ベースのセグメンテーション 78
非辞書ベースの分析 78
本製品のアクセシビリティ機能 101

[マ行]

見出し語 80
文字の正規化 83

[ラ行]

ランキング調整ワード辞書
 検索アプリケーション・サポート 73
 DIC ファイルの作成 75

ランキング調整ワード辞書 (続き)

XML ファイルの作成 74
レンマタイゼーション 80

D

DIC ファイル

同義語 67
ユーザー定義のストップワード 71
ランキング調整ワード 75

E

esboostworddictbuilder.bat スクリプト 75
esboostworddictbuilder.sh スクリプト 75
esstopworddictbuilder.bat スクリプト 71
esstopworddictbuilder.sh スクリプト 71
essyndictbuilder.bat スクリプト 67
essyndictbuilder.sh スクリプト 67

J

JDBC 対応データベースへの 分析結果の
マッピング

ステップ 49
説明 48

JDBC 対応データベースへのカスタム分析
結果のマッピング

共通分析構造からデータベースへのマ
ッピング・ファイル 50

コンテナ・タイプ 55

「コンテナ・タイプ」のマッピング
55

ロード・ファイル・セットの使用 49

N

N-gram セグメンテーション

完全 79
数字 79
説明 78
標準 79

U

UIMA

カスタム・テキスト分析サポート 3

基本概念 4

正規表現アノテーターの使用 13

説明 3

データベース・コンシューマーへの共
通分析構造の使用 10

ベース・アノテーターとカスタム・テ
キスト分析結果の表示 13

UIMA (続き)

ベース・エンタープライズ・サーチ・
アノテーターの実行 7

ベース・エンタープライズ・サーチ・
アノテーター 7

Unicode の正規化 83

unicode ベースの空白文字のセグメンテー
ション 78

X

XML 文書構造から UIMA タイプへのマ
ッピング

説明 28

XML 文書構造から UIMA タイプへのマ
ッピング

XML エレメントから共通分析構造へ
のマッピング・ファイルの作成 30

IBM



SD88-6728-02



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